BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

SIMULACIÓN DE UN SISTEMA PARA

LA OPTIMIZACION DE OPERACIONES DE E/S EN CONSULTAS SQL

T E S I N A

QUE PARA OBTENER EL TITULO DE:

LIC. EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

JOSÉ SILVESTRE GARCÍA VALDEZ

ASESORA:

M.C. ERICA EUGENIA VERA CERVANTES.

PUEBLA, PUEBLA **2008**

CONTENIDO

Capítulo 1

Modelo relacional

Existen enfoques muy útiles y adecuados para describir y construir las estructuras de datos, por mencionar el orientado a objetos y el de entidad-relación, pero en la actualidad la mayoría de las aplicaciones que se han venido desarrollando sobre las bases de datos están basadas en la teoría formal denominado *el Modelo Relacional de datos* **[1]**. Más aun, la mayoría de las investigaciones que se han realizado durante los últimos años sobre el desarrollo de base de datos se basan en este *Modelo*, debido a esto, todas aquellas bases de datos que se han apegado a él son hoy en día las que han tenido éxito en los procesamientos comerciales.

Éste, ha sido y será uno de los *Modelos* más utilizados debido a que nos ofrece una manera única de representar a los datos y porque en él se han establecido los sistemas comerciales de base de datos más importantes. Con frecuencia los diseñadores inician implementando un sistema aplicando el modelo entidad-relación o un modelo cimentado en objetos y posteriormente lo traducen al *Modelo Relacional* para su implementación.

El *Modelo Relacional* además se ocupa sólo de aspectos lógicos, no de aspectos físicos y los principales aspectos que aborda de los datos es su estructura, su integridad y la manipulación de los datos **[2]**.

1.1 Estructura de las Bases de Datos Relacional.

Las estructuras que son utilizadas para representar a las bases de datos deben tener la característica de ser flexible ante los constantes cambios que está expuesta toda organización. Cuando es necesario implementar algunos cambios

para mejorar el funcionamiento de la organización, las bases de datos deben almacenar nuevos datos y acomodar nuevas relaciones para apoyar posteriormente las nuevas decisiones, es por ello que las bases de datos deben tener opciones para permitir que estos cambios se puedan efectuar **[4]**.

En el M*odelo Relacional* una base de datos es percibida por los usuarios como una conjunto de variables de relación o de manera más informal como tabla bidimensional denominada relación **[1]**. En otras palabras las variables de relación representan una abstracción de la forma en que los datos están almacenados físicamente; una abstracción en la cual se le ocultan a los usuarios diversos detalles del nivel de almacenamiento, como la ubicación y la secuencia de los registros almacenados. A cada una de las variables de relación se le asigna un nombre único.

Una variable de relación consta de dos partes: Un encabezado y un cuerpo, el encabezado es un conjunto de atributos, estos, son utilizados para identificar a cada una de las columnas de la relación mientras que el cuerpo es un conjunto de tuplas que se apegan al encabezado. Al número de atributos del encabezado se le denomina grado, mientras que al número de tuplas del cuerpo se le conoce como cardinalidad, una tupla corresponde a una fila de dicha variable de relación y un atributo a una columna; y un dominio que es solo un tipo de datos (posiblemente integrado o definido por el sistema; para ser más específicos definido por el usuario).

Para hacer corresponder la definición anterior con las que se utilizan en la práctica se puede considerar que una variable de relación puede percibirse como un archivo pero con las siguientes restricciones **[5]**:

- Cada “archivo” contiene sólo un tipo de tuplas.
- Los atributos no tienen un orden específico, de izquierda a derecha.
- Las tuplas no tienen un orden específico, de arriba hacia abajo.

- Cada campo tiene un sólo valor.
- Las tuplas poseen un campo identificador único (o combinación de campos) llamado llave primaria.

1.2 Restricciones de Integridad.

El problema de la integridad lo relacionamos al aseguramiento de que los datos que se van a almacenar en las bases de datos sean exactos o correctos, es decir que los datos sean verificados de inmediato en el momento de almacenarlos y rechazarlos si este no es del tipo apropiado de los atributos en la base de datos.

Una base de datos relacional está sujeta a un conjunto de restricciones de integridad (en general) de una complejidad arbitraria. Por lo tanto, todos los sistemas administradores de base de datos relacionales deben estar bien informados de todas estas restricciones en que las bases de datos están sujetas para poderlas hacerlas cumplir de alguna manera (básicamente rechazando cualquier actualización que de otro modo las violaría).

1.2.1 Restricciones de tipo.

Con este tipo de restricciones nos referimos a la enumeración de los valores válidos de tipo. En consecuencia, podemos decir que una relación que cuente con este tipo de restricciones todos los valores que se traten de introducir son verificados de inmediato y por lo tanto, ninguna relación puede adquirir nunca un valor para ningún atributo de cualquier tupla si éste no es de tipo apropiado (por supuesto, en un sistema que maneje las restricciones de tipo).

Cualquier valor dado siempre será exactamente un tipo y no podrá cambiar jamás su tipo. En otras palabras si tipo es un valor, entonces se puede pensar en tipo

como si portara una bandera que anuncia “soy entero” o “soy un número de cliente”, etc.

1.2.1.1 Restricciones de atributo.

El inicio de tipo de restricciones es en el momento de la definición de éste con los valores válidos para cada atributo y nunca deberán ser violados (dando por hecho que las restricciones de tipo están verificadas).

En otros términos, las restricciones de atributo son parte de la definición del atributo en cuestión y pueden ser identificadas por medio del nombre del atributo correspondiente. De aquí que una restricción de atributo solo puede ser eliminado mediante la eliminación del propio atributo (lo cual en la práctica significa generalmente eliminar la relación que lo contiene).

1.2.1.2 Restricciones de varrel.

Este tipo de restricciones son las que se le asignan a una relación individual (ésta se expresa solamente en términos de la relación en otros aspectos puede ser compleja). Es decir una varrel con éste tipo de restricciones solamente podrá admitir los tipos de valores correctos y exactos, visto de una forma nunca registrarán información acerca de algo que no se pueda identificar **[5]**.

Las restricciones de varrel siempre al igual que las restricciones de tipo anteriores son verificadas de inmediato (en realidad, como parte de la ejecución de cualquier instrucción que pudiera ocasionar que fueran violadas). Por lo tanto, cualquier instrucción que intente asignar un valor a una relación dada que viole cualquier restricción para esa relación, será rechazada.

1.2.1.3 Restricciones de base de datos.

Una restricción de este tipo es aquella que relaciona dos ó mas relaciones distintas, es decir realiza una junta natural entre las relaciones para posteriormente realizar una selección de valores de atributos que satisfagan una restricción.

1.2.1.4 Clave.

El uso de este tipo de estricciones son para que cada una de las tuplas que forman parte de una relación posean un identificador único o también se puede definir como una restricción de una relación.

Es importante señalar del *Modelo Relacional* que con el uso de claves no se permitirá las redundancia de la información, es decir no existirán tuplas duplicadas en una relación.

Todas las relaciones en *Modelo Relacional* tiene por lo menos una clave llamada clave primaria, ésta posee dos propiedades importantes la de unicidad, que hace que ningún valor de una relación contenga dos tuplas distintas con la misma clave y la de irreductibilidad, esto es que ningún subconjunto de atributos de la relación tiene la propiedad de unicidad, es decir, con esto el sistema manejador de la base de datos estará al tanto del estado de las cosas y con esto hace cumplir las restricciones de integridad.

En una relación podemos encontrar que existen más de un identificador único, en este caso especial se dice que la relación posee varias claves candidatas; se selecciona a una de estas claves coma la clave primaria, llamando al resto claves alternativas. Así una clave alternativa es una clave candidata que no es la clave primaria **[5]**.

1.3. Álgebra Relacional.

En esta sección se presenta un panorama general muy breve de la tercera y última parte que se encuentra relacionado con la parte manipulativa del *Modelo Relaciona.* Éste ha evolucionado considerablemente desde que se realizaron las primeras publicaciones originales hasta la actualidad. Sin embargo, todavía el principal componente de este aspecto es lo que se denomina álgebra relacional, que básicamente consiste de un conjunto de operadores de alto nivel que operan sobre las relaciones . Analizaremos de forma más detalladamente al álgebra relacional en el siguiente capítulo.

1.4 Cálculo Relacional.

Así como el *Modelo Relacional* está basado en el álgebra relacional, podemos decir que también está basada al cálculo relacional, por el hecho de que el álgebra y el cálculo relacional son lógicamente equivalentes, porque para cada expresión algebraica existe una equivalencia del cálculo, y para cada expresión del cálculo existe una algebráica equivalente. Existe una correspondencia de uno a uno entre ambos. Por los tanto una diferencia entre ellos es en realidad una diferencia de estilo, es decir mientras el álgebra proporciona un conjunto de operadores explícitos que son utilizados para indicar al sistema como construir un relación deseada a partir de relaciones dadas, el cálculo simplemente proporciona una notación para establecer la definición de esa relación deseada en términos de dichas relaciones dadas [2]. Analizaremos en forma más detallada al cálculo relacional en el siguiente capítulo.

1.5 Referencias.

[1] Ullman Jefrey D., Widom Jennifer., “Introducción a los sistemas de Base de Datos”, Prentice Hall, México 1999.

[2] C.J. Date, “Introducción a los sistemas de bases de datos”, Addison Wesley Longman, 1998.

[3] Korth Henry F., Silberschatz Abraham, “Fundamentos de base de datos”, Mc Graw Hill, 1996.

[4] Awryszkiewycz I. T., “Análisis y Diseño de Base de Datos”, Megabyte, 1994.

[5] García Fernández Jesús, Análisis Y Diseño de Base de Datos”, Diplomado en computación. 1ª. Edición, Abril/97, Pue., Méx.

Capítulo 2

Consultas sobre Base de Datos Relacional

El propósito de este capítulo es hacer referencia a tres lenguajes de consultas, clasificados como procedimental o no procedimental, así como también de un lenguaje de consulta que posea características de estos dos enfoques. Con el uso de un lenguaje procedimental, el usuario solamente proporciona las instrucciones al sistema para que éste lleve acabo la secuencia de las operaciones sobre la base de datos y de esta forma obtener el resultado deseado. Mientras que con el uso de un lenguaje no procedimental, el usuario solamente describe la información deseada sin dar el procedimiento específico para obtener esa información **[3]**.

2.1. Álgebra relacional.

*Lenguaje de consulta catalogado como procedimental, básicamente consiste de una colección de operadores de alto nivel que actúan en una o dos relaciones como parámetros, obteniéndose con ello una nueva relación como salida **[3]**. De acuerdo a las primeras publicaciones del álgebra relacional los operadores especiales son* restringir, proyectar, junta natural y división. *Además de éstos existen otros denominados operadores tradicionales como:* unión, intersección, diferencia y producto cartesiano *(todos estos aplicados a las relaciones) **[1], [5]**.*

A los operadores esenciales *restringir y proyectar* se les conoce como operadores *unarios* debido a que actúan sobre una relación. El resto de ellos son clasificados como *binarios* por operar sobre pares de relaciones **[3]**.

Se hace mención que quizás algunos de los operadores se describen de manera muy sencilla, otros, tal vez de forma más detallada, debido a que no todos se les dará uso en la elaboración de las consultas a optimizar (operaciones de E/S).

A manera general y muy breve proporcionaremos una visión más clara de dichos operadores.

Restringir.

Conocido hoy en día como *seleccionar* genera una relación compuesta por todas las tuplas de una relación especificada que satisfagan un predicado (es decir una condición especificada). La denotaremos con el símbolo "**σ**".

A manera de ejemplo consideremos una consulta muy sencilla que estaremos utilizando con el álgebra relacional y para esto tomemos la relación ***C*** (numc#, nombre, dirección, ciudad): cliente, que se utilizará en el desarrollo de ésta tesina. Así, para *seleccionar* todas aquellas tuplas de la relación *cliente* en las que el valor del atributo ciudad es "Puebla", simplemente la escribiremos:

$$\sigma_{\text{ciudad = "Puebla"}} (C)$$

Más aun, este operador nos permite realizar una gama de comparaciones utilizando los símbolos $=, \neq, <, \leq, >,$ y $\geq$, en el predicado de dicho operador, además también la combinación de predicados en un predicado más complejo utilizando los conectores lógicos $\wedge$ y $\vee$. A manera de un ejemplo muy sencillo retomemos el ejemplo anterior y supongamos ahora que deseamos obtener aquella tupla pertenecientes a la relación *cliente* y cuyo valor del atributo "numc#" tenga como valor de 130, escribiremos:

$$\sigma_{\text{ciudad = "Puebla"} \wedge \text{numc\#= 130}} (C)$$

Proyectar.

La proyección de una relación sobre un subconjunto de sus atributos es una relación definida sobre ellos formada por todas aquellas tuplas o subtuplas que se originan después de eliminar los atributos especificados, no existiendo además las tuplas duplicadas que hubieran podido resultar. Este operador lo identificaremos con la letra griega "Π" . Los atributos que se desean que pertenezcan a la relación resultante se listarán como subíndice de Π.

Tomando como referencia las proyecciones que utilizaremos en el desarrollo de ésta tesina, así como también de la relación "**C**", supongamos que deseamos obtener una relación que presente los atributos de "nombre" y "dirección" de la relación "**C**" que ha realizado un pedido. Simplemente escribimos

$$\Pi_{\text{nombre, dirección}} (c)$$

Ahora supongamos que deseamos obtener el nombre del cliente que tiene asignado el número 130 y que ha solicitado un pedido. Escribimos

$$\Pi_{\text{nombre}} (\sigma_{\text{numc\# = 130}} (c))$$

Observemos que en vez de proporcionar el nombre de una relación como argumento de la operación proyección, se escribe una expresión que evalúa a una relación.

Producto cartesiano.

Producto cartesiano o también conocido simplemente como Producto de dos relaciones de cardinalidad m y n es una relación cuyo esquema estará determinado sobre la unión de los atributos de ambas relaciones, y cuya cardinalidad estará constituida por las n x m tuplas formadas concatenando cada tupla de la primera

relación con cada una de las de las tuplas de la segunda. A este tipo de operador lo denotaremos con la letra “x”

En el *producto cartesiano* de dos relaciones que tengan nombres de atributos comunes, lo recomendable es renombrar dichos atributos utilizando simplemente el operador *renombrar*.

Renombrar.

Operación relacional que es utilizado simplemente para renombrar a los atributos dentro de una relación especificada. Las aplicaciones que se le dan a esta operación es para cuando se desea generar el producto cartesiano de dos relaciones que tiene nombres de atributos comunes, primeramente se debe usar este operador para renombrar estos atributos en forma adecuada. Evitando así el problema de generar un encabezado con dos atributos iguales.

Unión.

La Unión de dos relaciones compatibles en su esquema es otra relación generada sobre el mismo esquema de relación cuya extensión estará formada por las tuplas que pertenezcan a una de las dos relaciones o a ambas(no existiendo las tuplas duplicadas debido a que son relaciones). A este tipo de operador lo denotaremos con el símbolo “$\cup$”.

Diferencia.

La diferencia entre dos relaciones compatibles en su esquema es otra relación determinada sobre el mismo esquema de relación, cuya extensión estará dada por el conjunto de tuplas que pertenecen a la primera relación y no a la segunda. A este operador se denota con el símbolo “-“.

Las operaciones fundamentales del álgebra relacional descritas son suficientes para poder realizar cualquier consulta en álgebra relacional. Es importante señalar que al utilizar solamente a estas operaciones, algunas consultas comunes serian largas de expresar. Por tanto, hablaremos de operaciones adicionales que no añaden ninguna potencia al álgebra relacional, pero simplifican a las consultas comunes.

Intersección.

La intersección de dos relaciones compatibles en sus esquemas es otra relación originada por el mismo esquema de relación, cuya extensión estará formada por las tuplas que pertenecen a ambas relaciones. A este operador se le denota con el símbolo "∩".

Junta natural.

La junta natural de dos relaciones es una combinación por igualdad donde se ha eliminado de la relación obtenida uno de los atributos duplicados. Esta operación es una de las más utilizadas para combinar aquellas relaciones que poseen atributos comunes.

División

La división de dos relaciones genera a una relación formada por todas las tuplas que al completarse con las tuplas de la segunda permiten obtener la primera. Se denota por el símbolo "**:**".

2.2 Cálculo relacional de tuplas.

Lenguaje no procedimental denominado también como lenguaje predicativo, por establecerse en una rama de las matemáticas denominada cálculo de predicados, aunado a esto, otra de las características fundamentales es la utilización de una variable de tuplas cuyos valores únicos y permitidos son tuplas de la relación especificada, es decir, si la variable de tuplas al recorrer toda una relación, en un tiempo determinado dicha variable representará una tupla de la relación **[3].**

2.3 Cálculo relacional de dominios.

Íntimamente relacionado con el cálculo relacional de tuplas, difiere en el uso de variables de dominio en vez de las variables de tuplas, es decir, variables que hacen el recorrido a dominios tomando dichos valores de un atributo en vez de hacerlo a valores de una tupla completa, otras de las diferencias y más evidente entre los cálculos de dominio y de tuplas, es que los cálculos de dominios permiten el uso de una condición de pertenencia, dicho de otra manera dada una relación con sus respectivos atributos y una variable de dominio, al evaluar la condición se obtendría un valor verdadero si y solo sí, la relación contiene los valores especificados en los atributos indicados **[3].**

2.4 SQL ("Structured Query Languaje")

El lenguaje SQL es una evolución del lenguaje SEQUEL desarrollado por IBM, posteriormente fue normalizado por el Instituto Americano de Normalización (ANSI) **[5]**.

El SQL utiliza una combinación de construcciones tanto del álgebra como del cálculo relacional, esto lo hace muy amigable para los usuarios que trabajan actualmente con base de datos relacionales el cual es soportado prácticamente por todos los productos en el mercado que manejan bases de datos. Otras de las capacidades del SQL además de realizar consultas a base de datos, están las de definir la estructura de los datos, para actualizar los datos de la base de datos y para determinar las restricciones de seguridad **[3]**. Cabe hacer mención que únicamente presentamos algunas de las características que posee este lenguaje y que han sido tomadas como bases y referencias para entender y desarrollar el tema de la tesina.

2.4.1 Estructura básica

Una consulta en el SQL es comparada a una selección en el álgebra relacional, dicha consulta como todas las permitidas son muy simples, las cuales solo constan de tres cláusulas claves que la caracterizan **select, from y where [3], [5]**.

- **select**, concierne a la operación de proyección del álgebra relacional, es decir determina los atributos de las tuplas que cumplen las condiciones.
- **from**, concierne a la operación del producto cartesiano del álgebra relacional, es decir, contiene la relación o relaciones a la especificada por la consulta.
- **Where**, condición semejante a la característica del operador selección del álgebra relacional, que deben cumplir las tuplas para corresponder a la consulta.

En una consulta típica en SQL primeramente se forma el producto cartesiano de las relaciones nombradas en la cláusula **from**, después se realiza una selección del álgebra relacional usando el predicado de la cláusula **where** y finalmente se proyecta el resultado a los atributos de la cláusula **select**. El resultado de esta consulta, es por supuesto, una relación.

2.4.2. Operaciones de conjuntos y tuplas duplicadas

Los lenguajes de consulta formales se basan en la noción matemática de relación como un conjunto. Por ello nunca aparecen tuplas duplicadas en las relaciones. SQL (y la mayoría de todos los lenguajes de consulta comerciales) permiten duplicados en las relaciones. Para los casos en los que se deseen eliminar la tuplas duplicadas, se agrega la palabra clave **distinct** en seguida de **select**, quedando de la siguiente manera tomando en cuenta la relación "**C"**:

select distinct nombre
from C

SQL también permite el uso de la palabra clave **all** para especificar explícitamente que no se eliminan los duplicados.

select all nombre
from C

SQL incluye las operaciones *unión*, *intersect* (*intersección*) y *minus (diferencia)*, del álgebra relacional que operan sobre las relaciones.

Analicemos los siguientes ejemplos de consultas en SQL, tomando en cuenta las relaciones de **C**(numc#, nombre, dirección, ciudad) y **PE**(numpe#, numc#, cant):

Primeramente encontraremos a todos los clientes que radican en la ciudad de Puebla:

select distinct nombre
from C
where ciudad = ″ Puebla″

Ahora para encontrar los números de cada cliente que radican en la cuidad Puebla o que adquieren la pieza P_2 escribimos:

(select numc
from C
where ciudad = " Puebla ")
union
(select numc
from PE
where numpe# = " P_2 ")

De manera similar, encontrar los números de cada cliente que radican en la cuidad de Puebla y que adquieren la pieza P_2 escribimos:

(select distinct numc
from C
where ciudad = " Puebla ")
intersect
(select distinct numc
from PE

where P# = "P_2")

La operación **union** por omisión realiza la eliminación de las tuplas duplicadas, ahora si deseamos conservar los duplicados se debe escribir **union all** en vez de solamente **union**. Hemos observado que la omisión en la eliminación de duplicados para **union** difiere de la omisión en **select.** Con el uso de **select distinct** si tiene la certeza de no obtener duplicados en la intersección anterior.

Para encontrar los números de lo clientes que radican en Puebla pero que no adquieren la pieza P_2 escribimos:

(select distinct numc
from cliente
where ciudad = "Puebla")
minus
(select distinct numc
from pedido

where P# = "P_2")

2.4.3 Predicados y conectores

El SQL no cuenta con una representación inmediata del producto natural. Sin embargo, como el producto natural se define en términos de un producto cartesiano, una selección y una proyección es relativamente fácil escribir una expresión en SQL para el producto natural.

Analicemos la siguiente expresión del álgebra relacional

$$\Pi_{\text{nombre, } P\#} (\mathbf{PE} \times \mathbf{C})$$

La consulta “Obtener nombre y número de un cliente que ha adquirido cierta pieza”. En SQL, esto lo podemos escribir así.

```
select distinct C.nombre, numc
   from PE, C
    where PE.numc = C.numc
```

Obsérvese que el SQL usa la notación nombre-relación.nombre-atributo, como el álgebra relacional, para evitar ambigüedad en los casos en los que un atributo aparece en el esquema de más de una relación.

El SQL también permite el uso de los conectores lógicos **and, or** y **not** símbolos equivalentes en matemáticas a $\wedge$, $\vee$, y $\neg$ respectivamente. Además permite el uso de expresiones aritméticas como operando de los operadores de comparación, donde cada expresión puede implicar a los operadores, + , - , * y /, operando sobre constantes o valores de tuplas.

2.5 Referencias.

[1] Ullman Jefrey D., Widom Jennifer., “Introducción a los Sistemas de Base de Datos”, Prentice Hall, México 1999.

[2] C.J. Date, “Introducción a los Sistemas de Bases de Datos”, Addison Wesley Longman, 1998.

[3] Korth Henry F., Silberschatz Abraham, “Fundamentos de Base de Datos”, Mc Graw Hill, 1996.

[4] Awryszkiewycz I. T., “Análisis y Diseño de Base de Datos”, Megabyte, 1994.

[5] García Fernández Jesús, Análisis Y Diseño de Base de Datos”, Diplomado en computación. 1ª. Edición, Abril/97, Pue., Méx.

Capítulo 3

Optimización de Consultas

Cuando se hace mención de la optimización de operaciones de E/S sobre consultas de un sistema administrador de base relacional, nos referimos a el número de lecturas y escrituras que éste emplearía en emitir los resultados de dichas consultas, debido a que las optimizaciones son procesos que alteran el funcionamiento de un sistema con el objetivo de mejorar su eficiencia así como también de reducir el uso de los recursos disponibles.

La optimización de las operaciones en consultas es todo un reto para un sistema administrador de base de datos porque para que éste tenga un buen desempeño debe incluir la optimización de operaciones de consultas y una oportunidad para sistemas de ésta naturaleza porque en el enfoque relacional las expresiones relacionales están representadas en un nivel semántico muy alto para que la optimización sea factible [2].

Un sistema administrador de base de datos relacional con un buen optimizador puede superar muy bien a un sistema no relacional, porque en éstos las peticiones de los usuarios están expresadas en un nivel semántico muy bajo y cualquier “optimización” es realizada en forma manual por los usuarios, en éstos sistemas es el usuario y no el sistema quien decide que operaciones de bajo nivel se necesitan y la secuencia en que estas deben ser ejecutadas [2].

3. 1. Problemas de optimización.

Para ilustrar la necesidad y el potencial de la optimización en los sistemas administradores de base de datos relacionales, supongamos que se tienen las relaciones C (clientes) con 100 clientes y PE (pedidos) con 10000 pedidos y consideremos la consulta "obtener los nombres de los clientes que solicitan el pedido “P2". Consideraremos que solamente 50 tuplas de PE corresponden al pedido “P2”. Una posible solución en SQL sería:

```
SELECT DISTINCT C.NOMBRE
FROM C, PE
WHERE C.C# = PE.C#
AND PE.P# = " P2" ;
```

La serie de pasos a seguir por un sistema sin la inclusión de una herramienta para optimizar consultas sería:

1. Obtener el producto cartesiano entre las relaciones **C** y **PE**. Este proceso consta de 100 lecturas de cada una de las 10,000 tuplas, en total 1’ 000, 000 lecturas de tuplas, en este evento intermedio recalcamos dos casos: uno sería la cantidad de memoria requerida para almacenar a las tuplas el otro sería que probablemente no se contará con los recursos para almacenar dichas tuplas y por tanto implique 1’ 000, 000 escrituras para este resultado..
2. Efectuar la operación de selección como se indica en la condición especificada en la cláusula WHERE, lo que implicaría la lectura de 1’ 000, 000 tuplas, emitiendo el resultado a tan solo 50 tuplas, el cual creemos que es lo suficientemente pequeño para mantenerlo en la memoria principal.

3. Efectuar la operación de proyección sobre **C.NOMBRE**, generando como resultado un máximo de 50 tuplas, los cuales pueden permanecer en memoria principal.

Otra de las soluciones para la anterior consulta, es decir con el mismo resultado o algebraicamente equivalente sería:

1. Restringir **PE** solamente a las tuplas de los pedidos "P_2". Esto implicaría la lectura de 10, 000 tuplas, pero produce una tabla de sólo 50 tuplas, las cuales suponemos que pueden mantener en memoria principal.
2. Realizar el JOIN de la tabla anterior con la relación **C** mediante C#. Esto implica la lectura de sólo 100 tuplas. El resultado contiene 50 tuplas que bien pueden permanecer en memoria principal.
3. Proyección sobre C.NOMBRE con un resultado de un máximo de 50 tuplas.

Si consideramos el "rendimiento" como el número de operaciones de E/S de tuplas, el segundo procedimiento es unas 100 veces aproximadamente mejor que el primero tomando en cuenta la cantidad de E/S de tuplas, ya que el primero realiza 2,000,000 operaciones de E/S contra las 10100 del segundo [2].

3.2. ¿Donde incurre la optimización?

En los procedimientos de optimización se tiene que poner atención en el buen funcionamiento general de las base de datos y por tanto, una eventual optimización de una consulta tendrá que considerar el exceso que pueda generarle al sistema y no sólo con la eficiencia de dicha consulta en particular. Para ello los métodos de optimización deberán incidir en los siguientes puntos:

- El costo de comunicación de acceso a almacenamiento secundario.
- El costo de almacenamiento.
- El costo del software.

Estos puntos van a cambiar dependiendo de la estructura de la base de datos, de la memoria principal disponible, de las características que posea el procesador o procesadores y la configuración del sistema.

Para esto el optimizador de consultas debe contar con las siguientes características **[2]**:

- Información estadística, como la cardinalidad de cada dominio, la cardinalidad de cada relación, el número de valores en cada columna, el número de veces que aparece cada valor distinto en cada columna, etc. (Esta información se conservará en el catálogo del sistema),
- Independencia de las estrategias de acceso con respecto de la organización física de la base de datos, debido a que si las estadísticas de la base de datos cambian de manera significativa, quizá se deba elegir una estrategia distinta.
- Potencia de cálculo en la toma de decisiones frente a la optimización manual. El optimizador es capaz de tomar en consideración cientos de estrategias diferentes para realizar una solicitud dada.
- Disponibilidad del optimizador para todos los usuarios del sistema, ofreciéndoles en forma eficiente y económica, un conjunto de recursos que de otro modo serían difíciles de conseguir.

El optimizador interviene no solo en los procesos de consulta, sino también en las actualizaciones y borrados.

3.3. Procesamiento de consultas.

Las etapas principales del procesamiento de consultas son **[6]**:

1. Trasladar la consulta a su estructura interna.
2. Cambiar a una estructura canónica.
3. Selección de procedimientos de bajo nivel.
4. Generación y elección de consultas.

3.3.1 Convertir la consulta a su forma interna.

El primer paso en el procesamiento de las consultas es la conversión de la consulta original en una representación interna que sea más adecuada para manejarla en el sistema (por lo general en un árbol de consulta o árbol de sintaxis abstractos, aunque esa representación pueda ser considerada sólo como una forma interna del álgebra relacional o del cálculo relacional), eliminando así consideraciones meramente externas. La representación interna deberá cumplir las siguientes características:

- Ser relacionalmente completo.
- Suministrar un punto de partida sólido para las siguientes fases.
- Proporcionar un grado de libertad suficiente para realizar las posibles optimizaciones.

Los sistemas de representación más extendidos son: álgebra relacional, cálculo relacional (orientado a tuplas o a dominios) y el árbol sintáctico abstracto o árbol de consulta.

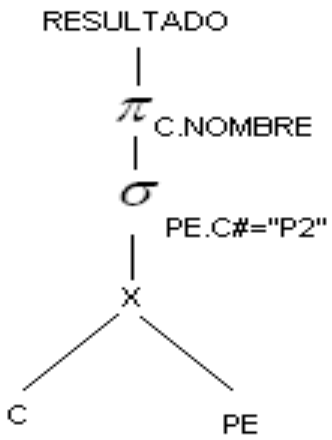

Figura 3.1 Árbol de consulta "obtener los nombres de los proveedores que suministran la pieza P_2".

En el desarrollo de este ejemplo utilizaremos el álgebra relacional como sistema de representación interna, teniendo en cuenta sólo los operadores básicos del álgebra relacional. Con estas condiciones, la operación de JOIN se expresará como un producto cartesiano y a continuación una selección. La consulta anterior se expresará entonces de la forma:

$$\pi_{NOMBRE}(\sigma_{P\#="PE"}(\sigma_{C.C\#=PE.C\#}(C\ x\ PE)))$$

3.3.2. Conversión a forma canónica.

En esta fase, se efectúan una secuencia de optimizaciones seguras y garantizadas., sean cuales sean los valores reales de los datos y las rutas de acceso existentes. Estas optimizaciones se basan en que la mayoría de los lenguajes de consulta de sistemas relacionales permiten expresar una misma consulta de varias formas distintas. Se trata, por tanto, de encontrar una expresión equivalente de una consulta dada en la que se mejore de alguna manera el rendimiento. Esta expresión equivalente será la FORMA CANÓNICA de dicha consulta.

3.3.2.1 Reglas de transformación.

El paso del resultado de la primera fase a la forma canónica correspondiente se realiza mediante un conjunto de reglas de transformación (que serán utilizadas en el algoritmo de optimización de expresiones relacionales en la sección 3.4) bien definidas que son **[7]**:

1. Cascada de proyecciones:
   Si $\{A_{1,\dots,}A_n\} \subset \{B_{1,\dots,}B_m\}$, entonces:
   $$\pi_{A_{1,\dots}A_n}(\pi_{B_{1,\dots}B_m}(E)) \equiv \pi_{A_{1,\dots}A_n}(E)$$

2. Cascada de selecciones:
   $\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_1 \wedge F_2}(E)$

   Como $F_1 \wedge F_2 = F_2 \wedge F_1$, entonces $\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_2}(\sigma_{F_1}(E))$

3. Conmutación de selección y proyección:
   Si F involucra atributos sólo de $A_{1,\dots,}A_n$, entonces:
   $$\pi_{A_{1,\dots}A_n}(\sigma_F(E)) \equiv \sigma_F(\pi_{A_{1,\dots,}A_n}(E))$$

4. Conmutación general de selección y proyección:
   Si F involucra a los atributos $B_{1,\dots,}B_m$ que no están entre los $A_{1,\dots}A_n$, entonces:
   $$\pi_{A_{1,\dots}A_n}(\sigma_F(E)) \equiv \pi_{A_{1,\dots}A_n}(\sigma_F(\pi_{A_{1,\dots}A_nB_{1,\dots}B_m}(E)))$$

5. Conmutación de selección con producto cartesiano (I):
   Si todos los atributos que aparecen en F son atributos de E_1, entonces:

   $$\sigma_F(E_1 \; x \; E_2) \equiv \sigma_F(E_1) \; x \; E_2$$

6. Conmutación de selección y producto cartesiano (II):

Si F es de la forma $F_1 \wedge F_2$, donde F_1 involucra sólo atributos de E_1 y F_2 involucra sólo de E_2, entonces:

$$\sigma_{F_1 \wedge F_2}(E_1 \; x \; E_2) \equiv (\sigma_{F_1}(E_1)) \; x \; (\sigma_{F_2}(E_2))$$

7. Conmutación de selección y producto cartesiano (III):
   Si F es de la forma $F_1 \wedge F_2$, donde F_1 involucra sólo atributos de E_1 y F_2 involucra atributos de E_1 y E_2, entonces:

$$\sigma_{F_1 \wedge F_2}(E_1 \; x \; E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \; x \; E_2)$$

8. Conmutación de selección y unión:
   Una selección sobre una unión equivale a dos selecciones y una unión.

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_F(E_1) \cup \sigma_F(E_2)$$

9. Conmutación de selección con una diferencia:
   Una selección sobre una diferencia equivale a dos selecciones y una diferencia.

$$\sigma_F(E_1 - E_2) \equiv \sigma_F(E_1) - \sigma_F(E_2)$$

10. Conmutación de proyección con un producto cartesiano:
    Sea $\{A_{1,\ldots,} A_n\}$ un conjunto de atributos de los que los K primeros son de E_1 y los restantes de E_2, entonces:

$$\pi_{A_{1,\ldots,} A_n}(E_1 \; x \; E_2) \equiv \pi_{A_{1,\ldots} A_K}(E_1) \; x \; \pi_{A_{K+1,\ldots,} A_n}(E_2)$$

11. Conmutación de proyección con una unión:
    Una proyección de una unión equivale a una unión de dos proyecciones.

$$\pi_{A_{1,\ldots,} A_n}(E_1 \cup E_2) \equiv \pi_{A_{1,\ldots} A_n}(E_1) \cup \pi_{A_{1,\ldots,} A_n}(E_2)$$

Estas transformaciones van dirigidas a conseguir que las relaciones se vean reducidas en su tamaño antes de la realización de JOIN de manera que se realicen menos operaciones de lectura /escritura de tuplas y buscando mantener los resultados parciales en memoria, sin que se tengan que volcar la información a disco.

Las estrategias generales de optimización serán las siguientes **[7]**:

1. Realizar las selecciones tan pronto como sea posible.
2. Combinar ciertas selecciones con un producto cartesiano anterior, para realizar una asociación o un JOIN.
3. Combinar secuencias de operaciones unarias, como selecciones y proyecciones.
4. Detectar subconsultas con resultados equivalentes que puedan reutilizarse a lo largo de una misma consulta.
5. Mediante cascada de selecciones, sustituir una selección compuesta en varias simples.
6. Mediante sus propiedades, desplazar cada selección simple lo más abajo posible del árbol.
7. Mediante sus propiedades, desplazar las proyecciones lo más abajo posible del árbol.
8. Usar las propiedades de proyección y selección para combinar cascadas de proyecciones y selecciones en una selección sencilla, una proyección sencilla o una selección seguida de una proyección. Con esta estrategia se persigue hacer todas las proyecciones y a continuación todas las selecciones que afecten a una misma relación en vez de alternar dichas operaciones.
9. Dividir los nodos interiores del árbol resultante en grupos de la siguiente manera:

Todo nodo interior que presenta una operación binaria estará en un grupo junto con sus antecesores correspondientes a una operación unaria. El grupo incluirá también toda cadena de descendientes correspondientes a operaciones unarias y que terminan en una hoja del árbol; excepto en el caso en el que la

operación binaria sea un producto cartesiano no seguido de una selección para formar un JOIN.

10. Evaluar cada grupo en cualquier orden, siempre que se respete la jerarquía del árbol.

3.3. 3. Elección de procedimientos de bajo nivel.

Una vez transformada la representación interna de la consulta a una forma (canónica) más deseable. En esta etapa, el optimizador debe decidir cómo evaluar la consulta previamente transformada, basándose en elementos tales como:

- Existencia de índices u otras rutas de acceso.
- Distribución de los valores de los datos almacenados.
- Agrupamiento físico de los registros.

El optimizador considerará la consulta como un conjunto de expresiones de "bajo nivel" (join, selección, etc.) con ciertas interdependencias entre ellas. El optimizador dispondrá de un conjunto de procedimientos predefinidos, de bajo nivel, para llevar a la práctica cada una de estas operaciones de bajo nivel **[8] , [9]**.

Ejemplo:

Para la eliminación de valores repetidos, una proyección requerirá que los valores estén en un cierto orden, cosa que deberá cumplir como resultado la consulta inmediatamente anterior.

El optimizador tendrá una serie de procedimientos predefinidos para realizar cada una de estas operaciones de "bajo nivel" según la situación de los datos.

3.3.4 Generación y elección de consultas más económicos.

La generación de los distintos planes de consulta se construye mediante la combinación de los procedimientos de bajo nivel candidatos, uno por cada de las operaciones de bajo nivel de la consulta.

Hay que tener en cuenta que los planes de consulta generados pueden tener un número considerable, por lo que puede no ser conveniente generarlos todos. Si el número de planes generados es muy elevado, el proceso de selección puede ser retrasado.

La eficiencia de un optimizador dependerá, por tanto, de la eficacia de la técnica heurística empleada para la generación de planes de consulta **[6]**.

La estimación de costos de un plan dependerá de:

- El número de operaciones de E/S de disco requeridas.
- Utilización de CPU.

El problema está en que una consulta suele implicar la generación de resultados intermedios. Esto hace que la estimación de las operaciones de E/S dependerá del tamaño de las tablas con estos resultados, lo cual depende mucho de los valores reales de los datos.

3.4 Algoritmo de Optimización de expresiones relacionales.

La salida de éste es algoritmo consiste de los pasos siguientes:

1. La aplicación de una sola selección o proyección.
2. La aplicación de una selección y proyección, o

3. La aplicación de un producto cartesiano, unión o diferencia de conjuntos para dos operadores, quizás precedida por selecciones y/o proyecciones aplicadas a uno o ambos operadores, y posiblemente seguidas por esas operaciones.

Entrada: Un árbol que representa una expresión de él álgebra relacional.

Salida: Un programa para evaluar esa expresión.

Metódo:

1. Use la regla (2) para separar cada selección $\sigma_{F_1 \wedge ... \wedge F_n}(E)$ en la cascada $\sigma_{F_1}(...(\sigma_{F_n}(E))...)$
2. Para cada selección, use la regla (2) - (9) para mover la selección hasta debajo de él árbol lo más posible.
3. Para cada proyección, use la regla(1), (10), (11), y la regla generalizada (3) para mover la proyección lo más abajo del árbol. Note que la regla (3) ocasiona algunas proyecciones a desaparecer, mientras que la regla generalizada (5) divide una proyección en dos proyecciones, una que puede emigrar hacia abajo del árbol si es posible. También elimina una proyección si proyecta una expresión en todos sus atributos.
4. Use la regla (1)-(3) para combiner cascadas de selecciones y proyecciones en una sola selección, una sola selección o una selección seguida por una proyección.
5. La partición de los nodos interiores del árbol resultante en grupos, como se indica, cada nodo interior representa a un operador binario $X, \cup, o -$, esta enb un grupo junto con cualquiera de sus anteriores inmediatos que son etiquetados por un operador unario $\sigma \, or \, \pi$). También incluye en el grupo cualquier cadena de descendientes etiquetados por los operadores unarios y terminados en una hoja, except en el caso que el operador binario sea un producto cartesiano y ninguno seguido por una selección que con el producto para formar una junta.
6. Produce un programa que consiste de unos pasos para evaluar cada grupo de consultas.

3.5 Referencias.

[1] Ullman Jefrey D., Widom Jennifer., “Introducción a los Sistemas de Base de Datos”, Prentice Hall, México 1999.

[2] C.J. Date, “Introducción a los Sistemas de Bases de Datos”, Addison Wesley Longman, 1998.

[3] Korth Henry F., Silberschatz Abraham, “Fundamentos de Base de Datos”, Mc Graw Hill, 1996.

[4] Awryszkiewycz I. T., “Análisis y Diseño de Base de Datos”, Megabyte, 1994.

[5] García Fernández Jesús, Análisis Y Diseño de Base de Datos”, Diplomado en computación. 1ª. Edición, Abril/97, Pue., Méx.

[6 Mathias Jarve y Juergen Koch, “Quero Optimization in Database Systems”. ACM Comp. Surv. 16,nùm. 2 (junio de 1984).

[7] Ullman Jefrey D., “Principles of DATABASE SYSTEMS”, Second Edition, Computer Science Press, 1982.

[8] S.B. Yao. “Optimization of Quero Evaluation Algorithms”. ACM TODS 4, Núm. 2 (junio de 1979).

[9] Leonard D. Shapiro. “Join Processing in Databse Systems with Large Main Memories”. ACM TODS ¡!, núm. 3 (septiembre 1986).

Capítulo 4

Consultas basada en SQL

En el capítulo anterior se presentaron los conceptos y bases teóricas del algoritmo optimización de expresiones relacionales. Retomando estos conceptos podemos ahora implementar este algoritmo para la optimización de las operaciones E/S en las consultas a base de datos relacionales.

Con las reglas de transformación y las estrategias generales de optimización contamos con las herramientas para llevar acabo la optimización de operaciones de E/S de las consultas planteadas y de esta manera poder analizar el comportamiento que presentan a las variaciones de estas reglas y estrategias.

La implementación del algoritmo ha sido realizado en **C** con algunos pequeños cambios para poder trabajarse en **Visual C++ [1]**, el cual trabaja esencialmente con arreglos de estructuras e instrucciones de repetición.

4.1 Desarrollo de consultas.

Considerando la cuarta etapa del procesamiento de consultas, en el proceso de la optimización **[2]**, se involucran la construcción de un conjunto de consultas candidatas, seguida de una selección de la mejor de ellas. Cada consulta es construida por medio de la combinación de una serie de procedimientos candidatos. Observamos que generalmente existirán muchas consultas posibles (tal vez demasiados) para una consulta dada. En la práctica no es una buena idea generar todas las consultas, ya que en combinación habrá muchos y la tarea de elegir el mejor o sea el más económico (con relación al número de

operaciones de E/S) bien puede llegar a ser excesivamente cara por sí misma **[3]**. Las consultas que se analizarán en este capítulo como ejemplos con el objeto de ilustrar y comprender mejor el proceso de la optimización de las operaciones de E/S de consultas son: **1)** Listar títulos de libros que se han solicitado en una biblioteca un día en particular, **2)** Obtener la descripción de las piezas adquiridas en una cantidad superior a 100 unidades por los clientes que radican en la Ciudad de "Puebla."

Ejemplo 1).

Consideremos la base de datos de una biblioteca que consiste de las relaciones siguientes **[4]**:

BOOKS (TITLE, AUTHOR, PNAME, LC_NO)

PUBLISHERS (PNAME, PADDR, PCITY)

BORROWERS (NAME, ADDR, CITY, CARD_NO)

LOANS (CARD_NO, LC_NO, DATE)

Planteamos la siguiente consulta: "Listar los títulos de libros que se han solicitado antes del día 9 / 06 / 08"

Consulta No. 1.

La consulta planteada se construye en álgebra relacional como:

$$\pi_{TITLE}(\sigma_{DATE<9/06/08}(XLOANS))$$

Donde:

$XLOANS$ Es el producto cartesiano entre las relaciones $BOOKS, BORROWERS\ y\ LOANS$, que contiene información adicional sobre los libros que han sido prestados, este puede ser definido como:

$$\pi_S(\sigma_F(LOANS\ x\ BORROWERS\ x\ BOOKS))$$

Mientras:

$$F = BORROWERS.CARD\_NO = LOANS.CARD\_NO$$
$$AND \quad BOOKS.LC\_NO = LOANS.LC\_NO$$

y

$$S = TITLE,\ AUTHOR,\ PNAME,\ LC\_NO,\ NAME,$$
$$ADDR,\ CITY,CARD\_NO,DATE$$

Después de ser sustituido $XLOANS$ en la consulta, ésta expresión tiende a ser analizada por el árbol siguiente:

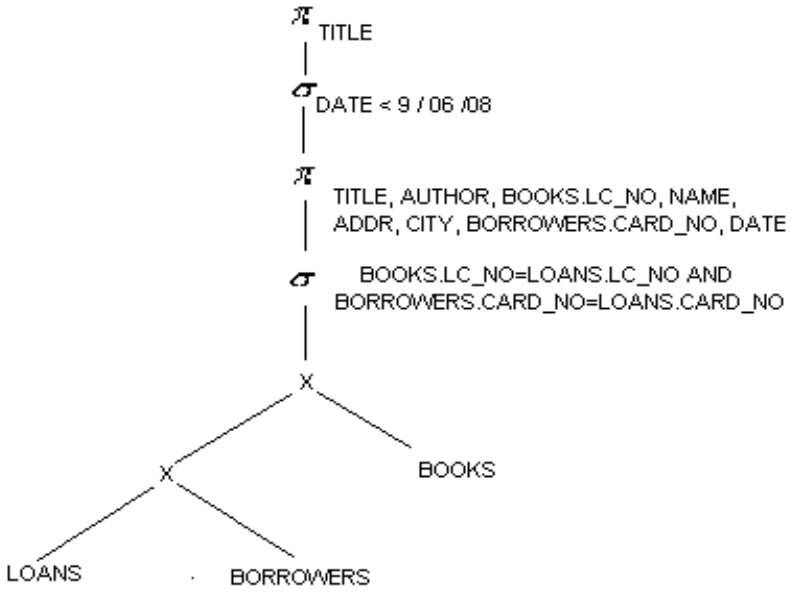

figura 4.1 Árbol de la primera consulta ejemplo 1).

Consulta No. 2.

En ésta consulta realizaremos los siguientes cambios: el primer paso por la estrategia 5 es dividir la selección F compuesta en dos selecciones simples **[4]** $BOOKS.LC\_NO = LOANS.LC\_NO$ y $BORROWERS.CARD\_NO = LOANS.CARD\_NO$

La selección correspondiente $\sigma_{DATE < 9/06/08}$ puede desplazarse hasta LOANS aplicando la estrategia 6 **[4]**.

Las selecciones correspondientes al producto cartesiano pueden desplazarse de la siguiente forma: la correspondiente a $\sigma_{BORROWERS.CARD\_NO=LOANS.CARD\_NO}$ se conmuta con el producto cartesiano inmediatamente inferior. La selección correspondiente a $\sigma_{BOOKS.LC\_NO=LOANS.LC\_NO}$ no puede conmutarse por hacer referencia a atributos de diferentes relaciones **[4]**.

El árbol para ésta segunda consulta quedaría entonces de la forma siguiente:

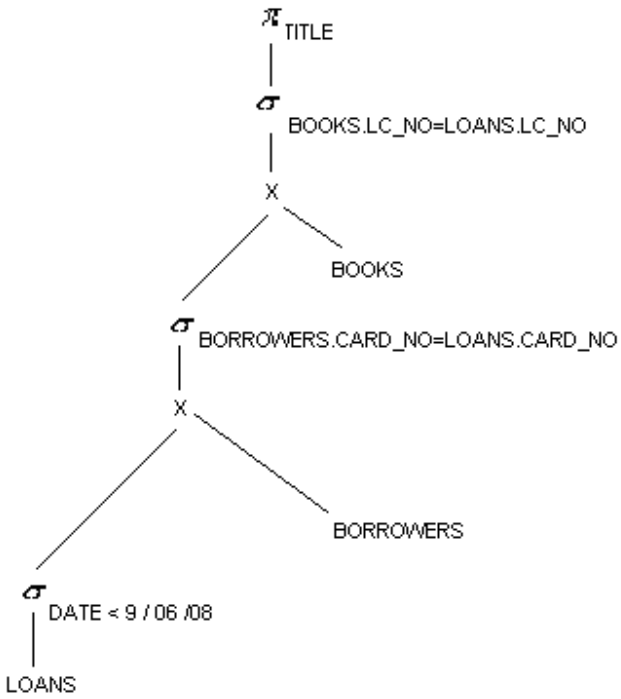

figura 4.2 Árbol de la segunda consulta ejemplo 1).

Consulta No. 3.

Continuando con las construcciones de las consultas y mejorando la consulta anterior, aplicaremos las siguientes reglas o estrategias:

Utilizando la regla 5 **[4]**, se pueden sustituir:

π_{TITLE} y $\sigma_{BOOKS.LC\_NO=LOANS.LC\_NO}$ por la cascada

π_{TITLE}
$\sigma_{BOOKS.LC\_NO=LOANS.LC\_NO}$
$\pi_{TITLE,BOOKS.LC\_NO,LOANS.LC\_NO}$

La regla 11 **[5]** permite conmutar la $\pi_{TITLE,BOOKS.LC\_NO}$ con el producto cartesiano inmediatamente inferior de la siguiente forma:

$\pi_{TITLE.BOOKS.LC\_NO}$ sobre BOOKS y $\pi_{LOANS.LC\_NO}$ sobre el resultado de $\sigma_{BORROWERS.CARD\_NO=LOANS.CARD\_NO}$.

El árbol para la tercera consulta quedaría entonces de la forma siguiente:

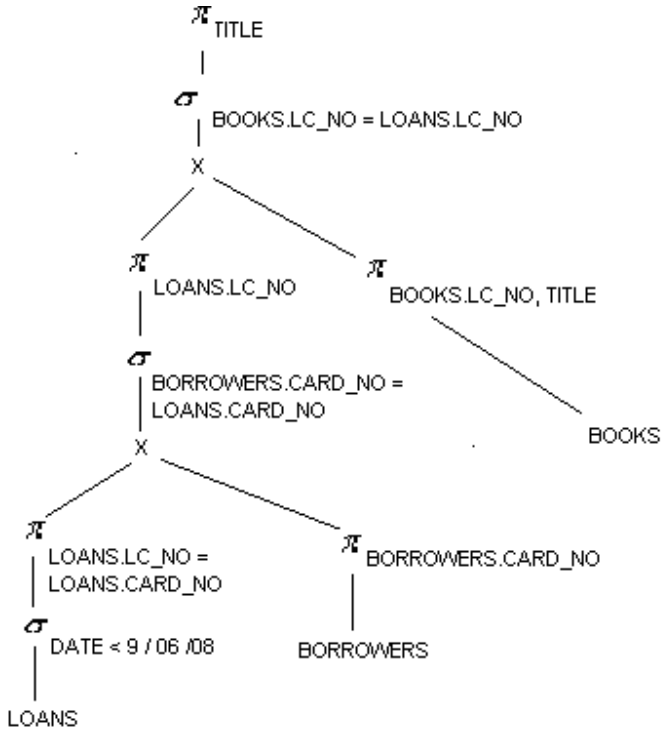

figura 4.3 Árbol de la tercera consulta ejemplo 1).

Ejemplo 2).

En este ejemplo se generarán solamente cuatro consultas candidatas, llevándose a cabo la tarea de seleccionar como se menciona anteriormente el mejor **[3]**.

Consideremos la Base de Datos con las siguientes relaciones:

P (P#, DESCRIPCION, FABRICA) P: Pieza

C (C# ,NOMBRE, DIRECCION, CIUDAD) C: Cliente

PE (P#, C#, CANTIDAD) PE: Pedido

Planteamos la siguiente consulta: "Obtener la descripción de las piezas adquiridas en una cantidad superior a 100 unidades por los clientes que radican en la Ciudad de “Puebla."

Consulta No. 1.

Esta consulta consiste en lo siguiente:

La consulta planteada se construye en álgebra relacional como:

$$\pi_{DESCRIPCION}(\sigma_{CANTIDAD>100\ AND\ CIUDAD="PUEBLA"}(P\ x\ C\ x\ PE))$$

Para construir ésta primera consulta expresaremos el JOIN como producto cartesiano y selección, quedando una expresión de la forma:

$$\pi_{DESCRIPCION}(\sigma_{CANTIDAD>100\ AND\ CIUDAD="PUEBLA"}(\sigma_{P.P\#=PE.P\#\cdot\ AND\ C.C\#=PE.C\#}(P\ x\ C\ x\ PE))$$

El árbol para ésta consulta es:

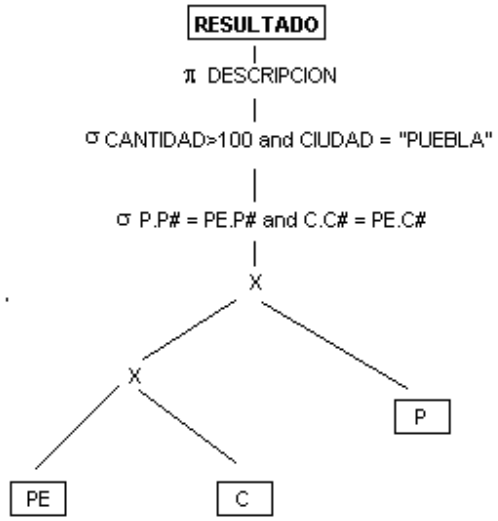

figura 4.4 Árbol de la primera consulta ejemplo 2).

Consulta No. 2.

En ésta segunda consulta realizaremos los siguientes cambios a la anterior :

Primero, separaremos las dos selecciones compuestas en selecciones simples, por lo tanto $\sigma_{CANTIDAD>100\ AND\ CIUDAD="PUEBLA"}$ se descompone en $\sigma_{CANTIDAD>100}$ y $\sigma_{CIUDAD="PUEBLA"}$. La selección correspondiente al producto cartesiano $\sigma_{P.P\#=PE.P\#\ \ AND\ \ C.C\#=PE.C\#}$ se descompone en $\sigma_{P.P\#=PE.P\#}$ y $\sigma_{C.C\#=PE.C\#}$. De esta forma se podrán desplazar cada una de las selecciones lo más abajo posible en el árbol **[4]**.

La selección correspondiente a $\sigma_{CANTIDAD>100}$ puede desplazarse hasta PE aplicando la estrategia 6 **[4]**.

La selección correspondiente a $\sigma_{CIUDAD="PUEBLA"}$ puede desplazarse hasta C aplicando la estrategia 6 **[4]**.

Las selecciones correspondientes al PRODUCTO CARTESIANO pueden desplazarse de la siguiente forma: la correspondiente a $\sigma_{C.C\#=PE.C\#}$ se conmuta con el producto cartesiano inmediatamente inferior. La selección correspondiente a $\sigma_{P.P\#=PE.P\#}$ no puede conmutarse por hacer referencia a atributos de diferentes relaciones **[5]**.

El árbol para ésta segunda consulta quedaría entonces de la forma siguiente:

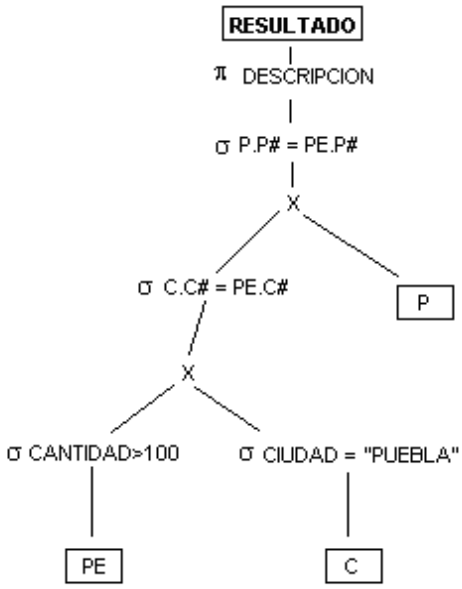

figura 4.5 Árbol de la segunda consulta ejemplo 2).

Consulta No. 3.

Continuando con las construcciones de las consultas y mejorando la consulta anterior, aplicaremos las siguientes reglas o estrategias:

Utilizando la regla 5 **[4]**, se pueden sustituir:

$\pi_{DESCRIPCION}$ y $\sigma_{P.P\# = PE.P\#}$ por

$\pi_{DESCRIPCION}$

$\sigma_{P.P\# = PE.P\#}$

$\pi_{DESCRIPCION,\, P.P\#,\, PE.P\#}$

La regla 11 permite conmutar la $\pi_{DESCRIPCION,\, P.P\#,\, PE.P\#}$ con el producto cartesiano inmediatamente inferior de la siguiente forma **[4]**:

$\pi_{DESCRIPCION,\,P.P\#}$ sobre P y $\pi_{PE.P\#}$ sobre el resultado de $\sigma_{C.C\# = PE.C\#}$.

El árbol para la tercera consulta quedaría entonces de la forma siguiente:

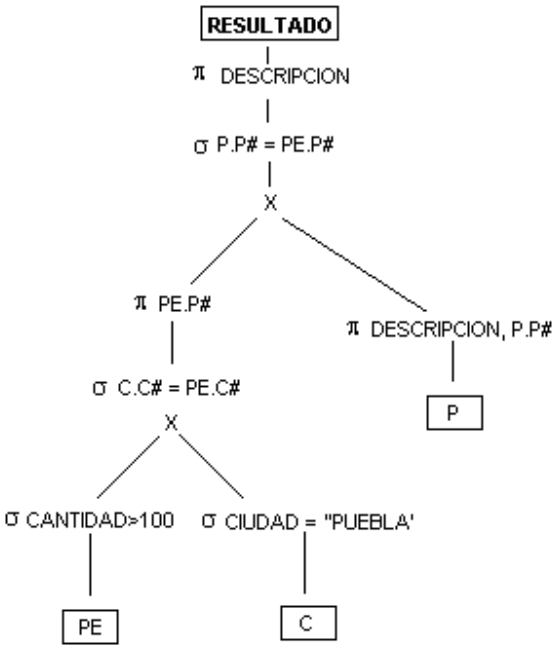

figura 4.6 Árbol de la tercera consulta ejemplo 2).

Consulta No. 4.

De forma análoga a la anterior la consulta quedaría y por la estrategia **[4],** podemos sustituir

$\pi_{PE.P\#}$ y $\sigma_{C.C\# = PE.C\#}$ por

$\pi_{PE.P\#}$

$\sigma_{C.C\# = PE.C\#}$

$\pi_{PE.P\#, C.C\#, PE.C\#}$

La regla 11 permite conmutar $\pi_{PE.P\#,\, C.C\#,\, PE.C\#}$ con el producto cartesiano inmediatamente inferior de la siguiente forma **[4]**:

$\pi_{PE.C\#,\, PE.P\#}$ sobre el resultado de $\sigma_{CANTIDAD>100}$ y $\pi_{C.C\#}$ sobre el resultado de $\sigma_{CIUDAD="PUEBLA"}$.

El árbol para ésta consulta “optimizada” queda de la forma siguiente:

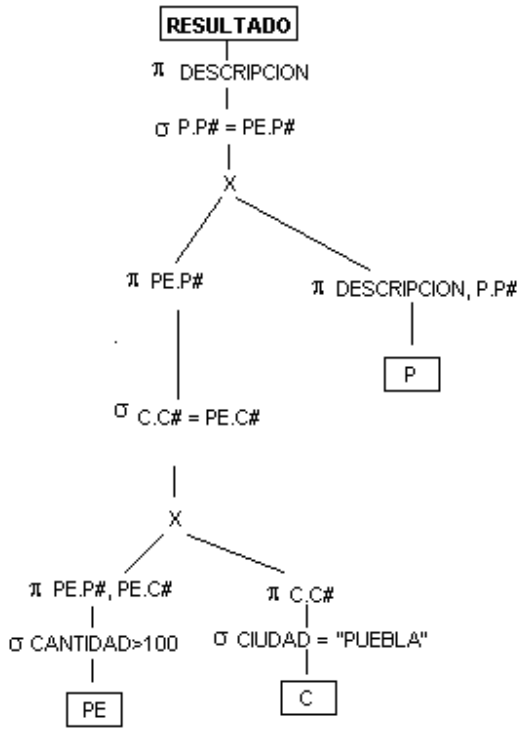

figura 4.7 Árbol de la primer a consulta ejemplo 2).

4.2 Referencias.

[1] User' s Guide C, Visual C++.

[2] Mathias Jarve y Juergen Koch, "Query Optimization in Database Systems". ACM Comp. Surv. 16,nùm. 2 (junio de 1984).

[3] C.J. Date, "Introducción a los Sistemas de Bases de Datos", Addison Wesley Longman, 1998.

[4] Ullman Jefrey D., "Principles of DATABASE SYSTEMS", Second Edition, Computer Science Press, 1982.

Capítulo 5

Resultados

Los resultados obtenidos por la optimización de operaciones de E/S en las consultas se presentan en este capítulo. Estas optimizaciones se basan en que la mayoría de los lenguajes de consulta de los sistemas administradores de base de datos relacional permiten expresar una misma consulta de varias formas distintas, obteniendo el mismo resultado para cada una de ellas. Nuestro propósito con la simulación del sistema para optimizar las operaciones de E/S en las consultas es no solamente obtener los mismos resultados para cada una de ellas sino también evaluar la mejor de éstas, es decir la más económica en lo que se refiere a el número de operaciones de E/S. Dicho sistema consta de un proceso para cada consulta que opera dándole el seguimiento al algoritmo, con esto se llegan a obtener los datos, los cuales únicamente sirven para la comparación de resultados y, por tanto , el sistema no necesita especificaciones rigurosas para su ejecución, solamente se genera un programa ejecutable, el cual no es muy grande y puede ser ejecutado prácticamente en cualquier computadora con una memoria mayor de 512 kb. y un sistema operativo Windows 98 o superior, solamente es necesario un usuario para su manejo, sencillamente se puede tener una copia de los archivos como respaldo y con esto sería suficiente.

En realidad el sistema no requiere demasiado tiempo para su desarrollo, se necesita cierto tiempo para entender el seguimiento del algoritmo (4 semanas), un tiempo para la programación (8 semanas) y por último un cierto tiempo para las pruebas (5 semanas).

5.1 Resultados analíticos

Los resultados obtenidos de forma analítica por las distintas consultas planteadas mediante el álgebra relacional y con las aplicaciones de las reglas de transformación presentadas son los mismos, estos resultados obtenidos son positivos, aceptables y seguros, de esta forma podemos realizar la comparación con cada uno de los resultados que se genera al implementarlo en el sistema.

Los datos de entrada necesarios para la simulación del sistema, son:

Ejemplo 1).

Una base de datos de una biblioteca con las relaciones siguientes:

BOOKS (TITLE, AUTHOR, PNAME, LC_NO)

PUBLISHERS (PNAME, PADDR, PCITY)

BORROWERS (NAME, ADDR, CITY, CARD_NO)

LOANS (CARD_NO, LC_NO, DATE)

Donde:

TITLE	AUTHOR	PNAME	LC_NO
CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
ARQUIETECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4

figura 5.1 La relación de BOOKS.

PNAME	PADDR	PCITY
WILLIAM ANTHONY GRANVILLE	432 LOGWOOD AVENUE	BOSTON E. U. A

M. MORRIS MANO	7635 N LA CHOLLA BLVD.	LOS ANGELES CAL. E. U. A
JEFREY D. ULLMAN	2083 RIDER UNIVERSITY	NEW JERSEY E. U . A
M. MORRIS MANO	1307 NEW YORK AVENUE	LOS ANGELES CAL. E.U.A

figura 5.2 La relación de PUBLISHERS.

NAME	ADDR	CITY	CARD_NO
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013

figura 5.3 La relación de BORROWERS

CARD_NO	LC_NO	DATE
0010	1	8 / 06 /08
0011	2	10 / 06 /08
0012	3	7 /06 /08
0013	4	11 / 06 / 08

figura 5.4 La relación de LOANS.

Los resultados obtenidos de forma analítica por las tres consultas analizando sus expresiones y siguiendo las sintaxis de cada uno de ellos son:

Consulta No. 1.

Tomando como referencia el árbol de esta consulta (figura 4.1) podemos observar las siguientes secuencias de eventos:

1. Calcular el producto cartesiano de las relaciones BORROWERS y LOANS. Este paso implica leer 4 veces cada una de las 4 tuplas de LOANS (una vez por cada tupla de BORROWERS). El producto contendrá un total de 16 tuplas (figura 5.5) y aproximadamente se necesitan 48 kb. para su almacenamiento. Si en cierto momento llegará a ser un número de tuplas demasiado grande y éstas no pudieran mantener en la memoria principal, sería necesario almacenarlo en disco, esto aumentaría el número de operaciones tanto de lecturas como escrituras. Debemos recordar que se hace mención en el capitulo 2 sección 2.1 que en el producto cartesiano de dos relaciones que tengan nombres de atributos comunes como en este caso, lo recomendable es renombrar dichos atributos utilizando simplemente el operador ***renombrar***.

NAME	ADDR	CITY	CARD_NO	CARD_NO1	LC_NO	DATE
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0011	2	10 / 06 /08
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 /06 /08
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0013	4	11 / 06 / 08
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 / 06 /08
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0012	3	7 /06 /08
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0013	4	11 / 06 / 08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0011	2	10 / 06 /08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0013	4	11 / 06 / 08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0011	2	10 / 06 /08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08

figura 5.5 BORROWERSX LOANS

2. Calcular el producto cartesiano entre la relación obtenida en el paso 1 con la relación BOOKS. Este implica leer 16 veces cada una de las 4 tuplas de BOOKS. Éste producto contendrá un total de 64 tuplas (figura 5.6) y aproximadamente se necesitan 184 kb. para su almacenamiento. De igual manera al llegar a ser un número de tuplas demasiado grande para mantenerlo en la memoria principal, sería necesario almacenarlo en disco, esto aumentaría el número de operaciones tanto de lecturas como escrituras.

NAME	ADDR	CITY	CARD_NO	CARD_NO1	LC_NO	DATE	TITLE	AUTHOR	PNAME	LC_NO1
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0011	2	10 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1

JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0011	2	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0011	2	10 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0011	2	10 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0013	4	11 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0013	4	11/ 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0013	4	11 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0013	4	11 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 /06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2

ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0012	3	7 /06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0012	3	7 /06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0012	3	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0012	3	7 /06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0013	4	11 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0013	4	11 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0013	4	11 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3

ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0013	4	11 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0011	2	10 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0011	2	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0011	2	10 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0011	2	10 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4

DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0013	4	11 / 06 / 08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0013	4	11 / 06 / 08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0013	4	11 / 06 / 08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0013	4	11 / 06 / 08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0011	2	10 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0011	2	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0011	2	10 / 06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0011	2	10 / 06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1

JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4

Figura 5.6 (BORROWERS X LOANS) X BOOKS

3. Restringir el resultado del paso 2 según se representa en la figura 4.1. Este paso implica leer las 64 tuplas , produciendo una relación formada por sólo 4 tuplas (figura 5.7) y aproximadamente se necesitan 32 kb. para su almacenamiento:

NAME	ADDR	CITY	CARD_NO	CARD_NO1	LC_NO	DATE	TITLE	AUTHOR	PNAME	LC_NO1
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0011	2	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2

DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0013	4	11 / 06 / 08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4

Figura 5.7 Restricción de la relación del paso 2.

4. Proyectar la relación del paso 3 sobre TITLE, AUTHOR, BOOKS.LC_NO, NAME, ADDR, CITY, BORROWERS.CARD_NO, DATE, según se representa en la figura 4.1, a fin de reducir la relación a ocho atributos quedando de la siguiente manera (figura 5.8) y aproximadamente se necesitan 28 kb. para su almacenamiento:

NAME	ADDR	CITY	CARD_NO	DATE	TITLE	AUTHOR	LC_NO
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	1
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	10 / 06 /08	INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	2
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	3
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	11 / 06 / 08	ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	4

Figura 5.8 Proyección de la relación del paso 3.

5. Restringir la relación del paso 4 según lo especificado por DATE ***< 9 / 06 /08*** según se representa en la figura 4.1. Este paso implica leer 4 tuplas , produciendo una relación formada

por sólo 2 tuplas (figura 5.9) y aproximadamente se necesitan 24.5 kb. de espacio para su almacenamiento :

NAME	ADDR	CITY	CARD_NO	DATE	TITLE	AUTHOR	LC_NO
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	8 / 06 /08	CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	1
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	7 /06 /08	INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	3

Figura 5.9 Restricción de la relación del paso 4.

6. Proyectar el resultado del paso 5 sobre ***TITLE*** a fin de producir el resultado final deseado.

TITLE
CÁLCULO DIFERENCIAL E INTEGRAL
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS

Consulta No. 2.

Tomando como referencia el árbol de esta consulta (figura 4.2) podemos observar las siguientes secuencias de eventos:

1. Restringir la relación LOANS a las tuplas donde DATE < 9 / 06 /08. Este paso implica leer 4 tuplas pero produce una relación formada sólo por 2 tuplas (figura 5.10) y aproximadamente se necesitan 24.5 kb. de espacio para su almacenamiento.

CARD_NO	LC_NO	DATE
0010	1	8 / 06 /08
0012	3	7 /06 /08

Figura 5.10 Restricción de la relación LOANS.

2. Calcular el producto cartesiano entre la relación obtenida en el paso 1 con la relación BORROWERS. Este implica leer 4 veces cada una de las 2 tuplas de la relación obtenida en el paso 1. Éste producto contendrá un total de 8 tuplas (figura 5.11) y aproximadamente se requieran 35.5 kb. para su almacenamiento . Debemos recordar también que en el producto cartesiano de dos relaciones que tengan nombres de atributos comunes como en este caso, lo recomendable es renombrar dichos atributos utilizando simplemente el operador ***renombrar***.

NAME	ADDR	CITY	CARD_NO	CARD_NO1	LC_NO	DATE
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0012	3	7 /06 /08
ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	0011	0010	1	8 / 06 /08
ANA I. BALLESTEROS	AV. SANTA ANA	TUXTLA GTZ.	0011	0012	3	7 /06 /08

REYES	1930					
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0010	1	8 / 06 /08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0010	1	8 / 06 /08
JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	0013	0012	3	7 /06 /08

Figura 5.11 Producto cartesiano entre la relación del paso 1 con BORROWERS

3. Restringir el resultado del paso 2 según se representa en la figura 4.2. Este paso implica leer las 8 tuplas, produciendo una relación formada por sólo 2 tuplas (figura 5.12) y aproximadamente se requieran 25 kb. para su almacenamiento:

NAME	ADDR	CITY	CARD_NO	CARD_NO	LC_NO	DATE
JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08

Figura 5.12 Restricción BORROWERS.CARD_NO =LOANS.CARD_NO

4. Calcular el producto cartesiano entre las relaciones del paso 3 y BOOKS. Este paso implica leer 2 veces cada una de las 4 tuplas de BOOKS. El producto contendrá un total de 8 tuplas (figura 5.13) y aproximadamente se requieran 38.5 kb. para su almacenamiento:

TITLE	AUTHOR	PNAME	LC_NO	NAME	ADDR	CITY	CARD_NO	CARD_NO	LC_NO	DATE
CÁLCULO	WILLIAM	LIMUSA	1	JOSÉ SILVESTRE	RIO PANUCO	PUEBLA	0010	0010	1	8 / 06 /08

DIFERENCIAL E INTEGRAL	ANTHONY GRANVILLE			GARCIA VALDEZ.	5934					
CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	M. MORRIS MANO	PRENTICE HALL	2	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08
ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
ARQUITECTURA DE COMPUTADORAS	M. MORRIS MANO	PERNTICE HALL	4	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08

Figura 5.13 Producto cartesiano entre la relación del paso 3 con BOOKS

5. Restringir la relación del paso 4 según se representa en la figura 4.2. Este paso implica leer las 8 tuplas, produciendo una relación formada por sólo 2 tuplas (figura 5.14) y aproximadamente se requieran 25 kb. para su almacenamiento:

TITLE	AUTHOR	PNAME	LC_NO	NAME	ADDR	CITY	CARD_NO	CARD_NO	LC_NO	DATE
CÁLCULO DIFERENCIAL E INTEGRAL	WILLIAM ANTHONY GRANVILLE	LIMUSA	1	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	0010	0010	1	8 / 06 /08
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	JEFREY D. ULLMAN	PRENTICE HALL	3	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	0012	0012	3	7 /06 /08

Figura 5.14 Restricción BOOKS.LC_NO = LOANS.LC_NO

6. Proyectar el resultado del paso 5 sobre ***TITLE*** a fin de producir el resultado final deseado.

TITLE
CÁLCULO DIFERENCIAL E INTEGRAL
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS

Consulta No. 3.

Tomando como referencia el árbol de esta consulta (figura 4.3) podemos observar las siguientes secuencias de eventos:

1. Restringir la relación LOANS a las tuplas donde DATE < 9 / 06 /08. Este paso implica leer 4 tuplas pero produce una relación formada sólo por 2 tuplas (figura 5.15) y aproximadamente se requieran 24.5 kb. para su almacenamiento :

CARD_NO	LC_NO	DATE
0010	1	8 / 06 /08
0012	3	7 /06 /08

Figura 5.15 Restricción de la relación LOANS.

2. Proyectar la relación del paso 1 a los atributos CARD_NO y LC_NO. Este paso produce la relación (figura 5.16) y aproximadamente se requieran 24.5 kb. para su almacenamiento:

CARD_NO	LC_NO
0010	1

0012	3

Figura 5.16 Proyección de la relación del paso 1.

3. Proyectar la relación de BORROWERS al atributo CARD_NO. Este paso produce la relación siguiente (figura.517) y aproximadamente 24.5 kb. para su almacenamiento:

CARD_NO
0010
0011
0012
0013

Figura 5.17 Proyección de la relación BORROWERS.

4. Calcular el producto cartesiano entre las relaciones del paso 2 y 1. Este paso implica leer 2 veces cada una de las 4 tuplas del paso 3. El producto contendrá un total de 8 tuplas (figura 5.18) y aproximadamente 28.5 kb. para su almacenamiento:

CARD_NO	LC_NO	CARD_NO
0010	1	0010
0010	1	0011
0010	1	0012
0010	1	0013
0012	3	0010
0012	3	0011
0012	3	0012
0012	3	0013

Figura 5.18 Producto cartesiano entre las relaciones del paso 2 y 3.

5. Restringir la relación del paso 4 donde los valores de los atributos CARD_NO coincidan. Este paso implica leer 8 tuplas produciendo la relación siguiente(figura 5.19) y aproximadamente 24.5 kb. para su almacenamiento:

CARD_NO	LC_NO	**CARD_NO**
0010	1	0010
0012	3	0012

Figura 5.19 Restricción de la relación paso 4.

6. Proyectar la relación del paso 5 al atributo LC_NO. Este paso produce la relación siguiente (figura 5.20) y aproximadamente se requieran 24 kb. para su almacenamiento:

LC_NO
1
3

Figura 5.20 Proyección de la relación del paso 5.

7. Proyectar de la relación BOOKS a los atributos LC_NO y TITLE. Este paso produce la relación siguiente (figura 5.21) y aproximadamente se requieran 25.5 kb. para su almacenamiento:

TITLE	**LC_NO**
CÁLCULO DIFERENCIAL E INTEGRAL	1
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	2
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	3
ARQUITECTURA DE COMPUTADORAS	4

Figura 5.21 Proyección de la relación BOOKS.

8. Calcular el producto cartesiano entre las relaciones del paso 6 y 7. Este paso implica leer 2 veces cada una de las 4 tuplas del paso 7. El producto contendrá un total de 8 tuplas (figura 5.22) y aproximadamente 28.5 kb. para su almacenamiento:

TITLE	**LC_NO**	LC_NO
CÁLCULO DIFERENCIAL E INTEGRAL	1	1
CÁLCULO DIFERENCIAL E INTEGRAL	1	3
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	2	1
INGENIERÍA COMPUTACIONAL DISEÑO DEL HARDWARE	2	3
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	3	1
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS	3	3
ARQUITECTURA DE COMPUTADORAS	4	1
ARQUITECTURA DE COMPUTADORAS	4	3

Figura 5.22 Producto cartesiano entre las relaciones del paso 6 y 7.

9. Restringir la relación del paso 8 donde los valores de los atributos LC_NO coincidan. Este paso implica leer 8 tuplas produciendo la relación siguiente(figura 5.23) y aproximadamente se requieran 24.5 kb. para su almacenamiento:

TITLE	**LC_NO**	LC_NO
CÁLCULO DIFERENCIAL E INTEGRAL	1	1
INTRODUCCIÓN A LOS	3	3

SISTEMAS DE BASE DE DATOS		

Figura 5.23 Restricción de la relación del paso 8.

10. Proyectar el resultado del paso 9 sobre ***TITLE*** a fin de producir el resultado final deseado.

TITLE
CÁLCULO DIFERENCIAL E INTEGRAL
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS

Ejemplo 2).

La Base de Datos con las siguientes relaciones:

P (P#, DESCRIPCION, FABRICA) P: Pieza

C (C# ,NOMBRE, DIRECCION, CIUDAD) C: Cliente

PE (P#, C#, CANTIDAD) PE: Pedido

Donde:

C#	NOMBRE	DIRECCION	CIUDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA

Figura 5.24 Relación CLIENTE.

P#	C#	CANTIDAD
50	130	500
20	160	50
3	30	160
30	130	200
10	10	30
60	130	200

Figura 5.25 Relación PEDIDO

P#	DESCRIPCION	FABRICA
50	TUERCA	ACEMEX
20	TORNILLO	STANLEY
30	MARTILLO	STANLEY
60	PINZAS	STANLEY
3	CLAVOS	ACEMEX
10	CHIAPAS	SINMAR

Figura 5.26 La relación de PIEZA.

Los resultados obtenidos de forma analítica por las cuatro consultas analizando sus expresiones y siguiendo las sintaxis de cada uno de ellos son:

Consulta No. 1.

Tomando como referencia el árbol de esta consulta (figura 4.4) podemos observar las siguientes secuencias de eventos:

1. Calcular el producto cartesiano entre las relaciones CLIENTE y PEDIDO. Este paso implica leer 4 veces cada una de las 6 tuplas de PEDIDO. El producto contendrá un total de 24 tuplas (figura 5.27) y aproximadamente se requieran 67 kb. para su almacenamiento. Si en cierto momento llegará a ser un número de tuplas demasiado grande para mantenerlo en la memoria principal, sería necesario almacenarlo en disco, esto aumentaría el número de operaciones tanto de lecturas como escrituras.

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30

30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200

Figura 5.27 Producto cartesiano entre las relaciones CLIENTE y PEDIDO.

2. Calcular el producto cartesiano entre las relaciones del paso 1 y PIEZA. Este paso implica leer 24 veces cada una de las 6 tuplas de PIEZA. El producto contendrá un total de 144 tuplas (figura 5.28) y aproximadamente se requieran 129 kb. para su almacenamiento.

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD	P#1	DESCRIPCION	FABRICA
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	10	CHIAPAS	SINMAR
130	JOSÉ	RIO PANUCO	PUEBLA	20	160	50	50	TUERCA	ACEMEX

	SILVESTRE GARCIA VALDEZ.	5934							
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	20	160	50	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE	RIO PANUCO 5934	PUEBLA	30	130	200	50	TUERCA	ACEMEX

	GARCIA VALDEZ.								
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	10	10	30	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE GARCIA	RIO PANUCO 5934	PUEBLA	60	130	200	50	TUERCA	ACEMEX

	VALDEZ.								
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	20	TORNILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	30	MARTILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	50	130	500	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	20	TORNILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	30	MARTILLO	STANLEY
160	ANA I.	AV. SANTA	TUXTLA	20	160	50	60	PINZAS	STANLEY

	BALLESTEROS REYES	ANA 1930	GTZ.						
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	20	TORNILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	30	MARTILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	3	30	160	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	20	TORNILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	30	MARTILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	30	130	200	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	20	TORNILLO	STANLEY

	REYES								
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	30	MARTILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	10	10	30	10	CHIAPAS	SINMAR
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	50	TUERCA	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	20	TORNILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	30	MARTILLO	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	3	CLAVOS	ACEMEX
160	ANA I. BALLESTEROS REYES	AV. SANTA ANA 1930	TUXTLA GTZ.	60	130	200	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500	10	CHIAPAS	SINMAR

30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	20	160	50	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200	60	PINZAS	STANLEY
30	DANIEL DE J.	2ª. PTE NORTE	PUEBLA	30	130	200	3	CLAVOS	ACEMEX

	ESTRADA CAMACHO	650							
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	10	10	30	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500	30	MARTILLO	STANLEY
10	JOSE EDUARDO	RIO VERDE	TLAXCALA	50	130	500	60	PINZAS	STANLEY

	GARCIA MOLINA	6010							
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	50	130	500	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	30	MARTILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	60	PINZAS	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	20	160	50	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	30	MARTILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	60	PINZAS	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	3	30	160	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	30	MARTILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	60	PINZAS	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	30	130	200	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	30	MARTILLO	STANLEY
10	JOSE EDUARDO	RIO VERDE	TLAXCALA	10	10	30	60	PINZAS	STANLEY

	GARCIA MOLINA	6010							
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	10	CHIAPAS	SINMAR
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	50	TUERCA	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	20	TORNILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	30	MARTILLO	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	60	PINZAS	STANLEY
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	60	130	200	10	CHIAPAS	SINMAR

Figura 5.28 Producto cartesiano entre las relaciones (CLIENTE X PEDIDO) X PIEZA.

3. Restringir la relación del paso 2 donde los valores de los atributos C# y P# coincidan. Este paso implica leer 144 tuplas pero produce una relación formada sólo por 6 tuplas (figura 5.29) y aproximadamente se requieran 32.5 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD	P#1	DESCRIPCION	FABRICA
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	60	PINZAS	STANLEY
160	ANA I. BALLESTEROS	AV. SANTA ANA 1930	TUXTLA GTZ.	20	160	50	20	TORNILLO	STANLEY

	REYES								
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	3	CLAVOS	ACEMEX
10	JOSE EDUARDO GARCIA MOLINA	RIO VERDE 6010	TLAXCALA	10	10	30	10	CHIAPAS	SINMAR

Figura 5.29 Restricción de la relación del paso 2.

4. Restringir la relación del paso 3 donde CANTIDAD > 100 y CIUDAD = "PUEBLA". Este paso implica leer 6 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.30) y aproximadamente se requieran 28 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD	P#1	DESCRIPCION	FABRICA
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	3	CLAVOS	ACEMEX

Figura 5.30 Restricción de la relación del paso 3.

5. Proyectar el resultado del paso 4 sobre ***DESCRIPCION*** a fin de producir el resultado final deseado.

DESCRIPCION
TUERCA

MARTILLO
PINZAS
CLAVOS

Consulta No. 2.

Tomando como referencia el árbol de esta consulta (figura 4.5) podemos observar las siguientes secuencias de eventos:

1. Restringir la relación PEDIDO a las tuplas donde CANTIDAD > 100. Este paso implica leer 6 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.31) y aproximadamente se requieran 25.5 kb. para su almacenamiento:

P#	C#	CANTIDAD
50	130	500
3	30	160
30	130	200
60	130	200

Figura 5.31 Restricción de la relación PEDIDO.

2. Restringir la relación CLIENTE a las tuplas donde CIUDAD = “PUEBLA”. Este paso implica leer 4 tuplas pero produce una relación formada sólo por 2 tuplas (figura 5.32) y aproximadamente se requieran 25 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA

Figura 5.32 Restricción de la relación CLIENTE.

3. Calcular el producto cartesiano entre las relaciones del paso 1 y 2. Este paso implica leer 4 veces cada una de las 2 tuplas de la relación del paso 2. El producto contendrá un total de 8 tuplas (figura 5.33) y aproximadamente se requieran 35 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200

Figura 5.33 Producto cartesiano entre las relaciones del paso 1 y 2.

4. Restringir la relación del paso 3 donde los valores de los atributos C# coincidan. Este paso implica leer 8 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.34) y aproximadamente se requieran 28.5 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160

Figura 5.34 Restricción del la relación del paso 3.

5. Calcular el producto cartesiano entre las relaciones del paso 4 y PEDIDO. Este paso implica leer 4 veces cada una de las 6 tuplas de la relación PEDIDO. El producto contendrá un total de 24 tuplas (figura 5.35) y aproximadamente se requieran 68.5 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#	CANTIDAD	P#	DESCRIPCION	FABRICA
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE	RIO PANUCO	PUEBLA	60	130	200	20	TORNILLO	STANLEY

	GARCIA VALDEZ.	5934							
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	10	CHIAPAS	SINMAR
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	20	TORNILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	30	MARTILLO	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	3	CLAVOS	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	10	CHIAPAS	SINMAR
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	50	TUERCA	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	20	TORNILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	60	PINZAS	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	3	CLAVOS	ACEMEX
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	10	CHIAPAS	SINMAR

Figura 5.35 Producto cartesiano entre la relación del paso 4 y PEDIDO.

6. Restringir la relación del paso 5 donde los valores de los atributos P# coincidan. Este paso implica leer 24 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.36) y aproximadamente se requieran 28 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD	P#1	DESCRIPCION	FABRICA
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500	50	TUERCA	ACEMEX
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200	60	PINZAS	STANLEY
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200	30	MARTILLO	STANLEY
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160	3	CLAVOS	ACEMEX

Figura 5.36 Restricción del la relación del paso 5.

7. Proyectar el resultado del paso 6 sobre ***DESCRIPCION*** a fin de producir el resultado final deseado.

DESCRIPCION
TUERCA
MARTILLO
PINZAS
CLAVOS

Consulta No. 3.

Tomando como referencia el árbol de esta consulta (figura 4.6) podemos observar las siguientes secuencias de eventos:

1. Restringir la relación PEDIDO a las tuplas donde CANTIDAD > 100. Este paso implica leer 6 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.37) y aproximadamente se requieran 25.5 kb para su almacenamiento:

P#	C#	CANTIDAD
50	130	500
3	30	160
30	130	200
60	130	200

Figura 5.37 Restricción de la relación PEDIDO.

2. Restringir la relación CLIENTE a las tuplas donde CIUDAD = “PUEBLA”. Este paso implica leer 4 tuplas pero produce una relación formada sólo por 2 tuplas (figura 5.38) y aproximadamente se requieran 25 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA

Figura 5.38 Restricción de la relación CLIENTE.

3. Calcular el producto cartesiano entre las relaciones del paso 1 y 2. Este paso implica leer 4 veces cada una de las 2 tuplas de la relación del paso 2. El producto contendrá un total de 8 tuplas (figura 5.39) y aproximadamente se requieran de 35 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD
130	JOSÉ SILVESTRE GARCIA	RIO PANUCO 5934	PUEBLA	50	130	500

	VALDEZ.					
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	3	30	160
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	50	130	500
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	30	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	60	130	200

Figura 5.39 Producto cartesiano entre las relaciones del paso 1 y 2.

4. Restringir la relación del paso 3 donde los valores de los atributos C# coincidan. Este paso implica leer 8 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.40) y aproximadamente se requieran 28.5 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD	P#	C#1	CANTIDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	50	130	500
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	60	130	200
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA	30	130	200
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA	3	30	160

Figura 5.40 Restricción de la relación del paso 3.

5. Proyectar la relación del paso 4 sobre el atributo ***P#*** . Este paso produce la relación siguiente (figura 5.41):

P#
50
60
30
3

Figura 5.41 Proyección de la relación del paso 4.

6. Proyectar de la relación PIEZA a los atributos **DESCRIPCION**, **P#**. Este paso produce la relación siguiente(figura 5.42):

P#	DESCRIPCION
50	TUERCA
20	TORNILLO
30	MARTILLO
60	PINZAS
3	CLAVOS
10	CHIAPAS

Figura 5.42 Proyección de la relación PIEZA

7. Calcular el producto cartesiano entre las relaciones del paso 5 y 6. Este paso implica leer 4 veces cada una de las 6 tuplas de la relación del paso 6. El producto contendrá un total de 24 tuplas (figura 5.43) y aproximadamente se requieran 42 kb. para su almacenamiento:

P#	P#	DESCRIPCION
50	50	TUERCA
50	20	TORNILLO

50	30	MARTILLO
50	60	PINZAS
50	3	CLAVOS
50	10	CHIAPAS
60	50	TUERCA
60	20	TORNILLO
60	30	MARTILLO
60	60	PINZAS
60	3	CLAVOS
60	10	CHIAPAS
30	50	TUERCA
30	20	TORNILLO
30	30	MARTILLO
30	60	PINZAS
30	3	CLAVOS
30	10	CHIAPAS
3	50	TUERCA
3	20	TORNILLO
3	30	MARTILLO
3	60	PINZAS
3	3	CLAVOS
3	10	CHIAPAS

Figura 5.43 Producto cartesiano de las relaciones del paso 5 y 6.

8. Restringir la relación del paso 7 donde los valores de los atributos P# coincidan. Este paso implica leer 24 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.44) y aproximadamente se requieran 25.5 kb. para su almacenamiento :

P#	P#	DESCRIPCION
50	50	TUERCA
60	60	PINZAS
30	30	MARTILLO
3	3	CLAVOS

Figura 5.44 Restricción de la relación del paso 7.

9. Proyectar el resultado del paso 8 sobre ***DESCRIPCION*** a fin de producir el resultado final deseado.

DESCRIPCION
TUERCA
PINZAS
MARTILLO
CLAVOS

Consulta No. 4.

Tomando como referencia el árbol de esta consulta (figura 4.7) podemos observar las siguientes secuencias de eventos:

1. Restringir la relación PEDIDO a las tuplas donde CANTIDAD > 100. Este paso implica leer 6 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.45) y aproximadamente se requieran 25.5 kb. para su almacenamiento:

P#	C#	CANTIDAD
50	130	500
3	30	160

30	130	200
60	130	200

Figura 5.45 Restricción de la relación PEDIDO.

2. Proyectar la relación del paso 1 sobre los atributos ***P#*** y **C#**. Este paso produce la relación siguiente (figura 5.46):

P#	C#
50	130
3	30
30	130
60	130

Figura 5.46 Proyección de la relación del paso 1.

3. Restringir la relación CLIENTE a las tuplas donde CIUDAD = “PUEBLA”. Este paso implica leer 4 tuplas pero produce una relación formada sólo por 2 tuplas (figura 5.47) y aproximadamente se requieran 25 kb. para su almacenamiento:

C#	NOMBRE	DIRECCION	CIUDAD
130	JOSÉ SILVESTRE GARCIA VALDEZ.	RIO PANUCO 5934	PUEBLA
30	DANIEL DE J. ESTRADA CAMACHO	2ª. PTE NORTE 650	PUEBLA

Figura 5.47 Restricción de la relación CLIENTE.

4. Proyectar la relación del paso 3 sobre el atributo **C#**. Este paso produce la relación siguiente (figura 5.48):

C#
130
30

Figura 5.48 Proyección de la relación del paso 3.

5. Calcular el producto cartesiano entre las relaciones del paso 2 y 4. Este paso implica leer 4 veces cada una de las 2 tuplas de la relación del paso 4. El producto contendrá un total de 8 tuplas (figura 5.49) y aproximadamente se requieran 28.5 kb. para su almacenamiento:

P#	C#1	C#
50	130	130
50	130	30
3	30	130
3	30	30
30	130	130
30	130	30
60	130	130
60	130	30

Figura 5.49 Producto cartesiano de las relaciones del paso 3 y 4.

6. Restringir la relación del paso 5 donde los valores de los atributos **C#** y **C#** coincidan. Este paso implica leer 8 tuplas pero produce una relación formada sólo por 3 tuplas (figura 5.50) y aproximadamente se requieran 25.5 kb. para su almacenamiento:

P#	C#1	C#
50	130	130
3	30	30
60	130	130
30	130	130

Figura 5.50 Restricción de la relación del paso 5.

7. Proyectar la relación del paso 6 sobre el atributo **P#**. Este paso produce la relación siguiente (figura 5.51):

P#
50
3
60
30

Figura 5.51 Proyección de la relación del paso 6.

8. Proyectar de la relación PIEZA a los atributos **DESCRIPCION**, **P#**. Este paso produce la relación siguiente(figura 5.52):

P#	DESCRIPCION
50	TUERCA
20	TORNILLO
30	MARTILLO
60	PINZAS
3	CLAVOS
10	CHIAPAS

Figura 5.52 Proyección de la relación PIEZA

9. Calcular el producto cartesiano entre las relaciones del paso 7 y 8. Este paso implica leer 4 veces cada una de las 6 tuplas de la relación del paso 8. El producto contendrá un total de 24 tuplas (figura 5.53) y aproximadamente se requieran 42 kb. para su almacenamiento:

P#	P#1	DESCRIPCION
50	50	TUERCA
50	20	TORNILLO
50	30	MARTILLO
50	60	PINZAS
50	3	CLAVOS
50	10	CHIAPAS
3	50	TUERCA
3	20	TORNILLO
3	30	MARTILLO
3	60	PINZAS
3	3	CLAVOS
3	10	CHIAPAS
60	50	TUERCA
60	20	TORNILLO
60	30	MARTILLO
60	60	PINZAS

60	3	CLAVOS
60	10	CHIAPAS
30	50	TUERCA
30	20	TORNILLO
30	30	MARTILLO
30	60	PINZAS
30	3	CLAVOS
30	10	CHIAPAS

Figura 5.53 Producto cartesiano de las relaciones del paso 7 y 8.

10. Restringir la relación del paso 9 donde los valores de los atributos **P**# y **P#** coincidan. Este paso implica leer 24 tuplas pero produce una relación formada sólo por 4 tuplas (figura 5.54) y aproximadamente se requieran 25.5 kb. para su almacenamiento:

P#	P#1	DESCRIPCION
50	50	TUERCA
3	3	CLAVOS
60	60	PINZAS
30	30	MARTILLO

Figura 5.54 Restricción de la relación del paso 9.

11. Proyectar el resultado del paso 10 sobre ***DESCRIPCION*** a fin de producir el resultado final deseado.

DESCRIPCION
TUERCA
CLAVOS
PINZAS
MARTILLO

5.2 Resultados generados por los procesos.

Como se ha mencionado al inicio de este capítulo lo que se busca es reducir las operaciones tanto de lecturas como escrituras, así como también reducir las relaciones en su número de tuplas antes de la realización del producto cartesiano, obteniendose así los mismos resultados de cada una de las consultas planteadas y como lo podemos observar los resultados han sido iguales en cada una de la consultas.

Ejemplo 1).

Considerando la misma información de cada una de las relaciones **BOOKS, PUBLISHERS**, **BORROWERS, LOANS,** almacenados en disco, los resultados emitidos por el sistema para cada una de las consultas son:

TITLE
CÁLCULO DIFERENCIAL E INTEGRAL
INTRODUCCIÓN A LOS SISTEMAS DE BASE DE DATOS

Que son los resultados obtenidos en forma analítica.

Ejemplo 2).

De manera similar al ejemplo anterior considerando la misma información de cada una de las relaciones **PE, P** y **C**, almacenados en disco, los resultados emitidos por el sistema para cada consulta son:

DESCRIPCION
TUERCA
CLAVOS
PINZAS

Que son los resultados obtenidos en forma analítica.

Capitulo 6

Conclusiones

En el desarrollo de ésta tesina se han involucrado a dos ejemplos de tres y cuatro consultas candidatos respectivamente para realizar la optimización de las operaciones de E/S de una consulta en particular. Cada consulta ha sido construida mediante la combinación de procedimientos candidatos, analizados e implementados en un sistema. Observamos generalmente que las consultas de los dos ejemplos son confiables y aceptables para optimizar a las consultas pero hemos seleccionado que el más barato, es decir el mejor de ellos en lo que se refiere al menor número de operaciones de E/S es:

Ejemplo 1).

La consulta No. 2 , considerando los siguientes criterios:

- Como podemos observar ésta consulta cumple con el objetivo de realizar el mínimo número de lecturas que son 36 lecturas en ella y,
- aproximadamente se requieran en total de las operaciones 148 kb. para su almacenamiento.

Ejemplo 2).

La consulta No. 4, considerando los siguientes criterios:

- Es la consulta que cumplen con nuestro objetivo de realizar el menor número de lecturas, ésta lleva a cabo un total de 74 lecturas y,
- aproximadamente se requieran en total de las operaciones 172 kb. para su almacenamiento

Como podemos observar éstas consultas son las mejores, es decir aquellas que manejan menos cantidades de información y por lo consiguiente menos operaciones de escrituras como lecturas, así como también menos espacio de memoria para las operaciones intermedias que se realizan.

Apéndice

```
captura()
 {
   FILE *fp,*fp1,*fp2;
   char * como, *ptrozo;
   char texto[100],texto1[30];
   if((fp=fopen("CLIENTE","r"))==NULL)
     {
      printf("NO SE PUEDE ABRIR EL ARCHIVO");
      exit(1);
     }
   i=0;
   while(!feof(fp))
    {
     como=fgets(&texto[0],100,fp);
     while(como!=NULL)
      {
       ptrozo=strtok(&texto,"|");
       sscanf(ptrozo,"%d",&client[i].numc);
       ptrozo=strtok(NULL,"|");
       sprintf(client[i].nombre,"%s",ptrozo);
       ptrozo=strtok(NULL,"|");
       sprintf(client[i].direccion,"%s",ptrozo);
       ptrozo=strtok(NULL,"|");
       sprintf(client[i].ciudad,"%s",ptrozo);
       i++;
       como=fgets(&texto[0],100,fp);
      }
    }
   fclose(fp);
   if((fp1=fopen("PEDIDO","r"))==NULL)
     {
      printf("\nNO SE PUEDE ABRIR EL ARCHIVO");
      exit(1);
     }
```

```
j=0;
while(!feof(fp1))
 {
  como=fgets(&texto[0],30,fp1);
  while(como!=NULL)
   {
    ptrozo=strtok(&texto,"|");
    sscanf(ptrozo,"%d",&pe[j].numpe);
    ptrozo=strtok(NULL,"|");
    sscanf(ptrozo,"%d",&pe[j].numc);
    ptrozo=strtok(NULL,"|");
    sscanf(ptrozo,"%d",&pe[j].cant);
    j++;
    como=fgets(&texto[0],30,fp1);
  }
}
fclose(fp1);
if((fp2=fopen("PIEZA","r"))==NULL)
  {
   printf("NO SE PUEDE ABRIR EL ARCHIVO");
   exit(1);
  }
y=0;
while(!feof(fp2))
 {
  como=fgets(&texto[0],30,fp2);
  while(como!=NULL)
   {
    ptrozo=strtok(&texto,"|");
    sscanf(ptrozo,"%d",&piezas[y].numpi);
    ptrozo=strtok(NULL,"|");
    sprintf(piezas[y].descripcion,"%s",ptrozo);
    ptrozo=strtok(NULL,"|");
    sprintf(piezas[y].fabrica,"%s",ptrozo);
    y++;
    como=fgets(&texto[0],30,fp2);
  }
```

```
   }
   fclose(fp2);
}

clienteXpedido()
{
  for(n=0;n<i;n++)
   {
    for(h=0;h<j;h++)
     {
      propexp[k].numc1=client[n].numc;
      strcpy(propexp[k].nombre,client[n].nombre);
      strcpy(propexp[k].direccion,client[n].direccion);
      strcpy(propexp[k].ciudad,client[n].ciudad);
      propexp[k].numped=pe[h].numpe;
      propexp[k].numc=pe[h].numc;
      propexp[k].cant=pe[h].cant;
      k=k+1;
     }
   }
}

clienXpediXpie()
{
 for(b=0;b<k;b++)
  {
   for(x=0;x<y;x++)
    {
      proxfinal[a].numclif=propexp[b].numc1;
      strcpy(proxfinal[a].nombref,propexp[b].nombre);
      strcpy(proxfinal[a].direccionf,propexp[b].direccion);
      strcpy(proxfinal[a].ciudadf,propexp[b].ciudad);
      proxfinal[a].numpef=propexp[b].numped;
      proxfinal[a].numcf=propexp[b].numc;
      proxfinal[a].cantf=propexp[b].cant;
      proxfinal[a].numpief=piezas[x].numpi;
      strcpy(proxfinal[a].descripcionf,piezas[x].descripcion);
```

```
      strcpy(proxfinal[a].fabricaf,piezas[x].fabrica);
       a=a+1;
     }
    }
  }

 consulta()
{
  int ban=0,xx=1;
  for (i=0;i<a;i++)
      {
       if(proxfinal[i].numpief==proxfinal[i].numpef &&
             proxfinal[i].numcf==proxfinal[i].numclif && proxfinal[i].cantf>100)
             {
              r=strlen(proxfinal[i].ciudadf);
              strcpy(compa,proxfinal[i].ciudadf);
              Mayusculas_minusculas(compa,r);
              res=strcmp(compa,"puebla\n");
              if(res==0)
              {
               ban=1;
               gotoxy(35,5+xx);printf("%s",proxfinal[i].descripcionf);
               xx++;
              }
              else
              if (ban==1) printf("NO EXISTE DESCRIPCION");
             }
      }
  }
```

```
selecpedido()

{
  for(k1=0;k1<j;k1++)
    {
     if(pe[k1].cant>100)
       {
            pec[k].numpe=pe[k1].numpe;
            pec[k].numc=pe[k1].numc;
            pec[k].cant=pe[k1].cant;
            k=k+1;
       }
    }
}



seleccliente()
{
  for (n=0;n<i;n++)
    {
     r=strlen(client[n].ciudad);
     strcpy(compa,client[n].ciudad);
     Mayusculas_minusculas(compa,r);
     res=strcmp(compa,"puebla\n");
     if (res==0)
       {
            clientcon[a].numc=client[n].numc;
            strcpy(clientcon[a].nombre,client[n].nombre);
            strcpy(clientcon[a].direccion,client[n].direccion);
            strcpy(clientcon[a].ciudad,client[n].ciudad);
            a=a+1;
       }
    }
  }
```

```
pedidoXcliente()
{
   for(x1=0;x1<k;x1++)
     {
      for(y=0;y<a;y++)
      {
       pexcliencon[b].numc1=clientcon[y].numc;
       strcpy(pexcliencon[b].nombre,clientcon[y].nombre);
       strcpy(pexcliencon[b].direccion,clientcon[y].direccion);
       strcpy(pexcliencon[b].ciudad,clientcon[y].ciudad);
       pexcliencon[b].numped=pec[x1].numpe;
       pexcliencon[b].numc=pec[x1].numc;
       pexcliencon[b].cant=pec[x1].cant;
       b=b+1;
      }
     }
 }

selecpedixclien()
{
 for(i=0;i<=b;i++)
     {
      for(x1=0;x1<=y;x1++)
             {
              producfinal[y1].numped=pexcliencon[i].numpe;
              producfinal[y1].numpi=piezas[x1].numpi;
              strcpy(producfinal[y1].descripcion,piezas[x1].descripcion);
              y1++;
             }
     }
}

consulpediXclien()
 {
  for(y=0;y<b;y++)
     {
      if(pexcliencon[y].numc==pexcliencon[y].numc1)
```

```
            {
             consulpexclien[y].numc=pexcliencon[y].numc;
             strcpy(consulpexclien[y].nombre,pexcliencon[y].nombre);
             strcpy(consulpexclien[y].direccion,pexcliencon[y].direccion);
             strcpy(consulpexclien[y].ciudad,pexcliencon[y].ciudad);
             consulpexclien[y].numped=pexcliencon[y].numped;
             consulpexclien[y].numc1=pexcliencon[y].numc1;
             consulpexclien[y].cant=pexcliencon[y].cant;
            }
     }
}
consulfinal()
{
  int ban1=0,x2=1;
 for(x1=0;x1<b;x1++)
    {
      for(x=0;x<=y;x++)
      {
       porfin[c1].numc=consulpexclien[x1].numc;
       strcpy(porfin[c1].nombre,consulpexclien[x1].nombre);
       strcpy(porfin[c1].direccion,consulpexclien[x1].direccion);
       strcpy(porfin[c1].ciudad,consulpexclien[x1].ciudad);
       porfin[c1].numped=consulpexclien[x1].numped;
       porfin[c1].numc1=consulpexclien[x1].numc1;
       porfin[c1].cant=consulpexclien[x1].cant;
       porfin[c1].numpi=piezas[x].numpi;
       strcpy(porfin[c1].descripcion,piezas[x].descripcion);
       strcpy(porfin[c1].fabrica,piezas[x].fabrica);
       if(porfin[c1].numpi==porfin[c1].numped)
            {
             ban1=1;
             gotoxy(35,5+x2);printf("%s",porfin[c1].descripcion);
             x2++;
            }
       c1=c1+1;
      }
     }
```

```
    if(ban1==0)
     {
      system("cls");
     gotoxy(20,5);printf("NO EXISTE DESCRIPCION");
    }
 }

ultimoproduc()
 {
  int x3=1,ban2=0;
  for(y=0;y<=y1;y++)
    {
            if (producfinal[y].numped==producfinal[y].numpi)
            {
             ban2=1;
             gotoxy(35,5+x3);printf("%s",producfinal[y].descripcion);
             x3++;
            }
     }
  if(ban2==0)
    {
    gotoxy(25,5);printf("NO EXISTE DESCRIPCION");
    }
 }

proyspedidoXcliente()
{
   for(j=0;j<k;j++)
     {
      for(i=0;i<a;i++)
      {
       pexcliencon[b].numpe=pec[j].numpe;
       pexcliencon[b].numc1=pec[j].numc;
       pexcliencon[b].numc=clientcon[i].numc;
       b=b+1;
      }
    }
```

```
 }
otro()
{
 for(i=0;i<b;i++)
    {
     if(pexcliencon[i].numc1==pexcliencon[i].numc)
            {
              proxante[d].numpe=pexcliencon[i].numpe;
            }
      d=d+1;
     }
   }

producotro( )
{
 int ban3=0,x4=1;
 for(i=0;i<d;i++)
 {
  for(x=0;x<y;x++)
  {
   prodxfinal[e].numpe=proxante[i].numpe;
   prodxfinal[e].numpi=piezas[x].numpi;
   strcpy(prodxfinal[e].descripcion,piezas[x].descripcion);
   if (prodxfinal[e].numpe==prodxfinal[x].numpi)
     {
      ban3=1;
      gotoxy(35,5+x4);printf("%s",prodxfinal[e].descripcion);
      x4++;
     }
   e=e+1;
  }
 }
 if(ban3==0)
  {
  printf("NO EXISTE DESCRIPCION");
  }
}
```