CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I. P. N.

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

REPRESENTACIÓN DE SISTEMAS DE CONSULTA EN TABLEAUX Y SU IMPLEMENTACIÓN EN UNA BASE DE DATOS RELACIONAL

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS EN LA ESPECIALIDAD DE INGENIERÍA ELÉCTRICA

P R E S E N T A:

JOSÉ ALEJANDRO MARTÍNEZ HERNÁNDEZ

DIRECTOR DE TESIS

DR. SERGIO VICTOR CHAPA VERGARA

MÉXICO, D.F. **1998**

C O N T E N I D O

Quisiera expresar un sincero agradecimiento al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional por la oportunidad que me dieron para avanzar en mi formación profesional y cumplir con uno mas de mis objetivos. En particular a la Sección de Computación del departamento de Ingeniería Eléctrica y al Dr. Sergio V. Chapa Vergara por su confianza y comprensión como maestro y amigo. A los Doctores Arturo Díaz Pérez y Juan Francisco Corona Burgueño por su interés y comentarios en este trabajo de tesis.

A mis padres, Sara Hernández Angeles y Luis Martínez Godinez por el amor, coraje, y trabajo infinito, porque con ello se puede lograr todo en la vida.

A mis Esposa Raquel Bueno Arias porque en ella encontre un nuevo sentido a la vida y me impulso para que continuar con lo que habia iniciado. A mis hijos Alejandro, María Fernanda y Guadalupe Martinez Bueno para que vean en este esfuerzo, un ejemplo de que las cosas si se pueden lograr.

A mis suegros Raquel y Enrique Bueno Padilla por su apoyo y a todos mis hermanos y demas familiares les dedico este esfuerzo y con ello solo una parte de mi vida, finalmente quisiera dedicar este trabajo a todos mis sobrinos para que vean que la vida es más interesante cuando se realiza un sacrificio y un esfuerzo por alcanzar sus metas.

Resumen

Basándonos en los fundamentos del modelo de datos relacional, un sistema de consulta formal puede ser llevado a su implementación en un Sistema Manejador de Base de Datos. El paradigma de tableaux descrito por Ullman y Maier es usado para representación de sistemas formales de consulta [Aho 1979][Ullman 1982][Maier 1983]. Tableaux se implanta con sus reglas de composición y sus estructuras de datos para permitir un medio de consulta eficaz tanto de los lenguajes de QBE como el de SQL a través de las reglas de transformación dadas por los fundamentos del modelo relacional de datos y considerando la estrategia de descomposición de procesamiento de consultas de Eugene Wong y Karel Youssefi [Wong 1976]. Proponemos el descriptor de archivos [Chapa 1985] como la estructura de datos principal que comunica al sistema manejador de datos con el sistema de consulta basado en tableaux.

Palabras clave: Modelo relacional de datos / Sistemas Formales de Consulta / Tableaux / Sistema Manejador de Bases de Datos / Estructuras de Datos / Descriptor de Archivos / QBE / SQL.

Introducción

El modelo de datos relacional presentado por Codd [Codd 1970] ha influido en el diseño de sistemas actuales de bases de datos, y un número de nuevos sistemas se construyen alrededor de este modelo. Sus principales virtudes son la independencia de datos y la capacidad de formular consultas no procedurales en términos del cálculo relacional y, en menor extensión, del álgebra relacional. Los sistemas de consulta no procedurales son de alto nivel porque liberan al usuario de tener que elaborar una respuesta deseada, dejando que el trabajo de esta determinación caiga fundamentalmente en el procesador del lenguaje de consulta del sistema de la base de datos, mientras que, en sistemas de consulta procedurales como el álgebra relacional se da un conjunto de operaciones en relaciones y el orden en que deben de ejecutarse. Basándonos en los fundamentos del modelo relacional un sistema de consulta puede ser llevado a su implementación en un Sistema Manejador de Base de Datos. Resultados de varios grados de generalidad en estrategias de construcción y procesamiento de sistemas de consulta formales han sido reportados por [Astrahan 1975][Wong 1976] [Ullman 1979] [Aho 1979], sin embargo la carencia de un enfoque general para la construcción de sistemas manejadores de bases de datos junto con sistemas de consulta y la interrelación con sus estructuras de datos internas mantienen un impedimento principal para alcanzar un grado satisfactorio de conocimiento de sistemas manejadores de base de datos [Chapa 1985] [Guzmán 1985]. El objetivo principal de nuestro trabajo es implantar el paradigma de tableaux para procesamiento de sistemas de consulta descrito por Tableaux [Aho 1979] [Ullman 1982] [Maier 1983] con sus reglas de composición y sus estructuras de datos para permitir un medio de consulta eficaz tanto de los lenguajes de QBE como el de SQL a través de las reglas de transformación dadas por los fundamentos del modelo relacional de datos y considerando la estrategia de descomposición de procesamiento de consultas de Eugene Wong y Karel Youssefi [Wong 1976]. Utilizamos la metodología Tableaux como un traductor de lenguajes de alto nivel y proponemos el descriptor de archivos [Chapa 1985] como la principal estructura de datos que comunican al sistema manejador de datos con el sistema de consulta basado en tableaux. Este sistema es la interfaz con un Sistema Manejador de Base de Datos, que incluye un modelo físico con un sistema de organización basado en árboles B desarrollado por Al Stevens [Stevens 1987].

Este trabajo fue hecho en el contexto de desarrollar un consultador para una base de datos que contiene información referente a coordenadas geográficas para la localización de pozos petroleros junto con la tesis de maestría "Sist*emas de Información Geográfica para la Exploración Petrolera*" que obtiene información gráfica de una área seleccionada [Fiorentino 1996] y es la base para nuevos desarrollos en *Openstep* y *Raphsody*. El trabajo contempla dos niveles de usuarios: *usuario administrador* donde se da un conjunto de herramientas para definir la base de datos, de tal manera que el sistema manejador sea la interfaz entre el administrador de la base de datos y la base de datos física; *usuario final* en donde se ofrece un lenguaje de consulta no procedural, con facilidades para seleccionar datos compartidos, utiliza vistas de tablas propias y externas definidas en otra base de datos.

El Lenguaje de consulta desarrollado en lenguaje "C" es un prototipo del lenguaje SQL (Structured Query Language) y es traducido a la metodología tableaux dentro del modelo relacional. La metodología tableaux, no es un sistema de consulta completo en comparación con el álgebra relacional por considerar solo los operadores selección proyección y junta, sin embargo, las consultas por tableaux abarcan un gran número de consultas que pudieran surgir en aplicaciones de bases de datos.

El trabajo se encuentra dividido en cuatro capítulos principales y, uno de implementación de una aplicación de base de datos y conclusiones.

El capítulo 1 analiza la importancia del modelo relacional para el desarrollo de sistemas de consulta basados en el álgebra relacional, cálculo relacional de dominios y cálculo relacional de tuplas. Codd creador del modelo relacional introduce los conceptos de lenguajes de alto nivel (sin una sintaxis específica), permitiendo expresar operaciones sobre grandes cantidades de información a la vez [Codd 1982]. Este capítulo también describe el lenguaje ***"Alpha"*** con su algoritmo de reducción el cual permite transformar sus expresiones en una secuencia de operaciones semánticamente equivalentes al álgebra relacional, usado también como la medida de selectividad sobre la cual cualquier lenguaje de consulta debe compararse basado en el modelo relacional [Codd 1972a].

El capítulo 2 se estudia las consultas por ***"Tableaux"*** que también son equivalentes a las expresiones del sublenguaje Alpha [Aho 1979][Ullman 1982][Maier 1983], utiliza un subconjunto de operadores del álgebra relacional, donde únicamente se consideran: la **S-*restricción*** con la comparación de igualdad, la **P-*proyección*** y la **J-*junta natural*** y es objeto de implementación en el desarrollo del presente trabajo. De este capítulo se llega a los resultados que la base es *SPJ* con la principal conjunción para resolver consultas. Sin embargo, estas no resultan completas aunque resuelven muchas de ellas. Para esto en este capítulo presentamos las diversas extensiones. La primera fue llegar a que sistemas algebraicos mantienen una equivalencia en Tableaux, y la segunda es que un Tableaux mantienen una equivalencia con sistemas algebraicos aspecto fundamental para validar nuestra transformación en la implementación de TabluSql. Por otro lado se ve Tableaux en conjuntos y uniones. En este capítulo se encuentra que en un lenguaje de alto nivel no esta determinado directamente el modo de realizar operaciones en la base de datos, sin embargo apoyados en la metodología tableaux se puede realizar una transformación formal de una expresión basado en sistemas de consulta del cálculo relacional en una expresión equivalente en tableaux con marca que consiste básicamente en una expresión que contiene los tres operadores SPJ. Estos operadores si tienen una implementación natural para procesamiento de consultas en sistemas computacionales que se pueden desarrollar, ya que internamente las relaciones de modelo relacional pueden manipularse en tablas con las operaciones SPJ.

El capítulo 3 presenta la implantación del paradigma de tableaux considerando el algoritmo descrito en el capítulo 2 y la estrategia de descomposición de procesamiento de

consultas de Eugene Wong y Karel Youssefi [Wong 1976]. También se esbozan las fases de construcción del módulo de consultas: la fase del analizador consiste en revisar el lenguaje de consulta haciendo un análisis sintáctico y semántico basado en un autómata de tipo *top-down;* la fase tableaux se obtiene un "*query-tableaux*" equivalente a la consulta del usuario y se establece una *matriz de ejecución*; y la fase de ejecución recupera renglones de datos para una declaración SELECT (*summary*) usando la *matriz tableaux*; y la fase recuperación (realiza lecturas físicas o lecturas/escrituras lógicas). Finalmente, se describe el procesamiento de las consultas con un algoritmo de búsqueda basado en reglas y las principales estructuras de datos que contienen la definición interna del descriptor de archivos y del diccionario de datos. El concepto de descriptor de datos viene a ser una de las piezas fundamentales para la comunicación con el sistema manejador de la base de datos presentado en el siguiente capitulo [Chapa 1985].

En el capítulo 4 se tratan las principales funciones del sistema manejador de la base de datos y las estructuras de datos que contienen la definición interna del descriptor de archivos y del diccionario de datos. Se describen las funciones del manejador de archivos de datos y las funciones del manejador de archivos b-tree desarrollado por Al Stevens [Stevens 1987] con las cuales, las consultas de datos son más rápidas al considerar la navegación por llaves primarias, secundarias, terciarias, cuaternarias usadas generalmente en una operación equi-junta entre dos o más tablas-archivo. Finalmente se presenta la interrelación de las estructuras de datos que contienen la organización interna del modelo de bases de datos relacional implantado.

El capítulo 5 presenta el sistema de consulta TabluSql y la implantación de una aplicación, con dos niveles de usuarios: ***usuario administrador*** que crea la base de datos, define tablas, atributos de campos, e índices; Y ***usuario final*** que carga una base de datos, imprime el contenido de las tablas, actualiza datos con funciones de insertar, borrar y modificar, consulta la información de la base de datos en SQL, consulta tablas externas (vistas) definidas por el administrador, y consulta la definición de la base de datos (diccionario).

Capítulo 1

Los Sistemas de Consulta Formales

En este capítulo cubriremos un análisis de los sistemas de consulta formales: procedurales; y no procedurales del modelo relacional [Codd 1970] [Codd 1972a][Ullman 1982][Maier 1983]. Describiremos el *álgebra relacional* que expresa selecciones, restricciones, y combinaciones de relaciones en una base de datos. El *cálculo relacional de tuplas* que consiste en el cálculo de predicados de la lógica de predicados de primer orden. El *cálculo de dominios* que es similar, excepto que el rango de variables es sobre dominios simples en vez de tuplas enteras. Analizaremos también el lenguaje Alpha que se forma de expresiones del cálculo relacional más la operación de proyección. Finalmente se analiza el algoritmo de reducción de Alpha, que transforma las expresiones Alpha en una secuencia de operaciones semánticamente equivalentes del álgebra relacional, usado también como la medida de selectividad sobre la cual cualquier lenguaje de consulta debe compararse basado en el modelo relacional [Codd 1972a]. Siendo nuestro propósito obtener del modelo relacional y de los sistemas de consulta formales la base para la implantación de un sistema de consulta basado en los fundamentos de bases de datos relacionales

En sistemas de base de datos una consulta es una computación sobre relaciones para derivar nuevas, como lo contempla la definición bien fundamentada y la completitud del modelo relacional. Un sistema de consulta, es relacionalmente completo si permite, que la expresión de cualquier consulta *(query)* pueda ser expresada en un simple predicado del llamado cálculo relacional (que se basa en la lógica de predicados de primer orden), y más concretamente en expresiones Alpha. El sistema de consulta sirve para expresar las preguntas en un lenguaje específico de programación que permite formular comandos en un sistema manejador de base de datos. Los sublenguajes de consulta propuestos por Codd permiten que el manejo de la información se lleve a cabo mediante un procesamiento en conjunto a los datos tabulares o relaciones enteras. Resultados importantes que se encuentran en el artículo *"Relational Base: A Practical Foundation for Productivity"* [Cood 1982], han servido para introducir los conceptos de lenguajes de alto nivel (sin una sintaxis específica) que permiten expresar operaciones sobre grandes cantidades de información a la vez. *El sistema de consulta transforma las peticiones de los usuarios en requerimientos de un Sistema Manejador de Base de Datos (SMBD).* Un SMBD entre otras funciones consta de un conjunto de rutinas para recuperar y almacenar los datos almacenados de la base de datos.

Los sistemas de consulta relacionales caen en dos categorías: Los sistemas de consulta procedurales que usan el álgebra relacional y los sistemas de consulta no procedurales que se basan en el cálculo relacional, tanto el álgebra relacional como el cálculo relacional fueron introducidos por Codd [1970]. El álgebra relacional usa una serie de operadores procedurales de alto nivel como son la "***junta***" y la "***proyección***", sus consultas se establecen usando estos operadores en relaciones con un orden de ejecución. El cálculo relacional esta constituido de $\{\Sigma, R\}$ donde Σ es el alfabeto que consta de un conjunto de símbolos $\{\forall, \exists\}$ y R={conjunto de reglas sintácticas}, la forma de expresar sus consultas son de alto nivel porque liberan al usuario de tener que elaborar una respuesta deseada, dejando que el trabajo de esta determinación caiga fundamentalmente en el

procesador del lenguaje de consulta del sistema de la base de datos. Para analizar las características de estos sistemas de consulta, daremos la definición del concepto de relación y después describiremos los sistemas de consulta formales.

Formalmente un esquema de relación R es un conjunto finito de nombres de atributos $\{A_1, A_2, \ldots, A_n\}$. Correspondiendo a cada nombre de atributo A_i un conjunto de valores D_i llamado el dominio de A_i o $dom(A_i)$. Los nombres de atributo son también llamados símbolos atributos o simplemente atributos. Los dominios son arbitrarios, conjuntos no vacíos, finitos o contablemente infinitos. Sea $D = D_1 \cup D_2 \cup \ldots \cup D_n$. Una relación *r* sobre un esquema de relación R es un conjunto de transformaciones $\{t_1, t_2, \ldots, t_p\}$ de R a D con la restricción de que cada transformación $t \in r$, $t(A_i)$ debe estar en D_i, $1 \leq i \leq n$. Las transformaciones son llamadas tuplas.

1.1 El Algebra Relacional

Los sistemas de consulta basados en el álgebra relacional se consideran procedurales de bajo nivel , debido a que las consultas se establecen por el usuario usando un conjunto de operadores en relaciones con un orden en la ejecución de estos, esto limita a que el procesador de la consulta de la base de datos optimice la recuperación de los datos, ya que el orden al evaluar literalmente estas expresiones no siempre resultan ser lo más adecuado, llegando a realizar procesos con un resultado exponencial, inesperado e inconsistente, además requiere que los usuarios de bases de datos tengan un conocimiento previo de los operadores del álgebra relacional.

Nos referiremos a los operadores unión, intersección, diferencia, selección, junta natural, división, y theta-junta, con relaciones constantes y relaciones regulares, como el álgebra relacional. Cualquier expresión legalmente formada usando estos operadores y relaciones es una expresión algebraica.

La unidad básica de manipulación en el álgebra relacional es la relación (denominada también tabla), siempre que las relaciones cumplan con las reglas establecidas por el modelo relacional y, dado que las tablas son conjuntos, todas las operaciones usuales del álgebra de conjuntos aplicadas a las relaciones y otras propias de bases de datos relacionales, forman el álgebra relacional. Las expresiones del álgebra relacional, constan de operadores relacionales y de operandos que pueden llegar a ser relaciones constantes, o variables de grado fijo. Las operaciones en relaciones tienen como regla principal, la derivación de nuevas relaciones, las operaciones básicas son: unión, diferencia, producto cartesiano, proyección, restricción (figura 1.1), y los operadores derivados de los anteriores: junta natural, división e intersección (figura 1.2). La notación del álgebra relacional de F. Codd, se puede observar en el lenguaje ISBL (Information System Base Language) [Todd 1978], donde se tiene la implantación pura del álgebra relacional y ha servido como lenguaje intermedio para lenguajes de consulta de más alto nivel como en el pictografico LIDA "Lenguaje iconográfico para el desarrollo de aplicaciones" [Chapa 1981].

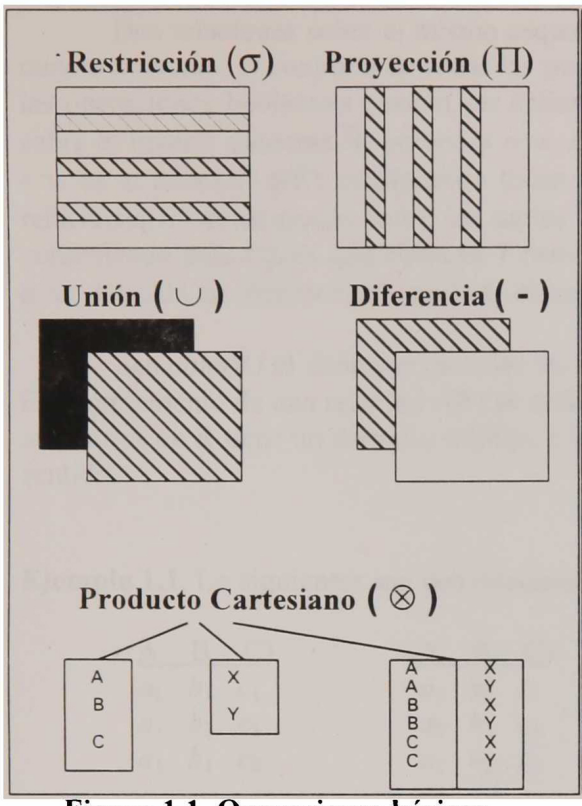


Figura 1.1. Operaciones básicas.

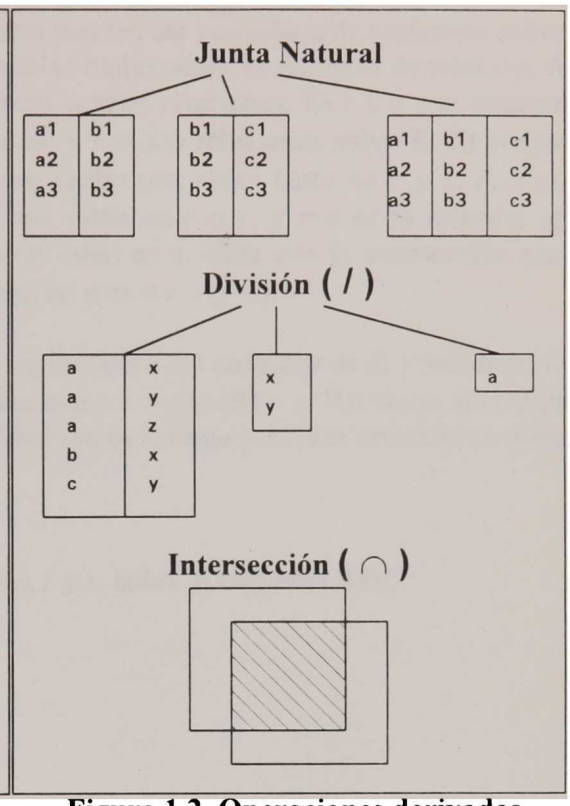


Figura 1.2. Operaciones derivadas.

Definición 1.1. Sea U un conjunto de atributos llamado el universo. Sea D un conjunto de dominios, y sea *dom* una función total de U a D. Sea $R=\{R_1, R_2,..., R_p\}$ un conjunto de esquemas de relación distintos, donde $R_i \subseteq U$ para $1 \leq i \leq p$, Sea $d=\{r_1, r_2,..., r_p\}$ un conjunto de relaciones, tal que r_i es un relación sobre R_i para $1 \leq i \leq p$. Sea θ un conjunto de comparadores sobre dominios en D, incluyendo al menos comparadores de igualdad y desigualdad para cada dominio. El álgebra relacional sobre todo el universo de atributos con un conjunto de dominios, un conjunto de esquemas de relación, y un conjunto de comparadores se expresa como la 7-tupla **R**=(U, D, *dom*, R, d, θ, O), donde O es el conjunto de operadores unión, intersección, diferencia, proyección, junta natural, y divide, la selección usando comparadores en θ, y conectivos lógicos y Theta junta usando comparadores en θ. Una expresión algebraica sobre R es cualquier expresión formada legalmente (de acuerdo a las restricciones sobre los operadores) de las relaciones en d y relaciones constantes sobre esquemas en U, usando los operadores en O.

Los paréntesis en las expresiones son permitidos, y se asume que no hay precedencia de los operadores binarios, excepto para la precedencia usual de ∩ y ∪. También se omiten paréntesis para cadenas de relaciones conectadas por el mismo operador, si la operación es asociativa. Analicemos en la sección 1.1.1 como las operaciones booleanas sobre conjuntos se aplican en relaciones, y en las secciones siguientes algunos ejemplos de los tres operadores: selección, proyección, y junta.

1.1.1 Operaciones Booleanas

Dos relaciones sobre el mismo esquema pueden ser considerados conjuntos sobre el mismo universo, el conjunto de todas las posibles tuplas sobre el esquema de relación. Así, las operaciones booleanas pueden ser aplicadas a tales relaciones. Si r y s son relaciones sobre el mismo esquema R, entonces $r \cap s$, $r \cup s$, y $r - s$ son relaciones sobre R. El conjunto $r \cap s$ es la relación q(R) conteniendo todas las tuplas que están tanto en r y s, $r \cup s$ es la relación q(R) conteniendo todas las tuplas que están en r o s, y $r - s$ es la relación q(R) conteniendo esas tuplas que están en r pero no están en s. Note que la intersección puede estar definida en términos del conjunto diferencia: $r \cap s = r - (r - s)$.

Sea *dom*(R) el conjunto de todas las tuplas sobre los atributos de R y sus dominios. El complemento de una relación r(R) se define como $r = dom(R) - r$. Por tanto, si cualquier atributo A en R tiene un dominio infinito, r también es infinito y no una relación en nuestro sentido.

Ejemplo 1.1. Lo siguientes son dos relaciones, r y s, sobre el esquema ABC:

r(A	B	C)
a_1	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_2

s(A	B	C)
a_1	b_2	c_1
a_2	b_2	c_1
a_2	b_2	c_2

Los resultados de las operaciones $r \cap s$, $r \cup s$, $r - s$ son:

$r \cap s$= (A B C)

a_1	b_2	c_1

$r \cup s$= (A B C)

a_1	b_1	c_1
a_1	b_2	c_1
a_1	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2

$r - s$ = (A B C)

a_1	b_1	c_1
a_2	b_1	c_1

Dado $\{ a_1, a_2 \}$, $\{ b_1, b_2, b_3 \}$, y $\{ c_1, c_2 \}$ como los dominios de A, B, y C, el dominio de R y el complemento de r derivado del dominio de R es como se muestra:

dom(R) = (A B C)

a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_1	b_3	c_1
a_1	b_3	c_2
a_2	b_1	c_1
a_2	b_1	c_2
a_2	b_2	c_1
a_2	b_2	c_2

r=*dom*(R)-r = (A B C)

a_1	b_1	c_2
a_1	b_2	c_2
a_1	b_3	c_1
a_1	b_3	c_2
a_2	b_1	c_1
a_2	b_2	c_1
a_2	b_2	c_2
a_2	b_3	c_1
a_2	b_3	c_2

a_2 b_3 c_1
a_2 b_3 c_2

El conjunto de todas las relaciones sobre un esquema dado es cerrado bajo la unión, intersección, conjunto diferencia, y complemento activo. Sin embargo no todas las operaciones preservan las llaves.

1.1.2 El Operador Selecciona

Selecciona es un operador unario sobre relaciones, aplicado a una relación *r*, dá otra relación que es el subconjunto de tuplas de *r* con un cierto valor sobre un atributo especificado. Sea *r* una relación sobre el esquema R, A un atributo en R, y *a* un elemento del *dom*(A). Usando una notación de transformación, $\sigma_{A=a}(r)$ ("selecciona A igual *a* sobre *r*") es la relación $r'(R) = \{t \in r \mid t(A) = a\}$.

Tabla 1.1 VUELOS.

NUMERO	ORIGEN	DESTINO	SALIDA	LLEGADA
84	O'Hare	JFK	03:00p	05:55p
109	JFK	Los Angeles	09:40p	02:42a
117	Atlanta	Boston	10:05p	12:43a
213	JFK	Boston	11:43a	12:45p
214	Boston	JFK	02:20p	03:12P

Ejemplo 1.2. Consideremos los vuelos de una aerolínea con el horario mostrado en la tabla 1.1. La consulta que obtiene los vuelos que parten de "JFK", se expresa en álgebra relacional como:

$\sigma_{ORIGEN="JFK"}$ en VUELOS.

La Tabla 1.3. es el resultado de aplicar $\sigma_{ORIGEN="JFK"}$ en VUELOS

Tabla 1.3 $\sigma_{ORIGEN="JFK"}$ a VUELOS.

NUMERO	ORIGEN	DESTINO	SALIDA	LLEGADA
109	JFK	Los Angeles	09:40p	02:42a
213	JFK	Boston	11:43a	12:45p

Los operadores selecciona conmutan bajo la composición. Sea *r*(R) una relación y sea A y B atributos en R, con $a \in dom(A)$ y $b \in dom(B)$, entonces

$$\sigma_{A=a}(\sigma_{B=b}(r)) = (\sigma_{B=b}(\sigma_{A=a}(r))$$

Esta propiedad se sigue inmediatamente de la definición de selecciona:

$\sigma_{A=a}(\sigma_{B=b}(r)) = (\sigma_{A=a}(\{t \in r \mid t(B) = b\}) =$
$\{t' \in \{t \in r \mid t(B)=b\} \mid t'(A) = a\} = \{t \in r \mid t(A) = a \text{ y } t(B) = b\} =$
$\{t' \in \{t \in r \mid t(A)=a\} \mid t'(B) = b\} = \sigma_{B=b}(\sigma_{A=a}(r))$.

Dado que el orden de selecciona no es importante, escribimos $\sigma_{A=a}$ o $\sigma_{B=b}$ como $\sigma_{A=a\,B=b}$ y $\sigma_{A_1=a_1}$ o $\sigma_{A_2=a_2}$ o ... o $\sigma_{A_n=a_n}$ como $\sigma_{A_1=a_1,A_2=a_2 \cdots A_n=a_n}$ (Las A_i's no son necesariamente distintas). Si x es un X-valor, $\sigma_{X=x}$ es una notación legítima, si interpretamos x como una secuencia de valores más que una transformación.

Selecciona también es distributivo sobre las operaciones booleanas binarias:

$$\sigma_{A=a}(r \gamma s) = \sigma_{A=a}(r) \gamma \sigma_{A=a}(s)$$

donde $\gamma = \cap, \cup, -$, y r y s son relaciones sobre el mismo esquema.

1.1.3 El Operador Proyecta

Proyecta es también un operador unario sobre relaciones, contrariamente a selecciona que elige un subconjunto de renglones en una relación, proyecta elige un subconjunto de columnas, Sea r una relación sobre el esquema R, y sea X un subconjunto de R. La *proyección* de r en X, escrita $\pi_X(r)$, es la relación $r'(X)$ obtenida de sacar columnas correspondientes a los atributos en R – X y eliminar las tuplas duplicadas. En notación de transformación, $\pi_X(r)$ es la relación $r'(X) = \{t(X) \mid t \in r\}$.

Ejemplo 1.3 En los siguientes ejemplos se hacen proyecciones en la relación vuelos (Tabla 1.1), el resultado es otra relación con los atributos especificados en la proyección.

1. $\pi_{\{SALIDA, LLEGADA\}}$ (*vuelos*) = (SALIDA LLEGADA)

03:00p	05:55p
09:40p	02:42a
10:05a	12:43a
11:43a	12:45p
02:20p	03:12p

2. $\pi_{SALIDA}(\pi_{\{SALIDA, LLEGADA\}}$ (*vuelos*)) = π_{SALIDA}(*vuelos*).

03:00p
09:40p
10:05p
11:43a
02:20P

La operación de proyección, conmuta con la selección, cuando el atributo o atributos para la selección están entre los atributos en el conjunto sobre el cual la proyección esta tomando lugar. Si $A \in X$, $X \subseteq R$, y r es una relación sobre R, entonces

$\pi_x(\sigma_{A=a}(r)) = \pi_x(\{t \in r \mid t(A)=a\}) = \{t'(X) \mid t' \in r \{t \in r \mid t(A)=a\}\} =$
$\{t(X)\ t \in r \text{ y } t(A)= a\} = \sigma_{A=a}\pi_x(r))$.

1.1.4 El Operador Junta

Junta es un operador binario que combina dos relaciones, según el siguiente ejemplo. Suponga que nuestra aerolínea contiene una lista de los tipos de aviones que pueden usarse en cada vuelo y una lista de los tipos de aviones que cada piloto puede volar. Estas listas son almacenadas como las relaciones *utilizable*(VUELO EQUIPO) y *certificado*(PILOTO EQUIPO). La Tabla 1.4 muestra los mismos ejemplos de esas relaciones.

Tabla 1.4 Relaciones *utilizable*(VUELO EQUIPO) y *certificado*(PILOTO EQUIPO).

utilizable(VUELO	EQUIPO)
83	727
83	747
84	727
84	747
109	707

certificado(PILOTO	EQUIPO)
Simmons	707
Simmons	727
Barth	747
Hill	727
Hill	747

Si buscamos la lista que muestre los pilotos utilizados en cada vuelo, se crea una relación *opciones* sobre el esquema {VUELO, EQUIPO, PILOTO} de las relaciones *utilizable* y *certificado* combinando renglones con el mismo valor para EQUIPO. Opciones son mostradas en la Tabla 1.5.

Tabla 1.5 La relación opciones sobre el esquema {VUELO, EQUIPO, PILOTO}.

opciones(VUELO	EQUIPO	PILOTO)
83	727	Simmons
83	727	Hill
83	747	Barth
83	747	Hill
84	727	Simmons
84	727	Hill
84	747	Barth
84	747	Hill
109	707	Simmons

Una vez que las relaciones son combinadas, nosotros podemos calcular $\pi_{\{VUELO, PILOTO\}}$ (*opciones*), como mostramos en la Tabla 1.6.

En general, la junta combina dos relaciones con todos sus atributos comunes. Inicia con las relaciones r(R) y s(S), con RS=T, la junta de r y s, escrita $r \bowtie s$, es la relación q(T) de todas las tuplas t sobre T tales que hay tuplas $t_r \in r$ y $t_s \in s$ con $t_r = t$(R) y $t_s = t$(S). Dado que R$\cap$S es un subconjunto de ambos R y S, como una consecuencia de la definición t_r(R$\cap$S) = t_s(R$\cap$S). Así, cada tupla en q es una combinación de una tupla de r y una tupla de s con igual (R$\cap$S)-valores.

Tabla 1.6 Cálculo de $\Pi_{\{\text{VUELO, PILOTO}\}}$ (*opciones*).

$\pi_{\{\text{VUELO, PILOTO}\}}$ (*opciones*) = (VUELO PILOTO)

VUELO	PILOTO
83	Simmons
83	Hill
83	Barth
84	Simmons
84	Hill
84	Barth
109	Simmons

1.1.5 El Operador Divide

El operador divide tiene una definición más compleja, pero con algunas aplicaciones en situaciones naturales, funciona generalmente para encontrar datos de una cierta clase.

Definición 1.2. Sea r(R) y s(S) relaciones con S$\subseteq$R. Sea R′=R−S. Entonces r dividido por s, escrito $r \div s$, es la relación

$$r'(R') = \{t \mid \text{para cada tupla } t_s \in s \text{ hay una tupla } t_r \in r \text{ con } t_r(R')=t \text{ y } t_r(S)=t_s\}.$$

La relación r' es el cociente de r dividido por s. Otra forma para establecer la definición es que $r \div s$ es el subconjunto máximal r' de $\pi_{R'}(r)$ tal que $r' \bowtie s$ está contenido en r. La junta en este caso es un producto cartesiano. El siguiente ejemplo aclara la definición.

Ejemplo 1.5. La Tabla 1.7 es otra instancia de la relación *certificado*(PILOTO EQUIPO) dada en la Tabla 1.4. Suponga que deseamos encontrar los pilotos que pueden volar todos los tipos de aeronaves en algún conjunto. Sea q(EQUIPO) y s(EQUIPO) como sigue:

q(EQUIPO)
707
727
747

s(EQUIPO)
707

Tabla 1.7 Una instancia de la relación *certificado*(PILOTO EQUIPO).

Certificado	(PILOTO	EQUIPO)
	Desmond	707
	Desmond	727
	Desmond	747
	Doyle	707
	Doyle	727
	Davis	707
	Davis	727
	Davis	747
	Davis	1011
	Dow	727

La división puede entonces ser usada para reunir información sobre que pilotos pueden volar los tipos de aviones en q, o para encontrar que pilotos pueden volar los aviones en s.

certificado ÷ q = q'(PILOTO)

Desmond
Davis

certificado ÷ s = s'(PILOTO)

Desmond
Doyle
Davis

1.2 El Cálculo Relacional de Tuplas

Los sistemas de consulta no procedurales basados en el cálculo relacional, se consideran de alto nivel, porque expresan el resultado deseado y no la manera en que debe obtenerse, dejando que el trabajo de esta determinación caiga fundamentalmente en el procesador del lenguaje de consulta del sistema de la base de datos. El cálculo relacional de tuplas y el sublenguaje llamado ***Alpha***, propuestos por Codd [Codd 1972a], consisten en el cálculo de predicados de la lógica de primer orden. Su notación matemática abstracta ha servido de lenguaje anfitrión para el desarrollo de un sin número de lenguajes como: QUEL [Stonebraker 1975], SQL[Chamberlin 1976] y QBE [Zloof 1977], los cuales incorporan ciertos elementos de ese cálculo.

La unidad de manipulación del cálculo relacional a diferencia del álgebra relacional que hace uso de la relación, es la variable tupla. Precisamente la noción fundamental en los lenguajes basados en el cálculo relacional, es el uso de la variable tupla. Una variable tupla es una variable que ***varía sobre*** alguna relación ***r***, es decir, una variable cuyos únicos valores permitidos son tuplas de esa relación. En otras palabras, si la variable tupla x varía sobre la relación ***r***, entonces en cualquier instante dado, x representa alguna tupla individual de ***r***, originalmente se le llamo ***término rango*** $P_j x$ y se interpreta como, la variable tupla x que tiene a la relación r_j como su rango, P es un predicado monádico y su función es establecer una correspondencia uno a uno con las relaciones de la base de datos dada (tantos como sean necesarios).

Las expresiones del cálculo relacional de tuplas tienen la forma

$$\{ x(R) \mid f(x) \}$$

donde f es un predicado Booleano sobre la variable tupla x. La expresión denota la relación r(R) que consiste de todas las tuplas t(R) donde $f(t)$ es verdadera. Antes de dar su definición formal del conjunto de fórmulas legales, daremos algunos ejemplos informales.

Ejemplo 1.6 La consulta: *Obtenga las tuplas de todos los vuelos que llegan a Boston"* Se expresa en el cálculo relacional como:

$$\{vuelos(t) \mid x \in r \wedge x(DESTINO) = \text{"Boston"} \}$$

Aquí la variable tupla es x, que varía sobre la relación ***r***. La consulta puede leerse así: *para cada valor posible de la variable tupla x, recupere la tupla **t**, si y sólo si el componente destino tiene el valor "Boston".*

Ejemplo 1.7 La pregunta "Que vuelos puede hacer Simmons?" puede expresarse como:

$\{x(t) \mid x \in r \wedge$ hay un $y \in s$
donde x(EQUIPO) = y(EQUIPO) $\wedge$ y(PILOTO)="Simmons"$\}$

donde r y s se sustituyen por las relaciones *utilizable* y *certificado* de la Tabla 1.4.

En este ejemplo la variable tupla es x, que varía sobre la relación ***r***. La consulta puede leerse así: para cada valor de la variable tupla x, recupere una tupla y que varia sobre la relación s, si y sólo si y(piloto)="Simmons" y los valores de x(EQUIPO), y(EQUIPO) son iguales.

1.2.1 Fórmulas del Cálculo Relacional de Tuplas

El conjunto de fórmulas legales del cálculo de tuplas se define relativo a

1. Un conjunto universal de atributos U, con un dominio *dom*(A), por cada atributo en U;
2. Un conjunto de comparadores binarios θ, sobre dominios; y
3. Un conjunto d de nombres de relación $\{r_1, r_2, ..., r_p\}$ sobre esquemas $R_1, R_2, ..., R_p$ de todos los subconjuntos de U.

Los bloques de construcción básicos de las fórmulas son átomos, de cual hay tres clases:

C1. Para cualquier nombre de relación r en d, y para cualquier variable tupla x, $r(x)$ es un átomo; $r(x)$ se liga para $x \in r$.

C2. Para cualquier variable tupla x,y (no necesariamente diferentes), cualquier comparador θ y cualquier atributo A y B en U estos son *θ-comparables*, $x(A)\,\theta\,x(B)$ son átomos.

C3. Para cualquier variable tupla x, cualquier comparador θ y cualquier atributo A y B en U que son *θ-comparables*, si c es una constante en *dom(A)*, entonces $c\,\theta\,x(B)$ es un átomo; si c es una constante en *dom(B)*, entonces $x(A)\,\theta\,c$ es un átomo.

En el artículo "Relational Completness of Data Base Sublenguages" presentado por E.F.Codd [Codd 1972a] se determina que los términos del cálculo relacional son únicamente de dos tipos: *términos rango*, y *términos join*. Un ***término rango*** es una variable tupla x que se restringen a variar sobre alguna relación r_i. Un ***término join*** son los comparadores de tipo $x(A)\,\theta\,x(B)$, y los del tipo $c\,\theta\,x(B)$ y $x(A)\,\theta\,c$ definidos en c2, y c3 respectivamente. La expresión x(A) que representa al componente A, donde A es un atributo de la relación sobre la cual varía x, se le denomina ***tupla indexada***.

Ahora usaremos los conectivos $\neg$(negación), $\wedge$(conjunción), $\vee$(disyunción), $\exists$(existe), y $\forall$(para todo) para construir a partir de átomos fórmulas recursivamente, de acuerdo a las seis reglas siguientes: Las fórmulas son similares a las del cálculo de predicados de primer orden usando r_1, r_2,..., r_p como símbolos de relaciones unarias. Se definen recursivamente como:

F1. Cualquier átomo (término join, término rango) es una fórmula.

F2. Si f es una fórmula, entonces $\neg f$ es una fórmula; $\neg f$ es verdadera cuando f es falsa.

F3. Si f y g son fórmulas entonces $f \wedge g$ y $f \vee g$ son fórmulas; $f \wedge g$ es verdadera cuando f y g son verdaderas, $f \vee g$ es verdadera cuando f es verdadera o g es verdadera.

F4. Si x es una variable tupla, f es una fórmula envolviendo x, y R es un subconjunto de U, entonces $\exists x(R) f$ es una fórmula. Esta fórmula es verdadera si hay alguna tupla t sobre R que hace que f sea verdadera cuando se sustituye x en R.

F5. Si x es una variable tupla, f es una fórmula que envuelve x, y R es un subconjunto de U, entonces $\forall x(R) f$ es una fórmula. Esta fórmula es verdadera si para cada tupla t sobre R, f es verdadera cuando t es sustituida por x.

F6. Si f es una fórmula, entonces (f) es una fórmula

Los paréntesis son usados para alcanzar la precedencia de los conectivos. Nosotros asumimos que $\exists$ y $\forall$ son de alta e igual precedencia, seguido por $\neg$, $\wedge$, y $\vee$ en precedencia decreciente. Ahora definamos el concepto de cuando una ocurrencia de x es libre o ligada en una fórmula.

1.2.2 Variables Libres y Ligadas

La idea de ocurrencias libres o ligadas de variables tupla es análoga a las variables globales y locales de un lenguaje de programación con la declaración de variables en procedimientos anidados Figura 1.3.

```
  proc MAIN;
1 decl X, Y, Z;
    [cuerpo del Main]
...
      proc SUB1;
2     decl X, W;
      [cuerpo de SUB1]
        ...
        proc SUB12;
3           decl Z;
              [cuerpo de SUB12]
        end SUB12;
      end SUB1;
  end MAIN.
```

Figura 1.3 Analogía de ocurrencias de variables libres (globales) y ligadas (locales).

Cualquier mención de X, Y, Z en el cuerpo del MAIN se refiere a las variables declaradas en 1. Cualquier mención de Y o Z en el cuerpo de SUB1 se refiere a la declaración en 1, mientras que cualquier mención de X o W se refieren a la declaración en 2. Y Z son globales originadas desde el MAIN y mantienen la misma localidad de almacenamiento. X y W son locales en SUB1 y globales en SUB12 y están ocultas fuera de SUB1, pero pueden ser vistas por procedimientos internos de SUB1. En el cuerpo de SUB12, X, Y, W son globales, y solo Z es local. Nótese que si cambiamos en el procedimiento y en el cuerpo de SUB12, cada ocurrencia de Z en la declaración 3 por otra nueva variable no cambia el significado del programa, con tal de que la nueva variable no sea global a SUB12. Pero si cambiamos cada ocurrencia de W en SUB12 se podría alterar substancialmente el significado del programa, dado que esas ocurrencias de W son globales a SUB12. Note también que esas *ocurrencias* de variable son globales o locales. Una ocurrencia de Z en el cuerpo de SUB12 es local, mientras que una ocurrencia en SUB1 es global.

En una fórmula, las ocurrencias de variables libres o ligadas corresponden a ocurrencias de variables globales y locales de un programa. Los conectivos $\exists$ y $\forall$, llamados *cuantificadores* corresponden a declaraciones; que unen o ligan ocurrencias de variables en su alcance. Los cuantificadores sirven también para escribir variables en nuestras fórmulas, como cuando se hacen las declaraciones en los programas.

Cada ocurrencia de una variable tupla dentro de una fórmula es libre o ligada. Una *ocurrencia* de una variable tupla, es una aparición del nombre de la variable dentro de la fila de símbolos que constituye la fórmula bajo consideración. Una variable tupla *ocurre* dentro de una fórmula f en el contexto de una expresión de la forma x(A) (donde x es una variable tupla y A es un atributo de la relación asociada), o como la variable que sigue a uno de los símbolos de cualificación para todo ($\forall$) y existe ($\exists$). Las reglas que gobiernan las ocurrencias libres o ligadas son:

1. Dentro de una condición, todas las ocurrencias de las variables tupla son libres

2. Las ocurrencias de las variables tupla en la fórmula $(\neg f_1)$ son libres/ligadas según sean libres/ligadas en f_1. Las ocurrencias de las variables tupla en las fórmulas $(f_1 \wedge f_2)$, $(f_1 \vee f_2)$ son libres/ligadas según sean libres/ligadas en f_1 o f_2 (cualquiera de las dos en donde aparezcan).

3. Las ocurrencias de x que sean libres en f son ligadas en las fórmulas $\exists x(f)$, $\forall x(f)$. Otras ocurrencias de variables tupla en f son libres/ligadas en estas fórmulas según sean libres/ligadas en f.

Ejemplo 1.8 La consulta "*Que vuelos salen antes de las 3:00p*" se escribe como

$$f = \forall x(R_1)(\ vuelos(x) \vee (x(\text{SALIDA}) \leq 3{:}00\text{p}).$$

donde *vuelos* = R_1 = (NUMERO#, ORIGEN, DESTINO, SALIDA, LLEGADA)

Todas las ocurrencias de x están ligadas; ellas están en el alcance de $x(R_1)$. Esta fórmula es verdadera si para cada tupla t en *vuelos*, t(SALIDA) ≤ 3:00p.

Ejemplo 1.9 Sea f la fórmula

$$\forall x(R_1) \in vuelos(\exists y(R_1) \in vuelos\ (x(\text{ORIGEN}) = y(\text{ORIGEN}) \wedge x(\text{NUMERO\#}) = z(\text{NUMERO\#}).$$

Todas las ocurrencias de x, y están ligadas. Cada una de las x esta ligada por $\forall x(R_1)$; cada y esta ligado por $\forall y(R_1)$. La única ocurrencia de z es libre, su esquema esta indefinido; y menciona a la variable NUMERO#.

Ejemplo 1.10 Sea f la fórmula

$$\begin{aligned}&\exists x(R_1) \in vuelos(x(\text{ORIGEN}) = \text{"JFK"} \wedge\\ &\quad \forall y(R_2) \in utilizable((x(\text{NUMERO\#}) \neq y(\text{VUELO\#}) \vee y(\text{EQUIPO}) \neq \text{"747"}) \vee\\ &\quad \exists x(R_1) \in vuelos\ (x(\text{NUMERO\#}) = y(\text{VUELO\#}) \wedge\\ &\qquad\qquad x(\text{ORIGEN}) = z(\text{ORIGEN})))).\end{aligned}$$

donde *vuelos* = R_1 = (NUMERO#,ORIGEN, DESTINO,SALIDA, LLEGADA)
utilizable = R_2 = (VUELO, EQUIPO)

Sean x_1, x_2, x_3, x_4, x_5, x_6 las seis ocurrencias de x en el orden que ellas ocurren en f. Todas las ocurrencias de x están ligadas, sin embargo, x_1, x_2, y x_3 están ligadas al primer $\exists x(R_1)$, mientras que x_4, x_5, y x_6 están ligadas al segundo $\exists x(R_1)$. Todas las ocurrencias de y están ligadas, pero la única ocurrencia de z es libre. Note que la subfórmula.

$$\begin{aligned}&\exists x(R_1) \in vuelos\ (x(\text{NUMERO\#}) = y(\text{VUELO\#}) \wedge\\ &\qquad\qquad x(\text{ORIGEN}) = z(\text{ORIGEN})))).\end{aligned}$$

Puede ser cambiada por

$$\exists w(R_1) \in vuelos\ (y(\text{NUMERO\#}) = w(\text{VUELO\#}) \wedge w(\text{ORIGEN}) = z(\text{ORIGEN})))).$$

que hace a *f* más fácil de entender, donde NUMERO# = VUELO

Finalmente, el cálculo de tuplas lo podemos denotar como una séxtupla (U, *D*, *dom*, **R**, *d*, θ, donde U es el universo de atributos, *D* es el conjunto de dominios, *dom* es una transformación "mapping" de U a *D*, **R** es un conjunto de esquemas de relación sobre U, *d* es una base de datos sobre el esquema en **R**, y θ es un conjunto de comparadores que incluyen al menos igualdad y desigualdad para cada dominio en *D*. Una expresión del cálculo de tuplas sobre la séxtupla tiene la forma

$$\{ x(R) \mid f(x) \}$$

donde: 1. *f* es una fórmula legal relativa a U, *D*, *dom*, y θ.
2. *x* es la única variable tupla que ocurre libre en *f*.
3. *R* es un subconjunto de U.
4. Si el esquema de *x* es definido, éste es igual a *R*, en otro caso es $R \supseteq$ subconjunto de atributos de *x*.

1.3 El Cálculo Relacional de Dominios

El cálculo relacional de dominios es completamente similar al cálculo relacional de tuplas, excepto que las variables representan dominios con valores simples más que tuplas completas. Para mantener la notación correcta, es necesario asumir un orden fijo de atributos en una relación. Para esto daremos únicamente algunos ejemplos y enseguida mencionaremos los conceptos para elaborar sus fórmulas.

Ejemplo 1.11 La expresión

$$\{(x_1, x_2,) \mid vuelos(x_1\ x_2\ x_3\ x_4\ x_5) \wedge x_3 = \text{"JFK"}\}$$

representa los valores NUMERO# y ORIGEN con destino "JFK". La expresión es igual a:

$$\{(x_1, x_2,) \mid vuelos\ (x_1\ x_2\ \text{"JFK"}\ x_4\ x_5)\ \}$$

Ejemplo 1.12 La siguiente expresión representa los vuelos y el origen que realiza el tipo de avión 727.

$$\{(x_1\ x_2) \mid \exists x_1\ \textit{vuelos}(x_1\ x_2\ x_3\ x_4\ x_5) \wedge \textit{utilizable}(x_1\ 727)\}$$

Un cálculo de dominios se denota de la misma forma como el cálculo de tuplas, llamémosle como una séxtupla (U, *D*, *dom*, **R**, *d*, θ). Las fórmulas del *cálculo de dominios* se construyen de dominios de variables usando, comparadores, y los conectivos ¬, ∧, ∨, ∃, y ∀. Los bloques básicos de construcción son átomos:

- Si **r** es una relación en la base de datos d con esquema A_1, A_2, ...,A_n, entonces $\mathbf{r}(a_1, a_2, \ldots, a_n)$ es un átomo, donde cada a_i es un dominio variable o un dominio constante del $\text{dom}(A_i)$.
  Si *x*, *y* son variables, θ es un comparador y c es una constante apropiada, entonces $x\theta y$, $x\theta c$ son átomos.
- Las constantes booleanas verdadero y falso son átomos.

Los átomos son combinados recursivamente en fórmulas:

F1. Cualquier átomo es una fórmula.
F2. Si *f* es una fórmula, entonces ¬*f* es una fórmula.
F3. Si *f* y *g* son fórmulas, también lo son $f \wedge g, f \vee g$.
F4. Si *f* es una fórmula, también lo son $\exists x(A)f$, donde A es un atributo en U (conjunto universal de atributos) y *x* es un dominio variable.
F5. Si *f* es una fórmula, así lo es $\forall x(A)f$, donde A es un atributo en U (Conjunto Universal de Atributos) y *x* es un dominio variable. La precedencia de conectivos es igual como en las fórmulas del cálculo de tuplas.

Las reglas de ocurrencias de variables libres o ligadas son análogas a las fórmulas del cálculo de tuplas. El tipo de una variable *x* en una fórmula *f*, es tanto un dominio en *D* o es indefinido. Como en el cálculo de tuplas, la legalidad de las fórmulas requiere consistencia para el tipo de una variable y que una variable cuantificada ocurra libre en la fórmula cuantificada.

1.4 Las Expresiones Alpha

Si las expresiones del cálculo relacional fueran usadas como un lenguaje de manipulación de datos para extraer información de la base de datos, estas no podrían tener la capacidad de realizar proyecciones en relaciones, es decir, sus expresiones sólo pueden seleccionar tuplas con las operaciones descritas en la sección 1.2, pero no pueden hacer proyección sobre los atributos de tuplas en las relaciones. El lenguaje ***Alpha*** fue el primer sublenguaje en notación matemática presentado como un lenguaje de consulta formal para la manipulación de bases de datos relacionales, que forma sus expresiones con la operación de proyección de atributos y las expresiones del cálculo relacional para realizar operaciones de comparación en las tuplas de una relación. Estas expresiones, se enriquecen todavía más

porque presentan una equivalencia de sus expresiones formadas del calculo relacional con expresiones del álgebra relacional vía una transformación, todo esto es demostrado en el teorema de reducción [Codd 1972a], el cual descompone una expresión Alpha en expresiones algebraicas simples. En esta sección describiremos la formulación de las expresiones Alpha y algunos ejemplos, finalmente daremos en la sección 1.5 una descripción del algoritmo de reducción de Codd que consiste de 7 pasos. El propósito es tener complementado la medida generalizada para la elección de un lenguaje que contenga básicamente la operación de proyección y las operaciones de selección de tuplas de una relación como lo considera el lenguaje Alpha. Y tener presente que cualquier lenguaje de base de datos basados en el cálculo relacional puede tener un equivalente en el álgebra relacional.

Definición 1.3 Sea Γ una fórmula determinada por el cálculo relacional de tuplas. Entonces se define una expresión alpha Z como:

$$Z = \{ (t_1, t_2, ..., t_k) : \Gamma \}$$

donde: *i*). $t_1, t_2, ..., t_k$ son variables tupla o componentes de variables tupla t(A);
ii). el conjunto de variable tupla que ocurren en $t_1, t_2, ..., t_k$ es precisamente el conjunto de variables libres en Γ.

La lista $(t_1, t_2, ..., t_k)$ es llamada la *lista fuente* y Γ es la *expresión de calificación.*

Supongamos que $\rho_1,\rho_2,...,\rho_n$ son las diferentes variables tupla en el orden de su primera ocurrencia en la lista fuente de una expresión ***alpha* Z**:

$$Z = \{ (t_1, t_2, ..., t_k): \Gamma \}$$

y las relaciones $r_1, r_2,..., r_p$ (no necesariamente diferentes) son los rangos de $\rho_1,\rho_2,...,\rho_n$ respectivamente. Entonces, el valor de **Z** denota una cierta proyección del subconjunto del producto cartesiano $R_1 \otimes R_2 \otimes ... \otimes R_n$ de aquellas tuplas donde Γ se evalúa como verdadera, o si Γ se omite se hace una proyección del producto cartesiano completo. La proyección en cuestión, se toma sobre los componentes (atributos indicados) por las entradas en la *lista fuente* $(t_1, t_2, ..., t_k)$.

Para dar algunos ejemplos de consultas en la forma de simple expresiones alpha, ampliaremos a nuestra base de datos dos tablas más que describen partes de avión que suministran los proveedores.

r = *proveedores*(PROVEEDOR#, NOMBRE, CIUDAD)
s = *surten* (PROVEEDOR#, PARTE#)

y supondremos que *t* y *u* son las variables tupla sobre las relaciones **r** y **s** respectivamente.

Ejemplo 1.13 *Encuentre el número de proveedor de los proveedores que suministran la parte 15.*

$$u(\text{PROVEEDOR\#}) : u\mathbf{s} \wedge u(\text{PARTE\#}) = 15$$

Aquí la variable tupla es *u*, que varía sobre la relación **s-***surten*. La consulta puede leerse así: *para cada valor posible de la variable u, recupere el valor del componente* u(PROVEEDOR#) , si y sólo si, el componente u(PARTE#) tiene el valor 15.

Ejemplo 1.14 *Encuentre el nombre y la ciudad de los proveedores que suministran la parte 15.* En sintaxis ***Alpha*** se expresa:

$$(t(\text{NOMBRE}), t(\text{CIUDAD})) : t\mathbf{r} \wedge \exists u\, \mathbf{s}\, (\, u(\text{PARTE\#}) = 15 \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$$

Esto se lee así: Para cada valor posible de la variable tupla ***t*** en la relación *r*, recupere el valor del componente t(NOMBRE) y u(CIUDAD), si y sólo si, existe una variable tupla ***u*** en la relación *r*, cuyo componente u(PARTE#) tiene el valor *15* y los valores de u(PROVEEDOR#) y t(PROVEEDOR#) son iguales.

Ejemplo 1.15 *Encuentre la ciudad de los proveedores y las partes que ellos suministran (omitiendo los proveedores que no suministran partes).*

$$(t(\text{NOMBRE}), u(\text{PARTE\#})) : t\mathbf{r} \wedge u\mathbf{s} \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$$

Las expresiones ***Alpha*** simples, se pueden **generalizar** sin perder su rango de propiedades. Una expresión ***Alpha*** se define recursivamente como sigue:

1. Cada expresión ***Alpha*** simple, es una expresión ***Alpha***
2. Si $t: w_1$ y $t: w_2$ son expresiones ***Alpha***, también lo son:
   $t: (w_1 \vee w_2)$, $t: (w_1 \wedge \neg w_2)$, y $t: (w_1 \wedge w_2)$
3. No hay otras expresiones que sean expresiones ***Alpha***.

1.5 El Algoritmo de Reducción

En esta sección se describimos el proceso general del algoritmo de transformación de expresiones Alpha en expresiones equivalentes del álgebra relacional, junto con su formalización que consiste de 7 pasos presentado en el artículo "Relational Completness of Data Base Sublenguages" por E.F.Codd en 1972 [Codd 1972a].

El algoritmo de reducción para expresiones ***Alpha*** simple, puede ser mejor entendido suponiendo por el momento, que en vez de producir enteramente varias expresiones algebraicas, esas expresiones son evaluadas cuando son producidas. El efecto de ésta evaluación podría ser rigurosamente como sigue:

1. Los rangos de las variables tuplas citadas se generan por recuperación de ciertas relaciones de la base de datos con base en los conjuntos unión, intersección, y diferencia.

2. Se forma un producto cartesiano de esos rangos, del cual se extrae la relación final.

3. Las tuplas que no satisfacen la combinación de condiciones se eliminan del producto.

4. El producto restante se restringe bajo proyección y división para satisfacer la cuantificación en la expresión ***Alpha***.

5. Finalmente, se realiza una proyección especificada por la lista fuente y entonces se obtiene la relación requerida.

La expresión algebraica es obtenida a partir de la aplicación de las siguientes reglas:

1. El rango de las variables tuplas en una relación se traduce en forma general como la relación misma es decir: t_r se sustituye por r.
2. $\vee$ se sustituye por $\cup$;
3. $\neg\wedge$ se sustituye por $-$;
4. $\wedge$ se sustituye por $\cap$.

La descripción formal del algoritmo de reducción de expresiones ***Alpha*** se muestra en los pasos siguientes.

Sea la expresión ***Alpha***

$$z = t : w ,$$

donde 1. $t = (t_1, t_2, ..., t_k)$ es la *lista fuente*;
2. $w = U_1 \wedge U_2 \wedge .... \wedge U_p \wedge V$ es la expresión de calificación (con rangos claramente definidos en todas sus variables);
3. Hay $q \geq 0$ variables ligadas en V;
4. Toda las p de las variables libres en w ocurren en t (así, $k \geq p$).

Paso 1. Por conveniencia, se aplican cuatro transformaciones a z que dan una nueva expresión ***Alpha*** con la misma notación:

a). Convertir V a ***forma normal prenex*** (i.e. expresiones conjuntivas, si es que no estuviera en esta forma);
b). Mantener los cuantificadores principales sin cambio, convertir la subformula restante (también llamada matriz) a la forma normal disyuntiva;
c). Donde se encuentre un *término join* usando una relación θ que esta inmediatamente precedida por $\neg$, eliminar el símbolo $\neg$ y reemplazar θ por su complemento (los complementos de $=$, $\neq$, $<$, $\leq$, $>$, y $\geq$ son $\neq$, $=$, $\geq$, $>$, $\leq$, y $<$, respectivamente);

d). Aplicar sistemáticamente un cambio alfabético a las variables en la expresión ***Alpha*** resultante de 3. Así que las variables vienen a ser $t_1, t_2, ..., t_{p+q}$ en el orden de su primera ocurrencia en la calificación (note que la primera ocurrencia de una variable ligada es con su cuantificador).

Diremos que la expresión resultante de esas cuatro transformaciones es

$$z' = t : U'_1 \wedge U'_2 \wedge .... \wedge U'_p \wedge V'.$$

Las variables ligadas en V' son t_j donde j=p+1, p+2, ..., p+q. Suponga que el cuantificador ($\exists$ o $\forall$) asociado con t_j en V' es Q_j, y supongamos que el rango sobre t_j en V' es U'_j.

Paso 2. Para j= 1, 2, ..., p+q formar una ecuación para el rango $\mathbf{s}_j$ de variables tupla t_j La expresión algebraica de lado derecho de esta ecuación es obtenida de U'_j (el rango de fórmulas sobre t_j), por aplicación de las siguientes reglas en U'_j.

1. $P_i\, t_j \rightarrow R_i$
2. $\vee \rightarrow \cup$
3. $\neg\wedge \rightarrow -$
4. $\wedge \rightarrow \cap$

Por ejemplo, si $U'_j = P_3\, t_j \wedge \neg P_2\, t_j$, entonces la ecuación para S_j es $S_j = R_3 - R_2$, donde R_3 y R_2 son unión compatibles.

Paso 3. Asociar todas las relaciones R_i (i=1, 2, ..., N) que son unión-compatibles con R_i. Llamemos a esta unión como $\mathbf{U}(R_i)$, para cualquier subconjunto S_j de $\mathbf{U}(R_i)$, se define a $\mathbf{U}(S_j) = \mathbf{U}(R_i)$. Forman la ecuación

$$S = L_1 \otimes L_2 \otimes ... \otimes L_{p+q},$$

donde $L_j = S_j$ si t_j es libre o existencialmente ligada,
$\mathbf{U}(s_j)$ en otro caso.

El tratamiento especial acordado en el cuantificador universal permite manejar correctamente el caso en el cual $s_j=\phi$ y t_j es universalmente cuantificado. (En este sentido, un sistema de base de datos debe advertir a un usuario si encuentra un rango vacío S_j o un universo vacío $\mathbf{U}(s_i)$ en la interpretación de la consulta "*query*", y de esta manera poder informarle de la condición pertinente.

Sea n_j el grado de la relación L_j (j =1, 2, ..., p+q). Sea

$$\mu_j = \sum_{i=1}^{j-1} n_i.$$

Entonces, el dominio con la posición j en la relación L_j tiene posición $J+\mu_j$ en el producto cartesiano S.

Paso 4. Eliminar los cuantificadores rango acoplados (cualesquiera) de V´para obtener el cuantificador libre fórmula V". Si V" es nulo, forman la ecuación

$$C_{p+q} = S.$$

en otro caso, formar una ecuación para C_{p+q} , el lado derecho del cual es una expresión algebraica obtenida de V" por aplicación de las siguientes reglas:

a). $\vee \rightarrow \cup$

b). $\wedge \rightarrow \cap$

c). $(t_j[J]\theta\ t_k[K] \rightarrow S[\ (J+\mu_j)\ \theta\ (K+\mu_k)\ ]$

d). $(t_j[j]\theta\ a) \rightarrow S[\ (J+\mu_j)\ \theta\ 1]\ \{a\}$

donde *a* es una constante individual, y θ es uno de los siguientes símbolos relacionales $=, \neq, <, \leq, >, \geq$.. Note que el símbolo $\neg$ hace que no ocurra a todos en V" debido a la tercera transformación en el paso 1.

Paso 5. Formar *q* ecuaciones para C_j-1 (j=p+q, p+q-1, ..., p+1) el cual refleja los efectos de los cuantificadores de V´ iniciando con el mas interno Q_{p+q} y procediendo al más externo Q_{p+1}. La ecuación para C_{j-1} es

$$C_{j-1} = \begin{cases} C_j[1,2,...,\mu_j] & \text{si } Q_j = \exists \\ C_j[E_{j+1} \div D_j]S_j & \text{si } Q_j = \forall, \text{ donde } E_j = (\mu_{j+1}, \mu_{j+2}, ..., \mu_{j+nj}) \text{ y } D_j = (1,2,...,n_j). \end{cases}$$

Paso 6. Formar una ecuación (exacta) para C el cual toma en cuenta la proyección especificada en la lista fuente ***t*** original:

$$C = Cp[F_1 \circ F_2 \circ ... \circ F_k],$$

donde, para h=1,2,...,k

$$F_h = \begin{cases} (1+\mu_j, 2+\mu_j, ..., \mu_{j+1}) & \text{si } C_h = t_j \\ (J+\mu_j) & \text{si } t_h = t_j[J] \text{ y } \mu_j \text{ es definido como en el paso 2.} \end{cases}$$

Paso 7. Por medio de una simple sustitución, formar una ecuación que defina a C directamente en términos de R_1, R_2, ...,R_3 (y sus respectivos grados n_1, n_2, ..., n_N) por eliminación $L_1,L_2,...,L_{p+q},S_1,S_2$, ..., S_{p+q},S, C_{p+q}, ...,C_p de las ecuaciones generadas arriba. Nosotros podemos asumir que las ecuaciones para $U(R_i)$ (i=1,2,...,N) son dadas, y ellas representan una propiedad de la base de datos más que de los ***queries.***

Ejemplo 1.16 Un ejemplo de transformación usando el algoritmo anterior aplicado a una expresión Alpha, se determinaría de la siguiente forma.

$$(t(\text{NOMBRE}), t(\text{CIUDAD})) : tr \wedge \exists u\, s\, (\, u(\text{PARTE\#}) = 15 \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$$

se traduce por pasos en una expresión algebraica como sigue

1. Sustituimos los rangos de $t\mathbf{r}$ y us por **r** y **s** respectivamente
   donde:
   r = *proveedores*(PROVEEDOR#, NOMBRE, CIUDAD)
   s = *surten* (PROVEEDOR#, PARTE#)

2. Para $\exists u$s (u(PARTE#) = 15), se sustituye por la expresión algebraica

$$\sigma_{\text{PARTE} = 15}(s)$$

3. $tr \wedge \exists u\, s\, (\, u(\text{PARTE\#}) = 15 \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$, se sustituye por la equi-junta de la expresión algebraica, donde la equi-junta se establece sobre los dominios PROVEEDORES# de ambas relaciones.

$$r \bowtie \sigma_{\text{PARTE} = 15}(s)$$

4. Finalmente a toda la expresión

$$(t(\text{NOMBRE}), t(\text{CIUDAD})) : tr \wedge \exists u\, s\, (\, u(\text{PARTE\#}) = 15 \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$$

se le aplica la proyección y queda la expresión algebraica

$$\pi_{\text{NOMBRE, CIUDAD}}(r \bowtie \sigma_{\text{PARTE} = 15}(s))$$

El proceso hubiera sido más elaborado, si aplicamos el producto cartesiano desde el paso 1 y 2, así como aplicar unión e intersección en los siguientes pasos, tal y como se indica en el algoritmo de reducción.

En este capítulo hemos llegado a establecer los sistemas de consulta formales, en particular el sublenguaje Alpha, que con la operación de proyección junto con las expresiones del cálculo relacional se plantea como un lenguaje para la solución a problemas de consulta, basado en lógica de predicados de primer orden y de base de datos, usado también como medida de selección, con la cual, cualquier lenguaje propuesto para consultar base de datos debe compararse. Hemos establecido las unidades básicas de manipulación en los sistemas formales: el atributo, la tupla, y la relación conteniéndose respectivamente, para finalmente encontrar una transformación de tuplas a relaciones. Con esto se ha encontrado que los sistemas de consulta formales son la base de los lenguajes de consulta que hoy en día se están comercializando y es la base formal de *transformación de un lenguaje de consulta que permite formular las peticiones o requerimientos en un Sistema Manejador de Base de Datos (SMBD).*

Capítulo 2

Sistema de Consulta Tableaux

En este capítulo veremos como representar una transformación de expresiones definida por Selección, Proyección y Junta Natural en matrices especializadas llamadas tableaux [Aho 1979] [Ullman 1982] [Maier 1983]. Veremos también que la metodología utiliza dos tipos de tableaux: tableaux sin marca y tableaux con marca. Las consultas de tableaux sin marca parten de una junta natural de todas las tablas de la base de datos, y las consultas tableaux con marca se restringen a trabajar sobre tablas especificas que se van juntando. También describimos el algoritmo y los teoremas de equivalencia y transformación de expresiones algebraicas a tableaux presentados por [Maier 1983].

Las consultas a las bases de datos tiene una representación tabular muy especial basándose en la "*relación universal de las bases de datos*", que se caracteriza porque el usuario no necesita conocer la organización lógica de la base de datos que se refiere a nombres de tablas y atributos de estas. En vez de ello, la interfaz del usuario, es una gran tabla donde los resultados se construyen con la proyección de ciertas columnas. (*Tableaux* también ha sido considerado como una notación estilizada de un subconjunto del lenguaje "*Query by Example*" [Zloof 1977]). El objetivo de este capítulo es implementar en nuestro trabajo la transformación de consultas provenientes de un cálculo relacional en tableaux usando la metodología planteada por tableaux. Para esto analizaremos las secciones que consisten fundamentalmente en:

- Desarrollo de las consultas por tableaux
- Traducción de expresiones algebraicas a tableaux
- Consulta a tableaux que vienen de expresiones algebraicas
- Consultas tableaux en multirelaciones, en conjuntos y conjuntivas.

2.1 Desarrollo de Consultas por Tableaux

Las expresiones *Tableaux* se basan en: la ***selección*** con la comparación de igualdad, la ***proyección*** y la ***junta natural***. De esta manera describimos a Tableaux como una representación de expresiones *SPJ* **(Selección, Proyección, y Junta)**. Aunque este subconjunto no es relacionalmente completo, es suficiente para expresar muchas consultas comunes. Los teoremas de Tableaux aseguran que cada expresión *SPJ* tiene un *Tableaux* asociado. Una extensión de *Tableaux* puede tratar con dependencias funcionales que se utilizan para implicar equivalencias adicionales entre *Tableaux* [Aho 1979] [Ullman 1982], mecanismo que no tienen los otros sistemas de consulta. El poder encontrar sistemas equivalentes nos conduce a encontrar un *Tableaux* más óptimo que el original, de modo que el query sea extraído de manera más eficiente que el original.

El interés en las expresiones formadas por *Tableaux*, estriba en que pueden ser optimizadas a través de minimizar el número de los operadores junta (⋈), en la correspondiente expresión algebraica, aunque este proceso de optimización puede ser costoso.

Podemos decir que el método tableaux se usa para consultar bases de datos relacionales o relaciones simples. *Las expresiones algebraicas que se pueden representar en Tableaux sólo son válidas cuando se traten de expresiones restringidas.* Una expresión algebraica restringida E, es una expresión construida de relaciones (temporales), y relaciones constantes de tuplas simples que usan los operadores

1. Selección de la forma $\sigma_{A=c}$,
2. Proyección π y ,
3. Junta natural ⋈.

Ejemplo 2.1 La consulta: *Encuentre el número de proveedor de los proveedores que suministran la parte 15 que se localizan en Puebla*, puede ser expresada por la operación algebraica restringida.

$$\pi_{\text{PROVEEDOR\#}}(\sigma_{\text{CIUDAD}=\text{"PUEBLA"}}(\textit{proveedores} \bowtie \textit{surten})).$$

Una consulta por Tableaux consta de ***variables distinguibles*** (valores constantes) y variable ***no-distinguibles*** (blancos o espacios) denominadas en ambos casos símbolos. Las variables distinguibles denotadas con ***a***'es, corresponden a las *variables libres* de las expresiones del cálculo relacional y más precisamente de las expresiones *alpha* de (2,1). Las variables no distinguibles denotadas con ***b***'s son las *variables ligadas* de las expresiones de las expresiones *alpha* de (2.1). Las variables distinguibles en términos de álgebra relacional corresponden a las proyecciones que se representan en una expresión de Alpha, así mismo las variables no distinguibles son todas las demás variables o términos que se cuantifican en la expresión a se ligan a un cuantificador.

$$(t_1, t_2, ..., t_k): \Gamma, \qquad (2.1)$$

Ejemplo 2.1.a La consulta: *Encuentre el número de proveedor de los proveedores que suministran la parte 15 que se localizan en Puebla*, puede ser expresada en Alpha.

$$t(\text{PROVEEDOR}) : tr \wedge \text{t(CIUDAD)="Puebla"} \wedge \exists u \text{ s } (\ u(\text{PARTE\#}) = 15 \wedge u(\text{PROVEEDOR\#}) = t(\text{PROVEEDOR\#}))$$

donde $(t_1, t_2, ..., t_k)$ **es** t(PROVEEDOR)

y Γ tr $\wedge$ t(CIUDAD)="Puebla" $\wedge$ $\exists u$ s (u(PARTE#) = 15 $\wedge$ u(PROVEEDOR#) = t(PROVEEDOR#))

Una consulta por Tableaux para una relación $r(A_1\ A_2\ ...A_k)$ es una arreglo bidimensional (Figura 2.1), que debe satisfacer siete condiciones:

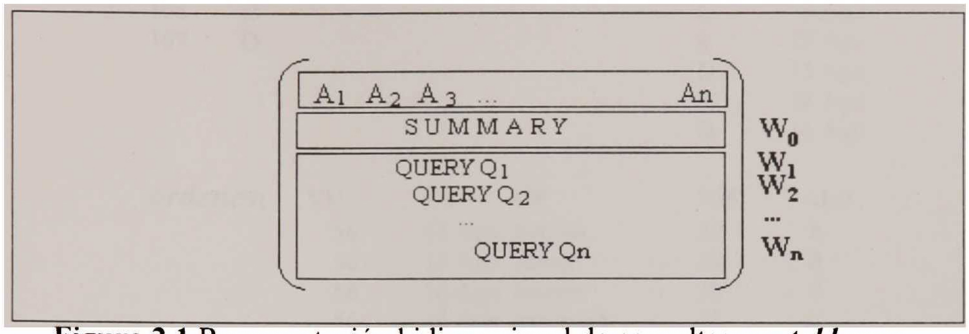


Figura 2.1 Representación bidimensional de consultas por ***tableaux***

1. El primer renglón de un tableaux contiene la lista de los atributos $A_1\ A_2\ ...A_k$ de la base de datos para cada columna.

2. El siguiente renglón, denominado ***summary***, consiste solo de variables distinguibles (o símbolos distinguibles),

3. Los renglones restantes w_1, w_2,... w_n están asociados con los atributos de las relaciones y en ellos, se expresan los predicados o los criterios de selección de la consulta para cada relación.

4. Cada variable debe aparecer en una columna.

5. Si una variable distinguible aparece en una columna de un renglón (diferente al sumario), esta variable debe aparecer también en la columna del sumario.
6. Si una constante c aparece en la columna A_i, entonces c pertenece al $dom(A_i)$.

7. Denominaremos a w_0 como el nombre dado al sumario y w_1, w_2,... w_n los usaremos para denominar a los renglones restantes del tableaux.

Con el objetivo de ir mostrando el uso de las consultas mediante Tableaux nos apoyaremos en una base de datos, con un esquema de relaciones R={*servicios*, *opciones*}, la cual ha sido diseñada para mantener y gestionar los datos de alimentos que se sirven en una aerolínea. El conjunto de atributos son vuelos, alimentos, fecha, opción y número los cuales serán abreviados como VU, AL, FE, OP, NM.

De esta manera los esquemas de relación serán determinados como *servicios*(VU, AL), *opciones*(AL, FE, OP) con datos que consisten según la figura 2.2.

Servicios(**VU** **AL**)

VU	AL
56	B
56	L
57	D
106	L
106	D
107	D

opciones(**AL** **FE** **OP**)

AL	FE	OP
B	15 Ago	huevos
B	15 Ago	cereal
B	16 Ago	huevos
L	15 Ago	sandwich
L	16 Ago	pasta
L	15 Ago	carne
D	15 Ago	frutas
D	16 Ago	ensalada
D	16 Ago	queso

ordenes(

VU	FE	OP	NM	AL)
56	15 Ago	huevos	27	B
56	15 Ago	cereal	23	B
56	16 Ago	huevos	50	B
56	15 Ago	sandwich	50	L
56	15 Ago	carne	50	L
57	15 Ago	frutas	55	D
57	16 Ago	frutas	60	D
106	16 Ago	sandwich	80	L
106	16 Ago	pasta	45	L
106	16 Ago	carne	35	L
106	15 Ago	frutas	40	D
106	15 Ago	ensaladas	40	D
106	16 Ago	queso	40	D
106	16 Ago	pollo	40	D
107	15 Ago	frutas	60	D
107	16 Ago	pollo	60	D

Figura 2.2 Relaciones *servicios*, *opciones*, y *ordenes*

La relación *servicios* indica los alimentos servidos normalmente en cada vuelo; la relación *opciones* dá las diferentes opciones de cada alimento en una fecha dada; y la relación *ordenes* según figura 2.2 indica la cantidad de alimentos de cada opción que se ordenaron en un vuelo y en una fecha determinada.

Supondremos que la porción de la línea punteada de la tabla *ordenes* es lo que ha sido ordenado, y que las tres relaciones dan justamente la junta completa. Sea *alimentos* la junta de las tres relaciones, y cuyo resultado se escribe en la columna de más a la derecha de la relación *ordenes*.

Ejemplo 2.2 Utilizando precisamente la relación *alimentos*, expresemos la consulta siguiente en el tableaux de la Tabla 1.1. "*Para cada alimento que se sirve en el vuelo 107, y para cada una de las opciones disponibles de ese alimento, ¿cuantas de esas opciones hacen que en el vuelo 106 se tenga ordenado ese alimento en cada día?*" La relación

resultante se forma de la proyección de las variables a_i que aparecen en el sumario (FE, OP, y NM).

Tabla 1.1 Consulta por Tableaux para la relación *alimentos*.

	Q(VU	FE	OP	NM	AL)
W_0		a_2	a_3	a_4	
W_1	b_1	b_2	a_3	a_4	b_5
W_2	*106*	a_2	a_3	a_4	b_5
W_3	*107*	b_6	b_7	b_8	b_4

Para evaluar una consulta Q Tableaux en una relación **r**(R), definimos a $\rho(Q)$ como una función de símbolos de Q a valores del dominio, es decir, si α es un símbolo en la columna A de Q, $\rho(\alpha) \in dom(A)$, y si c es una constante, entonces $\rho(c)=c$ esta en el dominio de la columna A. Por ejemplo, la Tabla 1.1 contiene los valores *106* y *107* que están contenidos en el domino de *VU*.

La evaluación de ρ también se mapea sobre conjuntos de dominios y de renglones $w_1, w_2, \ldots w_n$ de Q a tuplas en el *dom*(R), y mapea el sumario de Q al *dom*(S), donde S es el conjunto de columnas de Q cuando el sumario no contiene blancos y la denotamos por

$$\rho(Q) = \{\rho(w_i) \mid w_i \text{ es un renglón en } Q\}.$$

El Q Tableaux es semejante a la expresión algebraica

$$\pi_{FE\ OP\ NM}(\pi_{OP\ AL}(alimentos) \bowtie \pi_{FE\ OP\ NM}(\sigma_{VU=106}(alimentos) \bowtie \pi_{AL}(\sigma_{U=107}(alimentos)))).$$

La expresión algebraica es equivalente a:

$$\pi_{FE\ OP\ NM}(\pi_{OP\ AL}(opciones) \bowtie \pi_{FE\ OP\ NM}(\sigma_{VU=106}(ordenes)) \bowtie \pi_{AL}(\sigma_{VU=107}(servicios))).$$

Esta última expresión, es una partición de ***alimentos*** que es la **junta** de *servicios*, *opciones* y *ordenes* (es decir, son las proyecciones de la instancia común ***alimentos***).

Una expresión equivalente del cálculo de tuplas de la consulta Q Tableaux para una relación ***r***(R) con sumario w_0 y renglones $w_1, w_2, \ldots, w_n$, donde S es un conjunto de atributos en w_0 con elementos no blancos es:

$$E = \{\ x(S) \mid \exists y_1(R) \in r, \exists y_2(R) \in r \ldots \exists y_n(R) \in r \wedge f(x, y_1, y_2, \ldots, y_n)\ \}.$$

donde f es la conjunción de átomos

1. $x(A) = y_i(A)$ donde w_0 y w_i tiene la variable distinguible para la columna A.
2. $y_i(A) = y_j(A)$ donde $w_i(A) = w_j(A)$ y $w_i(A)$ no es una constante.
3. $x(A) = c$ donde $w_0(A) = c$, c es una constante, y

4. $y_i(A) = c$ donde $w_i(A) = c$, c es una constante.

La expresión E es equivalente a una expresión algebraica que involucra a lo más *n-1* juntas. Nótese que las elecciones para y_1, y_2 ,..., y_n en **r** corresponden a $\rho(w_1),\rho(w_2),...,\rho(w_n)$ para alguna evaluación ρ tal que $\rho(Q) \subseteq \mathbf{r}$.

Ejemplo 2.3 Una expresión del cálculo de tuplas equivalente al Tableaux de la Tabla 1.1, donde R_m = (VU, FE, OP, NM, AL) es:

$$\{\ x(\text{FE OP NM}) \mid \exists y_1(R_m)\ \exists y_2(R_m)\ \exists y_3(R_m)$$
$$(\ x(\text{FE}) = y_2(\text{FE}) \wedge x(\text{OP}) = y_1(\text{OP}) \wedge x(\text{OP}) = y_2(\text{OP}) \wedge$$
$$x(\text{NM}) = y_2(\text{NM}) \wedge y_1(\text{AL}) = y_3(\text{AL}) \wedge$$
$$y_2(\text{VU}) = 106 \wedge y_3(\text{VU}) = 107\ \}.$$

Nótese que el átomo $y_1(\text{OP}) = y_2(\text{OP})$ puede ser incluido, pero sería redundante, debido a que se entiende implícitamente que ambos atributos son iguales tanto en su nombre como en el dominio que utilizan, aunque aparezcan en diferentes renglones en la Tabla 1.1.

Definición 2.1 Sea Q una consulta Tableaux con esquema R y sea w un renglón de Q. Si A es un atributo en R, entonces w empata con la columna A si $w(A)$ = c, para alguna constante c, $w(A) = w_0(A)$ para el sumario w_0, o bien $w(A) = w'(A)$ para otro renglón w' en Q. Llamamos a $w(A)$ un símbolo "*match*".

$$match(w) = \{A \mid w \text{ empata en la columna A}\}.$$

Definición 2.2 Sea Q una consulta por Tableaux sobre el esquema $\boldsymbol{R}$. Sea BD una base de datos sobre $\boldsymbol{R}$ con esquema de relaciones $\boldsymbol{R} = \{R_1, R_2, ..., R_p\}$, Q se aplica a la base de datos BD, si para cada renglón w en Q, hay un esquema de relación $R_i \in \boldsymbol{R}$ con $match(w) \in R_i$.

Ejemplo 2.4 Para la consulta Q Tableaux de la Tabla 1.1,

$$match(w_1) = \text{FE OP AL}$$
$$match(w_2) = \text{VU FE OP NM}$$
$$match(w_3) = \text{VU AL}.$$

Q se aplica a la base de datos de la Figura 2.2, y Q no puede aplicarse a la base de datos BD con esquema {FE OP , OP AL , VU FE OP NM , VU AL }, debido a que hay un renglón w en Q, que no tiene esquema en FE,OP o en OP,AL.

Sea Q una consulta Tableaux con sumario w_0 y renglones $w_1,w_2,..,w_n$ que se aplican a la base de datos BD. Si las relaciones de BD son las proyecciones de una instancia **r** común, entonces Q puede ser evaluada con BD como sigue. Para cada renglón w_i en Q, sea $r_i(R_i)$ una relación en BD tal que $match(w_i) \subseteq R_i$, para una evaluación ρ en Q,

damos $\rho(Q) \subseteq \mathbf{d}$ que significa $\rho(w_i(R_i)) \subseteq r_i$. Esto es, ρ mapea una porción de w_i conteniendo $match(w_i)$ a una tupla en r_i. Podemos entonces dar

$$Q(\text{BD}) = \{\rho(w_0) \mid \rho \text{ es una evaluación para } Q \text{ donde } \rho(Q) \subseteq \text{BD}\}.$$

debiendo ser claro que $Q(\text{BD}) = Q(\mathbf{r})$. Para alguna evaluación ρ tal que $\rho(Q) \subseteq \mathbf{r}$, esto también es parecido a $\rho(Q) \subseteq$ BD. Para cualquier evaluación ρ tal que $\rho(Q) \subseteq$ BD, hay una evaluación ρ' tal que $\rho(w_0) = \rho'(w_0)$ y $\rho'(Q) \subseteq \mathbf{r}$. *La evaluación de ρ' siempre existe, dado que si $\rho(w_i(R_i)) = t_i \in \mathbf{r}$, hay una tupla $\mathbf{t} \in \mathbf{r}$ tal que $t(R_i) = t_i$. Podemos elegir ρ' tal que $\rho'(w_i) = t$.*

2.2 Traducción de Expresiones Algebraicas a Tableaux

En esta sección describiremos el algoritmo para traducir una expresión algebraica restringida a una consulta equivalente por Tableaux. Por la naturaleza de los operadores en una expresión algebraica restringida, si cualquier subexpresión es idénticamente la relación vacía, entonces la expresión completa es idénticamente la relación vacía. Usaremos el símbolo $Q\phi$ para denotar una consulta de Tableaux especial que evalúa la relación vacía.

Algoritmo: Construcción de un Tableaux por expresiones Selección-Proyección-Junta.

Entrada: Una expresión algebraica que contiene relaciones variables como argumentos y *selección*, *proyección* y *junta natural* como operadores.

Salida: Un Tableaux, que es tratado como una consulta conjuntiva del cálculo relacional de dominios, que es equivalente a la expresión algebraica de entrada.

Método: Se construye un Tableaux por cada subexpresión de la expresión algebraica dada, que aparece en cada nodo del árbol, descendiendo hasta llegar a las hojas.

Dada una expresión algebraica restringida E para la relación **r**, definimos recursivamente un Tableaux equivalente Q. Sea $(A_1\ A_2 \ldots A_m)$ el esquema de relación de **r**.

Caso 1. La base es una hoja, es decir, una relación variable $\mathbf{r}(A_1, \ldots, A_n)$. Si E es **r**. Entonces Q es:

	A_1	A_2	A_m
w_0	a_1	a_2	a_m
w_1	a_1	$a_2 \ldots$	a_m

En otras palabras la base es un tableaux para cada relación, que se crea con un sumario y un renglón, marcado con el nombre del archivo w_1. Ambos renglones deben tener en sus columnas a las variables distinguibles A_i's y blanco en otro lado.

Caso 2. E es $<c_1:B_1\ c_2:B_2 \ldots\ c_k:B_k>$, entendiéndose como una expresión con c_i-constante en el rango de valores de B_i, donde $B_1\ B_2 \ldots\ B_k \subseteq A_1\ A_2 \ldots\ A_m$ y c_i es una constante en $dom(B_i)$. Q es el Tableaux que consiste sólo del sumario w_0, donde w_0 tiene c_i en la columna B_i, $1 \leq i \leq k$; y blanco en otro caso.

Caso 3. E es $\sigma_{A_i=c}(E')$. Sea Q' el Tableaux para E'. Hay tres posibilidades para Q.

a. Si $Q' = Q\phi$, entonces $Q = Q\phi = \varnothing$.
b. Si w_0' es el sumario para Q' y $w_0'(A) = c'$, entonces si $c = c'$; se tiene que $Q = Q'$. En otro caso, cuando $c \neq c'$, Q es $Q\phi$.
c. Si w_0' es el sumario de Q', y $w_0'(Ai) = a_i$, entonces Q es Q' con cada ocurrencia de a_i reemplazada por c. La selección $\sigma_{Ai=c}$, se expresa en el tableaux Q, cambiando la variable distinguible A_i por c.

Caso 4. E es $\pi_x(E')$. Sea Q' el Tableaux para E'. Si $Q' = Q\phi$, entonces $Q = Q\phi$. En otro caso, sea w_0' el sumario de Q'. El sumario w_0 para Q tiene $w_0(A_i) = w_0'(A_i)$ para $A_i \in X$ y es blanco en otro caso. Los renglones de Q son los renglones de Q', excepto si a_i es la variable distinguible para la columna A_i, y $A \notin X$, entonces a_i es reemplazada por una nueva variable no distinguible. Es decir, la proyección (π) para la subexpresión: $E = \pi_{A1, A2, \ldots, An}(E')$, se representa en un tableaux Q, eliminando del sumario los símbolos distinguibles (a_i), en las columnas que no pertenecen a las A_i's. Los demás símbolos y las restricciones de los renglones marcados pasan a ser variables no distinguibles (b_i).

Caso 5 E es $E' \bowtie E''$. Sean Q' y Q'' los Tableaux para E' y E''. Asumamos que no aparecen variables no distinguibles en E' como en E. Hay tres posibilidades para Q.

a. Si Q' o Q'' es $Q\phi$, entonces $Q = Q\phi$.

b. Si w_0' y w_0'' son los sumarios de Q' y Q'', y $w_0'(A_i) = c'$ y $w_0''(A_i) = c''$ para algún A_i, y las constantes c' y c'' son diferentes, entonces Q es $Q\phi$.

c. En otro caso el sumario w_0 tiene

i. $w_0(A_i) = c$ si tanto $w_0'(A_i) = c$ o $w_o''(A_i) = c$ (si a_i es cambiada a c en el sumario, es cambiada en todas partes).
ii. $w_0(A_i) = a_i$, si i no se aplica y tanto $w_0'(A_i) = a_i$ o $w_o''(A_i) = a_i$, y
iii. $w_0(A_i)$ = blanco en cualquier otro lado y los renglones de Q son los renglones de Q y Q''.

Es decir, si se tiene la subexpresión E'⋈E", sean Q' y Q'' los Tableaux para E' y E". Ambos con símbolos distinguibles en la columna del sumario para el atributo A (tienen símbolos en común). Entonces el tableaux para E'⋈E" tiene un sumario, en el cual una columna tiene un símbolo distinguible *a*, si *a* aparece tanto como un símbolo distinguible en la columna del sumario Q' o Q'', o ambos. En otro caso el sumario tiene un blanco en esa columna. El nuevo tableaux tiene como renglones todos los renglones de Q' y Q'', y tiene como restricciones todas las restricciones de los dos tableaux.

Ejemplo 2.5 Sea E la expresión algebraica restringida

$$\pi_{VU\ FE}(\pi_{FE\ AL}(\pi_{VU\ OP\ AL}(\sigma_{VU=56}(alimentos)) \bowtie \pi_{FE\ OP}(alimentos)) \bowtie alimentos)$$

para la relación *alimentos* esta es la junta de las relaciones de la Figura 2.2. Traducimos E a un Tableaux, combinando los pasos vistos. El Tableaux $\sigma_{VU=56}$ (*alimentos*) aparece en la Figura 2.3. El Tableaux para $\pi_{VU\ OP\ AL}$ ($\sigma_{VU=56}$(*alimentos*)) se muestra en la Figura 2.4. La Figura 2.5 da el Tableaux para $\pi_{VU\ OP\ AL}(\sigma_{VU=56}(alimentos)) \bowtie \pi_{FE\ OP}(alimentos)$. La Figura 2.6 da el Tableaux para $\pi_{FE\ AL}(\pi_{VU\ OP\ AL}$ ($\sigma_{VU=56}$ (*alimentos*)) ⋈ $\pi_{FE\ OP}(alimentos)) \bowtie$ (*alimentos*), y la Figura 2.7 da el Tableaux para toda la expresión E.

(VU	FE	OP	NM	AL)
56	a_2	a_3	a_4	a_5
56	a_2	a_3	a_4	a_5

Figura 2.3

(VU	FE	OP	NM	AL)
56		a_3		a_5
56	b_1	a_3	b_2	a_5

Figura 2.4

(VU	FE	OP	NM	AL)
56	a_2	a_3		a_5
56	b_1	a_3	b_2	a_5
b_3	a_2	a_3	b_4	b_5

Figura 2.5

(VU	FE	OP	NM	AL)
a_1	a_2	a_3	a_4	a_5
56	b_1	b_6	b_2	a_5
b_3	a_2	b_6	b_4	b_5
a_1	a_2	a_3	a_4	a_5

Figura 2.6

(VU	FE	OP	NM	AL)
a_1	a_2			
56	b_1	b_6	b_2	b_7
b_3	a_2	b_6	b_4	b_5
a_1	a_2	b_8	b_9	b_7

Figura 2.7

Teorema 2.1. Maier demuestra en el siguiente teorema la equivalencia de E y Q [Maier 1983]. Si E es una expresión algebraica restringida para la relación **r**, y Q es el Tableaux obtenido por el método dado arriba, entonces E es equivalente a Q.

Ejemplo 2.6 Considerar el Tableaux Q de la Figura 2.8. Q no es la traducción de ninguna expresión algebraica restringida usando el método previamente dado. Supongamos que Q viene de la expresión E. E debe ser de la forma E' ⋈ E". Sean Q' y Q'' los Tableaux de E' y E" ¿Cuales renglones de Q vienen de Q' y cuales vienen de Q''?

Q(A	B)
a_1	a_2
a_1	b_2
b_1	b_2
b_1	a_2

Figura 2.8

Supongamos sin perdida de generalidad que el renglón w_1 viene de Q' El renglón w_2 viene de Q', dado que se ajusta con w_1 sobre un símbolo no distinguible. Realmente, w_3 viene de Q'. No hay renglones de Q'', así E" debe ser una relación constante, lo cual es una contradicción, dado que Q no contiene constantes. Entonces Q no viene de ninguna expresión algebraica restringida.

El método para la traducción de expresiones algebraicas restringidas sobre una relación simple, también trabaja para expresiones de una base de datos BD, si todas las relaciones en la base de datos BD son las proyecciones de una instancia común ***r***. Si E es una expresión algebraica restringida sobre la base de datos BD, debemos reemplazar cada ocurrencia de r_i por $\pi R_i(\boldsymbol{r})$, donde R_i es el esquema de r_i, y proceder como antes.

El Tableaux Q resultante garantiza que puede aplicarse a la base de datos BD. Cualquier renglón w en Q puede ser trazado para una subexpresión $\pi R_i(\boldsymbol{r})$, lo cual significa que $match(w) \subseteq R_i$. Supongamos, por ejemplo, que ***r*** es la relación sobre ABCD y R_i = AC.

La tabla consulta para $\pi R_i(\boldsymbol{r})$ es

Q'	(A	B	C	D)
w_0	a_1		a_2	
w'	a_1	b_1	a_2	b_2

El renglón w' pertenece eventualmente de w en Q. Los símbolos no-distinguibles en w' nunca pueden estar empatados, ni en la selección ni en la junta. Entonces *match(w)* en Q está contenido en R_i = AC.

Ejemplo 2.7 Sea E expresión

$\pi_{OP\ AL}\ (\sigma_{VU=106 \wedge FE=15AGO}\ (servicios \bowtie opciones)$ un Tableux equivalente a la expresión E es:

(VU	FE	OP	NM	AL)
		a_3		a_5
106	b_1	b_2	b_3	a_5
b_4	*15 Ago*	a_3	b_5	a_5

Figura 2.9 Tableaux equivalente a
$\pi_{OP\ AL}(\sigma_{VU=106 \wedge FE=15AGO}(servicios \bowtie opciones)$

2.3 Consultas Tableaux que Vienen de Expresiones Algebraicas

Teorema 2.2. Sea Q un Tableaux para la relación $r(R)$. Si Q tiene a lo más un símbolo que empata en cada columna, entonces Q puede ser derivada de alguna expresión algebraica por el método de traducción dado.

Ejemplo 2.8 Sea r una relación sobre el esquema ABCD y sea Q el Tableaux para r mostrado en la Figura 2.10. Q puede ser derivado de la expresión algebraica restringida, $\pi_{AB}(\pi_{ACD}(r) \bowtie \pi_C(\sigma_{B=2}(\sigma_{D=3}(r)) \bowtie \pi_{BD}(r)$.

(A	B	C	D)
a_1	a_2		
a_1	b_1	b_2	b_3
b_4	2	b_2	3
b_5	a_2	b_6	b_3

Figura 2.10

$Q1$	(A	B	C	D)
a)	a_1		a_3	a_4
	a_1	b_1	a_3	a_4

$Q2$	(A	B	C	D)
b)			a_3	
	b_4	2	a_3	3

$Q3$	(A	B	C	D)
c)		a_2		a_4
	b_5	a_2	b_6	a_4

Figura 2.11

los Tableaux Q_1, Q_2 y Q_3 son usadas en las subexpresiones a) $\pi_{ACD}(r)$, b) $\pi_C(\sigma_{B=2}(\sigma_{D=3}(r))$, y c) $\pi_{BD}(r)$ y son mostrados en la figura 2.11

2.4 Consultas Tableaux en Multi Relaciones

Ahora modificaremos los Tableaux de forma que representen expresiones algebraicas restringidas sobre bases de datos donde las relaciones no necesariamente son la proyección de una instancia común. Asociaremos una relación particular con cada renglón en la consulta. Asumamos que todas las relaciones en la base de datos tienen diferentes esquemas, y asociemos los esquemas de relación con los renglones.

Sea BD una base de datos con esquema R sobre R. Un Tableaux con marca Q para BD es similar a un Tableaux regular con esquema R, excepto que los renglones de Q pueden contener blancos y cada renglón tiene una marca. La marca del renglón w, denotada por $tag(w)$, es el mismo esquema de relación $S \in R$. El renglón w es blanco exactamente en las columnas R-S. Veamos una evaluación ρ para Q como el mapeo del renglón w a una tupla sobre $tag(w)$. La notación $\rho(Q) \subseteq$ BD significa que para cada renglón w en Q, $\rho(w) \in \boldsymbol{S}$, donde $\boldsymbol{s}$ es la relación en $\boldsymbol{d}$ con esquema $tag(w)$. Si w_0 es el sumario de Q, tenemos

$$Q(\mathbf{d}) = \{\rho(w_0) \mid \rho \text{ es la evaluación de } Q \text{ tal que } \rho(Q) \subseteq \boldsymbol{d}\}.$$

es decir, el resultado obtenido esta contenido en la base de datos.

Ejemplo 2.9. Tomando la base de datos de la Figura 2.12. Los Tableaux con marca (Figura 2.13a,b,c), se pueden construir aplicando el algoritmo de traducción para la consulta ***"Encuentre el nombre y la ciudad de los proveedores que surten la parte 15".***

proveedores (NOMBRE, CIUDAD, PROVEEDOR#)
surten (PROVEEDOR#, PARTE#)

Figura 2.12 Base de datos de Proveedores

Su expresión algebraica es: $\pi_{nombre,ciudad}(\sigma_{parte=15}(proveedores \bowtie surten))$. Iniciemos construyendo los tableaux para las expresiones básicas *R* y *S*.

El tableaux base de *proveedores*:

Nombre	ciudad	proveedor#	
a_1	a_2	a_3	
a_1	a_2	a_3	*proveedores*

El tableaux base de *surten*:

proveedor#	parte#	
a_4	a_5	
a_4	a_5	*surten*

Figura 2.13a

Ahora construyamos los tableaux para las expresiones compuestas

El tableaux *proveedores* ⋈ *surten*:

nombre	ciudad	proveedor#	parte#	
a_1	a_2	a_3	a_5	
a_1	a_2	a_3		*proveedores*
		a_3	a_5	*surten*

El tableaux $\sigma_{parte\#=15}$(*proveedores* ⋈ *surten*)):

nombre	ciudad	proveedor#	parte#	
a_1	a_2	a_3	15	
a_1	a_2	a_3		*proveedores*
		a_3	15	*surten*

Figura 2.13b

Finalmente construimos el tableaux de toda la expresión:

El tableaux de $\pi_{nombre,ciudad}(\sigma_{parte\#=15}(proveedores \bowtie surten))$.

nombre	ciudad	proveedor#	parte#	
a_1	a_2			
a_1	a_2	b_1		*proveedores*
		b_2	15	*surten*

Figura 2.13c

La expresión E es equivalente a la expresión algebraica dada, que involucra a lo más n-1 juntas. Y representa el conjunto a_1 tal que para algún b_1, la tupla $(a_1\ a_2\ b_1)$ esta en *proveedores* y la tupla $(b_2\ 15)$ está en *surten*.

Su expresión es equivalente en **ALPHA** a:

$$(t[\text{nombre}],t[\text{ciudad}]) : t\text{R} \wedge \exists u\text{S} \ (\ u[\text{parte\#}=15] \wedge u[\text{proveedor\#}] = t[\text{proveedor\#}])$$

La traducción de expresiones algebraicas restringidas a Tableaux con marca es esencialmente lo mismo que el caso de relación simple. La única diferencia es la tabla consulta para una relación simple **s** con esquema $A_1\ A_2\ ...\ A_k$. El Tableaux para **s** tiene un sumario con símbolos distinguibles en las columnas correspondiendo a $A_1\ A_2\ ...\ A_k$ y blancos en otro lugar. También tiene un renglón simple que es idéntico al sumario.

Para una base de datos, donde todas las relaciones son la proyección de una instancia común, la consulta por Tableaux con marca derivado de una expresión algebraica es equivalente a una versión sin marca. Las dos Tableaux son casi idénticos, excepto que, un mismo símbolo en la versión sin marca, puede estar con símbolos en blanco en la versión con marca.

2.5 Consultas Tableaux en Conjuntos

Si permitimos que las consultas Tableaux denoten una consulta es posible extender y tener una diversidad de expresiones algebraicas que pueden estar modelados por Tableaux. Para ésto regresamos a Tableaux regulares, aunque la extensión descrita trabaja también para Tableaux con marca.

Definición 2.3 Una expresión algebraica E se dice que es monotónica si se construye de relaciones (intermedias) y relaciones constantes con tuplas simples, usando:

1. Selección de la forma $\sigma_{A=c'}$
2. Proyección π,
3. Junta Natural $\bowtie$, y
4. Unión

Nótese que en este caso, la unión, intersección y selecciones de condición con los conectivos $\wedge$ y $\vee$, son todos ellos posibles. También la unión puede ser usada para construir relaciones constantes de múltiples tuplas.

Definición 2.4 Decimos que los Tableaux Q_1 y Q_2 son compatibles si tienen el mismo esquema y sus sumarios son blancos en las mismas columnas. Un conjunto de Tableaux es compatible si cada par de consultas en el conjunto son compatibles.

Definición 2.5 Se define un conjunto Q Tableaux sobre el esquema R como $Q=\{Q_1, Q_2, ..., Q_m\}$ de Tableaux compatibles, en donde todos tienen esquema R. Para una relación **r**(R), el valor de Q en **r**, denotado por $Q(\mathbf{r})$ es

$$Q_1(\mathbf{r}) \cup Q_2(\mathbf{r}) \cup ... \cup Q_m(\mathbf{r}).$$

Ejemplo 2.10 Sea *alimentos* la junta de las relaciones de la Figura 2.2. Sea Q el conjunto de Tableaux $\{Q_1, Q_2\}$, donde Q_1 y Q_2 son las de la Figura 2.14. $Q(alimentos)$ aparece en la Figura 2.15.

Q_1

(VU	FE	OP	NM	AL)
a_1		a_3	a_4	
56	15 Ago	a_3	b_2	b_3
a_1	15 Ago	a_3	a_4	b_3

$Q2$

(VU	FE	OP	NM	AL)
a_1		a_3	a_4	
56	15 Ago	a_3	b_2	b_3
a_1	16 Ago	a_3	a_4	b_3

Figura 2.14

$Q(alimentos)$

(VU	OP	NM)
56	huevos	27
56	cereal	27
56	huevos	23
56	sandwich	50
56	carne	50
56	sandwich	80

Figura 2.15

Teorema 2.3 Sea E una expresión algebraica monotónica, entonces existe un conjunto de Tableaux Q que es equivalente a E.

Demostración: Por el Teorema 2.1, es suficiente mostrar que cualquier expresión algebraica monotónica E es equivalente a una expresión algebraica monotónica,

$$E_1 \cup E_2 \cup ... \cup E_m$$

Si tomamos en cuenta la propiedad de distribución de la selección, proyección y junta sobre la unión, entonces la equivalencia se mantiene. Donde cada E_i es una expresión algebraica. De la sección 2.2 conocemos que

$$\sigma_{A=c}(E_1 \cup E_2) = \sigma_{A=c}(E_1) \cup \sigma_{A=c}(E_2)$$

para las expresiones E_1 y E_2. Tenemos $\pi x\ (E_1 \cup E_2) = \pi x\ (E_1) \cup \pi x\ (E_2)$
No es tan directo mostrar que

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

es cierto. Aplicaciones repetidas de estas identidades transforman una expresión algebraica monotónica en expresiones algebraicas unión restringida. Nótese que si E contiene *k* uniones, *m* puede ser tan grande como 2^k

Ejemplo 2.11 El conjunto de Tableaux *Q* del Ejemplo 2.10 puede ser derivado de la expresión

$$\pi_{VU\ OP\ NM}(\pi_{FE\ OP\ AL}(\sigma_{VU=56}(\sigma_{FE=15\ AGO}(alimentos) \cup \sigma_{FE=16\ AGO}(alimentos))) \bowtie alimentos).$$

2.6 Consultas Conjuntivas

Las consultas conjuntivas heredan su nombre del hecho que los conectivos *"or"*, no son admitidos en su especificación. Por ejemplo, la consulta ***"Obtenga los proveedores que viven en Puebla y que surten la parte 15".*** Es una consulta conjuntiva. Las consultas conjuntivas son un conjunto de expresiones del cálculo de dominios. Aunque no son tan expresivas como el propio cálculo de dominios, pueden expresar muchas consultas usuales y ocurren como subexpresiones en expresiones del cálculo de dominios. El principal interés, como con las tablas consultas, es porque pueden optimizarse efectivamente.

Una consulta conjuntiva para una base de datos BD es una expresión del cálculo de dominios de la forma

$$\{ x_1(A_1)\, x_2(A_2) \ldots x_n(A_n) \mid \exists y_1(B_1)\, \exists y_2(B_2) \ldots \exists y_m(B_m)\ f(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_m) \}$$

tal que *f* es la conjunción de átomos de la forma $\boldsymbol{r}(a_1\ a_2 \ldots a_m)$ donde **r** está en BD y cada a_i es tanto una constante, x_j para alguna j, o y_k para alguna k.

Cada consulta conjuntiva es una expresión del cálculo de dominios. Estas son tan expresivas como los Tableaux con marca que no contienen constantes en el sumario.

Finalmente, se tiene que los Tableaux son básicamente tableaux con marca donde un símbolo puede aparecer en múltiples columnas, con consultas conjuntivas que pueden ser traducidas en expresiones algebraicas equivalentes.

Los trabajos realizados en la teoría de bases de datos relacionales tratan de crear sistemas de bases de datos más fáciles al usuario. De tal forma que, el usuario perciba la base de datos como una simple relación universal que permita la interpretación de sus consultas en esta relación en términos de los esquemas actuales de la base de datos. La metodología ***Tableaux*** se presenta como el mecanismo para que los datos sean almacenados en la representación de una relación universal lo que hace que usuario perciba los datos más amigablemente. Esto tiene la ventaja de permitir al usuario final olvidar los detalles de separar cuales atributos están agrupados en tal o cual esquema de relación. Finalmente, la idea es eliminar del usuario final no solo lo concerniente a la organización física, sino también a la organización lógica. Esto trae como consecuencia que el DBMS trabaje más para interpretar consultas y actualizaciones sobre la base de datos, que si el usuario especifica el *query* en términos de la misma base de datos conceptual.

En este capítulo hemos llegado a establecer un sistema formal de consulta basado en tableaux. Con el objetivo de llevar a cabo la implementación de nuestro sistema, se llega a resultados cuya base es *SPJ* (Selección, Proyección, y Junta Natural) con la principal conjunción para resolver consultas. Sin embargo, las consultas por tableaux resultan no completos por considerar un subconjunto de operadores del álgebra relacional, aunque con estos se pueden resolver muchas de las consultas. Para esto en este capítulo presentamos las diversas extensiones. La primera fue llegar a que sistemas algebraicos mantienen una equivalencia en Tableaux, aspecto fundamental para validar nuestra transformación en la implementación de TabluSql. La segunda se ve Tableaux en conjuntos y uniones. Con esto hemos encontrado dos características de tableaux: los tableaux que parten de la junta de todas las relaciones; y los tableaux con marca que va formando sus expresiones de una consulta establecida. Los tableaux con marca son precisamente nuestro objetivo de implementación para transformación de expresiones provenientes de un lenguaje de alto nivel en expresiones tableaux. La conclusión de esto, es porque en un lenguaje de alto nivel no esta determinado directamente el modo de realizar operaciones en la base de datos, sin embargo apoyados en la metodología tableaux se puede realizar una transformación formal de una expresión basado en sistemas de consulta provenientes del cálculo relacional en una expresión equivalente en tableaux con marca que consiste básicamente en una expresión basado en los tres operadores SPJ. Estos operadores si tienen una implementación natural para procesamiento de consultas en sistemas computacionales que se pueden desarrollar, ya que internamente las relaciones de modelo relacional pueden manipularse en tablas con las operaciones SPJ.

Capítulo 3

Implementación del Sistema de Consulta TabluSql

En este capítulo describiremos el módulo traductor tableaux del Sistema TabluSql con base a la teoría descrita del capítulo 2. Describiremos las fases de construcción del módulo de consultas: la fase del analizador consiste en revisar el lenguaje de consulta haciendo un análisis sintáctico y semántico basado en un autómata de tipo *top-down;* la fase tableaux se obtiene un "*query-tableaux*" equivalente a la consulta del usuario y se establece una *matriz de ejecución*; y la fase de ejecución recupera renglones de datos para una declaración SELECT (*summary*) usando la *matriz tableaux*; y la fase recuperación (realiza lecturas físicas o lecturas/escrituras lógicas). Finalmente se describe el procesamiento de las consultas con un algoritmo basado en reglas y las principales estructuras de datos que contienen la definición interna del descriptor de archivos y del diccionario de datos. El Sistema TabluSql se desarrolla en lenguaje "*C*", y forma parte de los proyectos en la línea de investigación de bases de datos de la sección de Ingeniería Eléctrica del CINVESTAV. En particular, de bases de datos geográficas junto con la tesis de maestría "*Modelo Físico de un Sistema de Información Geográfica para la Exploración Petrolera*" en CIEA del I.P.N [Fiorentino 1996]. La figura 3.1, muestra los componentes del sistema.

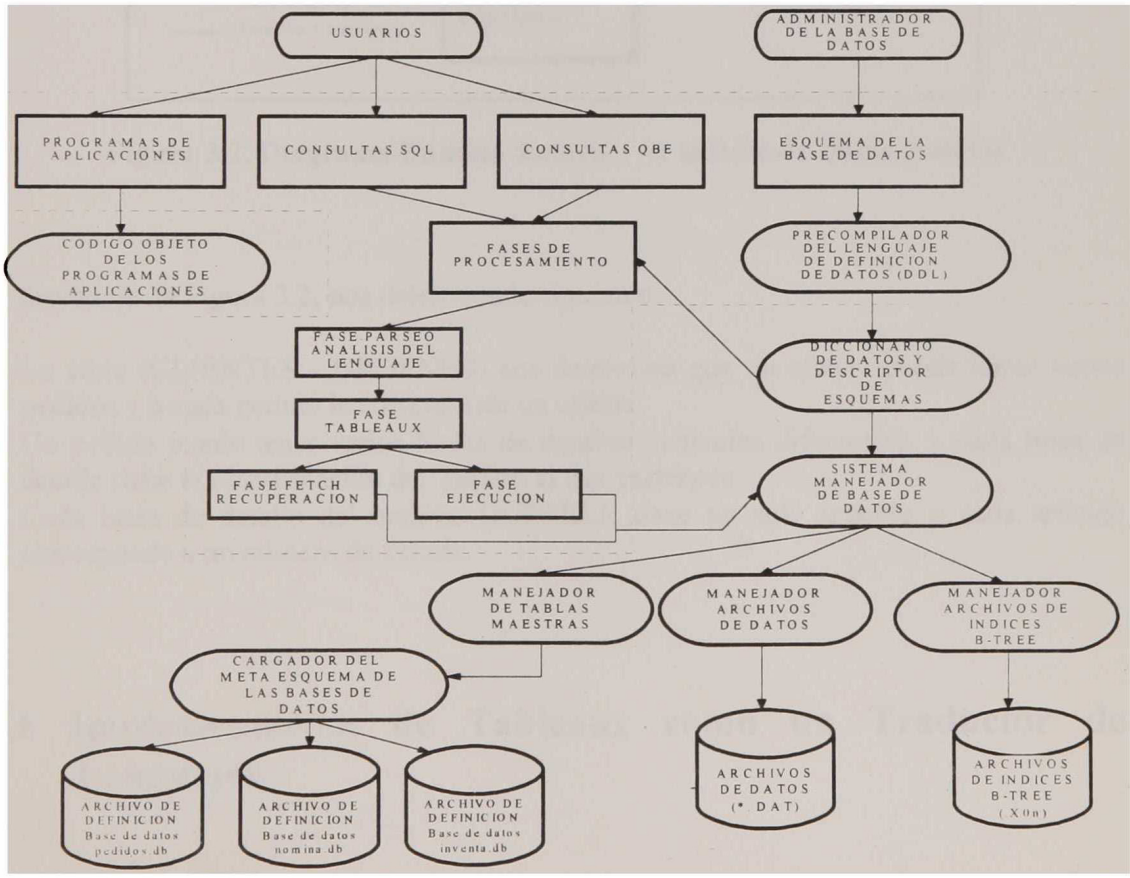


Figura 3.1. Esquema global del Sistema de TabluSql

Dejamos para el capítulo siguiente la descripción del descriptor de archivos con el sistema manejador de archivos. El sistema manejador de datos se encarga de manejar las peticiones de los usuarios a través del módulo traductor tableaux, proporciona una interfaz al usuario y al procesador de esquemas, *con* ***rutinas*** *para la manipulación de estructuras de datos lógicas del diccionario de datos, definidas por el mismo SMBD.*

La base de datos que usaremos para describir el trabajo es "***pedidos***"; contiene todas las órdenes que realizan los clientes de una distribuidora de artículos de cómputo y esta representada en la Figura 3.2.

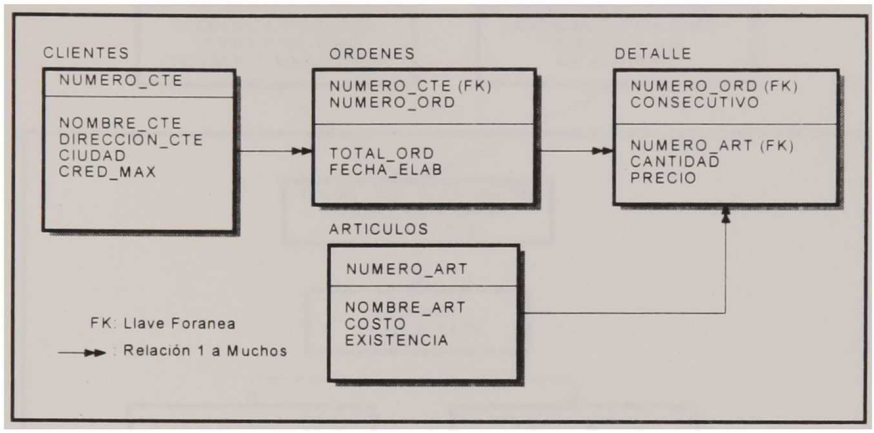


Figura 3.2. Diagrama Entidad-Relación de la Base de Datos Pedidos

El diseño de la Figura 3.2, nos determina lo siguiente:

- La tabla (CLIENTES - ORDENES) nos determina que un cliente puede hacer varios pedidos y a cada pedido le corresponde un cliente.
- Un pedido puede tener varias líneas de detalles (artículos diferentes), y cada línea de detalle tiene la identificación del pedido al que pertenece.
- Cada línea de detalle del archivo DETALLE tiene un solo artículo y cada artículo corresponde a un número de detalle.

3.1 Implementación de Tableaux como un Traductor de Lenguajes.

El proceso de implementación del módulo traductor tableaux tiene cuatro fases como se muestra en la figura 3.3 : *análisis*, *tableaux*, *ejecución* y *recuperación*. La fase del

analizador o "*revisión*" consiste en revisar el lenguaje de consulta haciendo un análisis sintáctico y semántico basado en un autómata de tipo *top-down;* en la fase tableaux se obtiene un "*query-tableaux*" equivalente a la consulta del usuario y se establece una *matriz de ejecución*; la fase de ejecución establece un plan para recuperar renglones de datos para una declaración SELECT (*summary*); y la fase *fetch* (realiza lecturas físicas o lecturas/escrituras lógicas).

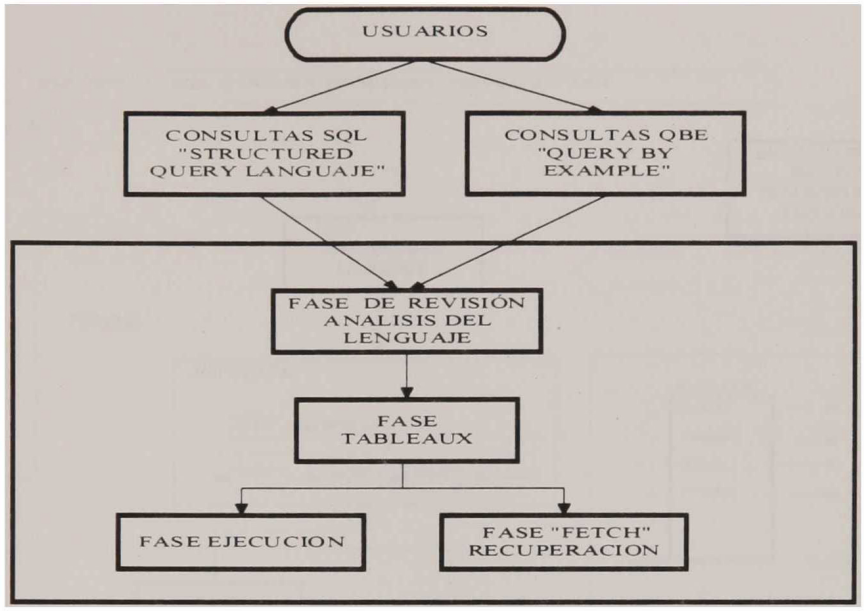


Figura 3.3 Fases de implementación del módulo traductor Tableaux

Las fases del módulo traductor, envuelven un conjunto procesos con funciones que se auxilian de: el diccionario de datos, el sistema manejador de la base de datos, y de la base de datos. Con la base de datos activa el procesamiento inicia obteniendo las consultas proporcionadas desde un editor de textos y almacenadas en un área de memoria temporal denominada ***"buffer de revisión"*** representada en la figura 3.4. El buffer de revisión interactúa con todas las fases de análisis, tableaux, ejecución y recuperación (fetch), validando sus procesos con el diccionario de datos y el descriptor de datos. Finalmente se realiza la recuperación de datos usando las funciones del sistema manejador de datos y del sistema operativo.

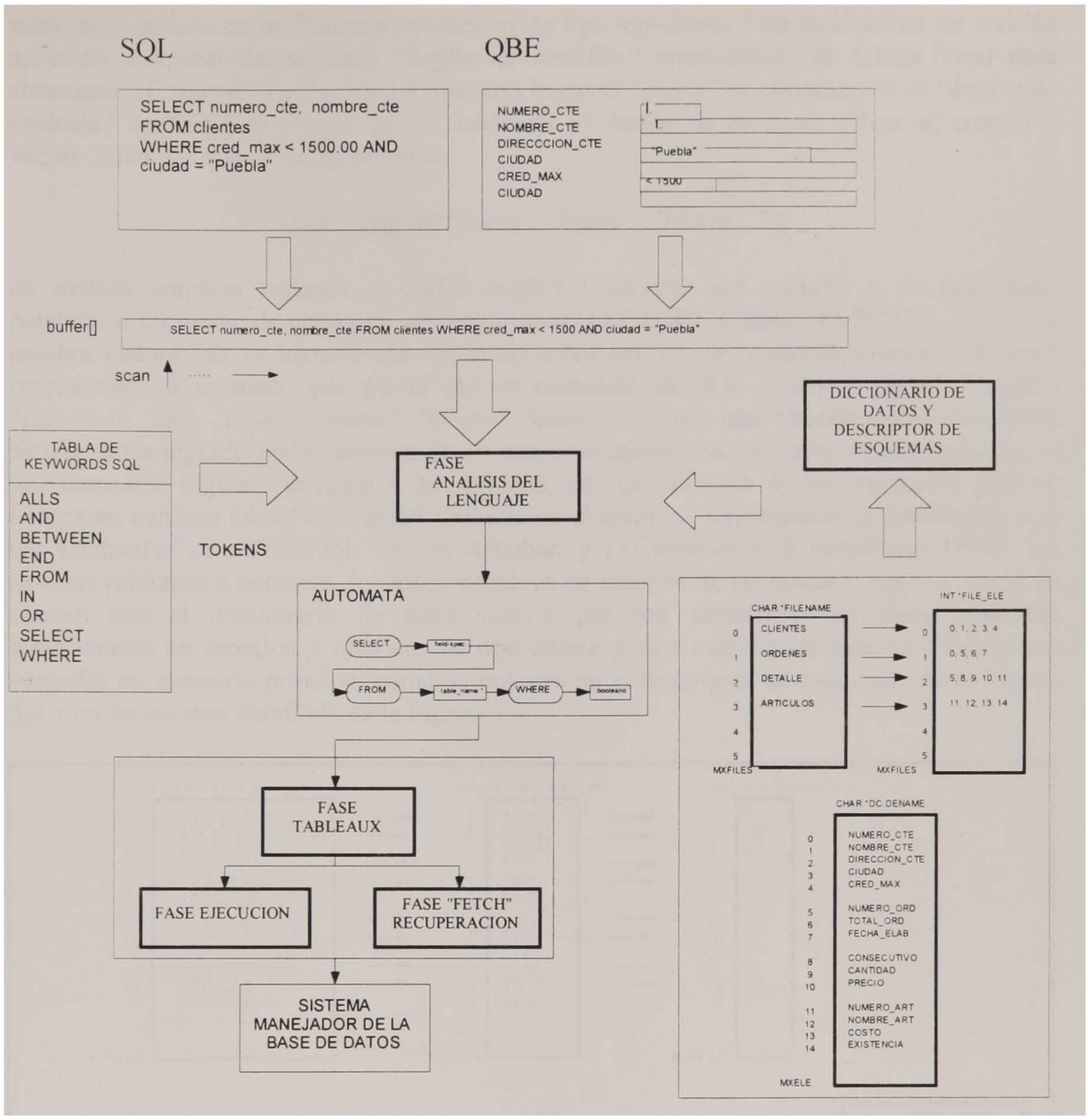


Figura 3.4. Procesamiento de consultas

3.2 Fase de Análisis "*o de revisión*".

La fase de análisis "o de *revisión*" tiene como propósito realizar el análisis sintáctico de una expresión formulada en SQL a través del editor de textos, el resultado de este análisis es llevado a la siguiente fase tableaux, en la cual se inicia el llenado de la

matriz tableaux. Para la revisión del lenguaje de consulta se realiza un análisis sintáctico y semántico basado en un autómata recursivo de tipo *top-down*. Esta fase utiliza un área de memoria temporal denominada ***"buffer de revisión"*** representada en forma lineal para almacenar el "*query*" La Figura 3.4, muestra como el "*query*" se almacena en el ***"buffer de revisión" buffer[ ]*** y el paso por el autómata. El buffer de revisión utiliza un arreglo o vector dinámico que es definido como

char * buffer["Select ... From ... Where ..."],

su análisis empieza leyendo el buffer ***buffer[ ]*** caracter por caracter y va agrupando palabras al momento de encontrar un delimitador (ESPACIO, COMA, PUNTO, ... etc.). La palabra encontrada es identificada como un *token* que es un concepto generalizado para representar un término, que puede ser un comando de SQL (*"alls", "and", "begin", "between", "by", "from", "select", "where", "query", ..., etc)*, una variable como el nombre de una tabla seguido por el nombre de un campo o ambos, un operador de comparación, o una constante lógica o decimal o de otro tipo, etc. La sintaxis de los comandos *SQL* se comparan con una tabla "*keywords*" definida en el archivo "keywords.h" y establecida con los estándares de ANSI-SQL en M.Astrahan y D.Chamberlin [Chamberlin 1974]. La sintaxis referente a nombres de tablas, nombres de campos de las tablas y tipo de datos se checan con el diccionario de datos activo que son estructuras de datos maestras almacenadas en arreglos y matrices de tipo entero y tipo carácter de tamaño dinámico y cargadas en memoria principal como se muestra en el descriptor de esquemas de la Figura 3.4 y en forma más detallada en la Figura 3.5.

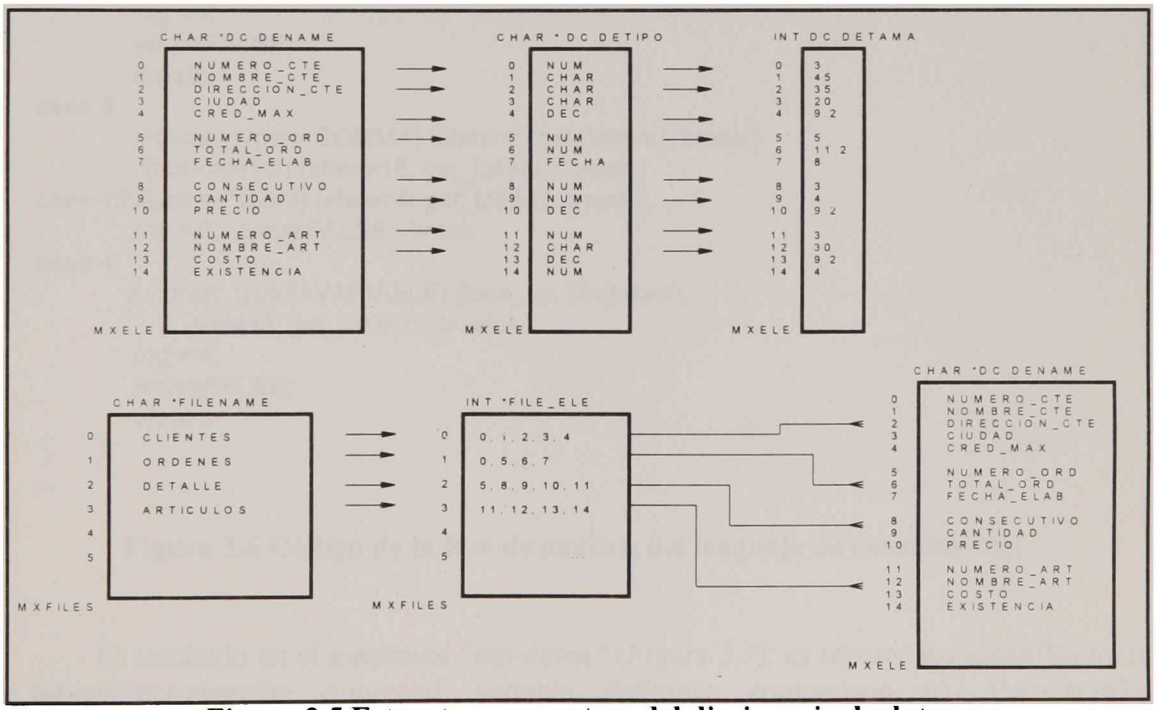


Figura 3.5 Estructuras maestras del diccionario de datos

Otras validaciones de sintaxis como operadores de comparación (<, >, ≤, ≥, =, ≠) se realizan dentro del autómata. La programación del autómata en lenguaje "C" (figura 3.6) consiste básicamente de declaraciones **CASE** y una variable de estado **state**. Cada declaración CASE activa otras posibles alternativas o estados. Por ejemplo, en un *query-block* SQL que inicia con el comando **SELECT** activa un **state=1** con la alternativa de tomar un CASE de tres posibles tokens "**UNIQUE, ALLS, o VARIABLE**" que activa un **state=2, 3, 3** respectivamente.

```
query_expr {...
  switch(state) {
   case 0:
            if(tok==SELECT) { state=1; get_token(); break;}
            mg = 0;
            error=FALSE;
            break;
   case 1:
            if(tok==ALLS) {state=3;  TODOS=TRUE; get_token(); break;}
            if(tok==UNIQUE)  {state=2; get_token(); break;}
            if(token_type==VARIABLE) {breakelem(token, SUMMARY);
                                      state=3; get_token(); break;}
            mg = 1;
            error=FALSE;
            break;
   case 2:
            if(token_type==VARIABLE) { breakelem(token, SUMMARY);
                                       state=3; get_token(); break;}
            mg = 2;
            error=FALSE;
            break;
   case 3:
            if(token_type==COMMA) {state=2; get_token(); break;}
            if(tok==INTO) {state=18; get_token(); break;}
   case 19:if(tok==FROM) {state=4; get_token(); break;}
            mg = 3;  error=FALSE;  break;
   case 4:
           if (token_type==VARIABLE) {look_up_file(token);
                state=5; get_token(); break;}
            mg = 4;
            error=FALSE;
            break;
   }
   ...
}
```

Figura 3.6 Código de la fase de análisis del lenguaje de consulta SQL

El resultado en el ***autómata "top-down"*** (Figura 3.7), es separar los identificadores o **tokens** (por ejemplo: _command, _variable, _delimiter, _comparison, etc.). Por ejemplo, si el siguiente token es **ALLS** (case 1) este asigna un **state=3** el cual advierte que su siguiente alternativa debería ser un token **FROM** esto depende definitivamente de lo que se obtenga del *buffer de revisión*. Pero si el token fuera una **VARIABLE** empezaría a

construir el nombre de una tabla "*file-name*" seguido de (.) y un campo "*field-name*" seguido por un delimitador COMMA y un nombre de una tabla "*file-name*" seguido de (.) y "*field-name*" etc., hasta obtener el token **FROM**, como se aprecia en el extracto de código de la función query_expr().

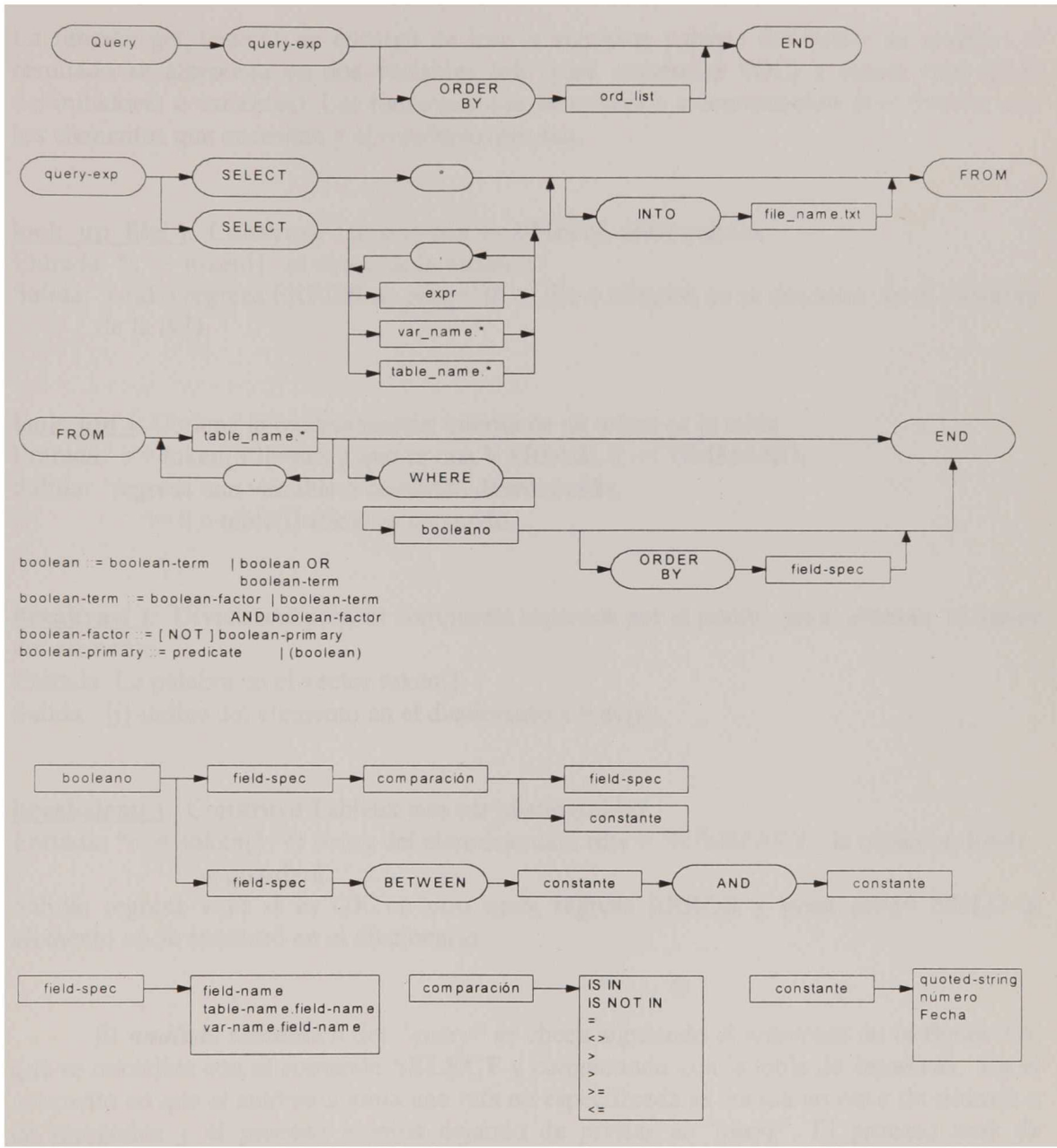


Figura 3.7 Autómata del Lenguaje de Consulta SQL

query_expr( ) : Checa la expresión del query, hace análisis sintáctico, semántico, valida, optimiza.
Entrada: Archivo file.sql, filename[], file_ele[], index[]
Salida: OK o ERROR, y la construccion, optimi[][]

La función get_token(), se encarga de leer la siguiente palabra del buffer de revisión el resultado se almacena en dos variables **tok** (para comandos SQL) y **token_type** (para delimitadores o variables). Las funciones que describimos a continuación describen su uso, los elementos que necesitan y el resultado que dan.

look_up_file( ): Construye Tableux con variables no distinguibles.
Entrada: *s = token[] el string de la relación
Salida: void o regresa ERROR en prog='\0' si file o relación no se encontró en el esquema de la B.D.

look_up( ): Obtiene la representación interna de un token en la tabla
Entrada: s = token[80]=string que es una VARIABLE o COMMAND;
Salida: regresa una variable o comando desconocido,
!= 0 o table[i].tok si es comando.

breakvar( ): Divide una palabra compuesta separada por el punto, para obtener el índice del elemento.
Entrada: La palabra en el vector token[];
Salida: [j]-indice del elemento en el diccionario s = dc[j];

breakelem( ): Construye Tableux con var_distinguibles.
Entrada: *r = token[] el string del elemento dato row = SUMMARY la posición donde se guardará
Salida: regresa void si es OK en otro caso, regresa ERROR y pone prog= NULO si elemento no se encontró en el diccionario

El ***análisis semántico*** del "*query*" se checa siguiendo el autómata de la figura 3.6, que se inicializa con el comando **SELECT** y comparando con la tabla de *keywords*. En el momento en que el autómata toma una ruta no especificada se manda un error de sintaxis o de semántica y el proceso termina dejando de revisar el "*query*". El proceso trata de realizarse como en Oracle v7.2, en donde cada declaración SQL es trabajada en un área de contexto o CURSOR y en nuestro caso el trabajo se realiza en un *buffer de revisión*. El CURSOR de oracle se utiliza cuando dos usuarios utilizan la misma declaración SQL, entonces se les asigna el mismo cursor. El cursor contiene información sobre la declaración SQL a analizar, el plan de ejecución y otros atributos.

3.3 Fase Tableaux

Esta fase comienza cuando el análisis sintáctico del lenguaje de consulta es aceptado por el autómata del lenguaje, aprovecha el proceso de descomposición del análisis sintáctico para ir llenando las columnas en la matriz poniendo una marca de la tabla que esta construyendo en el correspondiente renglón del tableaux. El resultado que se obtiene al final de la fase es una matriz que puede ser procesada con operaciones SPJ del álgebra relacional.

La equivalencia que existe entre los lenguajes QBE, SQL, y Tableaux permiten que se puedan agrupar en tres instancias de selección: selección de columnas, selección de relaciones o tablas, y la selección de criterios. La figura 3.8, muestra las secciones principales de las tres notaciones que mantienen una equivalencia paralela. Por ejemplo, para la instancia de selección de datos la cláusula **SELECT** es equivalente al *operador proyección (Π)* del álgebra relacional, que se representa en el *"summary" de la matriz Tableaux*. La cláusula WHERE es análoga al operador σ-restricción y se representa en nuestra implementación tableaux en una caja de condición contenida en una estructura de datos independiente *where[i]*, cada uno de sus elementos ***i*** esta en correspondencia con el diccionario de datos. Finalmente la selección de tablas declaradas en la cláusula FROM es semejante a las tablas se guardan en los siguientes renglones de la matriz ***tableaux***, ocupando para ello ***tableaux*** con marca. La marca ayuda para que no haya dos renglones en el tableaux para una misma tabla, esto facilita que se eliminen esquemas repetidos en el proceso de llenado. En el momento del cálculo de la consulta "*fase de ejecución*" o interacción con la base de datos física, sólo se recorre una sola vez.

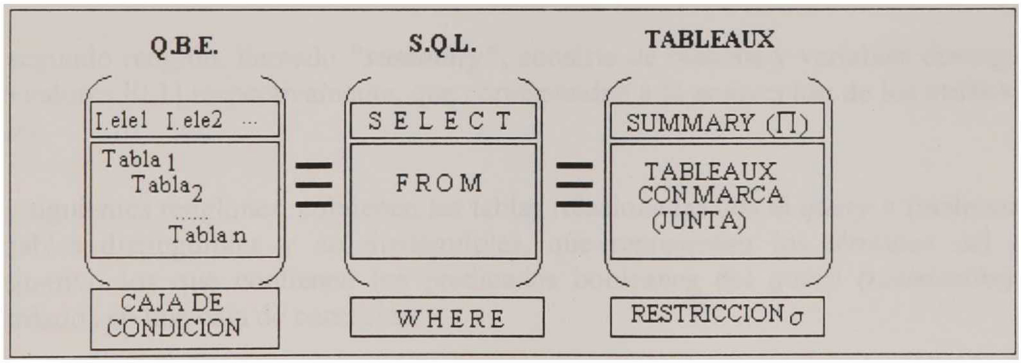


Figura 3.8 Equivalencias QBE, SQL y Tableaux

Ellas tienen su origen de las expresiones descritas en el capítulo 2, que tienen la forma de la expresión E

$$E = \{ a_1 \dots a_n / \ (\exists b_1)\dots(\exists b_2)(P_1 \dots P_k)\}$$

la cual consta de variables distinguibles *a*'s y variables no distinguibles *b*'s. Las variables distinguibles corresponden a las variables libres de las expresiones del cálculo relacional (las cuales instanciamos como la selección de columnas) y las variables no distinguibles que corresponde a las variables ligadas junto con los *P's* son los predicados donde la expresión *"E"* se hace verdadera (instanciado como la selección de relaciones y criterios).

En la implementación de Tableaux, usamos como estructura principal una matriz *tableaux[i][j]* que se representa en un arreglo bidimensional con valores (0-ausente y 1-presente), con las divisiones representadas en la figura 3.9:

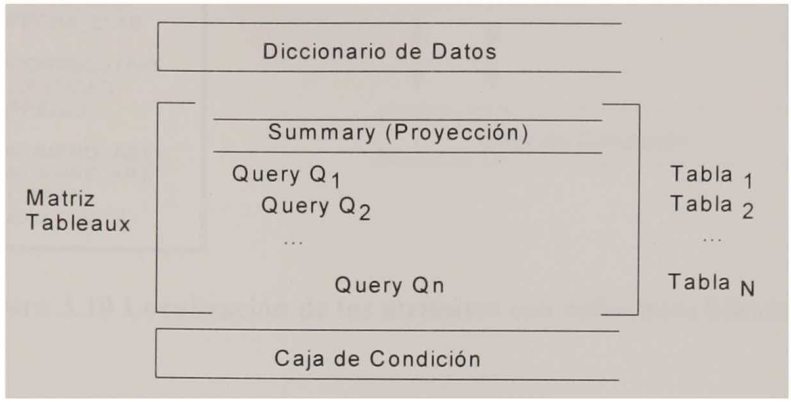


Figura 3.9 Estructura de la matriz tableaux.

- El primer renglón en la matriz *tableaux* corresponde al universo de datos (***diccionario de datos)***. Este renglón solo es abstracto ya que sería redundante con el diccionario de datos.

- El segundo renglón, llamado ***"summary"***, consiste de blancos y variables distinguibles con valores [0,1] respectivamente, que corresponden a la ***proyección*** de los atributos del *query*.

- Los siguientes renglones, contienen las tablas relacionadas con el *query* y finalmente las variables distinguibles y no distinguibles, que representan los términos del *query* conjuntivo los que contienen los predicados booleanos del *query* ***(restricción)***, son guardados en una caja de condición

En el llenado de cada de las secciones de nuestra matriz tableaux: summary, tablas, y caja de condición se usa la posición que guardan los nombres de los elementos en el diccionario de datos, tomando en cuenta que el nombre es único en el universo del diccionario de datos. Es decir, la posición de los campos en el ***summary***, tiene también referencia biunívoca con el diccionario de datos *"universo de datos"* y con la caja de condición de la Figura 3.10. Entonces, para cada uno de los atributos especificados en una declaración **SELECT** primero se busca la posición que le corresponde en el diccionario de los datos, esta misma posición es el valor de desplazamiento sobre las columnas del

SUMMARY y de la caja de condición de la matriz ***tableaux.*** Una vez posicionado en la columna *i* del summary se le asigna un valor de 1 al summary o cero por omisión.

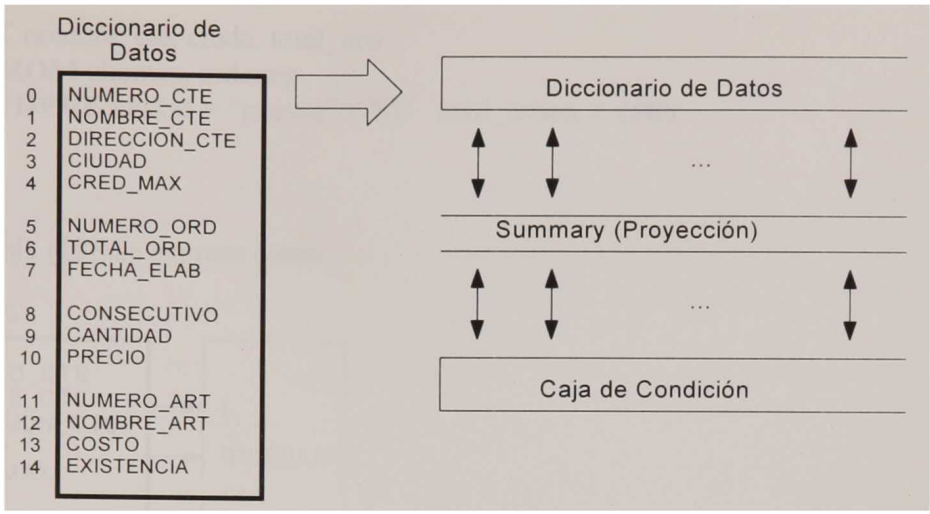


Figura 3.10 Localización de los atributos con referencia biunívoca.

Ejemplo 3.1 El proceso de encontrar el campo CIUDAD en la tabla de CLIENTES.

SELECT ciudad FROM clientes

Se busca el campo ciudad en el diccionario dc.dname[ciudad] lo que da
⇒ *i*=3, este mismo índice es usado en el summary para asignarle el valor de 1
⇒ summary[*i*=3] = 1.

La siguiente función breakelem() muestra el proceso de llenado de la matriz .

breakelem(): Constuye Tableux con var_distinguibles.

ENTRADA - *r = token[] el string del elemento dato
row = SUMMARY la posición donde se guardará
SALIDA - regresa void si es OK,
ERROR en otro caso y pone prog= Fin si elemento no se encontró en el diccionario

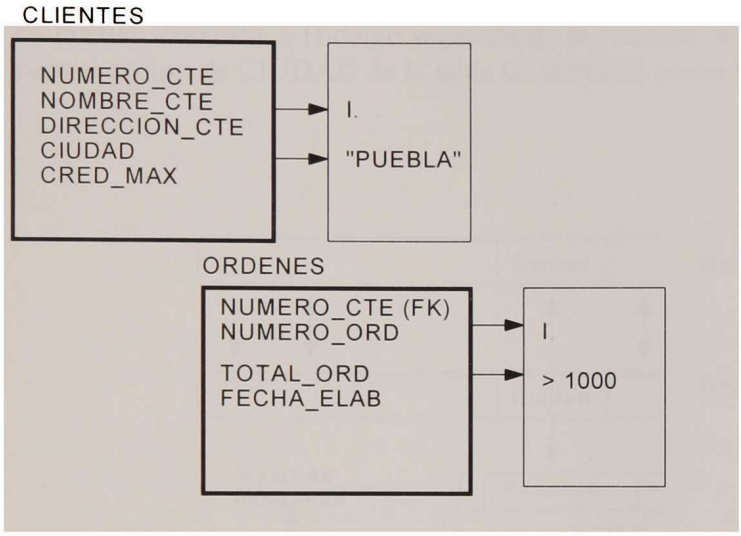
PSEUDO-CODIGO

1. Se descompone una cadena de nombres de tablas y atributos separados por (.)
2. La parte izquierda del punto es definida como el nombre de una tabla.
3. La parte a la derecha del punto se define como el atributo de una tabla
4. Busca el nombre del atributo en el diccionario de datos
5. Si, lo encontró marca un renglón de matriz tableaux con el nombre de la tabla y coloca el valor de 1 en la columna que corresponde en el diccionario pero en tableaux
6. Checa si ya existe para no remarcar el summary

Ejemplo 3.2 La consulta <<*Encuentra los clientes, y el número de la orden de los clientes que viven en Puebla y compraron más de 1000 pesos*">>, de la base de datos pedidos. En lenguaje *SQL* se expresa como:

SELECT nombre_cte, ciuda, total_ord
FROM clientes, ordenes
WHERE ciudad = "puebla" AND total_orden > *1000.*

En lenguaje *QBE* se expresa como:



La transformación de esta consulta en nuestra matriz tableaux *tableaux[i][j]* queda:

numero_cte	nombre_cte	direccion_cte	ciudad	numero_ord	total_ord	....	
0	1	0	0	1	0		Summary
1	1	1	1	0	0		clientes
1	0	0	0	1	1		ordenes
			"PUEBLA"		> 1000		Caja de Condiciones

Diccionario

La caja de condición es una estructura de datos adicional que crece dinámicamente respecto a las restricciones (σ) del *query*. Y se define internamente como:

```
Struct where {
        int cmp;          /* Operador de comparación <, ≤ , =, >, ≥, ≠, ...*/
        char *valor;      /* Valor a encontrar */
};
Struct elemto{
```

```
        int el;                          /* Contador de restricciones */
        struct where times[MXRES];       /* Límite de restricciones permitidas */
} atributo[MXELE];                       /* Limite de predicados en WHERE */
```

Figura 3.11 Estructura del operador restricción (σ) en la matriz Tableaux.

Cada elemento de esta estructura es un arreglo de MXELE posibles restricciones, con una estructura de tres componentes: el primero contiene un contador de restricciones; el segundo guarda el operador de comparación o lógico; y el tercero guarda el valor que se compara con la base de datos. Por ejemplo si deseamos encontrar los clientes que radican en Puebla, Tlaxcala, e Hidalgo seguidos de la cláusula WHERE, se obtendrá una estructura para el campo de CIUDAD de la tabla CLIENTES como lo muestra la Figura 3.12.

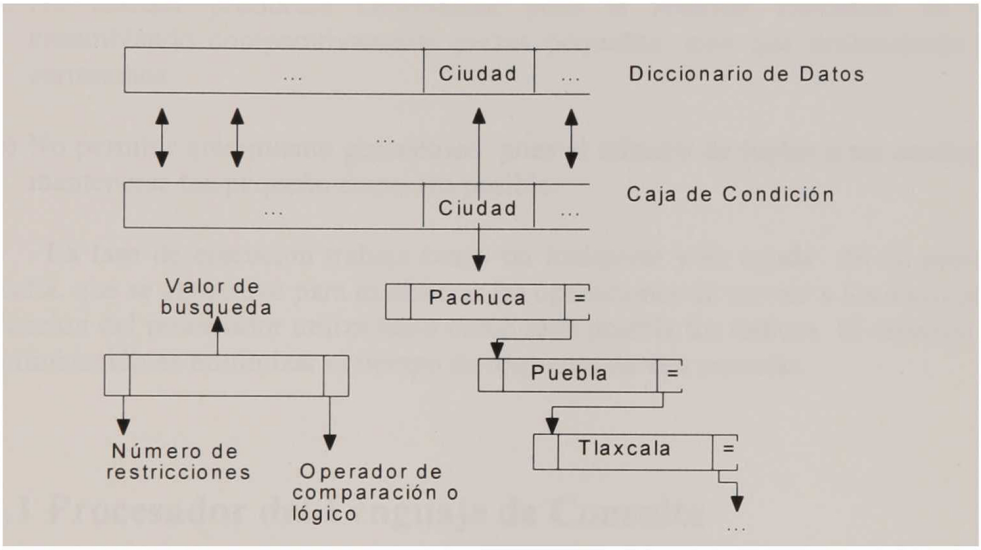


Figura 3.12 Estructura del operador restricción. El operador (=), puede ser sustituido por cualquier otro operador lógico o de comparación (OR, AND, <, > , ≥ , ≤ , ...).

3.4 Fase de Ejecución

La fase de ejecución, toma la matriz tableaux resultante del análisis y traducción del lenguaje de consulta a tableaux, y aplica un plan para ejecutar las operaciones relacionales (*Restricción, Proyección y Junta*) ayudado con técnicas recuperación de registros por índices. El procesamiento es inmediato ya que la *matriz tableaux* representa un conjunto de

llamadas a funciones del *SMBD* que abren y crean archivos, localizan registros, y manipulan campos mediante operaciones relacionales. El propósito en la *Matriz Tableaux* es formar un plan para fijar y controlar el conjunto de operaciones relacionales que darán la respuesta a las consultas.

En el llenado de la matriz el optimizador elimina esquemas repetidos del *tableaux*, lo que permite reutilizar la primera definición del esquema en el *Tableaux*. En el momento del cálculo de la consulta (interacción con la base de datos física), este se recorre una sola vez, ya que se conoce completamente el *query* restringido para este esquema. Esto es semejante a lo propuesto por [Wong 1976] en su algoritmo para procesamiento de consultas desarrollado para QUEL el cual es un lenguaje de datos para el Sistema INGRES que descompone las expresiones en consultas parciales como lo usamos en nuestro *tableaux* para procesar *queries* en forma independientemente.

El procedimiento general es reducir un *queries* multivariable, en una secuencia de *queries* simples. La estrategia de procesamiento de los *queries*, tiene dos objetivos generales:

a) No realizar productos cartesianos, pues la relación resultante es construida ensamblando comparativamente piezas pequeñas, más que deshaciendo productos cartesianos.

b) No permitir crecimiento geométrico, pues el número de tuplas a ser analizadas, debe mantenerse tan pequeño como sea posible.

La fase de ejecución trabaja como un intérprete y se ayuda de un procesador de consulta, que se construyó para minimizar las operaciones de acceso a los datos requeridos. La función del procesador utiliza tanto como sean posible los índices. El objetivo básico de la optimización es minimizar el tiempo de respuesta de una consulta.

3.4.1 Procesador del Lenguaje de Consulta

Con la utilización de los índices, la localización de los registros se puede obtener en dos movimientos uno al índice y otro a la localidad dada en la tabla. Sin embargo, los costos de tener índices consumen espacio y tiempo adicional en las actualizaciones. El espacio adicional de los índices, no es un problema grave, si consideramos los segundos perdidos para actualizar una tabla.

En las consultas que involucran más de una tabla (JUNTA), existe la posibilidad de una respuesta lenta, ya que la cabeza del disco se tiene que mover al lugar que ocupa una tabla, leer, moverse a otra tabla, leerla y juntarla en una tupla (registro o renglón), moverse de regreso a la posición original, y así respectivamente. Por ello, sugerimos usar el número mínimo de índices para una adecuada optimización y que se tengan presentes las siguientes recomendaciones.

1. El identificador clave de cada tabla, debe tener creado un índice único. Esto se hace para mantener la unicidad del valor de la llave, y es muy importante para un acceso E/S inmediato.
   a) La llave es el argumento de búsqueda más frecuentemente, usado para recuperar.
   b) La llave es realmente el elemento usado para hacer la JUNTA de una tabla con otras.

2. Las llaves foráneas deben estar indizadas (Las llaves foráneas elementos dato que son llaves de otras tablas, algunas veces referidas como "*llaves de otras relaciones*").

3. EL elemento dato más usado para una respuesta rápida, debe considerarse llave candidata.

4. Las columnas usadas para ordenamiento, en una cláusula ORDER BY .. son candidatas a ser indexadas.

El intérprete y el procesador usan el Lenguaje de Definición de Datos, para saber como está conformados el descriptor de archivos y el diccionario de datos y para determinar en que tablas están contenidos los elementos a consultar y como están almacenados (entendiéndose por ello, su tipo, tamaño e índices). Estas estructuras se mantienen en memoria dinámica y el intérprete las utiliza a la vez que recorre la matriz Tableaux. El cálculo de la consulta se establece conforme al algoritmo siguiente, que contempla el manejo de los índices de los puntos anteriores.

Algoritmo de Cálculo de Consultas a *query*'s TABLUSQL.

Entrada: Una matriz Tableaux que contiene los elementos que proyectarán (**Π**), las tablas del *query* en cuestión (**junta**) y los predicados o restricciones de cada tabla (**σ**).

Salida: Una forma de calcular este tipo de *query*'s, usando los operadores del álgebra relacional (*proyección, restricción y junta*) aprovechando los índices de las tablas.

Método: Consideremos la siguiente lista de métodos para obtener el *query* representado en la matriz Tableaux, junto con sus predicados. Para estos métodos, se tiene una elección del índice o predicado a usar, y sobre cual tabla o relación procesar primero.

Un caso especial se da cuando hay una operación binaria *JUNTA* que tiene *restricciones* y *proyecciones*. La *proyección* se establece, tomando una imagen idéntica a los elementos que aparecen en el *summary* de la matriz Tableaux, en forma de registro de un archivo temporal que se crea dinámicamente. Posteriormente se aplica la *restricción* y la *junta*. Si todas las *restricciones* llevan a conjunciones verdaderas en un ciclo de registro la tupla es aceptada y guardada en el registro creado.

Base: La base de datos, debe estar diseñada mínimamente en primera forma normal de Codd [Codd 7], considerando dependencias funcionales, y vínculos uno a muchos (1-n: Padre-Hijos) o muchos a uno (n-1: Hijos-Padre).

1. Obtener las tuplas de cada *query* de la matriz Tableaux que satisfacen el predicado de la forma $A_i = C_i$ (A_i-elementos dato, C_i-constante), que empate con un agrupamiento de índices. Posteriormente, aplicar el resto de los predicados a las tuplas obtenidas.

2. Si el *query* es A=c o Aθc y no se pueden usar los índices, debemos leer simplemente todas las tuplas de la tabla especificada en el *query* y aplicar los predicados a cada uno de los elementos requeridos.

3. Si el *query* tiene una operación binaria JUNTA entre las tablas f y g, se toman en cuenta los siguientes criterios.

i. Se determina el (los) elemento(s) común(es) de la JUNTA.
ii. Se determina la relación uno a muchos (1-n) entre las tablas f y g.
iii. Se determina si hay condiciones de búsqueda sobre los elementos de las tablas f y g.
iv. Se determina quien lleva el proceso de lectura de los registros externo e interno, para evitar accesos repetidos o redundantes. Aplicando para ello las funciones *find_childs()* y *find_father()*.

La función *find_childs()* obtiene de dos tablas diferentes, las tuplas hijos de cada tupla padre. Usada en consultas del tipo <*Obtén las ordenes del cliente Juan Perez*>. Para esto debe existir una correspondencia uno-muchos.

El pseudo-código de la función find_childs es:

a. Por cada una de las tuplas del archivo "Papá".
b. Buscar los "*n*" tuplas hijo que cumplan con la tupla "papá".
c. Copiar la tupla "papá" a la proyección para preparar la salida
d. Copiar la tupla "hijo" a la proyección para preparar la salida
e. Continuar buscando en la tabla papá la siguiente tupla que satisfaga el filtro.

- La función *find_father()* obtiene de dos tablas diferentes, la tupla padre de cada hijo. Usada en consultas del tipo <*Obtén los clientes que compraron más de 1000 pesos* >. Para esto debe existir una correspondencia uno a muchos.

El pseudo-código de la función find_father es:

a. Por cada una de las tuplas del archivo "hijo".
b. Buscar los "*n*" tuplas papá que cumplan con la tupla "hijo".
c. Copiar la tupla "hijo" a la proyección para preparar la salida
e. Copiar la tupla "papá" a la proyección para preparar la salida
f. Continuar buscando en la tabla hijo la siguiente tupla que satisfaga el filtro.

4. Aplicar los puntos 1, 2, y3 a las funciones *find_childs()* y *find_father()*.

Ejemplo 3.3 Supongamos que hay un único número de orden, y un cliente tiene varios pedidos, esto representa vínculos uno-muchos. Definamos a los elementos numero_cte y numero_ord como la llave primaria de ORDENES, y a numero_cte la llave primaria de CLIENTES. La tabla CLENTES es padre de ORDENES y ORDENES es el hijo de CLIENTES, en el sentido que no hay órdenes sin antes tener clientes. El *SMBD* es el encargado de supervisar que no haya hijos sin antes tener el padre. Analicemos algunas de las consultas que se pueden derivar de este diseño:

Q1. Para consultas *"Obtener los clientes con número de clave 201, 402, y 603"* El acceso es sobre la llave primaria con elemento **numero_cte**. El procedimiento anterior no se alteraría, si ahora, utilizáramos un predicado sobre cualquier otro elemento que no es llave como **ciudad**. La búsqueda sería: sobre el elemento **numero_cte** que es la llave primaria y después sobre el predicado ***ciudad="Pachuca"***. Por ejemplo:

Q2. *"Obtener los clientes con clave del 201 al 603 que viven en la ciudad de Pachuca".*

Sin embargo, si los predicados en la consulta son sobre elementos que no son llaves, tales como Q3. Se utiliza el paso 2 de nuestro algoritmo, que lee secuencialmente el archivo, tupla por tupla.

Q3. *"Obtener los clientes que viven en la ciudad de Pachuca".*

En consultas que involucran dos o más relaciones (*JUNTA*), el sistema analiza como están los vínculos del *query*, para ir navegando de una relación en otra. Los siguientes resultados son algunas ideas de aplicar el paso 3.

Q4. En la consulta *"Dame las ordenes de cada cliente"*. Observe que se presentan las dos relaciones CLIENTES y ORDENES donde no hay condiciones sobre la información a extraer, y no se pueden usar las llaves. El resultado de la consulta se puede obtener de dos maneras:

1. Si partimos de la tabla CLIENTES y a cada tupla le aplicamos el *JOIN* con la tabla ORDENES. Es decir, partimos de una tupla-hijo y obtenemos su tupla-padre. El resultado se presenta en forma ordenada por empleado.

2. Sin embargo si primero usamos la tabla ORDENES y buscamos todos sus CLIENTES. Es decir, partimos de una tupla-padre y obtenemos sus tuplas-hijos. El resultado se presenta en forma ordenada por departamento.

Nota: El SMBD se encarga de validar que las tuplas hijos siempre tengan tupla padre. Para el caso 1, se usa la función *find_father()* y en el segundo la función *find_childs()*.

Q5. En la consulta *"Dame los nombres de los clientes que tienen compras superiores a $1000, y que viven en la ciudad de Puebla y Pachuca"*, se tienen condiciones de búsqueda en ámbas tablas. El resultado se obtiene de la siguiente forma.

- Si el proceso inicia con la tabla ORDENES, que obtiene las tuplas donde el total de orden > 1000, y si no hay tuplas que empaten con las condiciones, el proceso termina, sin necesidad de leer la tabla CLIENTES para obtener los clientes de la ciudad de *"Puebla"* y *"Pachuca"*. Este caso permite un procesamiento más rápido si el total de las tuplas es menor a clientes.

- Esto es más directo que partir de la tabla CLIENTES, ya que puede haber muchos clientes que son de la ciudad de *"Puebla"* o *"Pachuca* y no realizaron compras mayores a 1000. Este proceso lee dos tablas completas lo que hace lento el procesamiento del resultado.

Algunas de las funciones para el procesamiento de consultas se dan en la Tabla 3.1.

Tabla 3.1 Funciones del módulo de procesamiento de la consulta

FUNCIÓN	DESCRIPCIÓN
separa_rel()	Separa las relaciones con JOIN y Producto Cruz
relate_files()	Checa si una tabla tiene vinculación
restriccion(f)	Checa si la tabla *f* tuvo restricción
informa(f,g)	Informa si el elemento de la tabla es llave primaria foránea.
accesa_key_prim(f)	Busca el registro (tupla) por la llave primaria.
barre_rel(f)	Barre archivo sin uso de llaves
project_query()	Realiza la proyección de la relación resultante.
cump_elemen_rest(f,e)	Checa si el elemento *e* del archivo *f* cumple con la condición.
prueba_rcd(f)	Checa si el registro de tabla *f* cumplió con las condiciones.
elecmp(valbd, op, valq, tipo)	Compara el valor del query *op* con el valor de base de datos.

El pseudo-código para procesar queries que usan una tabla es:

query_simple

1. Inicia el SMBD, abre B-tree
2. Checa si *query* tiene restricción
3. Checa si se pueden usar las llaves primarias o alternas
4. Realiza la gestión de registros aplicando restricciones
4. Prepara la salida, realiza la proyección
5. Cierra el SMBD

3.5 Fase de Recuperación "*o fetch*"

En esta sección describimos la última de las fases del módulo traductor, la fase de recuperación "*o fetch*" recupera los datos almacenados en las tablas. Esta función es procesada dentro del sistema manejador de la base de datos y más específicamente por el manejador de archivos B-treee y el manejador de archivos y físico de la base datos, trabaja en el nivel más bajo en el que se tiene la forma como se están almacenando los datos.

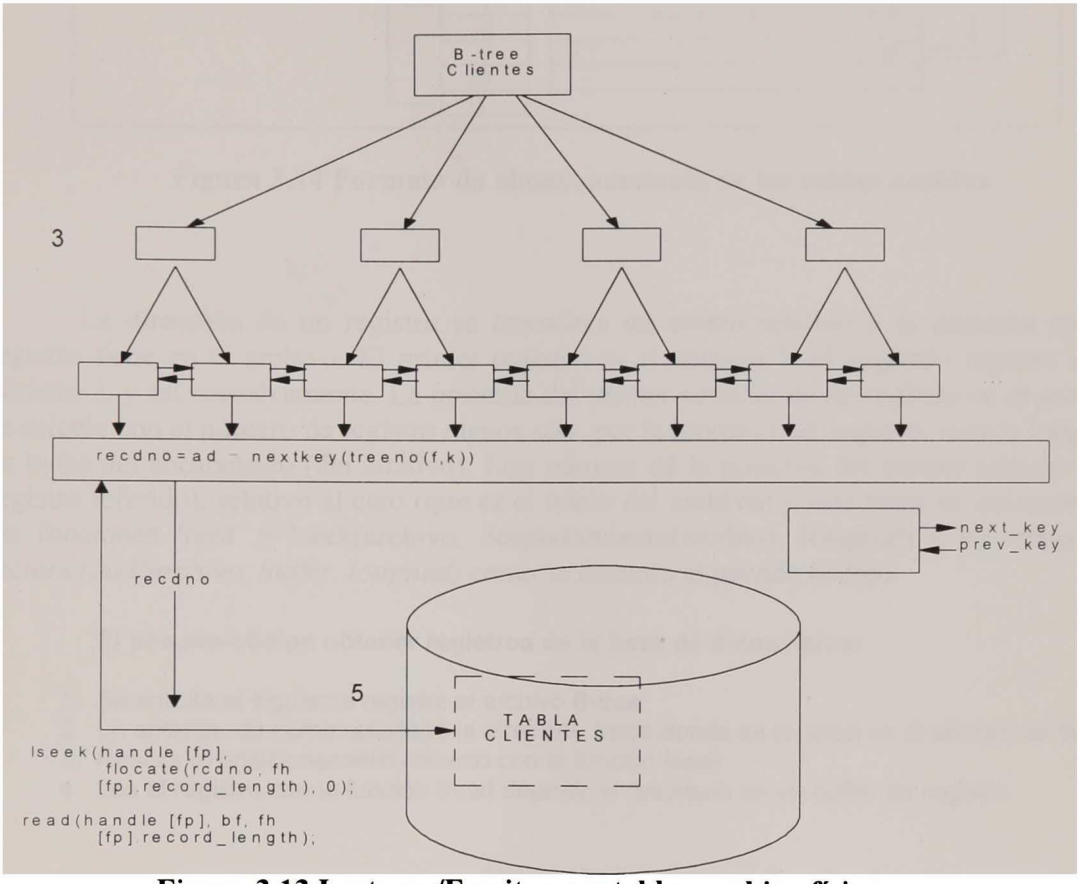


Figura 3.13 Lecturas/Escrituras a tablas-archivo físicos

Un tabla-archivo está compuesto por un encabezado Figura 3.14, y una colección de registros formados mediante un algoritmo de listas. Los registros de datos de las tablas son una colección de elementos de datos (*campos*), organizados en un formato de longitud de registros fija (**len**) y se determina con la suma de las longitudes de los elementos dato, definida en la estructura del diccionario de datos.

La posición del primer carácter de un registro en el archivo se calcula con el número de registro menos uno, por la longitud del registro, mas la longitud en bytes del encabezado (del archivo). Este número dá la posición del primer carácter (del registro referido), relativo al cero (que es el inicio del archivo) y este valor es utilizado por las funciones *'lseek'*.

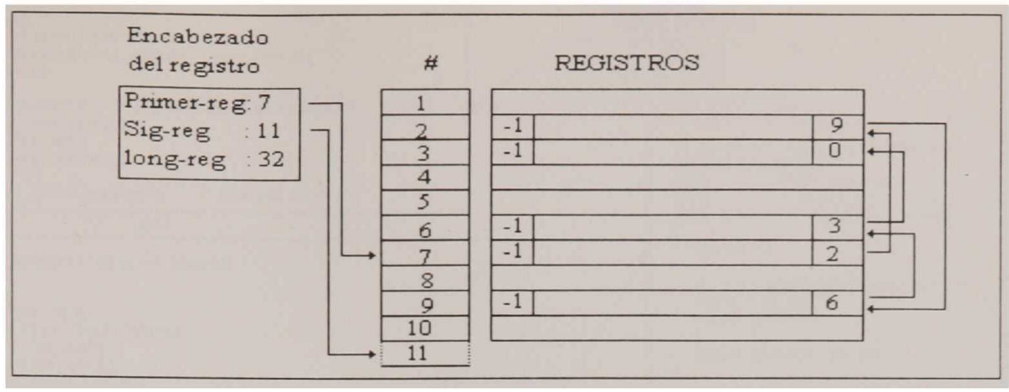


Figura 3.14 Formato de almacenamiento de las tablas-archivo

La dirección de un registro se considera un entero relativo a la posición que el registro tiene en el archivo. El primer registro es el número 1, el segundo registro es el número 2, y así sucesivamente. La posición del primer carácter de un registro en el archivo se calcula con el número de registro menos uno, por la longitud del registro, mas la longitud en bytes del encabezado (del archivo). Este número dá la posición del primer carácter (del registro referido), relativo al cero (que es el inicio del archivo) y este valor es utilizado por las funciones *'seek'* y lseek(archivo, desplazamiento(recdno), longitud) y su respectiva lectura *read(archivo, buffer, longitud)* como lo muestra el pseudo código.

El pseudo-código obtener registros de la base de datos física:

1. Se solicita el siguiente registro al archivo B-tree
2. Lo anterior, da como resultado la dirección física donde se localiza en el archivo de datos
3. Realiza el posicionamiento directo con la función lseek
4. Lee el registro con la función fread dejando el resultado en un buffer de registro.

Un proceso más detallado de la secuencia para obtener un registro de los archivos físicos de la base de datos se representa en la figura 3.15. El paso 1 inicia con la rutina que lee dos tablas f y g, el paso 2 solicita el siguiente registro a leer en el orden de la llave secuencial, el paso 3 busca el siguiente valor en el archivo b-tree, en el paso 4 y 5 se hace el posicionamiento en el archivo de datos con base a la dirección obtenida en el paso 3. Finalmente se regresa al paso 2, para ordenar el siguiente registro.

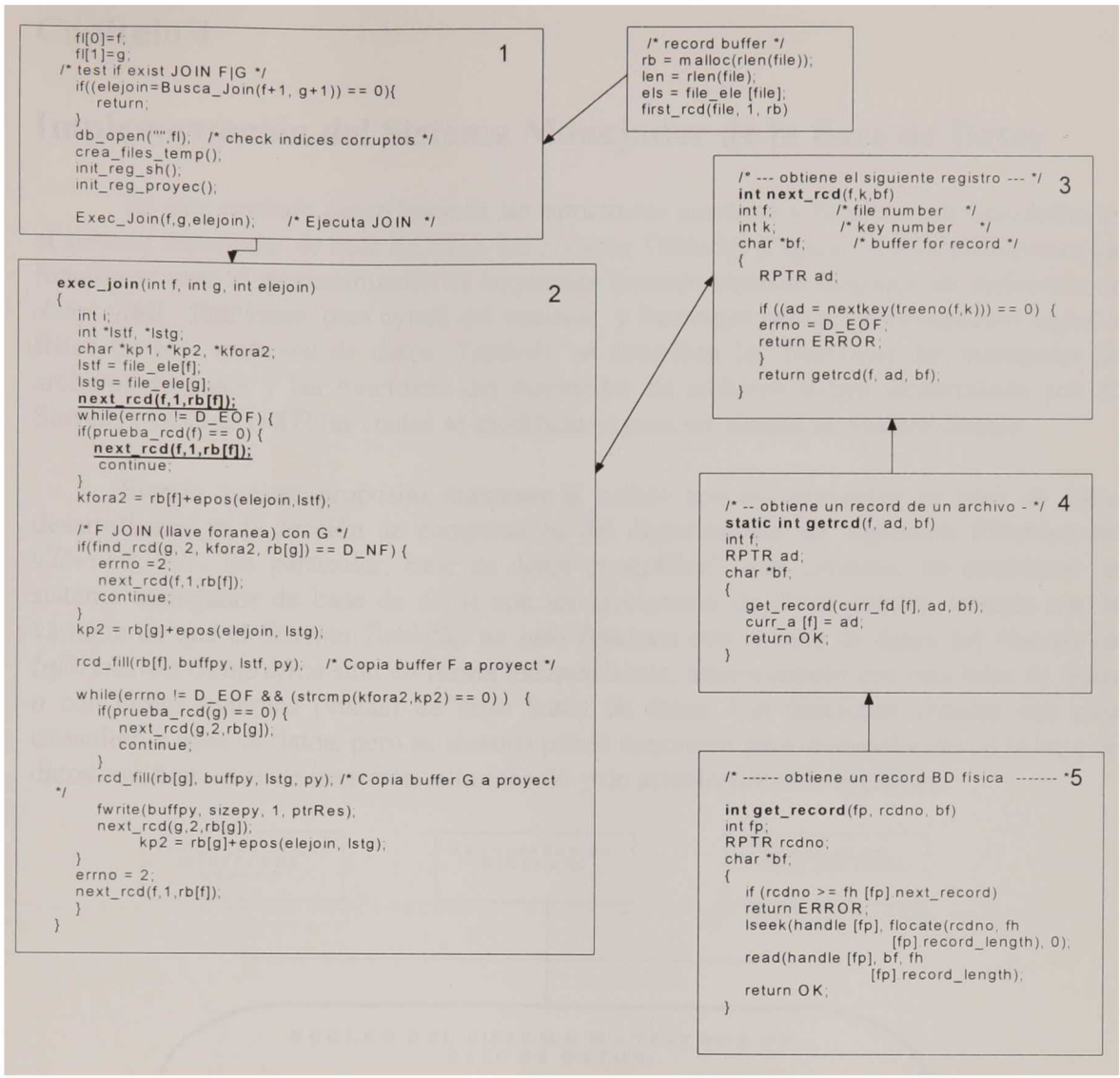


Figura 3.15 Proceso de recuperación de registros en tablas-archivo

En este capítulo hemos descrito el proceso general del Sistema TabluSql. También hemos establecido las fases para la implementación del módulo traductor tableaux: la fase del analizador o "*revisión*" consiste en revisar el lenguaje de consulta haciendo un análisis sintáctico y semántico basado en un autómata de tipo *top-down;* la fase tableaux se obtiene un "*query-tableaux*" equivalente a la consulta del usuario y se establece una *matriz de ejecución*; la fase de ejecución recupera renglones de datos para una declaración SELECT (*summary*) usando la *matriz tableaux*; y la fase *fetch* (realiza lecturas físicas o lecturas/escrituras lógicas). Finalmente describimos el procesamiento de las consultas con el uso de índices y las principales estructuras de datos que contienen la definición interna del descriptor de archivos y del diccionario de datos.

Capítulo 4

Implementación del Sistema Manejador de la Base de Datos

En este capítulo describiremos las estructuras maestras y funciones desarrolladas en el sistema manejador de base de batos del Sistema TabluSql (Figura 4.1). Describiremos las funciones: para el procesamiento de esquemas llamado también *lenguaje de definición de datos (ldd);* funciones para ayuda del sistema; y funciones para el procesamiento lógico y físico de los archivos de datos. También se describen las funciones del manejador de archivos de datos y las funciones del manejador de archivos B-tree desarrollado por Al Stevens [Stevens 1987] las cuales se modificaron para ser usadas en nuestro trabajo.

Siendo nuestro propósito mantener el enlace con los proyectos de base de datos desarrollados en la sección de computación del departamento de Ingeniería Eléctrica del CINVESTAV En particular, base de datos geográfica", que consistió en establecer un sistema manejador de base de datos con un diccionario de datos común, aunque con la variante de que el *Sistema TabluSql* no solo funciona con la base de datos del *Sistema de Información Geográfica* sino en forma independiente, interactuando con otra base de datos o con tablas externas (vistas) de otras bases de datos. Las funciones creadas son para consultar la base de datos, pero su sintaxis puede mejorarse para manipulación de la base de datos con funciones de inserción, eliminación y de actualización de registros.

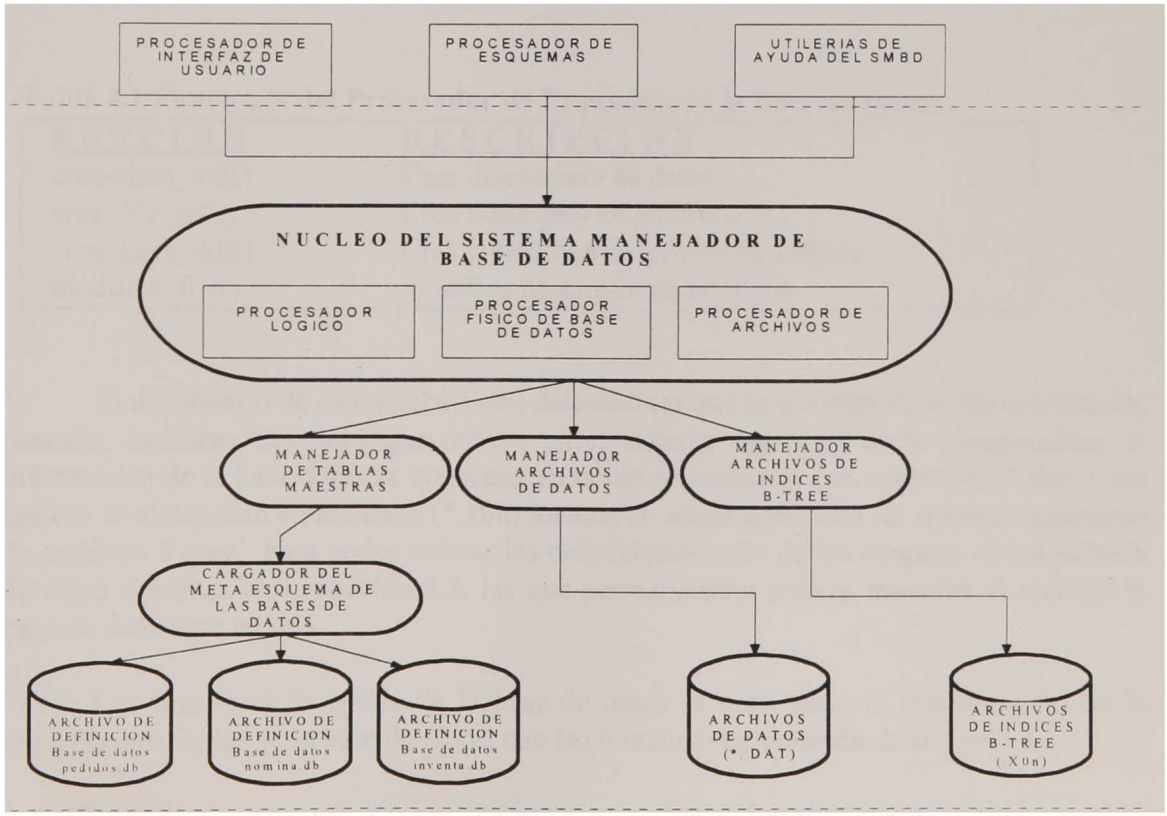


Figura 4.1 Arquitectura del Sistema Manejador de Bases de Datos

4.1 Funciones y Estructuras de Datos del Sistema Manejador de la Base de Datos

El software que maneja las rutinas de almacenamiento y recuperación de datos de una base de datos, es llamado sistema manejador de la base de datos o *SMBD*. Un Sistema Manejador de la Base de Datos implementa el diseño de los esquemas, los vínculos, las estructuras de datos, el tamaño de los elementos de almacenamiento, de los índices, las rutas de acceso y los algoritmos de ejecución y mantenimiento. Además establece las estructuras de la base de datos *-en base al modelo conceptual y lógico-* y realiza operaciones de almacenamiento y recuperación de datos vía lenguajes como: Definición de Datos o *LDD*; de Manipulación de Datos o *LMD* (por ejemplo: adicionar, modificar, y eliminar); y Lenguaje de Consulta.

El procesador de esquemas se utiliza para la definición y mantenimiento de la base de datos, vía el Lenguaje de Definición de Datos o *LDD*. Usa las funciones de la Tabla 4.1, para definir las estructuras del descriptor de archivos, los atributos de las tablas y los índices. El procesador de esquemas traduce datos *ldd* en formas internas, reconocidas por el *SMBD*, en la gestión de consultas, utilizando el núcleo del manejador para recuperar, actualizar y almacenar datos de control de la base de datos. Otros procesadores de esquemas pueden definir datos de control de las estructuras lógicas y físicas (como subesquemas, vistas, etc.), definición de restricciones de integridad semántica, de control de acceso y otra información determinada durante el proceso del diseño.

Tabla 4.1 Funciones del Procesador de Esquemas de la Base de Datos.

FUNCION	DESCRIPCION
crea-elem_ddl()	Crea diccionario de datos
crea_file_ddl()	Crea descriptor de archivos
crea_keys_ddl()	Crea y modifica descriptor de índices
modifica_file()	Modifica descriptor de archivos

El diccionario de datos trabaja con definiciones que se guardan en archivos (base.db, base.dic, base.inx, base.sch) que residen en el sistema operativo de la computadora, la información de la base datos se almacena en tablas relacionales con extensión (*.dat) y los índices se almacenan en archivos (*.x0n) los que se administran con un sistema manejador de archivos *B-tree*. Para poder utilizar las definiciones estas deben cargarse en estructuras de datos descritas en la sección 4.2, las que permanecerán activas mientras el sistema la base de datos este en uso.

Las funciones de ayuda de la base de datos se usan tener el conocimiento de la organización lógica de la base de datos, con las funciones de consulta de la Tabla 4.2

Tabla 4.2 Funciones de Ayuda del Administrador de la Base de Datos.

FUNCIÓN	DESCRIPCIÓN
help_diccionario()	Consulta diccionario
help_file()	Consulta la definición de las tablas
help_elem_keys()	Consulta índices
index()	Indexa la base de datos
init()	Inicializa la base de datos
indexfile()	Indexa una tabla de la base datos

Se adicionan también funciones de mantenimiento, para Cargar, Guardar, Cerrar e Imprimir la estructura de la base de datos como:

FUNCIÓN	DESCRIPCIÓN
carga_base_datos()	Carga una base de datos
guarda_base_datos()	Guarda la base de datos actual
nueva_base_datos()	Crea una nueva base de datos
cierra_base_datos()	Cierra la base de datos actual
imp_base_datos()	Imprime la definición de la base de datos

En el Núcleo Manejador se tienen las funciones para recuperar, almacenar las aplicaciones, y llevar el control de los datos almacenados en el sistema, manejando las peticiones de los usuarios a través del lenguaje de consulta con las funciones de la Tabla 4.3. Proporciona una interfaz al usuario y al procesador de esquemas. La interfaz del núcleo conectada a la interfaz de usuario, da facilidades para la manipulación de estructuras de datos lógicas, definidas por el SMBD del modelo de datos.

Tabla 4.3 Funciones del Sistema Manejador de la Base de Datos.

FUNCIÓN	DESCRIPCIÓN
Busca_reg (f, k, key, buffer)	Busca un registro
Ant_reg (f, k, buffer)	Busca el registro anterior
Sig_reg (f, k, buffer)	Busca el registro siguiente
Prim_reg (f, k, buffer)	Busca el primer registro
Ulti_reg (f, k, buffer)	Busca el último registro
Eli_reg (f)	Borra un registro
Add_reg (f, buffer)	Adiciona un registro
Verifi_reg (f, k, key)	Verifica si existe el registro

El Núcleo Manejador traduce las peticiones, en términos de esta interfaz lógica en comandos, usando las descripciones de la base de datos definida en el procesador de esquemas. Conceptualmente el núcleo del manejador de la base de datos consiste de tres componentes: El Procesador Lógico de la Base de Datos que traduce operaciones de la base de datos, expresadas en términos del modelo de datos relacional lógico, en operaciones

físicas sobre las estructuras de almacenamiento; El Procesador Físico de la Base de Datos que traduce operaciones físicas sobre estructuras de almacenamiento de la base de datos, en operaciones sobre archivos; Y el Procesador de Archivos, se refiere a las operaciones dadas por el sistema de archivos. Estas funciones incluyen varios métodos de ordenamiento y acceso a los sistemas de archivos del sistema operativo. El proceso de actualización de datos que proporciona el manejador de archivos para recuperación y borrado de registros usa las funciones de la Tabla 4.4. Las cuales utilizan el registro "lógico" (*o número de registro*) como argumento.

Tabla 4.4 Funciones del Procesador Físico de la Base de Datos.

FUNCIÓN	DESCRIPCIÓN
file_create(nombre, len)	Crea archivo
open_file(nombre)	Abre archivo
close_file(fp)	Cierra archivo
crea_reg(fp,bf)	Crea registro
get_record(fp,regno,bf)	Recupera registro
rewrite_record(fp,regno,bf)	Reescribe registro
delete_record(fp,regno)	Borra registro

Los archivos de datos se componen de un encabezado y una colección de registros formados mediante un algoritmo de listas. Cada uno de los registros tiene en sus extremos dos campos adicionales que guardan una dirección al siguiente y anterior registro como se muestra en la figura.

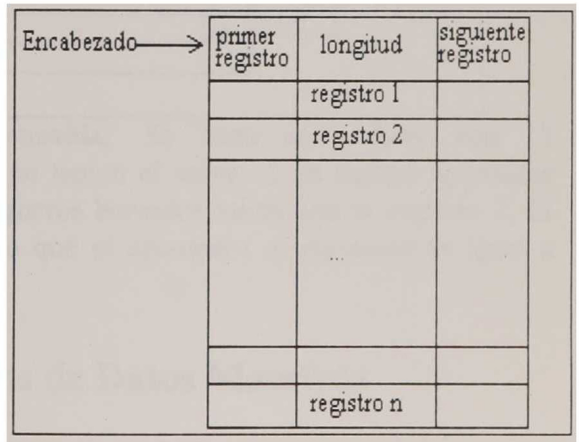


Figura 4.2 Organización de los archivos de datos.

Los archivos de datos usan un encabezado con tres valores importantes:

1. **Primer_reg**.- Guarda el número de registro inicial o actual en la lista (del espacio reusable o disponible). El valor es 0 si no hay registros borrados, en otro caso contiene el número del registro con que comienza la lista.
2. **Sig_reg**.- Da el número de registro a utilizar para insertar un registro en caso que la lista de disponibles esté vacía.
3. **Lon_reg**.- Da el tamaño en bytes del registro de longitud fija.

Las inserciones y borrados de registros, son como en las listas ligadas con encabezado HEAD (Figura 4.3). Estas se controlan en los extremos de la lista HEAD, situada en el encabezado (HEADER) del archivo. La dirección de un registro se considera un entero relativo a la posición que el registro tiene en el archivo y tomando en cuenta la longitud fija de este. El primer registro es el número 1, el segundo registro es el número 2, y así sucesivamente. La posición del primer carácter de un registro en el archivo se calcula con el número de registro menos uno, por la longitud del registro, mas la longitud en bytes del encabezado (del archivo). Este número da la posición del primer carácter (del registro referido), relativo al cero (que es el inicio del archivo) y este valor es utilizado por

Cuando un registro es borrado se le escribe una marca (o bandera) de borrado **-1,** y el número del registro que encabeza la lista de borrados se actualiza (***utilizando la variable primer_reg*),** con el número del registro actualmente borrado. La finalidad de este método es reutilizar los espacios disponibles que dejan los registros borrados. Cuando se quiere agregar un registro, se verifica si hay espacio disponible en los registros borrados, si es así, éste es usado para el nuevo registro. En caso contrario, el registro se agrega al final del archivo (***utilizando la variable sig_reg*),** y se actualiza la lista ligada.

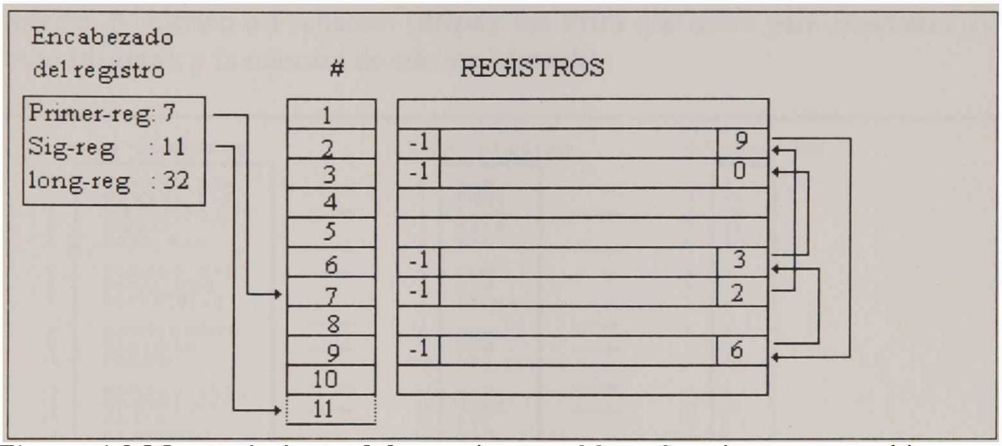


Figura 4.3 Mantenimiento del espacio reusable. Se tiene un archivo con 11 registros, los registros 2, 3, 6, 7 y 9 que tienen el valor –1 en campo apuntador primer-reg están borrados. La lista de registros borrados inicia con el registro 7. El registro 3 es el último de la lista, por lo que el apuntador al siguiente es igual a NULL.

4.2 Interrelación de las Estructuras de Datos Maestras

La definición de la base de datos o descriptor de archivos es contenida en una serie de estructuras de datos, las estructuras de datos se componen de los arreglos: *filename[]* que guarda los nombres de las tablas; *fileele[][]* y *ndxele[][][]* usadas para guardar los elementos y los índices de cada tabla respectivamente, y *dc[].name* que contiene los nombres de los campos y otros atributos como tamaño y tipo.

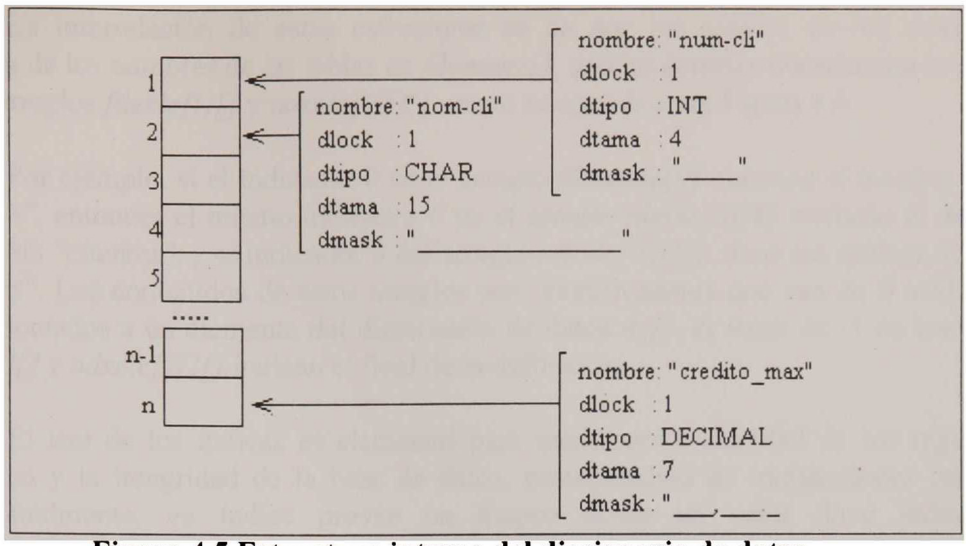


Figura 4.5 Estructura interna del diccionario de datos

En el diccionario de datos (Figura 4.5), se definen el universo de los elementos de la base de datos, cada elemento contiene los atributos: nombre (**dname**); el tipo -**A**lfanumérico, **N**úmerico o **F**echador- (**dtipo**); los bytes que usará para almacenar el valor del atributo (**dtama**); y la mascara de edición (**dmask**).

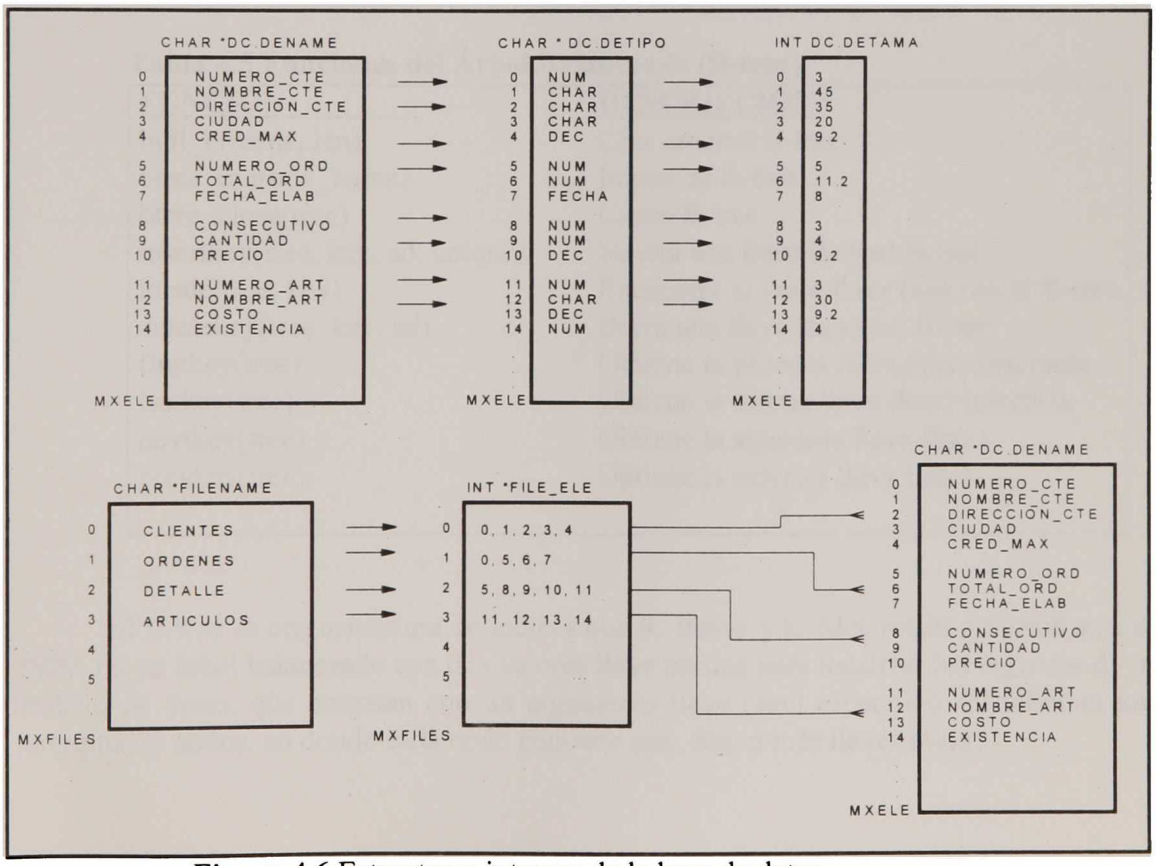


Figura 4.6 Estructuras internas de la base de datos.

La interrelación de estas estructuras se da por los índices en los arreglos. La posición de los nombres de las tablas en *filename[]*, está en correspondencia con la posición de los arreglos *fileele[i][]* y *ndxele[i][][]*, como se aprecia en la Figura 4.6.

Por ejemplo, si el indicador *0* en el arreglo *filename[0]* contiene el nombre de tabla *"clientes"*, entonces el mismo indicador *0* en el arreglo *fileele[0][k]* contiene el descriptor de la tabla *"clientes"*, y el indicador *0* del arreglo *ndxele[0][][]*, tiene los índices de la tabla *"clientes"*. Los contenidos de estos arreglos son identificadores que van de 0 a MXELE y están asociados a un elemento del diccionario de datos *dc[]*, el valor de -1 en los arreglos *fileele[][] y ndxele[][][]* indican el final de la definición,

El uso de los índices es elemental para mantener la unicidad de los registros, la seguridad y la integridad de la base de datos, protegiéndola de transacciones impropias. Conceptualmente, un índice provee un mapeo desde un valor llave indexado (o combinación de valores), a un (conjunto de) apuntador(es) al registro almacenado. El manejo de los índices es controlado por el SMBD mediante un grupo de funciones que manejan la base de datos, con archivos de índices B-tree. Las funciones del B-tree crean y buscan datos en el B-tree, adicionan y borran llaves en el B-tree, y navegan en el B-tree en secuencia de llaves ascendentes y descendentes. Estas funciones se muestran en la Tabla 4.5 y fueron implementadas, con mejoras y adiciones, de las que aparecen en Al Stevens [Stevens 1987].

Tabla 4.5 Funciones del Árbol Balanceado (B-tree)

FUNCIÓN	DESCRIPCIÓN
buil_b(name, len)	Crea archivo B-tree
btree_init(ndx_name)	Inicializa B-tree
btree_close(tree)	Cierra B-tree
insertkey(tree, key, ad, unique)	Inserta una llave (key)al B-tree
locate(tree, key)	Encuentra el valor llave (key) en el B-tree
deletekey(tree, key, ad)	Borra una llave (key) del B-tree
firstkey(tree)	Obtiene la primera llave (key) insertada
lastkey(tree)	Obtiene la última llave (key) insertada
nextkey(tree)	Obtiene la siguiente llave (key)
prevkey(tree)	Obtiene la anterior llave (key)

El B-tree es una estructura de índices que R. Bayer y E. McCreight desarrollaron en 1970. Es un árbol balanceado con dos valores llave usados para localizar los registros de un archivo de datos, que empatan con un argumento llave (*key*) específico. El árbol es una jerarquía de nodos, en donde cada nodo contiene una, dos, o más llaves (*keys*).

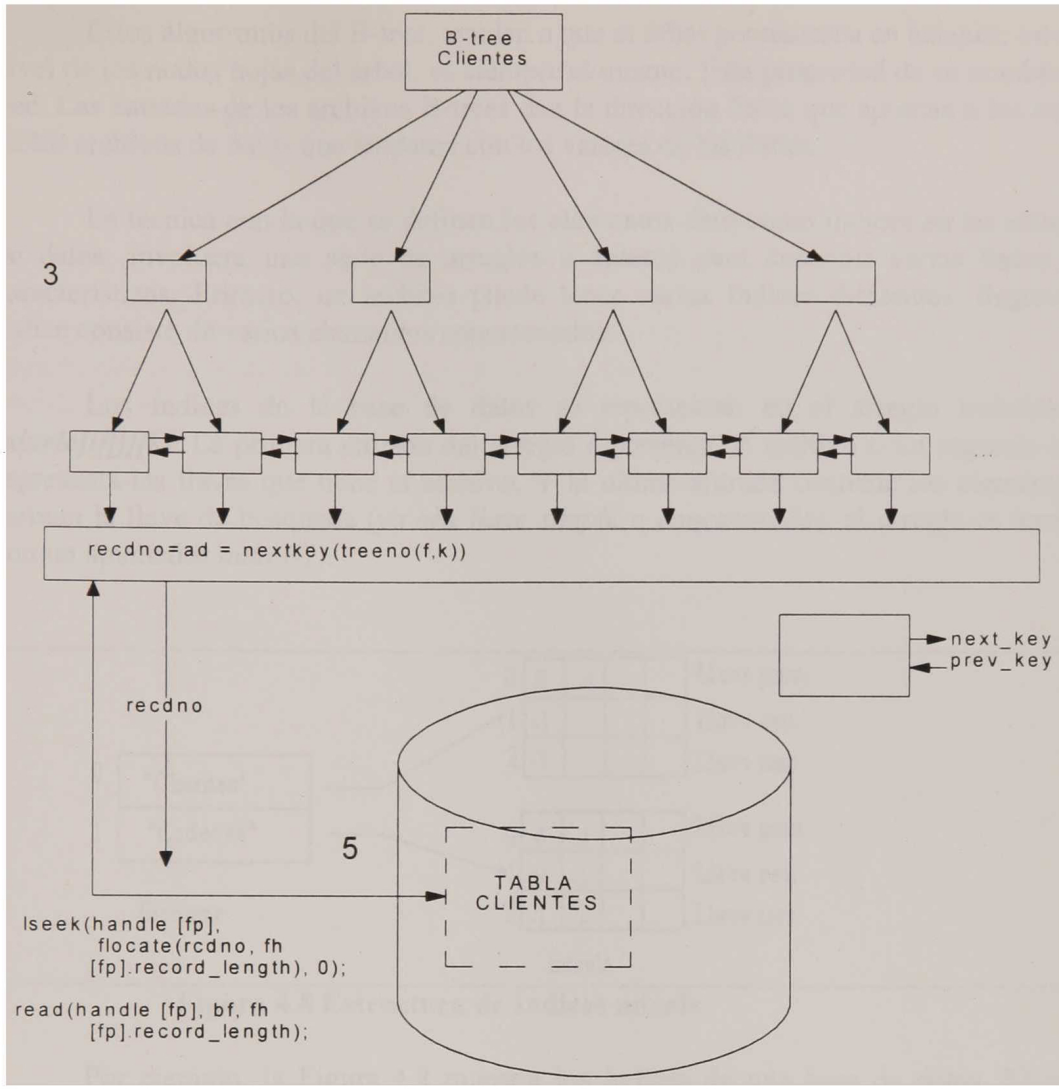


Figura 4.7 Estructura de los archivos B-tree.

Un B-tree consiste de un nodo raíz y dos o más nodos inferiores. Si el número total de llaves (*keys*) en el árbol, es igual o menor al número que un nodo puede contener, solo el nodo raíz existe. Cuando este número excede la capacidad del nodo, el nodo raíz se parte en dos nodos inferiores (hojas), reteniendo la llave (*key*) que esta lógicamente entre los valores de la llave (*key*) de los dos nuevos nodos. Los nodos son padres de los nodos inferiores. Los nodos almacenan en la secuencia del valor de la llave. Cuando el árbol tiene múltiples niveles, cada llave en el nodo padre apunta al nodo inferior que contiene una llave con valor mayor que la llave padre y menor que la llave adyacente, en el padre. Los nodos en los niveles más inferiores, son llamados hojas. Las llaves en un nodo hoja apuntan a los registros de los archivos que empatan con los valores indizados.

Estos algoritmos del B-tree, ayudan a que el árbol permanezca en balance: esto es, el nivel de los nodos hojas del árbol, es siempre el mismo. Esta propiedad da su nombre de B-tree. Las entradas de los archivos B-trees dan la dirección física que apuntan a los registros de los archivos de datos que empatan con los valores de las llaves.

La técnica con la que se definen los elementos dato como índices en las estructuras de datos, involucra una serie de arreglos a enteros para describir varias llaves y sus características. Primero, un archivo puede tener varios índices diferentes. Segundo, un índice consiste de varios elementos concatenados.

Los índices de la base de datos se representan en el arreglo tridimensional ***ndxele[i][j][k]***. La primera entrada del arreglo representa un archivo *i*. La segunda entrada representa las llaves que tiene el archivo. Y la ultima entrada contiene los elementos que forman la llave de búsqueda (*ya sea llave simple o concatenada*), el arreglo es terminado con un apuntador *nulo (-1)*.

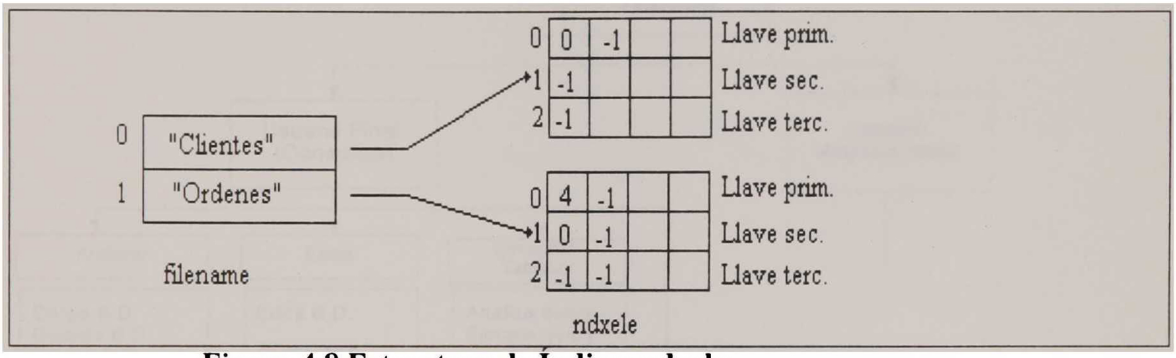


Figura 4.8 Estructura de Índices ndxele

Por ejemplo, la Figura 4.8 muestra los índices de una base de datos. El archivo ***"clientes"*** representado por el índice *0* en el arreglo *ndxele[0]*, contiene una llave (indicada en *0)* compuesta por el elemento *"numcli" (0)*, las otras llaves están sin uso. El archivo de ***"ordenes"*** tiene dos llaves: la primera llave es *"numorden" (4)* y la segunda *"numcli" (0)* respectivamente. Con esta configuración se puede saber que ordenes tiene un cliente usando la llave secundaria, y que cliente tiene la orden ***n*** usando la llave primaria. Esto permite que el procesador del lenguaje determine cual llave usar para accesar una tabla de datos.

En este capítulo hemos descrito las principales funciones del sistema manejador de la base de datos y las estructuras de datos que contienen la definición interna del descriptor de archivos y del diccionario de datos. El uso de las funciones del manejador de archivos b-tree nos ofrece un mecanismo de consulta de datos más rápido al considerar la navegación de llaves primarias, secundarias, terciarias, cuaternarias usadas generalmente en una operación equi-junta entre dos o más tablas-archivo. El concepto de descriptor de datos viene a ser una de las piezas fundamentales. La implementación dinámica de estas estructuras maestras, la creación, y manipulación se realizaron en el lenguaje "C".

Capítulo 5

Implementación de una Aplicación de Base de Datos

En este capítulo presentamos el diseño, la implementación, los resultados obtenidos de una base de datos y el funcionamiento del Sistema de Consulta TabluSql. La interfaz del sistema tiene dos niveles de usuarios: ***usuario administrador*** que crea la base de datos, define tablas, atributos de campos, e índices; Y ***usuario final*** que carga una base de datos, imprime el contenido de las tablas, actualiza datos con funciones de insertar, borrar y modificar, consulta la información de la base de datos en SQL y QBE, consulta tablas externas (vistas) definidas por el administrador, y consulta la definición de la base de datos (diccionario).

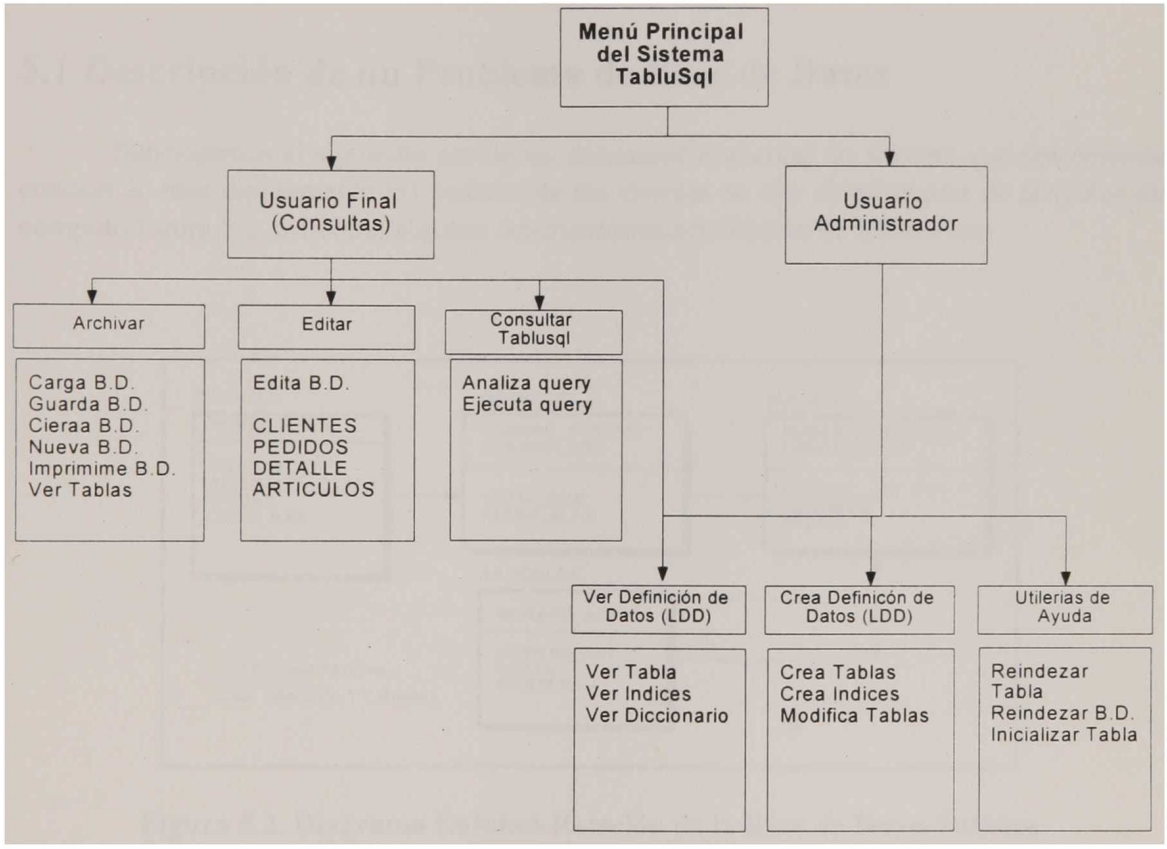


Figura 5.1 Menú principal del sistema TabluSql

El sistema desarrollado, se encuentra dividido en cuatro partes:

- **Lenguaje de Definición de Datos** (LDD) da el medio para definir diccionario, esquemas, subesquemas, e índices.

- **Lenguaje de Manipulación de Datos** (LMD) se usa para escribir aplicaciones de bases de datos (funciones para borrar, adicionar, modificar registros).
- **Lenguaje de Consulta** (TABLUSQL) usado para escribir consultas y reportes. Muchos lenguajes de bases de datos combinan consultas, manipulación, y definiciones de subesquemas. Sin embargo sólo usamos las consultas del lenguaje SQL (Structured Query Language).

- **Utilerias** utilizadas para el apoyo de la base de datos como: cargar y salvar la base de datos, desplegar esquemas, índices y diccionario de datos, inicializar la base de datos, reindezar la base de datos, etc...

5.1 Descripción de un Problema de Base de Datos

Supongamos el siguiente problema: deseamos implantar un sistema que nos permita conocer lo mas rápidamente los pedidos de los clientes en una distribuidora de artículos de computo figura 5.2, así como algunas descripciones adicionales de los clientes.

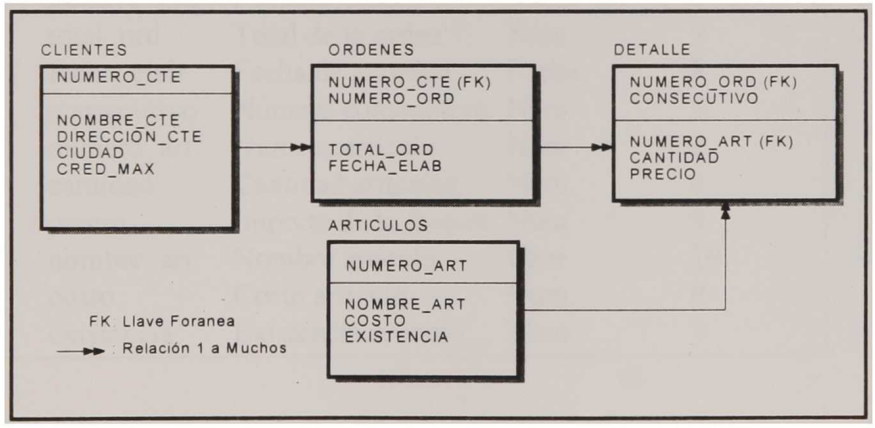


Figura 5.2. Diagrama Entidad-Relación de la Base de Datos Pedidos

El diseño se forma de los siguientes esquemas de relación:

Tablas	Elementos
CLIENTES	(numero_cte, nombre_cte, direccion_cte, ciudad, cred_max)
ORDENES	(numero_cte, numero_ord, total_ord, fecha_elab)

DETALLE	(numero_ord, consecutivo, cantidad, precio)
ARTICULOS	(numero_art, nombre_art, costo, existencia)

Los elementos llave de cada relación son:

Tablas	LLaves	Elementos
CLIENTES	1	(numero_cte)
ORDENES	1	(numero_cte, numero_ord)
	2	(numero_cte)
DETALLE	1	(numero_ord, consecutivo)
DETALLE	2	(numero_ord, numero_art)
ARTICULOS	1	(numero_art)

El diccionario de datos tiene los siguientes elementos:

Nombre	Descripción	Tipo	Tamaño
numero_cte	Número cliente	Num	3
nombre_cte	Nombre cliente	Char	20
direccion_cte	Dirección cliente	Char	20
ciudad	Ciudad cliente	Char	20
cred_max	Credito máximo	Num	9
numero_ord	Número orden	Num	3
total_ord	Total de la orden	Num	9
fecha_elab	Fecha de la orden	Fecha	8
consecutivo	Número consecutivo	Num	3
numero_art	Número artículo	Num	3
cantidad	Cantidad artículos	Num	4
precio	Importe de la factura	Num	9
nombre_art	Nombre artículo	Char	20
costo	Costo artículo	Num	9
existencia	Existencia artículo	Num	4

La tabla (CLIENTES - ORDENES) nos indica que un cliente puede hacer varias órdenes y a cada orden le corresponde un cliente. Es decir un mismo número de orden no puede tener dos clientes.

Una orden puede tener varias líneas de detalles (artículos diferentes), y cada línea de detalle tiene la identificación de la orden a la que pertenece.

Cada línea de detalle (consecutivo) del archivo DETALLE tiene un solo artículo y cada artículo corresponde a un número de detalle.

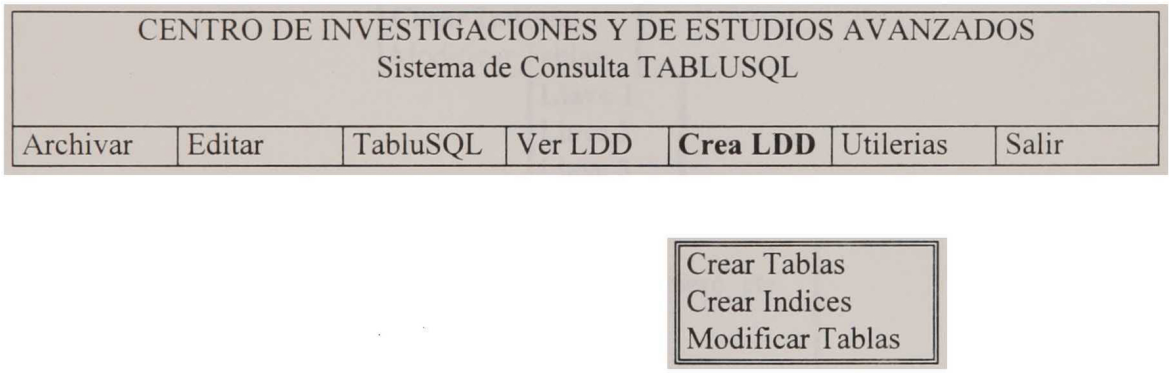
El diseño determina:

a). Todas las órdenes realizados por un cliente
b) Para una orden, la información correspondiente del cliente.
c) Para una orden, los artículos ordenados y la información de cada uno de ellos.
d) Para un artículo, las órdenes que lo contienen y los clientes que hicieron la orden.

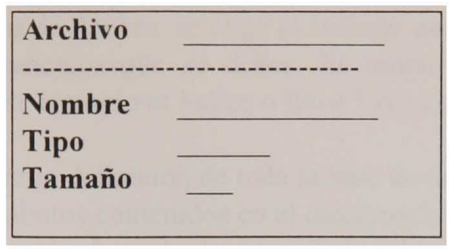
5.2 Implementación del Problema

El menú principal presenta cada uno de los lenguajes y sus diferentes opciones, que iremos utilizando. Los términos tabla y relación serán usados indistintamente en el presente documento.

Utilizando la opción *Crea LDD* del menú principal, definiremos la base de datos. Creando primero la tabla con sus atributos y posteriormente sus índices. Usaremos la tecla <ESC> para deshacer la operación, o la tecla <ENTER> para aceptar la operación



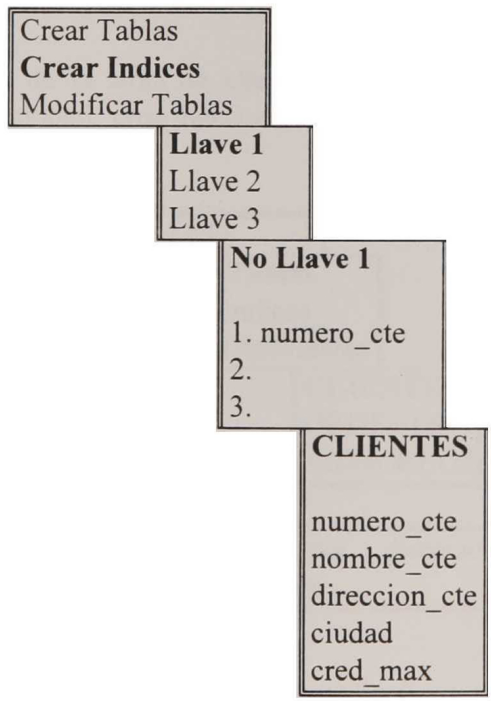
Al seleccionar *Crear Tablas* aparecerá el siguiente cuadro de dialogo para definir la tabla



Los campos utilizados en esta opción son:

Archivo: Se usa para dar el nombre a la tabla. Usa hasta 8 caracteres alfanuméricos.
Nombre: Se usa para dar los elementos (campo) de la tabla.Utiliza 8 caracteres.
Tipo: Se usa para dar el tipo del elemento. (E)ntero, (A)alfanumérico o (F)echa
Tamaño: Se usa para dar el tamaño que ocupara el elemento.

Para dar los índices a las tablas definidas anteriormente, se usa la opción *Crear Indices* del Menú *Crea LDD*. Esto dará la lista de todas las tablas definidas en la base de datos. Aquí se selecciona la tabla de una lista ya definida y se seleccionan los elementos llave, tal y como se muestra a continuación.



Con las teclas de flechas, se elige el atributo que corresponde al primer índice (llave 1) o al segundo índice, según se desee. El límite es hasta 4 índices con 4 campos concatenados. En este ejemplo el índice o llave 1 contiene el atributo numero_cte.

Para consultar la definición de toda la base de datos, ya sea las tablas, sus llaves, o la definición de sus atributos contenidos en el diccionario de datos usamos la opción *Ver LDD* y elegimos la sub-opción deseada. Los resultados de escoger estas opciones se presentan a continuación, partiendo del menú principal:

CENTRO DE INVESTIGACIONES Y DE ESTUDIOS AVANZADOS
Sistema de Consulta TABLUSQL

Archivar	Editar	TabluSQL	**Ver LDD**	Crea LDD	Utilerias	Salir

Ver Tablas
Ver Indices
Ver Diccionario

B.D: ORDENESS Desarrolló : José Alejandro Martínez Hernández

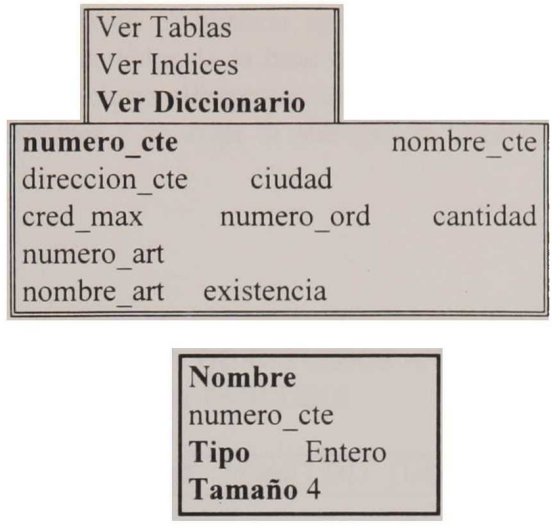
La consulta a la definición de la tabla de clientes, mostrará los atributos (numero_cte, nombre_cte, direccion_cte, ciudad, cred_max) como en el cuadro al final de la siguiente figura.

Ver LDD
Ver Tablas
Ver Indices
Ver Diccionario
CLIENTES
ORDENES
ARTICULOS

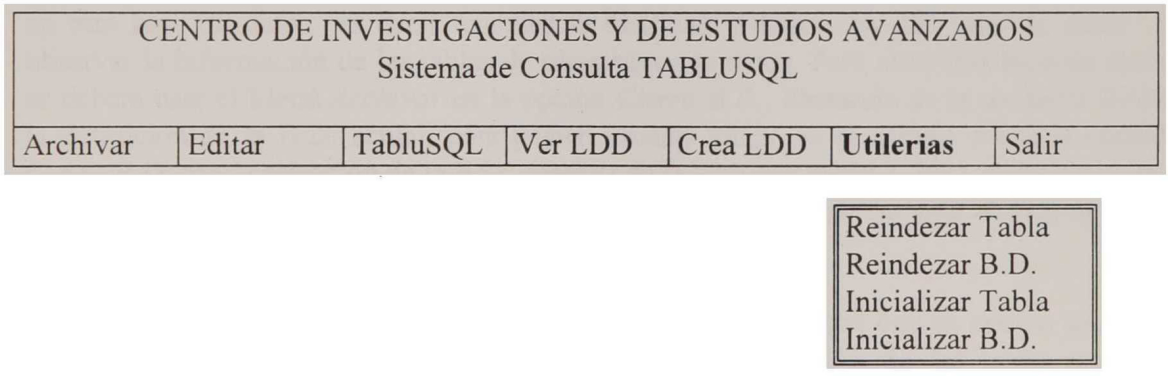
numero_cte nombre_cte direccion_cte
ciudad cred_max

La consulta al diccionario de la base de datos muestra el siguiente resultado. Al seleccionar el atributo *numero_cte* se observa:

Ver LDD

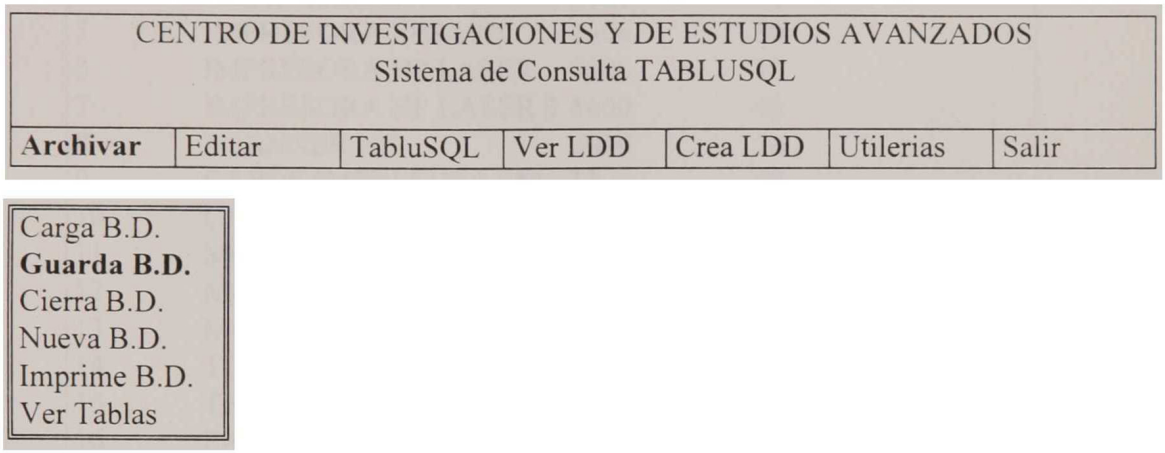


Una vez definidas todas las tablas de la base de datos y sus índices, el usuario debe proceder a crear la definición de las tablas, en un directorio de la computadora con cero registros. La creación de la tabla en el directorio elegido, se da por medio de un proceso que genera, para cada tabla definida en el LDD, una con el mismo nombre y con extensión.dat. También se generan otros archivos con el mismo nombre pero con extensiones (*.x0n), (donde n=1,5), según sean las llaves definidas a la tabla. Este proceso se ejecuta desde el Menú *Utilerias* en la opción *Inicializa la B.D.* Este proceso también se realiza al seleccionar algunas tablas del LDD, ya sea al inicializar, o al reconstruir los índices de la base de datos, proceso que preserva la información existente.



Para proteger la definición de toda la base de datos, se debe guardar con un nombre que no exceda del limite especificado por el sistema operativo. Como en nuestro caso

contamos con un sistema desarrollado en el sistema operativo DOS (Disk Operating System), compatible con IBM, tanto las tablas de la base de datos como el nombre de la base de datos no debe exceder de 8 caracteres alfanuméricos. Para realizar el respaldo de la base de datos se usa el Menú *Archivar* y se elige la sub-opción *Guarda B.D* como se muestra a continuación.



El sistema permite trabajar con diferentes bases de datos, entendiéndose que una base de datos consta de una colección de tablas e interactuar además con una tabla definida en otra base de datos. Es decir tenemos la facilidad trabajar con una base de datos y observar la información de las tablas de otras bases de datos. Para abrir otra base de datos se deberá usar el Menú *Archivar* en la opción *Cierra B.D.*, liberando de la memoria RAM la definición de la B.D. actual, para posteriormente elegir en el mismo menú la opción *Carga B.D.* con la selección de la B.D. deseada de la lista que aparece. Para poder consultar la información de una tabla foránea (de otra base de datos que no está activa) se debe agregar la definición de la tabla en la base de datos activa.

La definición de la base de datos activa puede imprimirse con la opción *Imprime BD.* del Menú *Archivar*. Esto da como resultado la impresión de las tablas con sus atributos, sus índices y la definición de sus elementos agrupados por nombre, tipo y tamaño. Finalmente la opción *Ver Tablas* despliega los registros de cada tabla seleccionada de la base de datos activa.

El resultado de solicitar la información que contiene la tabla ARTICULO de la base de datos de ORDENESS, daría el siguiente resultado.

Nombre Archivo: articulo

num_art	nombre_art	costo	existencia
1	COMP PC 286	2500	20
2	COMP PC 386 SX	5000	40
3	COMP PC 386 DX	6000	20
4	COMP 486 DX	7000	10
5	IMPRESORA EPS 1050	1300	50
6	IMPRESORA HP LASER	7000	30
7	IMPRESORA HP LASER 3	5600	40
8	SCANNER HP	6000	50
9	CABLE IMPRESORA 1M	12	50
10	CABLE IMPRESORA 3M	35	10
11	MONITOR COLOR VGA	1200	40
12	MONITOR COLOR SVGA	3000	30
13	MONITOR TTL	500	30
14	TERMINAL WY150	790	30
15	TERMINAL WYSE 30	550	3
16	PAPEL STOCK 8.5 X 11	130	90
17	CINTA EPSON LLG	23	100
18	PAPEL STOCK 15x20	26	20
19	MODEM HP M/4773	350	20
20	WINWORD 2.0	650	3

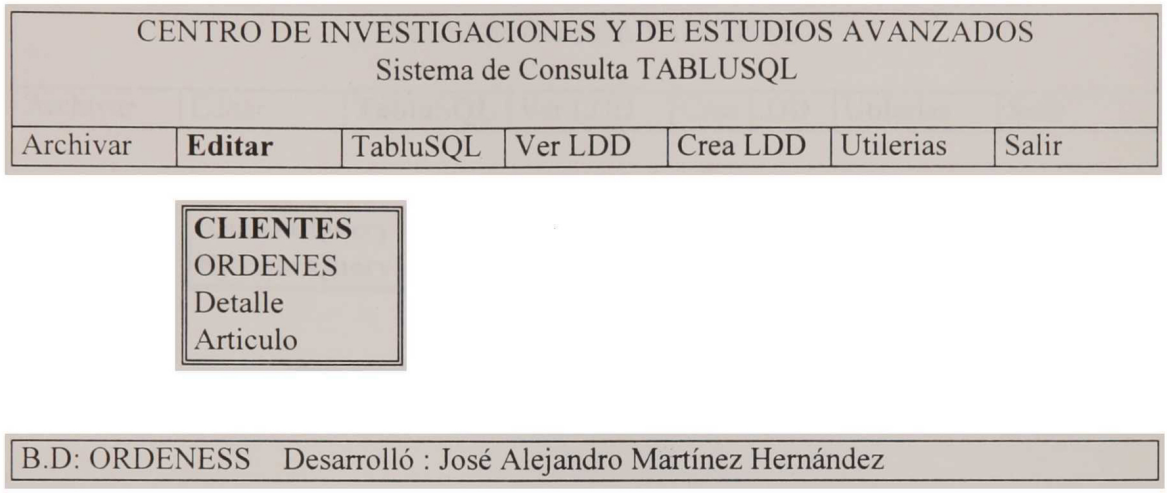
<Enter> para continuar...
Mostrando todos los elementos que alcance el monitor, los cuales no deben de exceder los 80 caracteres.

5.3 Resultados

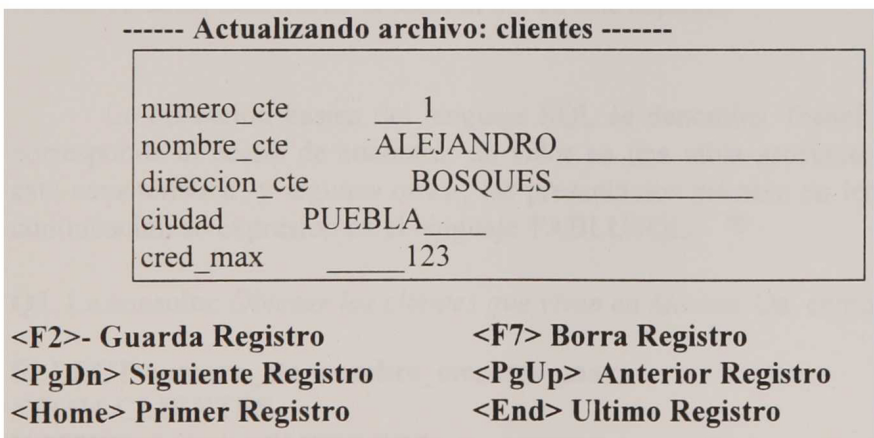
El sistema tiene como característica facilitar consultas de una tupla y de múltiples tuplas o registros a la vez. Facilitar la manipulación de datos con funciones de actualización para insertar, borrar y modificar una tupla a la vez. De esta manera para consultar nuestra base de datos necesitamos primero agregar información a nuestras tablas. Para realizar consultas en el nivel del usuario final solo tiene que conocer el diseño conceptual de la base de datos (esquema lógico) y no la forma en que almacenan.

Para adicionar información a una base de datos, elija del Menú *Editar* la(s) tabla(s) CLIENTES, ORDENES, DETALLE y ARTICULO, tomando en consideración el modelo

entidad-vínculo. Es decir que no puede existir un ORDENES sin tener antes un catálogo de clientes, de igual forma no puede registrarse un ORDENES sin la existencia de un catálogo de artículos. De hecho el SMBD protege la consistencia de la base de datos y evita tales situaciones.

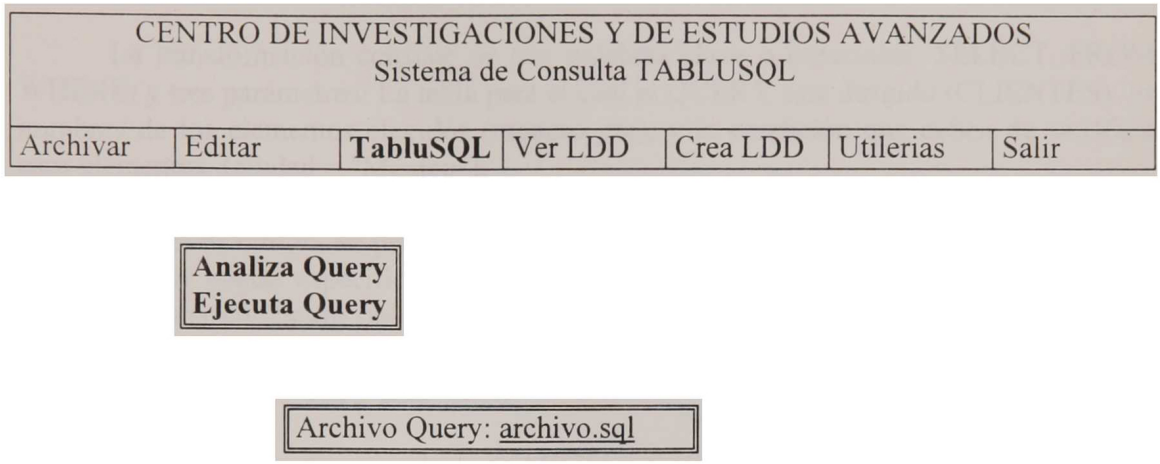


La figura anterior ilustra la selección de la tabla CLIENTES. La función de edición presentará los elementos que componen esta tabla. Se deben usar los espacios subrayados para agregar información. Existen también comandos para guardar la información del registro, para borrar o adicionar. A continuación se ilustra lo antes mencionado.



Las consultas en el lenguaje de consulta TABLUSQL, deberán ser construidas por medio de un editor de textos ASCII, ya que el sistema no contiene esa función. Para elaborar nuestras consultas hemos utilizado editores de textos.

Para que una consulta TABLUSQL sea ejecutada, se deben efectuar dos pasos: el primero **Analiza Query** solicita el archivo que contiene la expresión SQL y evalúa la expresión por medio del autómata descrito en el capitulo 3. En el segundo paso, **Ejecuta Query** se extrae la información requerida, aplicando el algoritmo del calculo de la consulta del capítulo 3. Las opciones mencionadas se pueden observar en la siguiente pantalla donde se resaltan las opciones en letras negrillas.



En el presente trabajo, únicamente implantamos las facilidades de Consulta de Datos, una descripción completa del lenguaje SQL aparece en [Chamberlin 1974][Lasardif 1987]. Las facilidades de TABLUSQL se presentan como una serie de ejemplos asociados a la base de datos descrita en la sección 5.2 de este capítulo.

La operación básica del lenguaje SQL se denomina Transformación (Mapping), y corresponde al hecho de encontrar un valor en una tabla asociado con otro. Para ilustrar esta característica -y algunas otras-, las presentamos primero en lenguaje natural **Qn,** y a continuación su expresión en el lenguaje TABLUSQL.

Q1. La consulta: *Obtener los clientes que viven en México*, Da como resultado:

```
SELECT numero_cte, nombre_cte, cred_max
FROM CLIENTES
WHERE ciudad = "MEXICO"
```

num_cte	nombre_cte	cred_max
8	ROBERTO	4373
9	RAQUEL B.	23882

14	RAMON	3990
15	ALICIA	3828
16-	PATRICIA	327
17	LETICIA	3834
22	MARCELINO	324
23	RAUL QUINT	323
24	LUISA MEZA	43277

La transformación consiste de tres palabras clave o especiales: SELECT, FROM, WHERE; y tres parámetros: La tabla para el cual el QUERY esta dirigido (CLIENTES), los nombres de los elementos elegidos (numero_cte), y la condición que deben de satisfacer esos elementos (ciudad = "México").

En TABLUSQL, se permite que el "mapping" básico se extienda en varias formas. Una consulta puede especificar más de un elemento a ser seleccionado, y así mismo especificar una condición booleana compleja en la cláusula WHERE, como se ilustra en Q2. También se ilustra la proyección de la relación CLIENTES sobre los atributos (numero_cte, nombre_cte, y cred_max).

Q2. *Obtenga los clientes que viven en México y tienen un crédito mayor a 1000*, daría el siguiente resultado.

SELECT numero_cte, nombre_cte, cred_max
FROM CLIENTES
WHERE cred_max > 1000 and ciudad = "MEXICO"

numero_cte	nombre_cte	cred_max
8	ROBERTO	4373
9	RAQUEL B.	23882
14	RAMON	3990
15	ALICIA	3828
17	LETICIA	3834
24	LUISA MEZA	43277

Si la cláusula WHERE se omite y se quieren solicitar todos los elementos de la tabla, la consulta Q3 regresa todos los valores de la tabla seleccionada.

Q3. *Obtenga toda la información de clientes*

SELECT alls
FROM CLIENTES

numero_cte	nombre_cte	direccion_cte	ciudad	cred_max
1	ALEJANDRO	BOSQUES	PUEBLA	123

2	RAQUEL	BOSQUES	PUEBLA	456
3	SARA	VOLCANES	PUEBLA	435
4	ARMANDO	VOLCANES	PUEBLA	594
5	GABRIELA	VOLCANES	PUEBLA	3883
6	SAMUEL	BOSQUES	PUEBLA	3273
7	GERARDO	VOLCANES	PUEBLA	4838
8	ROBERTO	LAS TORRES	MEXICO	4373
9	RAQUEL B.	LAS TORRES	MEXICO	23882
10	NOE	MAYORAZGO	PUEBLA	282
11	DAMASO	LOMA BELLA	TEHUACAN	454
12	JOSE LUIS	LA ROSA	PUEBLA	434
13	CARLOS	VOLCANES	PUEBLA	4334
14	RAMON	BALBUENA	MEXICO	3990
15	ALICIA	SATELITE	MEXICO	3828
16	PATRICIA	JARDIN BALB.	MEXICO	327
17	LETICIA	AZCATPOZ.	MEXICO	3834
18	MARIA LUISA	VOLCANES	PUEBLA	343
19	MARIO	BOSQUES	PUEBLA	433
20	ANTONIA	LOS ANDES	TEHUACAN	4384

numero_cte	nombre_cte	direccion_cte	ciudad	cred_max
21	JUAN JOSE	HIGO QUEM.	TUXTLA	3237
22	MARCELINO	TACUBAYA	MEXICO	324
23	RAUL QUINT.	LINDA VISTA	MEXICO	323
24	LUISA MEZA	LAS TORRES	MEXICO	43277
25	ROSA ALV.	PINO	CUERNAV.	3484
26	MANOLO	BOSQUES	PUEBLA	3283
27	MARIA DEL C.	VOLCANES	PUEBLA	32747
28	MARIA CONC.	HEROE NACO.	PUEBLA	34
29	MARIA DE LOS	GABRIEL P.	PUEBLA	4377
30	GUILLERMINA	SANTA MAR.	PUEBLA	4734
31	GUADALUPE	LORETO	PUEBLA	3477
32	ANA MARIA	BOSQUES	PUEBLA	43747
33	MARGARITA	LOS ARRO.	CUBA	233
34	ALFREDO	LA AZUCENA	PUEBLA	3474

Q4. La siguiente consulta es igual a Q3 pero sobre la tablas de órdenes. S expresa como *Obtenga la consulta de todas las órdenes*

SELECT alls
FROM ORDENES

numero_cte	numped	tipocred	fecha_elab
1	1	1	01/01/94
1	2	1	01/01/94
1	3	2	01/02/95

6	4	1	05/06/94
15	5	1	27/11/94
22	6	2	25/11/94
23	7	1	24/11/94
17	8	1	01/01/95
24	9	1	01/03/95
17	10	1	01/02/95
8	11	1	01/03/95

En TABLUSQL también se puede solicitar información contenida entre un rango de valores, para ello se deberá utilizar la cláusula BETWEEN.

Q5. La consulta: *Obtener los clientes que ganan entre 1000 y 40000*, daría el siguiente resultado.

```
SELECT numero_cte, nombre_cte, cred_max
FROM CLIENTES
WHERE cred_max BETWEEN 1000 and 4000
```

numero_cte	nombre_cte	cred_max
5	GABRIELA	3883
6	SAMUEL	3273
14	RAMON	3990
15	ALICIA	3828
17	LETICIA	3834
21	JUAN JOSE	3237
25	ROSA ALV.	3484
26	MANOLO	3283
31	GUADALUPE	3477
34	ALFREDO	3474

Para solicitar un conjunto de registros que "empaten" con ciertos valores en TABLUSQL, se utiliza la cláusula IN, como se muestra a continuación:

Q6. *Consultar los nombres de los clientes que tienen las claves 1, 3 y 5.*

```
SELECT numero_cte, nombre_cte, cred_max
FROM CLIENTES
WHERE numero_cte IN (1,3,5)
```

Nombre Archivo: salida.RES

numero_cte	nombre_cte	cred_max

1	ALEJANDRO	123
3	SARA	435
5	GABRIELA	3883

La capacidad de incluir una tabla adicional relacionada con un elemento común en TABLUSQL, se expresa de la siguiente forma:

Q7. *Obtenga los órdenes de todos los clientes*, daría el siguiente resultado

SELECT numero_cte, nombre_cte, ciudad, cred_max, numped, tipocred
FROM CLIENTES, ORDENES

numero_cte	nombre_cte	ciudad	cred_max	numped	tipocred
1	ALEJANDRO	PUEBLA	123	3	2
1	ALEJANDRO	PUEBLA	123	2	1
1	ALEJANDRO	PUEBLA	123	1	1
6	SAMUEL	PUEBLA	273	4	1
8	ROBERTO	MEXICO	4373	11	1
15	ALICIA	MEXICO	3828	5	1
17	LETICIA	MEXICO	3834	10	1
17	LETICIA	MEXICO	3834	8	1
22	MARCELINO	MEXICO	324	6	2
23	RAUL QUINT.	MEXICO	323	7	1
24	LUISA MEZA	MEXICO	43277	9	1

También podemos condicionar la salida de la consulta anterior, sobre uno o varios atributos, de las tablas clientes y órdenes, como se muestra en las consultas Q8 y Q9.

Q8. La consulta: *Obtenga los órdenes de los clientes, con crédito del tipo 1*, daría el siguiente resultado.

SELECT numero_cte, nombre_cte, cred_max, numped, tipocred
FROM CLIENTES, ORDENES
WHERE tipocred = 1

numero_cte	nombre_cte	credmx	nump	tipocred
1	ALEJANDRO	123	1	1
1	ALEJANDRO	123	2	1
6	SAMUEL	3273	4	1
15	ALICIA	3828	5	1
23	RAUL QUINT.	323	7	1

17	LETICIA	3834	8	1
24	LUISA MEZA	43277	9	1
17	LETICIA	3834	10	1
8	ROBERTO	4373	11	1

Q9. La consulta: *Obtenga los órdenes de los clientes con crédito del tipo 1 que viven en "México"*, daría el siguiente resultado.

```
SELECT numero_cte, nombre_cte, ciudad, numped, tipocred
FROM CLIENTES, ORDENES
WHERE ciudad = "MEXICO" AND tipocred = 1
```

numero_cte	nombre_cte	ciudad	numped	tipocred
8	ROBERTO	MEXIC	11	1
15	ALICIA	MEXIC	5	1
17	LETICIA	MEXIC	10	1
17	LETICIA	MEXIC	8	1
23	RAUL QUINT.	MEXIC	7	1
24	LUISA MEZA	MEXIC	9	1

Q10. La consulta: *Obtenga los órdenes con fecha de elaboración de aquellos clientes con crédito del tipo 1*, daría el siguiente resultado.

```
SELECT numero_cte, numped, tipocred, fecha_elab
FROM ORDENES
WHERE tipocred = 1
```

numero_cte	numped	tipocred	fecha_elab
1	1	1	01/01/94
1	2	1	01/01/94
6	4	1	05/06/94
15	5	1	27/11/94
23	7	1	24/11/94
17	8	1	01/01/95
24	9	1	01/03/95
17	10	1	01/02/95
8	11	1	01/03/95

En este capítulo hemos utilizado el sistema de consulta TabluSql basado en un prototipo de SQL y procesado con la metodología Tableaux, en donde se realiza la comunicación entre el usuario y la base de datos [Chapa 1985]. Con este sistema proponemos una solución que ofrece al administrador de bases de datos un conjunto de herramientas contenidas dentro del software del sistema manejador de bases de datos

relacional para definir la base de datos, y proporcionamos un lenguaje de consulta, con facilidades para seleccionar datos compartidos. El Lenguaje de consulta TABLUSQL implementado se basa en el modelo relacional al estilo del lenguaje SQL (Structured Query Language). Con este lenguaje intentamos que el usuario de sistemas: obtenga experiencia operacional en su uso, que evalúe la interfaz de diseño, desarrolle y pruebe las operaciones de recuperación de registros.

Conclusiones

En este trabajo hemos realizado una implementación de la metodología tableaux , la cual puede adaptarse también en otros lenguajes de consulta como QBE [Zloff 1977] y SEQUEL [Chamberlin 1976]. La implementación no es complicada ya que la estructura de matriz es sencilla de adaptar e interpretar sobre todo teniendo una organización de estructuras internas de datos como se ha establecido en este trabajo. Basados en la metodología tableaux hemos encontrado un modelo de transformación de consultas no procedurales a consultas procedurales del álgebra relacional con solo los operadores selección, proyección y junta. El propósito de esta transformación se debe a que en un lenguaje de alto nivel no está determinado directamente el modo de realizar operaciones en la base de datos. Sin embargo, apoyados en la metodología tableaux se puede realizar una transformación formal de una expresión basado en sistemas de consulta provenientes del cálculo relacional en una expresión equivalente en tableaux que consiste básicamente en una expresión basado en los tres operadores SPJ. Estos operadores si tienen una implementación natural que se pueden desarrollar en sistemas computacionales para procesamiento de consultas, ya que internamente las relaciones de modelo relacional pueden manipularse en tablas con las operaciones SPJ. Esto puede ser enriquecido con técnicas de acceso con las que se puedan medir el peso de trabajo con la que procesador de la consulta empezará a recuperar datos como algoritmos de B-Tree y Hashing. La estrategia para la ejecución de las consultas basado en reglas de índices usada no es la óptima, ya que nuestro propósito no es el de optimizar el procesamiento, sin embargo, puede ser un trabajo posterior el realizar una optimización.

Hemos de destacar que la intención de tableaux va mas allá de una transformación de sistemas de consulta ya que prevé una etapa siguiente en sistemas de consulta, precisamente la metodología ***Tableaux*** se puede adecuar para que los datos sean almacenados en la representación de una relación universal lo que hace que usuario perciba los datos más amigablemente. Esto tiene la ventaja de permitir al usuario final olvidar los detalles de separar cuales atributos están agrupados en tal o cual esquema de relación. El objetivo es eliminar del usuario final no solo lo concerniente a la organización física, sino también a la organización lógica. Esto trae como consecuencia que el DBMS trabaje más para interpretar consultas y actualizaciones sobre la base de datos, que si el usuario especifica el *query* en términos de la misma base de datos conceptual.

La metodología tableaux, no es un sistema de consulta completo en comparación con el álgebra relacional al considerar solamente los operadores de selección, proyección y junta, sin embargo, sus consultas abarcan un gran número de consultas que pudieran normalmente aparecer en aplicaciones de bases de datos. El tipo de consultas que dejaría de realizar son las que se refieren a encontrar datos de una clase como es el caso del operador relacional de división.

Los resultados de este trabajo que se obtuvieron fueron el poner al usuario en contacto directo con la información almacenada. Se eliminaron los problemas de manejo de archivos de entrada y salida. Se reemplazó el direccionamiento posicional por un direccionamiento asociativo con ayuda del descriptor de archivos, de forma que cada dato en la base de datos puede ser únicamente direccionado por el nombre de la relación, el valor de la llave primaria y el nombre del atributo. El direccionamiento asociativo permite que el sistema determine los detalles de colocación de una nueva pieza de información que será insertada en la base de datos y seleccione rutas de acceso apropiadas cuando recupere datos. Con esta máquina de base de datos hemos usado un lenguaje de consulta que maneja datos compartidos, con la posibilidad de seleccionar varias bases de datos en el sistema. Hemos establecido un algoritmo para accesar archivos utilizando la definición de sus llaves primarias y los vínculos que existen entre dos entidades (uno-a-muchos, muchos-a-uno, uno-a-uno).

[Aho 1979] A.V. AHO Bell Laboratories and YsaGIV and J.D. ULLMAN
Efficient Optimization of a Class of Relational Expressions
Princenton University. ACM Transactions of Database Systems,
Vol. 4, No. 4, December 1979, Pages 435-454.

[Astrahan 1975]
M.M. Astrahan and D.D. Chamberlin
Implementation of a Structured English Query Language
Communications of the ACM
October 1975, Volume 18, Number 10, Pags. 580-588.

[Barkakaty 1989]
BARKAKATY
The Waite Groups Turbo C Bible
Ed. HOWARD W. SAMS & COMPANY, 1989

[Boyce 1975]
Raymond F. Boyce, Donald D. Chamberlin, and W. Frank King III
Specifying Queries as Relational Expressions: The SQUARE
Data sublenguage. IBM Research Laboratory, San Jose and Michael M.
Hammer, Massachusettes Institute of Technology
Communications of the ACM, November 1975, Volume 18, Number 11,
Pags. 621-628.

[Chamberlin 1974]
D.D. Chamberlin and R.F. Boyce,
SEQUEL: A Structured English Query Languaje
Proc. ACM-SIGFIDET
Worshop, Ann Arbor, MI, May 1974

[Chamberlin 1976]
D.D. Chamberlin, M.M. Astraham, et.al.
SEQUEL2: A Unified Approach to Data Definition, Manipulation and
Control.
IBM JOURNAL OF RESEARCH AND DEVELOPMENT
November, 1976, Volume. 20, Number 6, Pags. 560-575

[Date 1981] Date C.J.
Referential Integrity
Proc Very Large Data Bases, Cannes France, September 9-11,
1981, pp 2-12

[Date 1986] C. J. Date
Introducción a los Sistemas de Base de Datos
Addison Wesley Iberoamericana, 1986

[Fiorentino 1992]
Fiorentino Pérez Fernando, México D.F. 1996.
Modelo Físico de un Sistema de Información Geográfica para la Exploración Petrolera
Tesis Maestría. CIEA del I.P.N., Departamento de Ingeniería Eléctrica.

[Guzmán 1985]
Adolfo Guzmán Arenas.
The Files Descriptor: Use of a descriptive tool to retrive general queryes to file. Second Edition, Serie: Amarilla Investigación,
Cinvestav, Diciembre 16, 1985, No 16A.

[Kernighan 1978]
Brian WKerninghan, Dennies M. Ritchie
The C Programming Language
Bell Laboratories,
Prentice-Hall, INC, 1978

[Lasardif 1987]
Lasardif
Database Expert's Guide to SQL, McGraw Hill, 1987

[Maier 1983] David Maier
The Theory of Relational Databases, Rockville, Maryland
Computer Science Press, 1983.

[Mathias 1985]
Matthias Jarke and Yannis Vassiliou
A Framework for Coosing a Database Query Language
Computing Surveys, Vol 17, No. 3, September 1985

[Raúl 1990] Raúl Stefanoni, Sergio V. Chapa Vergara
Normalización de Base de Datos en C
Serie Amarilla, Cinvestav, Octubre 1990, No 107.

[Stevens 1987]
Al Stevens
C Data Base Development
Hireland, Oregon, MIS - Management Information Source, Inc., 1987

[Stonebraker 1975]
Held, G.D. Stonebraker, M,. And Wong, E.
INGRES a relational data base management system
Proc. AFIPS 1975 NCC, Vol 44,AFIPS Press, Montvale, N.J., pag 408-416.

[Todd 1978] Todd S.J.P.
ISBL (Information System Base Language).
The Peterlee Relational Test Vehicle
A System Overview, IBM Syst.J.15,4, Pag 285-308.

[Ullman 1982]
Jeffrey D.Ullman
Principles of Database Systems
Stanford University, Second Edition, Computer Science Press, 1982.

[Wong 1976] Eugene Wong and Karel Toussefi
Decomposition- A Srategy for Query Processing
University of California, Berckleley
ACM Transactions of Database Systems,
Vol. 1, No. 3, September 1976, Pages 223-241

[Zloof 1977] M.M. Zloof
Query By Example:A Data Base Language
IBM Sys J.16 num 4, 1977

Los abajo firmantes, integrantes de jurado para el examen de grado que sustentará el **Lic. José Alejandro Martínez Hernández,** declaramos que hemos revisado la tesis titulada:

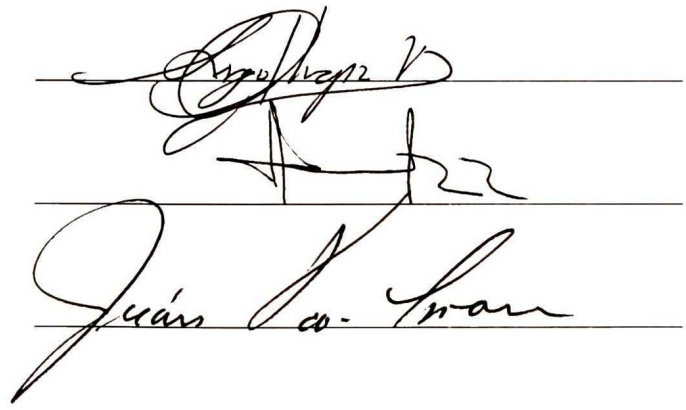
"REPRESENTACION DE SISTEMAS DE CONSULTA EN TABLEAUX Y SU IMPLENTACION EN UNA BASE DE DATOS RELACIONAL" consideramos que cumple con los requisitos para obtener el Grado de Maestro en Ciencias, con especialidad en Ingeniería Eléctrica.

Atentamente

Dr. Sergio V. Chapa Vergara

Dr. Arturo Díaz Pérez

Dr. Juan Francisco Corona Burgueño



AUTOR MARTINEZ HERNANDEZ, J.A.

TITULO REPRESENTACION DE SISTEMAS DE CONSULTA EN TABLEAUX Y SU...

CLASIF. XM 98.24 RGTRO. BI 15525

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
René Hdez. Sánchez	29-AG-00	5/5/00