UCLM-ESI

El Lenguaje QBE (Query By Example)

Olaya Alcañiz Ramos

Bases de Datos

Profesor: Francisco Ruiz

I. T. Informática de Gestión

Curso 1999 - 2000

ÍNDICE

1. INTRODUCCIÓN 3

2. INTRODUCCIÓN AL MODELO RELACIONAL 4

3. INTRODUCCIÓN AL CÁLCULO RELACIONAL 9

3.1. Introducción al cálculo relacional de dominios 9

4. EL LENGUAJE QUERY BY EXAMPLE 12

4.1. Introducción a QBE. 12

4.2. Operaciones de recuperación 13

4.3. Operaciones de recuperación sobre relaciones estructuradas en árbol 18

4.4. Funciones Integradas 21

4.5. Las operaciones de Actualización, Supresión e Insercción 23

4.6. Diccionario de QBE 26

4.7. Nuevas tendencias en QBE 29

5. EJEMPLOS QBE 30

6. QBE vs SQL 42

BIBLIOGRAFIA 44

1.- INTRODUCCIÓN.

En el trabajo que presentamos a continuación haremos un pequeño estudio acerca del lenguaje de consulta relacional QBE (Query By Example: consulta por ejemplo) creado por IBM research y que facilita la realización de consultas a una base de datos a cualquier usuario aunque éste sea inexperto.

Seguiremos un tratamiento muy parecido al que se da en la mayoría de los libros sobre bases de datos que tratan este tema, intentando dar más ejemplos con el fin de que el lector entienda completamente este lenguaje de consulta.

Empezaremos por dar una introducción al **modelo relacional**, ya que QBE es un lenguaje de consulta sobre tablas de relaciones. Como sabemos, con las bases de datos intentamos guardar datos del mundo exterior de forma estructurada y captar la semántica de los datos. Con el concepto de semántica, en este entorno, nos referimos a todas las relaciones que puede haber entre los datos, por ejemplo puede que no baste con que en nuestra base de datos se reflejen los valores de una serie de características de un individuo, sino que nos interesa también guardar información de las relaciones de unos datos sobre otros, por ejemplo, si un individuo está casado entonces quizás queramos almacenar también el número de hijos y el nombre de su mujer, si no lo es estos datos no tendrán interés. Al modelar una determinada zona cerrada del mundo exterior (universo de discurso) intentamos captar toda la información que es relevante para nosotros, en muchos casos esta información no son solo datos. Veremos una serie de características muy generales de como se enfoca el problema de la representación de datos del universo de discurso mediante el modelo relacional.

Después de esta introducción al modelo relacional abordaremos los conceptos del **cálculo relacional**, poniendo una especial atención al cálculo relacional de dominios ya que este lenguaje está basado en él.

A continuación estudiaremos las características principales del lenguaje, así como los tipos básicos de consultas que se pueden realizar con QBE; para una una mejor comprensión de estos conceptos introduciremos una pequeña colección de ejemplos.

Una ventaja significativa de este lenguaje (QBE), comparado con la mayoría de los lenguajes de consulta, es el grado de libertad de que disfruta el usuario, libertad para construir la consulta de una forma que se asemeja más a la manera natural que tenemos de consultar la información. En términos específicos la consulta se puede armar en cualquier orden que le agrade al usuario: el orden de los renglones dentro de una tabla de consulta carece totalmente de importancia; además, el orden en que el usuario rellena todas las entradas que constituyen estos renglones también es completamente arbitrario.

Por último, mostraremos las diferencias principales entre SQL y QBE, así como distintas consultas en los dos tipos de lenguajes realizando las comparaciones que sean pertinentes.

2.- INTRODUCCIÓN AL MODELO RELACIONAL.

Cuando se presentó el modelo relacional en 1970 se perseguían una serie de objetivos como la independencia lógica, la flexibilidad y otros. En aquel momento para poder realizar un buen diseño de una base de datos había que conocer bien el hardware donde se iba a implementar pues había que dar parámetros sobre tamaños de bloques de registros y ordenes de arboles de ordenación y otros muchos datos físicos de este estilo, algunos realmente complicados. Uno de los objetivos, y de las condiciones, del modelo relacional era el de la independencia lógica, es decir yo diseño la base de datos independientemente de la máquina sobre la que después se implantará, un diseño es válido para cualquier hardware también se pretendía que permitiera más complejidad de los datos y más tipos y formas de "restricciones" con las que poder añadir mayor semántica a una base de datos. Estos objetivos hoy día están conseguidos y no vamos a hacer un estudio detallado de ellos ya que esto solo es una introducción al modelo y no un estudio detallado sobre el mismo. Lo que si nos interesa resaltar es que hoy en día el modelo relacional es muy importante, los objetivos que se perseguían con su presentación se consiguieron y por ahora la inmensa mayoría de los productos comerciales están basados en ese modelo, aunque parece que el futuro nos dirige hacia el modelo relacional.

En un principio el modelo relacional fue puramente teórico, ya que no existían máquinas hardware donde se pudieran implementar Sistemas Gestores de Bases de Datos (SGBD) eficientes basados en él, estos problemas hoy en día están superados y el modelo relacional es actualmente el más usado en todo el mundo y de hay la importancia de este trabajo que nos enseñará como realizar buenos diseños de bases de datos para este modelo.

El éxito del modelo relacional quizás sea debido a que en principio era un modelo teórico. Codd lo planteo como un modelo que resolvía los principales problemas que tenían los modelos existentes en 1970, pero en aquellos momentos era irrealizable. Posteriormente se fueron desarrollando SGBD que cumplían más o menos con las restricciones (a esto nos referíamos antes al decir que la independencia lógica además de un objetivo era también una condición, ya que si un SGBD no la cumplía no podía ser considerado plenamente relacional) y la forma de trabajar del modelo, es decir, que no se intento un modelo que pudiera implementarse directamente en SGBD sino que se inventó un modelo con claras ventajas desde el punto de vista teórico y después se intentó implementar, cuando se consiguió esta implementación se pudo sacar partido de todas las ventajas del nuevo modelo y eso ha sido una de las causas de su éxito.

El modelo relacional esta basado fundamentalmente en el concepto de **relación.** Una relación es una colección de tuplas, cada una de las cuales esta formada por una serie de atributos, estos atributos son los mismos en cada una de las tuplas.

Una relación se puede representar como una tabla, en la cual cada atributo etiqueta las columnas de dicha tabla y cada fila es una tupla. Debido a este modelo de representación los términos relación y tabla se han convertido en sinónimos y a lo largo de este trabajo los usaremos indistintamente. También son sinónimos los términos atributo y campo.

Estas tablas deben de cumplir además las siguientes condiciones:

- No puede haber dos filas duplicadas.
- El orden de las columnas y de las filas no es importante. Si tenemos dos tablas con las mismas tuplas pero ordenadas de distinta forma realmente son lo misma tabla, lo mismo ocurre si se ordenan de forma distinta los atributos o si pasan las dos cosas a la vez.
- En la intersección de una columna y una fila , es decir en una "casilla", solo puede haber un valor. Esta condición también se expresa diciendo que no hay atributos multivaluados.

Estas últimas condiciones son llamadas **restricciones inherentes** al modelo. Veremos que hay mas tipos de restricciones, la mayoría de las cuales son definibles por el usuario y que le sirven para introducir en el esquema relacional más información que la que aportan los simples datos, información semántica de la que ya hablamos en la introducción.

Las relaciones tienen un nombre. Para denotar una relación se suele usar la siguiente notación:

Nombre_relación(Atributo1, Atributo2,...)

ponemos el nombre de la relación y a continuación entre paréntesis el conjunto de atributos que la forman. Si algún atributo se encuentra subrayado es que forma parte de la clave principal (ya veremos más adelante lo que es esto) y si alguno se encuentra en letra *itática* es que se trata de una clave ajena (también veremos este concepto más adelante).

Otro concepto importante en el modelo relacional es el **Dominio** .Un dominio es el conjunto del que un determinando atributo toma sus valores. A cada atributo se le asocia un dominio y para que un valor de ese atributo sea valido debe pertenecer al dominio correspondiente al atributo. La asignación de dominios a los atributos implica otra nueva restricción, en este caso una restricción que define el usuario al asignar un determinado dominio a un atributo, pues obligamos a que los valores de ese atributo deban pertenecer a un conjunto determinado. Ejemplos de dominios pueden ser: el conjunto de los números naturales, el conjunto de los números reales mayores de 3.5 ó el conjunto de las cadenas de caracteres de longitud 6.

Normalmente la notación de una relación incluye los dominios y es de la forma:

Nombre_relación(Atributo1 Dominio1, Atributo2 Dominio2,...)

pero nosotros usaremos la forma simplificada ya que para todo nuestro trabajo posterior el dominio de un atributo no es relevante.

El número de atributos que forman una relación es llamado **Grado** de esa relación y el número de tuplas que tiene una relación es llamado **Cardinalidad**, es decir el grado y la cardinalidad son el número de columnas y de filas de la tabla respectivamente.

Vamos a definir ahora un concepto fundamental en todo nuestro trabajo, el de **Clave Candidata :**

- Una clave candidata es un atributo o conjunto de atributos que identifica unívocamente a una tupla, es decir no hay dos tuplas con dos claves candidatas iguales.

Es claro que siempre existirá la clave candidata, ya que el conjunto de todos los atributos de la relación cumple la definición, ya que hemos dicho que no hay dos tuplas iguales como restricción fundamental del modelo.

La cuestión es que aunque el conjunto de todos los atributos siempre sirve para identificar unívocamente a cada tupla, en la realidad suele ocurrir que con un conjunto menor de atributos también podemos hacerlo. Veamos un ejemplo: imaginemos una relación llamada personas con las siguiente definición:

Personas (DNI,Nombre, Dirección, Edad, Fecha_nacimiento)

vemos que tiene 5 atributos, es decir, es una relación de grado 5. El conjunto de estos 5 atributos es una clave candidata ya que no hay dos tuplas que tengan iguales estos 5 campos, pero podemos ver que el atributo DNI también cumple esta condición, es decir, no hay dos tuplas en las que haya el mismo DNI, con lo cual el DNI nos identifica unívocamente a cada tupla y también es clave candidata.

Realmente en el ejemplo anterior el conjunto de todos los atributos no es una clave candidata, ya que estas deben cumplir una nueva condición de la que no hemos hablado hasta ahora y que es la de minimalidad, es decir si de un conjunto

de atributos que determina unívocamente a cada tupla se puede eliminar algún atributo y se sigue cumpliendo esta condición entonces el conjunto anterior no era clave candidata.

Vemos la nueva definición de clave candidata:

- Una **clave candidata** es un atributo o conjunto de atributos que identifica unívocamente a una tupla, es decir, no hay dos tuplas con dos claves candidatas iguales y además si se elimina algún atributo del conjunto la condición anterior deja de cumplirse, es decir, el conjunto es mínimo.

En una relación puede haber más de una clave candidata, por ejemplo imaginemos que en el caso anterior no puede haber dos personas con el mismo nombre, en ese caso el atributo **Nombre** también es clave candidata con lo cual tenemos dos posibles claves candidatas **DNI** y **Nombre.** Una de las claves candidatas se debe adoptar como **Clave Primaria**, que será única y las otras quedan como **Claves Alternativas**.

- Llamaremos **Clave Ajena** de una relación a un atributo o conjunto de atributos que deben coincidir con la clave principal de otra relación.

Imaginemos que además de la relación Personas tenemos otra en la que se hace referencia a estas, para referirnos a una persona usaremos la clave principal de la relación personas(por ejemplo el DNI) y el atributo que usemos para ello en la nueva relación será una clave ajena. Los valores que aparezcan en la clave ajena deben existir de antemano en la relación personas y los dos atributos deben tener el mismo dominio. Una clave ajena puede hacer referencia a al clave principal de la misma relación. En una relación puede haber claves ajenas o no, pero siempre habrá una clave primaria.

Ahora tenemos una nueva restricción inherente al modelo:

- Ningún atributo que forme parte de la clave primaria puede tomar valor nulo. Esta restricción es llamada **regla de integridad de entidad**.

También es una nueva restricción, llamada restricción de **Integridad Referencial**, el hecho de que los valores de una clave ajena deben ser valores que ya existen como claves principales en otra relación.

Para profundizar un poco más en el modelo relacional, realizaremos una pequeña introducción al proceso de **Normalización** y definiremos la cuatro primeras **formas normales** para los esquemas de relación.

La teoría de Normalización se desarrolló para que el diseño de las bases de datos relacionales, mejorara. La teoría de Normalización nos permite un diseño de base de datos riguroso y demuestra que existen métodos automáticos de diseño que permiten llegar a ciertas formas normales que garantizan un diseño de calidad. Es decir, siempre se puede asegurar el que una base de datos relacional esté al menos en forma normal de Boyce_Codd y con esto habremos eliminado gran parte de los problemas de redundancias y de ambigüedad que pueden surgir durante el diseño.

La **Normalización de los datos** puede considerarse como un proceso durante el cual los esquemas de relación insatisfactorios se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que poseen propiedades deseables. Un objetivo del proceso de normalización original es garantizar que no ocurran las *anomalías de actualización,* así como la eliminación de *información redundante en las tuplas.*

A continuación explicaremos brevemente las cuatro primeras formas normales, para ello nos ayudaremos del siguiente ejemplo que iremos pasando de una forma normal a otra;

R (código_prov, Nombre_prov, direccion_prov, Número_factura, Fecha, Importe)

Primera Forma Normal.

- Una relación se encuentra en primera forma normal cuando no hay grupos repetidos en sus atributos.

Esta condición es una restricción inherente al modelo relacional, ya que como hemos dicho anteriormente, en el cruce de una fila y una columna debe haber un único valor, no pueden existir atributos multivaluados. Como es una restricción inherente al modelo, está al menos en primera forma normal, por lo tanto la relación del ejemplo está en 1FN.

Segunda Forma Normal.

- Una relación se encuentra en segunda forma normal cuando está en 1FN y además todos los atributos que no forman parte de una clave candidata dan información sobre la clave principal.

Nuestro ejemplo en 2FN quedaría de la siguiente forma:

R1 (Código_prov, Nombre_Prov, Direccion_Prov)

R2 (Codigo_Prov, Numero_Factura, Fecha, Importe)

Tercera Forma Normal.

- Una relación se encuentra en tercera forma normal cuando está en 2FN y además los atributos que no forman parte de ninguna clave candidata dan información sobre la clave principal completa y sólo sobre la clave principal.

En el ejemplo las dos relaciones se encuentran en 3FN.

Forma Normal de Boyce_Codd.

Tomemos otro ejemplo, suponiendo que no se repiten los nombres de proveedores, es decir las claves candidatas son:

Código_prov, Número_factura.
Nombre_prov, Número_factura.

R(Código_prov,Nombre_prov, Numero_factura, Fecha, Importe)

Cómo vemos por la notación, hemos elegido como clave principal la primera opción. La relación está en FNBC.

- La relación se encuentra en FNBC cuando las claves candidatas son los únicos atributos o conjuntos de atributos sobre los que se facilita información por cualquier otro atributo.

Nuestra relación quedaría de la siguiente manera en FNBC:

R1(Código_prov, Número_factura, Fecha, Importe)

R2(Código_prov, Nombre_prov)

Esto es debido a que tenemos un atributo (Nombre_prov) que nos está dando información sobre otro atributo que no es clave candidata(Codigo_prov).

Se podría hablar mucho más sobre el proceso de Normalización y las relaciones que se dan entre los atributos(dependencias funcionales), pero no nos interesa profundizar excesivamente en este aspecto.

Si el lector estuviese interesado en profundizar más en el proceso de normalización, así como en el modelo relacional, la bibliografía es muy extensa habiendo multitud de libros que dedican uno o varios capítulos al tratamiento de estos temas.

3.- CÁLCULO RELACIONAL.

3.1.- INTRODUCCIÓN.

El concepto de cálculo relacional - es decir, un cálculo de predicados aplicado, propio de las bases de datos relacionales - fue propuesto en su forma original por Codd; un lenguaje basado explícitamente en ese cálculo, el sublenguaje de datos ALPHA (DSL ALPHA), también fue presentado por Codd. DSL ALPHA en sí nunca fue realizado, pero un lenguaje muy semejante a él, llamado QUEL, se usó como lenguaje de consulta en mayor detalle. SQL Y QBE, incorporan también ciertos elementos de cálculo.

Según el tipo de variables que se manejen, existen dos tipos de CR. El cálculo relacional de tuplas (CRT) emplea variables-tuplas, que designan a tuplas de relaciones(un lenguaje de manipulación de datos basado en CRT es SQL). Más recientemente, Lacroix y Pirotte han propuesto un cálculo relacional, el cálculo de dóminios (CRD), en donde las variables de tupla se reemplazan por las variables dominio - es decir, variables que varían sobre los dominios subyacentes en lugar de las relaciones(QUERY BY EXAMPLE **(QBE)** se considera un lenguaje de manipulación de datos basado en el CRD).

Al ser QBE un lenguaje basado en el **Cálculo Relacional de Dominios,** vamos a realizar una pequeña introducción al mismo.

3.2- Introducción al Cálculo Relacional de Dominios.

El Cálculo Relacional de Dominios es un tipo de lenguaje formal para bases de datos relacionales basado en el cálculo de predicados. Las variables en este lenguaje se denominan **Variables-dominio** ya que toman valores en el dominio asociado a alguno de los atributos de una relación en vez de representar una tupla entera. La definición de una variable-dominio es del tipo:

$$x{:}dom_k$$

de esta forma la variables x tomará valores en el dominio dom_k.Así pues, si queremos formar una relación de grado n como resultado de una consulta, debemos tener n de estas variables-dominio: una para cada atributo. Una expresión de Cálculo Relacional de Dominios tiene la forma

$$\{x_1,x_2,...,x_n \mid \text{COND}\ (x_1,x_2,...,x_n,x_{n+1},x_{n+2},...,x_{n+n})\}$$

donde x_1, x_2,..., x_n, x_{n+1}, x_{n+2}, ..., x_{n+m} son variables de dominio que abarcan dominios (de atributos) y COND es una **fórmula** del cálculo relacional de dominios. Las fórmulas se componen de **átomos.** Los átomos de una fórmula son un tanto diferentes de los del cálculo de tuplas y pueden ser cualquiera de los siguientes:

- Un átomo de la forma $R(x_1, x_2,...,x_n,x_{n+1}, x_{n+2},...,x_{n+m})$ donde R es el nombre de una relación de grado j y cada x_i, con $1<i<j$,es una variable de dominio. Este átomo expresa que una lista de valores de $< x_1, x_2,...,x_j >$ debe ser una tupla en la relación cuyo nombre es R, donde x_i es el valor del i-ésimo atributo de la tupla. A fin de hacer más concisas las expresiones del cálculo de dominio, se omiten las comas de la lista de variables; así pues, escribimos

$$\{x_1\ x_2\ ...x_n \mid R(\ x_1\ x_2\ x_3)\ \textbf{and}\ ...\}$$

en lugar de:

$$\{x_1,\ x_2,\ ...,\ x_n \mid R(\ x_1,\ x_2,\ x_3)\ \textbf{and}\ ...\}$$

- Un átomo de la forma x_i **op** x_j, donde **op** es uno de los operadores de comparación del conjunto {= ,< , > , ≤ , ≥ }, x_i y x_j son variables de dominio.

- Un átomo de la forma x_i **op** c o c **op** x_j, donde op es uno de los operadores de comparación del conjunto {= ,< , > , ≤ , ≥ }, x_i y x_j son variables de dominio y c es un valor constante.

Al igual que en el cálculo de tuplas, la evaluación de un átomo resulta ya sea TRUE o FALSE para un conjunto específico de valores, y estos resultados se denominan **valores lógicos** de los átomos. En el caso 1, si se asignan a las variables de dominio valores que corresponden a una tupla de la relación especificada R, el átomo será TRUE. En los casos 2y 3, si se asignan a las variables de dominio valores que satisfagan la condición, el átomo será TRUE.

De manera similar el cálculo relacional de tuplas, las fórmulas se componen de átomos, variables y cuantificadores.

A continuación daremos algunos ejemplos de consultas especificadas en el cálculo de dominios. Usaremos letras minúsculas l,m,n,...,x,y,z para las variables dominio, para ello utilizaremos el siguiente esquema de una base de datos.

proveedor (# prove, Nom, Ape, Ciudad, Calle, Num)
artículo (# articulo, Nom-art, Precio, Descrip)
pedido (# pedido, # articulo, # prove, cantidad)

consulta 1

Obtener el nombre de los proveedores que viven en ‘Madrid’.

$\{s \mid \exists x \exists t \exists v \exists w(\text{proveedor } (r,s,t,\text{‘Madrid’},v,w)\ \textbf{)}\}$

Necesitamos seis variables para la relación empleado, cada una para cubrir el dominio de un atributo en orden. De todas las variables, aquellas que están ligadas a un cuantificador existencial son x, t, v, w; la variable s es libre. Primero especificamos el atributo solicitado, *nombre* ,con la variable de dominio s. Luego después de la barra (|), especificamos la condición para seleccionar una tupla: es decir, que la secuencia de valores asignados a las variables rstuvw se una tupla de la relación Empleado y que los valores de **u** (ciudad) sean ‘Madrid’.

consulta 2

Nombre de los proveedores que suministran el articulo 20.

$\{x \mid \exists y \exists z \exists w \exists p \exists u \exists v(\text{proveedor } (p,x,y,z,w)\ \Lambda\ \text{pedido } (p,20,u,v)\ )\}$

consulta 3

Nombre de los proveedores que suministran artículos a un precio mayor que 10.

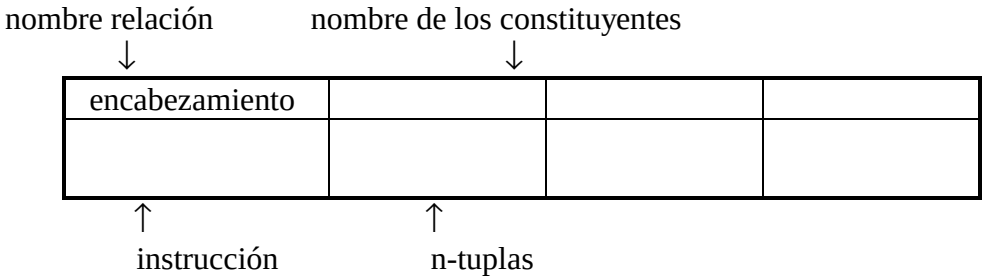
{x | ∃y∃z∃u∃v∃w∃k∃l∃m∃n∃ñ (proveedor (y,x,z,u,v) Λ articulo(w,k,l,m) Λ pedido (y,w,n,ñ) Λ l > 10)}

4.- EL LENGUAJE "QUERY BY EXAMPLE"(QBE).

4.1.- INTRODUCCIÓN A QBE.

QBE (Query By Example:consulta por ejemplo), es un producto de IBM ,que difiere de SQL y de QUEL en que el usuario no tiene que especificar explícitamente una consulta se formula llenando **plantillas** de relaciones que se exhiben en la pantalla de un terminal.

Podemos considerar que este lenguaje es un lenguaje de tipo predicativo con variables de dominio, contiene características específicas en las que se ha estudiado especialmente bien la interacción entre el hombre y la máquina. Efectivamente, el usuario de este lenguaje trabaja en un terminal de pantalla, y dispone de algunas instrucciones que le permiten mostrar en pantalla no sólo los esquemas de las relaciones (los nombres de las relaciones, los nombres de los constituyentes), sino también tablas que sirven para enmarcar el contenido de las relaciones, tal y como vemos a continuación.



Diremos que una tabla vacía de este tipo tiene la estructura de una relación. La estructura de una relación nos permite dividir la pantalla en varias zonas. En la zona de encabezamiento colocaremos el nombre de la relación y el de los constituyentes. Además, el usuario podrá añadir, si lo desea, columnas adicionales. La zona de n-tuplas situada bajo la zona de los nombres de los constituyentes tendrá una doble función: por un lado, lo hará posible que el usuario de fina su petición con la ayuda de convenciones y de ejemplos lo que justifica el nombre del lenguaje y, por otra, una vez que se ha formulado la petición, se puede recibir la respuesta en las columnas correspondientes. Por último, la zona de instrucciones colocada bajo el nombre de la relación permite indicar que operaciones de manipulación se desea ejecutar.

Para formular una petición, el usuario tendrá que hacer que en pantalla aparezca todas las estructuras que sean necesarias para llevar a cabo dicha formulación.

Por ejemplo, considérese la consulta "obtenga los números de proveedor de los proveedores en París". Primero, al oprimir una cierta llave de función en la terminal, el usuario puede tener una tabla "esqueleto" en blanco desplegada en la pantalla. Entonces, al saber que la respuesta a la consulta se puede hallar en la tabla S, el usuario inserta S como nombre de la tabla y hace que QBE responda rellenando con nombres de columna la zona superior del esqueleto. Ahora el usuario puede expresar la consulta haciendo entradas en dos posiciones de la tabla, como sigue:

S	# S	NOMS	ESTADO	CIUDAD
	P.S7			París

La " P " significa "imprima"; indica los blancos de la consulta, es decir, los valores que deben aparecer en el resultado. S7 es un "elemento de ejemplo", es decir, un ejemplo de una respuesta posible a la consulta; los elementos de ejemplo se introducen subrayados. París (no está subrayado) es un "elemento constante". La consulta se puede describir así: " imprima todos los valores de #S, tales como S7(por ejemplo), donde la ciudad correspondiente sea París". Nótese que S7 no necesita parecer en el conjunto resultante, ni siquiera en el conjunto original; el elemento de ejemplo es arbitraria, y pudo haberse utilizado **PIG**, **7** o **X** sin cambiar el significado de la consulta.

Después se explica por qué los elementos de ejemplo se usan para establecer enlaces entre los renglones en consultas más complicadas. Si ningún enlace es necesario, como en la consulta sencilla anterior, es posible omitir por completo los elementos de ejemplo(de modo que " P.S7" se reduciría a "P"), pero en general se incluirán por claridad.

Para ejecutar una petición en el lenguaje QBE procederemos de la siguiente forma:

1.- Llamaremos a pantalla la estructura de cada relación R_1,..., R_N.
2.- Por cada elemento c_{ij} de cada relación construiremos una línea:
-Si c_{ij} sólo aparece una vez, entonces podemos dejar en blanco la columna correspondiente.
-Si c_{ij} aparece como variable (se le menciona varias veces),
entonces colocaremos en las columnas correspondientes un valor a título de ejemplo, de manera que el valor vaya precedido de un signo de subrayado.
-Si c_{ij} aparece como un elemento a, entonces colocaremos el símbolo P en la columna correspondiente.
3.- Repetir el paso 2 para todas las relaciones.

A continuación presentamos las características principales de QBE, estudiaremos primeramente las operaciones del "Lenguaje de manipulación de datos"(DML) , seguidas de las operaciones del "Lenguaje de definición de datos"(DDL), porque como se verá después estas últimas son casos especiales de las primeras.

4.2.- OPERACIONES DE RECUPERACIÓN.

4.2.1.- Recuperación simple. Obtenga los números de parte de todas las partes suministradas.

SP	#S	#P	CTD
		P.PX	

A diferencia de SQL, QBE elimina automáticamente los duplicados redundantes del resultado de una consulta. Para suprimir esa eliminación , el usuario puede especificar la palabra clave "ALL".

La finalidad del símbolo ALL que hemos mencionado , escrito en una columna X y en una línea de la estructura de una relación R, es la de definir un conjunto de valores. Si no hay escrita otra información en la misma línea, el conjunto así definido es el de todos los valores de X en la relación o, dicho en términos más formales, la proyección de R sobre X. Si en la misma línea figuran otras relaciones, constantes o variables ejemplo ara definir un criterio de

calificación, el conjunto definido es el conjunto de valores de X que aparecen en las n-tuplas de R que satisfacen esos criterios de calificación.

Además, podemos escribir una variable ejemplo detrás del símbolo ALL, lo que nos permitirá dar un nombre al nuevo conjunto. Este nombre puede aparecer en la misma estructura de la relación o en otra estructura, para crear, en este segundo caso, nexos entre las diferentes estructuras. Mostramos con un ejemplo el uso del símbolo ALL.

SP	#S	#P	CTD
		P.ALLPX	

Si una columna es demasiado angosta en la pantalla como para contener la entrada deseada, el usuario puede ensancharla primero mediante otra llave de función. También se suministran llaves de función para añadir columnas o renglones, suprimir columnas o renglones, etc.

Ejemplo: Obtenga todos los detalles de todos los proveedores.

S	# S	NOMS	ESTADO	CIUDAD
	P.SX	P.SN	P.ST	P.SC

La siguiente es una representación abreviada de la misma consulta.

S	# S	NOMS	ESTADO	CIUDAD
P.				

4.2.2.- Recuperación calificada. Obtenga los números de proveedor de los proveedores en París con estado > 20.

S	# S	NOMS	ESTADO	CIUDAD
	P.SX		>20	París

Obsérvese el modo de especificar la condición " estado > 20". En general , se puede usar de esta manera cualquiera de los operadores de comparación =, ¬ =, <, <=, >, > = (excepto que a menudo se omite =, como en la columna CIUDAD anterior, y ¬ = por lo común se abrevia ¬ para simplificar).

También se puede realizar de la siguiente manera:

S	# S	NOMS	ESTADO	CIUDAD
	P.SX		y	París

CONDICIÓN
y > 20

En las estructuras de cada una de estas relaciones tenemos que indicar los valores de cada n-tupla, y en una tabla adicional CONDICIÓN indicaremos la relación que debe satisfacer

la variable y, que en este caso es y >20, para que se seleccione la n-tupla dada. Sólo se seleccionarán las n-tuplas que satisfagan esta condición, además de las otras condiciones que figuren en las estructuras. La tabla de condición puede contener varias líneas de expresen condiciones distintas.

Una línea de la tabla de condición puede utilizar los operadores de comparación (<, >, < =, > =, =, < >), los operadores booleanos (¬, ∧ , ∨) y las expresiones aritméticas.

La instrucción de negación ¬, se utiliza cuando queremos negar la existencia de una n-tupla, en este caso escribiremos el símbolo ¬ en la línea correspondiente y en la zona de la instrucción.

Además, "P." puede preceder a cualquiera de estos operadores si desea imprimir el valor en cuestión; por ejemplo, pudo haberse especificado " P. > 20" debajo de Estado si también se hubieran querido imprimir los valores de estado.

Ejemplo 1: Obtenga los números de parte de todas las partes suministradas por más de un proveedor.

SP	#S	#P	CTD
	SX	P.PX	
	¬SX	PX	

Esta consulta puede expresarse así: " imprima los números de parte PX tales que PX sea suministrada por algún proveedor SX y también por algún proveedor distinto de SX ".

ejemplo 2: Obtenga los números de proveedor de los proveedores que se localizan en París o tienen estado > 20 (o ambos).

S	# S	NOMS	ESTADO	CIUDAD
	P.SX			París
	P.SY		>20	

Se considera que las condiciones especificadas dentro de un solo renglón se vinculan mediante "AND". Para vincular dos condiciones usando "OR", por tanto, es necesario especificarlas en renglones separados. La consulta anterior, en efecto, está pidiendo la *unión* de todos los números de proveedores SX para los proveedores de París y de todos los números de proveedores SY con estado >20. Se necesitan dos elementos de ejemplo diferentes, porque si hubiera usado el mismo dos veces, ello habría significado que el mismo proveedor tendría que estar en París y tener estado >20.

Cuando una consulta comprende más de un renglón, como en el presente ejemplo, los renglones se pueden introducir en cualquier orden.

ejemplo 3: Obtenga los números de proveedor de los proveedores que suministran la parte P1 y la P2.

SP	#S	#P	CTD
	P. SX	P1	
	SX	P2	

Aquí el mismo elemento de ejemplo debe usarse dos veces; se necesitan dos renglones para expresar la consulta porque es preciso vincular dos condiciones mediante "AND" sobre la misma columna.

4.2.3.- Recuperación con ordenamiento. Obtenga el número de proveedor y el estado de los proveedores en París, en orden descendente de estado.

S	# S	NOMS	ESTADO	CIUDAD
	P.SX		P.DO.ST	París

El "DO." significa " orden descendente". "AO." se usa para indicar "orden ascendente". Cuando se requiere el ordenamiento en términos de varios campos, entonces el campo principal se indica por "AO(1)."[o "DO(1)."]; el siguiente, por "AO(2)."[o "DO(2)."], y así sucesivamente.

4.2.4.- Recuperación usando enlace. Obtenga los nombres de proveedor de los proveedores que suministran la parte P2.

S	# S	NOMS	ESTADO	CIUDAD
	SX	P.SN		

SP	#S	#P	CTD
	SX	P2	

El elemento de ejemplo SX se usa como enlace entre S y SP. La consulta se puede describir así:" imprima los nombres de los proveedores (como SN) donde el número de proveedor correspondiente, por ejemplo SX, aparezca en la tabla SP con número de parte igual a P2". En términos generales, los enlaces se usan en QBE donde en SQL usaría un SELECT anidado o un cuantificador existencial, o donde el álgebra usaría una reunión. En realidad, SX, en el ejemplo 3 del apartado 5.2.2 , también estaba actuando como enlace, pero los renglones se hallaban en la misma tabla.

ejemplo 1: Obtenga los nombres de proveedor de los proveedores que suministran al menos una parte roja.

S	# S	NOMS	ESTADO	CIUDAD
	SX	P.SN		

SP	#S	#P	CTD
	SX	PX	

P	#P	NOMPAR	COLOR	PESO	CIUDAD
	PX		ROJO		

4.2.5.- Recuperación usando la negación. Obtenga los nombres de proveedor que no suministran la parte P2.

S	# S	NOMS	ESTADO	CIUDAD
	SX	P.SN		

SP	#S	#P	CTD
	SX	P2	

Nótese el operador NOT(¬) contra el renglón de consulta de la tabla SP. La consulta puede escribirse así: "imprima los nombres de proveedor de los proveedores SX tales que SX no suministre la parte P2".

4.2.6.- Recuperación usando un enlace dentro de una sola tabla. Obtenga los números de proveedor de los proveedores que suministran al menos una parte suministrada por el proveedor S2.

SP	#S	#P	CTD
	P.SX	PX	
	S2	PX	

ejemplo 1: Obtenga los números de parte de todas las partes suministradas por más de un proveedor.

SP	#S	#P	CTD
	.SX	P.PX	
	¬SX	PX	

Esta consulta puede expresarse así: " imprima los números de parte PX tales que PX sea suministrada por algún proveedor SX y también por algún proveedor distinto de SX".

4.2.7.- Recuperación de más de una tabla. Obtenga el número de la parte y los nombres de todas las ciudades que suministran la parte, para cada parte suministrada.

El resultado de esta consulta no es una proyección de una tabla existente; por el contrario, es una proyección de una reunión de tablas existentes. Para formular esa consulta en QBE el usuario primero debe crear una tabla esqueleto de la misma forma del resultado esperado , es decir, con el número adecuado de columnas. Se puede asignar cualquier nombre que el usuario desee a esta tabla y a sus columnas (incluso puede dejarse sin nombre). El usuario puede expresar entonces la consulta usando la tabla de "resultados" y las dos tablas existentes de la manera siguiente:

S	# S	NOMS	ESTADO	CIUDAD
	SX			SC

SP	#S	#P	CTD
	SX	PX	

RESULTADO	#P	CIUDAD
	P.PX	P.SC

4.2.8.- Recuperación usando la tabla de condiciones (cuando aparece más de una tabla). Obtenga todas las parejas de números de proveedor tales que los dos proveedores se localicen en la misma ciudad.

S	# S	NOMS	ESTADO	CIUDAD
	SX			CZ
	SY			CZ

RESULTADO	#P	CIUDAD
P.	SX	SY

CONDICIÓN
SX > SY

A veces es difícil expresar una condición deseada dentro del marco de las tablas de la consulta. En esa situación, QBE permite al usuario insertar la condición en una "caja de condiciones" separada, como se refleja en el ejemplo anterior. La caja de condiciones se obtiene usando otra llave de función en la terminal.

4.3.- OPERACIONES DE RECUPERACIÓN SOBRE RELACIONES ESTRUCTURADAS EN ÁRBOL.

Las operaciones de QBE descritas en esta sección no están soportadas en la realización actual de IBM; sin embargo, merecen cierto análisis porque dirigen la atención a un área de

aplicación interesante que pocos lenguajes, relacionares o de otro tipo, tratan de modo adecuado.

Considérese la relación **RS** (estructura de reportamiento) que se muestra en la figura 1. Esta relación tiene dos atributos, #EMP y #ADM, ambos extraídos de un dominio subyacente de números de empleado. El significado de una tupla específica de **RS** es que el empleado(#EMP) indicado reporta directamente al administrador indicado (# ADM).

#ADM	#EMP
E1	E6
E1	E7
E1	E8
E6	E18
E8	E15
E8	E16
E15	E20
E15	E24
E24	E32

Se supone que la estructura de reportamiento representada por la relación **RS** satisface las siguientes restricciones:

1. Ningún empleado es su propio administrador.
2. Ningún empleado tiene más de un administrados inmediato.
3. Si **EX** es el administrador inmediato de **EY**, entonces **EY** no puede ser el administrador de **EX** a ningún nivel.

Una relación que posea dos atributos definidos sobre un dominio común y que satisfaga restricciones análogas a las anteriores se denominará *relación estructurada en árbol.* En QBE el usuario puede formular ciertas consultas sobre una relación estructurada en árbol, que lenguajes menos potentes, como el álgebra relacional, son incapaces de expresar. A continuación mostramos algunos ejemplos:

4.3.1.- Recuperación bajando un nivel. Obtenga los números de empleado de los empleados que reportan al empleado E8 en el primer nivel.

RS	#ADM	#EMP
	E8	P.EX

Por “en el primer nivel” se entiende que e8 es el administrador inmediato de los empleados que se desea. La repuesta a la consulta es los empleados E15 y E16. Este ejemplo es directo y no ilustra ningún punto nuevo.

4.3.2.- Recuperación bajando dos niveles. Obtenga los números de empleado de los empleados que reportan al empleado E8 en el segundo nivel.

RS	#ADM	#EMP
	E8 EY	EY P.EX

De nuevo la solución es directa, pero se ha tenido que introducir un enlace, EY , así como insertar dos veces el enlace en la tabla . En general, si se quisiera descender n niveles en el árbol, sería preciso insertar dos veces dad uno de los n-1 enlaces, proceso bastante pesado. Por tanto, QBE proporciona una abreviatura conveniente, que se ilustra en la siguiente formulación alterna de la consulta anterior.

RS	#ADM	#EMP
	E8	P.EY(2L)

El "(2L)" significa "segundo nivel". En general, una entrada de nivel puede componerse de cualquier entero seguido de la letra L, todo entre paréntesis. Siempre que se use una entrada de nivel, QBE incluye niveles relativos en el resultado tabulado(por ejemplo) como sigue:

#EMP
E20(2L)
E24(2L)

4.3.3.- Recuperación subiendo dos niveles. Obtenga el número de empleado del administrador que está dos niveles por encima del empleado E20.

RS	#ADM	#EMP
	P.MX(2L)	E20

Aquí la entra de nivel aparece en la columna #ADM. En general, la dirección de la búsqueda(hacia arriba o hacia abajo del árbol)se indica por la columna donde aparece la entrada de nivel. Sin embargo, en ciertas situaciones esta regla podría originar ambigüedad; para evitar este problema, QBE impone la restricción de que no pueden aparecer más de dos entradas en ningún renglón específico de la formulación de una consulta que comprenda niveles.

4.3.4.- Recuperación bajando a todos los niveles. Obtenga el número de empleado de los empleados que reportan al empleado E8 en cualquier nivel.

RS	#ADM	#EMP
	E8	P.EX(6L)

Resultado:

#EMP
E15(1L)
E20(2L)
E24(2L)
E32(3L)
E16(1L)

4.3.5.- Recuperación descendiendo al nivel más bajo. Obtenga el número de empleado de los empleados que reportan al empleado E8 al nivel más bajo.

RS	#ADM	#EMP
	E8	P.EX(MAX.6L)

MAX es una función integrada, que veremos mas delante. El significado de esta consulta es: "obtenga los números de empleado de los empleados cuyo novel relativo debajo de E8 es máximo".

4.3.6.- Recuperación bajando a los niveles terminales. Obtenga el número de empleado de los empleados que reportan al empleado E8 a quienes nadie los reporta.

RS	#ADM	#EMP
	E8	P.EX(LAST.L)

Se buscan empleados en los nodos terminales del árbol debajo de E8. Como estos empleados estarán en diferentes niveles relativos, en general, no se puede introducir un entero constante ni un entero de ejemplo(una constante significaría un nivel fijo, un ejemplo representaría todos los niveles); por tanto, QBE proporciona la función especial LAST.

4.3.7.- Recuperación de nivel. ¿ En que función esta el empleado E20 debajo del empleado E1?

RS	#ADM	#EMP
	E1	P.E20(7)

Resultado:

#EMP
E20(3L)

4.4.- FUNCIONES INTEGRADAS.

Al igual que SQL, QBE proporciona un conjunto de funciones integradas. En general, podemos decir que todos los lenguajes relaciones ofrecen la posibilidad de definir conjuntos de informaciones mediante expresiones muy potentes (en lenguaje predicativo o en lenguaje algebraico). Es posible que, una vez que hemos construido esos conjuntos, deseemos aplicar un operador que produzca un valor asociado a ese conjunto. Los operadores que se utilizan en el QBE son:

COUNT para enumerar los elementos.
SUM para sumar elementos.
AVEGARE para calcular la media.
MAX para buscar el máximo.
MIN para buscar el mínimo.
UNIQUE para eliminar valores duplicados en la respuesta a una pregunta.

Respecto al operador UNIQUE, tenemos que decir, que , los valores duplicados no se eliminan en ninguna de las posibilidades disponibles mediante el lenguaje QBE cuando imprimimos el resultado de una petición.

Desde un punto de vista formal, la respuesta a una petición en el lenguaje QBE no es un conjunto de valores, sino una lista de valores. El papel del operador UNIQUE consiste, en realizar el paso de la lista a conjunto.

El operador ALL (descrito anteriormente), se asocia con estos operadores del conjunto.

La palabra clave “ALL” siempre se especifica, mientras que “UNIQUE”, es opcional, sólo aparece en las funciones COUNT, SUM, AVERAGE.

4.4.1.- Recuperación simple usando una función.

a.- Obtenga el número total de proveedores.

S	# S	NOMS	ESTADO	CIUDAD
	P.CNT.ALL.SX			

b.- Obtenga el número total de los proveedores que suministran partes.

SP	#S	#P	CTD
	P.CNT.UNQ.ALL.SX		

4.4.2.- Recuperación calificada usando una función.

a.- Obtenga el número de remesas para la parte P2.

SP	#S	#P	CTD
	P.CNT.ALL.SX	P2	

b.- Obtenga la cantidad total suministrada de la parte P2.

SP	#S	#P	CTD
		P2	P.SUM.ALL.Q

4.4.3.- Funcion en la caja de condiciones. Obtenga los números de proveedor de los proveedores con estado menor que el estado máximo actual en la tabla S; obtenga también ese estado máximo.

S	# S	NOMS	ESTADO	CIUDAD
	P.SX		ST P.MAX.ALL.SS	

CONDICIÓN
ST<MAX.ALL.SS

4.4.4.- Recuperación con agrupamiento. Obtenga el número de parte y la cantidad total suministrada para cada parte suministrada.

SP	#S	#P	CTD
		P.G.PX	P.SUM.ALL. QX

La "G." es el equivalente de QBE del operador GROUP BY de SQL.

4.4.5.- Recuperación usando agrupamiento y la caja de condiciones. Obtenga los números de parte de todas las partes suministradas por más de un proveedor; también haga conteos de los proveedores correspondientes.

SP	#S	#P	CTD
	P.CNT.ALL.SX	P.G.PX	

CONDICIÓN
CNT.ALL.SX >1

4.5.- LAS OPERACIONES DE ACTUALIZACIÓN, SUPRESIÓN E INSERCIÓN.

4.5.1.- Las operaciones de actualización.

Las operaciones de actualización se representan mediante la instrucción UPDATE en la zona de instrucciones. Cuando deseemos actualizar una n-tupla, tenemos que decir de qué n-tupla se trata. Para ello debemos recordar que en todas las relaciones tenemos una clave, noción definida en el modelo relacional apartado 2. Recordaremos brevemente que la clave de una relación es un conjunto de constituyentes cuyos valores determinan unívocamente una única n-tupla de la relación. En todas las operaciones de actualización de una n-tupla de una

relación, tenemos que indicar los valores de la clave con la instrucción UPDATE, además del valor, o valores actualizados.

Ejemplo 1: **Actualización de un solo registro**. Cambie el color de la parte P2 a amarillo.

P	#P	NOMPAR	COLOR	PESO	CIUDAD
	P2		U. AMARILLO		

O también:

P	#P	NOMPAR	COLOR	PESO	CIUDAD
U.	P2		AMARILLO		

La operación de actualización es "U.". El registro que se va actualizar se identifica por el valor de su llave primaria. Los valores de la llave primaria no pueden actualizarse.

Ejemplo 2: **Actualización de un solo registro basada en el valor previo.** Aumente el peso de la parte P2 en 5.

P	#P	NOMPAR	COLOR	PESO	CIUDAD
U.	P2 P2			PSO PSO+5	

Para actualizar un registro con base en su valor anterior, el usuario inserta un renglón que representa la versión anterior y otro que representa la versión nueva. La "U." indica cual es la nueva.

Ejemplo 3: **Actualización de registros múltiples**. Doble el estado de todos los proveedores situados en Londres.

S	# S	NOMS	ESTADO	CIUDAD
U.	SX SX		ST 2* ST	LONDRES

Aquí el valor de la llave primaria (el número del proveedor) se especifica como elemento de ejemplo, no como constante.

Ejemplo 4: **Actualización con registros múltiples**. Ajuste la cantidad a cero para todos los proveedores situados en Londres.

SP	#S	#P	CTD
U.	SX		0

S	# S	NOMS	ESTADO	CIUDAD
	SX			LONDRES

Ejemplo 5: **Actualización de tablas múltiples**. Ajuste la cantidad y el estado a cero para todos los proveedores situados en Londres.

SP	#S	#P	CTD
U.	SX		0

S	# S	NOMS	ESTADO	CIUDAD
	SX			LONDRES
	SY			LONDRES
U.	SY		0	

Dado que la unidad de entrada de QBE es la pantalla completa, se pueden introducir varias actualizaciones al mismo tiempo.

4.5.2.- Las operaciones de Supresión.

La operación de supresión se representa mediante la instrucción UPDATE, se propone quitar las n-tuplas indicadas por el valor de la clave de la relación.

Ejemplo 1: **Supresión de un solo registro**. Suprima al proveedor S1.

S	# S	NOMS	ESTADO	CIUDAD
D.	S1			

"D." es el operador de suprimir. Si la tabla SP actualmente tienen cualesquiera remesas para el proveedor S1, esta supresión hará que la regla de integridad 2 sea violada.

Ejemplo 2: **Supresión de registros múltiples.** Suprima todos los proveedores situados en Londres.

S	# S	NOMS	ESTADO	CIUDAD
D.				LONDRES

Ejemplo 3: **Supresión de registros y tablas múltiples**. Suprima todos los proveedores situados en Londres y también todas las remesas para esos proveedores.

S	# S	NOMS	ESTADO	CIUDAD
	SX			LONDRES
D.				LONDRES

SP	#S	#P	CTD
D.	SX		0

Este ejemplo de QBE, suprimirá primero las remesas y después los proveedores.

4.5.3.- Las operaciones de inserción.

La operación de inserción se representa mediante la instrucción INSERT en la zona de instrucciones. Para llevar a cabo una operación de inserción en una relación, escribimos en la estructura de dicha relación una línea cuyo encabezamiento es la instrucción INSERT, seguida por todos los valores que queremos introducir.

Ejemplo 1: **Inserción de un registro individual**. Adicione la parte P7 (de nombre 'ARANDELA', color 'GRIS', peso 2, ciudad 'Atenas') a la tabla P.

P	#P	NOMPAR	COLOR	PESO	CIUDAD
I.	P7	ARANDELA	GRIS	2	ATENAS

"Y." es el operador para insertar. El registro nuevo debe tener un valor de llave primaria que sea no nulo y distinto de todos los valores de la llave primaria existentes en el atabla. Otros campos pueden dejarse en blanco en la tabla de esqueleto, en cuyo caso se ajustarán al valor nulo en la tabla de datos.

Ejemplo 2: **Inserción de registros múltiples**. Inserte en TEMP los números de parte de todas las partes suministradas por el proveedor S2.

SP	#S	#P	CTD
	S2	PX	

TEMP	#P
I.	PX

4.6.- DICCIONARIO DE QBE.

Query By Example, tiene un diccionario integrado primitivo que se le presenta al usuario como un conjunto de tablas. El diccionario incluye, por ejemplo una tabla TABLE y una tabla DOMAIN, que dan detalles de todas las tablas y de todos los dominios actualmente conocidos por el sistema. Las tablas del diccionario se pueden interrogar usando las operaciones de recuperación ordinarias de DML (lenguaje de manipulación de datos).

En QBE los operadores para consultar y actualizar el diccionario están integrados en el lenguaje, de forma que es consistente con los otros operadores (los que no son del diccionario).

En particular, QBE en realidad no incluye un DDL en sí, sino que usa formas especiales de los operadores de actualización del DML para proporcionar una función equivalente.

A continuación mostramos dos ejemplos de recuperación:

4.6.1.- Recuperación de nombres de tablas. Obtenga los nombres de todas las tablas conocidas por el sistema.

P.				

En lugar de construir un esqueleto para la tabla TABLE e insertar "P." en la columna NAME de ese esqueleto, el usuario puede formular esta consulta insertando la "P." en la posición del *nombre-de-tabla* de una tabla en blanco.

4.6.2.- Recuperación de nombres de columnas para una tabla específica. Obtenga los nombres de todas las columnas de la tabla S.

S P.				

El usuario inserta el nombre de la tabla (S) seguido por "P." frente al renglón de los nombres de columna (en blanco). QBE responde rellenando de manera adecuada esos blancos. Esta función a menudo se usa para armar la consulta "real".

Ahora se examinan las funciones de definición de datos.

4.6.3.- Creación de una tabla nueva. Cree la tabla S, suponiendo que todavía no existe.

I. S I.	**# S**	**NOMS**	**ESTADO**	**CIUDAD**

La primera "I." crea una entrada del diccionario para la tabla S; la segunda "I." crea entradas del diccionario para las cuatro columnas de la tabla S. Sin embargo, el proceso de creación de la tabla aún es incompleto; cierta información adicional debe especificarse para cada columna. Esta información incluye para cada columna, el nombre del dominio subyacente; el tipo de datos del dominio, si QBE no conoce ya el dominio; una indicación en cuanto a si la columna participa o no en la llave primaria, y una especificación de si se va a construir o no un índice sobre la columna.

QBE, supone que cada columna forma parte de la llave y que cada columna va a se indicada, a menos que se le informe lo contrario.

S	**# S**	**NOMS**	**ESTADO**	**CIUDAD**
DOMAIN I.	#S	NOMS	ESTADO	CIUDAD
TYPE I.	CHAR (5)	CHAR(20)	FIXED	CHAR(15)
KEY	Y	U.N	U.N	U.N
INVERSION	Y	U.N	U.N	U.N

En este ejemplo, la información de DOMAIN y de TYPE se suministra insertando un renglón completo de especificaciones; la información de KEY y de INVERSION se suministra actualizando los valores por omisión de QBE. Se han definido ciertos dominios para QBE, además de la tabla S y las columnas; por ejemplo , ahora QBE conoce a CIUDAD como un dominio de hileras de caracteres de longitud 15. Los tipos de datos soportados por QBE son CHAR, CHAR(n), FIXED, FLOAT, DATE y TIME.

4.6.4.- Creación de una instantánea. Para cada parte suministrada, obtenga el número de la parte y los nombres de todas las ciudades que suministran la parte. Conservese el resultado en la base de datos.

S	# S	NOMS	ESTADO	CIUDAD
	SX			SC

SP	#S	#P	CTD
		SX	PX

I. RESULTADO	#P	CIUDAD
P.I.	PX	SC

Se almacena en al base de datos (a causa de los operadores ”I.” una copia de esta tabla, con nombre de tabla RESULTADOS y nombres de columna #P y CIUDAD. Las especificaciones del dominio (y de los tipos de datos) para las columnas de esta tabla se heredan de las tablas subyacentes; otras especificaciones (por ejemplo, KEY e INVERSION) se suponen por omisión(pero se pueden cambiar, si se desea, usando “U.”; tales cambios se pueden especificar únicamente en el momento en que la tabla se crea, no en una operación subsiguiente).Se dice que la tabla almacenada recientemente es una instantánea de las tablas subyacentes. Una vez creada es independiente de esas tablas.

4.6.5.- Eliminación de una tabla. Elimine la tabla SP.
Una tabla se puede eliminar si está actualmente vacía.
a.- Suprima todas las remesas.

SP	#S	#P	CTD
D.			

b.- Elimine la tabla.

D. SP	#S	#P	CTD

4.6.6.- Ampliación de una tabla. Adicione una columna FECHA a la tabla SP.

QBE, no soporta directamente la adición dinámica de una columna nueva a una tabla existente, a menos que la tabla esté actualmente vacía. Por tanto, es necesario hacer lo siguiente:

1. Definir una tabla nueva de la misma forma de la tabla existente más la columna nueva.
2. Cargar la tabla nueva a partir de la vieja, usando una inserción de registros múltiples.
3. Suprimir todos los datos de la tabla vieja.
4. Eliminar la tabla vieja.
5. Cambiar el nombre de la tabla nueva por el de la tabla vieja.

Pasos 1 y 2:

SP	**#S**	**#P**	**CTD**
	SX	PX	QX

I.SPCOPIA I.	**#S**	**#P**	**CTD**	**FECHA**
DOMAIN				U.DATES
TYPE				U.DATE
KEY			U.N	U.N
INVERSION			U.N	U.N
I.	SX	PX	QX	

Paso 3:

SP	**#S**	**#P**	**CTD**
D.			

Paso 4:

D.SP	**#S**	**#P**	**CTD**

Paso 5:

I.SPCOPIA I.	**#S**	**#P**	**CTD**	**FECHA**

Todos los valores de FECHA en la nueva tabla SP son inicialmente nulos.

4.7. NUEVAS TENDENCIAS EN QBE

Actualmente se han desarrollado nuevas formas de consulta de la información basadas en los conceptos que se derivan de los lenguajes del tipo de QBE. En las nuevas bases de datos documentales o multimedia, se han implementado lenguajes de consulta basados en lenguajes formales, pero últimamente se están desarrollando nuevos lenguajes más orientados a la búsqueda mediante ejemplos, que se asemejan más a la forma natural de expresar las necesidades de información.

En los sistemas tradicionales de almacenamiento y recuperación de la información (bases de datos documentales), las preguntas se expresaban mediante un conjunto de términos que deben contener los documentos, que podían estar relacionados mediante operadores lógicos (AND, OR, NOT,...). Actualmente se pretende realizar la búsqueda mediante ejemplos, el usuario propone un documento relevante a la necesidad de información del usuario y el sistema busca documentos que se encuentren cercanos al documento propuesto por el usuario. También se usa este sistema para refinar la respuesta a una pregunta realizada con anterioridad.

En las bases de datos multimedia también se pretende realizar búsquedas basadas en ejemplos, proponemos una música, un video o una imagen y el sistema busca componentes que se encuentren cercanos al que damos por ejemplo. Para realizar esto se están implementando técnicas de almacenamiento de componentes multimedia mediante fractales.

Por tanto, actualmente los lenguajes basados en búsquedas por ejemplos están experimentando un auge, debido a las nuevas necesidades de búsqueda de información que se derivan de los sistemas multimedia, hipermedia, y a la gran revolución que ha supuesto Internet.

5.- EJEMPLOS QBE.

A continuación mostramos algunos ejemplos de cada una de las operaciones que se han descrito en el apartado anterior.

5.1.-Ejemplos de QBE sobre una relación basada en ESTUDIANTES DE INFORMÁTICA DE GESTIÓN.

El esquema de la relación es el siguiente:

ESTUDIANTES	ID	NOMBRE	CARRERA

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN

Ejemplo 1: “id y nombre de los estudiantes de ingenieros informáticos de gestión”.

ESTUDIANTES	ID	NOMBRE	CARRERA
	P._x	P._ y	ig

Ejemplo 2: “obtener los datos de todos los estudiantes”.

ESTUDIANTES	ID	NOMBRE	CARRERA
	P._x	P._ y	P._ z

o también

ESTUDIANTES	ID	NOMBRE	CARRERA
P.			

Ejemplo 3: “ Obtener los nombres de los estudiantes de informática de gestión”. Incluyendo duplicados.

ESTUDIANTES	ID	NOMBRE	CARRERA
		P.ALL.	ig

Ejemplo 4: "Obtener los id's de los estudiantes con número 441 y 323".

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN
	P._x _x	ig ig	441 323	

Ejemplo 5: "Obtener los nombres de los estudiantes de informática de gestión que han reprobado algún curso".

ESTUDIANTES	ID	NOMBRE	CARRERA
	_x	P.ALL.	ig

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN
	_x			< 7.5

Ejemplo 6: " Obtener los id's de estudiantes que han tomado cursos con el estudiante 123".

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN
	123 P._x	_ y _ y	_z _ z	

Ejemplo 7: " Obtener los nombres de los estudiantes que no son empleados".

ESTUDIANTES	ID	NOMBRE	CARRERA
	_x	P.ALL.	

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN
¬	_x			

Ejemplo 8: " Obtener el promedio de las calificaciones del estudiante 666 en cursos que no son de informática de gestión".

EST_CURSOS	ID	DPTO	NUM	CALIFICACIÓN
	666	¬ ig		P.AVG.ALL.

5.2.-Ejemplos de QBE sobre una relación basada en UNA ESTACIÓN FERROVIARIA.

Para ello necesitamos el siguiente esquema de relación:

RED	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	ESTACIÓN SIGUIENTE	NUM-LÍNEA

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN

LÍNEA	NUM-LÍNEA	RANGO	ESTACIÓN

TRÁFICO	NUM-TREN	NUM-LÍNEA	NUM-DÍA

TREN	NUM-TREN	NUM-VAGÓN

TRANSPORTE	CLIENTE	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	NUM-VAGÓN	TIPO-MERCANCIA

PESO-MERCANCIA	FECHA-CARGA

Ejemplo 1*:* suponemos que queremos responder a la pregunta " dar la lista de los vagones de tipo "frigo" que están disponibles en la estación de *tours* y cuya capacidad es superior a 10 T ".

Se trata de una pregunta típica de una expresión de selección sobre una relación. Si nos referimos a nuestro esquema relacional, tenemos que traer a la pantalla la relación cuyo nombre es VAGON. El usuario dispone para ello de una instrucción que le permite hacer que en pantalla aparezca la estructura. La forma de esta instrucción es P, que se refiere a la impresión(print en inglés).Una vez que tenemos la estructura en pantalla el usuario prepara su petición construyendo un criterio de calificación utilizando diferentes columnas. Si quiere seleccionar

las n-tuplas que tengan una cierta propiedad en un constituyente dado, colocará en la columna correspondiente el valor asociado a la condición. Si la condición requiere un operador que no la igualdad, esto es, <, >.≥, ≤, escribirá ese operador delante del valor. Si hay varias condiciones unidas entre si mediante un y lógico, las colocará en la misma línea. Así, en nuestro caso, las columnas TIPO-VAGON,CAPACIDAD, ESTADO y ESTACIÓN.

A continuación el usuario tiene que indicar que información quiere obtener a partir de las n-tuplas seleccionadas. Si quiere imprimir toda la n-tupla, colocará la letra P en la misma línea que las condiciones, y en la columna que corresponde al nombre de la relación. Si, por el contrario, solo quiere imprimir algunos valores de la relación seleccionada, indicará la letra P en las columnas correspondientes.

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
		frigo		>=10	libre	tours

Esta descripción equivale a una operación de selección seguida de una operación de proyección. La operación de selección está limitada a algunas formas de selección.

Ejemplo 2:" dar los prototipos de los vagones del tren 4002".

Este ejemplo nos permite ilustrar otra propiedad fundamental del lenguaje, esto es la forma de expresión de los nexos que hay entre varias relaciones.

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
		-w 106	P			

TREN	NUM-TREN	NUM-VAGÓN
	4002	-w 106

La idea básica para responder a esta pregunta consiste en seleccionar en la realción TREN las n-tuplas que corresponden al tren 4002. Para ello indicamos el valor 4002 en la columna correspondiente. Además, queremos seleccionar una n-tupla cuyo número de tren sea 4002, pero que tenga además un numero de vagón, por ejemplo w106, de forma que podamos seleccionar la relación VAGON los vagones que tengan este mismo número. Sólo nos falta imprimir el tipo de vagón, que es lo que indicamos mediante la letra P. Tenemos que señalar que hay una diferencia esencial entre el valor 4002 y w106. El primero corresponde a una constante , mientras que el segundo corresponde aun ejemplo de un valor tomado por una variable, razón por la que hablaremos de **variable ejemplo**. Para distinguir los dos casos, el segundo valor va precedido de un **signo de subrayado.** De hecho, podemos considerar w106 como una variable que establece un nexo entre las dos relaciones. Si examinamos la expresión correspondiente en un lenguaje predicativo con variables dominio

$\{t \mid \exists w \exists p \exists c \exists e \exists g \text{ (TREN (“4002”) } \Lambda \text{ VAGON(w,t, p, c, e, g))}\}$

vemos claramente que w es una variable común y que 4002 es una constante.

Con este ejemplo, apreciamos como podemos resolver un gran numero de peticiones de la misma clase expresadas en un lenguaje predicativo con variables de dominio.

Las expresiones de esta clase tienen la forma:

$$\{a_1, a_2,..., a_k \mid \exists b_1 \exists b_2 ... \exists b_p\ (R_1\ (C_{11},..., C_{1i1}) \Lambda\ R_2(C_{21},..., C_{2i2}) \Lambda\ ...R_n(c_{n1},..., c_{nip}))\}$$

en donde cada c_{ij} es un a, un b, o una constante y cada a y cada b aparece al menos una vez.

Ejemplo 3: " Dar la lista de las líneas que salen de la estación de tours".

En calculo relacional de dominios se expresa mediante la expresión:
$\{1 \mid \exists w \exists p \exists c$ (LÍNEA (1, w, "tours") Λ LÍNEA(1, p, c) Λ (w > p))}.

Para resolver esta petición en QBE hay que llamar a la estructura de la relación LÍNEA y construir dos líneas que corresponden a dos n-tuplas. En la primera línea, que corresponde a la primera n-tupla, indicaremos que debe verificar la condición ESTACIÓN = "tours". Además, tendrá que verificarse que las dos n-tuplas posean la misma línea, y que el rango de la segunda sea mayor que el de la primera. Para expresar esto utilizamos dos variables x e y, precedidas por el símbolo de subrayado. Finalmente, ña instrucción P en la columna NUM-LÍNEA nos permitirá imprimir los valores de esta columna.

LÍNEA	NUM-LÍNEA	RANGO	ESTACIÓN
	_x P._x	_y >_y	tours

Ejemplo 4:" Dar la lista de los trenes que saldrán de tours el 14.03.00".

Para responder a esta pregunta necesitamos de las estructuras de LÍNEA y TRÁFICO.

LÍNEA	NUM-LÍNEA	RANGO	ESTACIÓN
	_x _x	_y >_y	tours

TRÁFICO	NUM-TREN	NUM-LÍNEA	NUM-DÍA
	P.	_x	180300

Ejemplo 5: " Dar la lista de vagones que saldrán de tours el 18.03.00 y cuya estación de destino final es Beziers".

Utilizaremos las estructuras de las relaciones LINEA, TAFICO, TREN y TRANSPORTE.

LÍNEA	NUM-LÍNEA	RANGO	ESTACIÓN
		_y	Beiziers

TRÁFICO	NUM-TREN	NUM-LÍNEA	NUM-DÍA
	_t	_x	180300

TREN	NUM-TREN	NUM-VAGÓN
	_t	- w

TRANSPORTE	CLIENTE	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	NUM-VAGÓN	TIPO-MERCANCIA
			Beizbers	P._w	

PESO-MERCANCIA	FECHA-CARGA

Ejemplo 6: " Dar la lista de vagones que están cargados hasta la mistad en la estación de tours".

Para expresar esta petición en este tipo de lenguaje utilizaremos la estructura de las relaciones de TRANSPORTE y VAGON.

TRANSPORTE	CLIENTE	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	NUM-VAGÓN	TIPO-MERCANCIA
				_w 106	

PESO-MERCANCIA	FECHA-CARGA
P._x	

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
	P._w106			P._y		tours

CONDICIÓN

x < = y / 2

En las estructuras de cada una de estas relaciones tendremos que indicar los valores de cada n-tupla, y en una tabla adicional que llamamos CONDICION indicaremos la relación qye debe unir a las variables x e y , que en este caso es x <= y / 2, àra que se seleccione una n-tupla dada. Sólo se seleccionarán las n-tuplas que satisfacen la condición, además de las otras condicones que figuran en las estructuras.

Ejemplo 7: " cúal es la estación de llegada de la línea 10".

Si utilizamos la instrucción de negación, utilizaremos la estructura de la relación LINEA, señalando que si una n-tupla r contiene la estación de llegada, no puede existir otra n-tupla p tal que el rando de la n-tupla p sea mayor que el rango de la n-tupla r. Las dos líneas que figuran en el estructura representan las dos n-tuplas r y p.

LÍNEA	NUM-LÍNEA	RANGO	ESTACIÓN
	10	_x	P.
¬	10	>_x	

Ejemplo 8: "Dar los números de las líneas en las que ahy tren todos los días".

TRÁFICO	NUM-TREN	NUM-LÍNEA	NUM-DÍA
		P._1	ALL_ j
			ALL_ j

Ejemplo 9: " Dar la lista de las estaciones por las que pasará un vagón que vaya de *angers* a *beziers*".

RED	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	ESTACIÓN SIGUIENTE	NUM-LÍNEA
	angers	beziers	_g1	
	_g1	beziers	P._g2	

Si queremos obtener la 3ª estación, bastaría con introducir otra línea más.En general, si queremos obtener la n-ésima estación, basta con introducir n líneas. En estructura de arbole se puede resolver de la siguiente manera:

RED	ESTACIÓN ORIGEN	ESTACIÓN DESTINO	ESTACIÓN SIGUIENTE	NUM-LÍNEA
	angers	beziers	P._g(2L)	

Ejemplo 10: " Dar para todos los trenes el número de vagon de cada tren".

TREN	NUM-TREN	NUM-VAGÓN
	COUNT.ALL._x	P._y

Podemos interpretarlo de esta forma: Para cada variable, definimos el conjunto de los vagones de este tren mediante el operador ALL sobre el que aplicamos el operador COUNT. A continuación imprimimos los resultados.

Ejemplo 11: " Dar la lista de los números de los trenes que tienen más de 20 vagones".

TREN	NUM-TREN	NUM-VAGÓN
	P. t	COUNT.ALL._w

CONDICIÓN
COUNT.ALL._ w>= 20

Ejemplo 12: Para expresar la inserción de un nuevo vagón en la relación lo haremos de la siguiente forma:

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
INSERT	w105	frigo	10	50	libre	tours

Ejemplo 13: " suponemos que queremos cambiar la estación de un vagón cuyo número es w105 y sustituir tours por blois".

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
UPDATE	w105					blois

Ejemplo 14: Expresarmos la supresión del vagón cuyo número es w 105 simplemente mediante:

VAGÓN	NUM-VAGÓN	TIPO-VAGÓN	PESO-VAGÓN	CAPACIDAD	ESTADO	ESTACIÓN
DELETE	w105					

5.2.-Ejemplos de QBE sobre una relación basada en UNA SUCURSAL BANCARIA.

Para ello necesitamos el siguiente esquema de relación:

SUCURSAL	NOMBRE-SUCUR	CIUDAD	ACTIVO

CLIENTE	NOM-CLI	CALLE-CLI	CIUDAD-CLI

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE

PRESTATARIO	NOM-CLI	NUM-PRESTAMO

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO

IMPOSITOR	NOM-CLI	NUM-CUENTA

Ejemplo 1: " Obtener todos los números de prestamo de la sucursal Navacerrada"

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE
	Navacerrada	P._x	

Ejemplo 2: " Encontrar todos los números de prestamos de aquellos prestamos con una cantidad mayor a 140.000 ptas".

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE
		P.	> 140.000

Ejemplo 3: " Obtener todos los nombres de las sucursales que no tienen sede en Barcelona".

SUCURSAL	NOMBRE-SUCUR	CIUDAD	ACTIVO

	P.	¬ Barcelona	

Ejemplo 4: " Obtener los nombres de todos los clientes que tienen un prestamo en la sucursal de Navacerrada".

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE
	Navacerrada	_x	

PRESTATARIO	NOM-CLI	NUM-PRESTAMO
	P._y	_x

Para ejecutar esta consulta, primero localizamoslas tuplas en la relación prestamo que tienen el atributo *nombre sucursal* igual a " Navacerrada". Para cada una de estas tuplas el sistema busca las tuplas de la relación prestatario con el mismo valor para el atributo *número prestamo* que el mismo atributo de la tupla de la relación prestamo. Finalmente se muestra el valor del atributo *nombre del cliente* de todas las tuplas de la relación que cumplan las condiciones.

Ejemplo 5: " Obtener los nombres de todos los clientes que tienen una cuenta en el banco pero que no tienen un prestamo en el mismo".

IMPOSITOR	NOM-CLI	NUM-CUENTA
	P._x	

PRESTATARIO	NOM-CLI	NUM-PRESTAMO
¬	_x	

Ejemplo 6: " Obtener todos los números de cuenta con saldos entre 260.000 y 300.000 ptas".

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO
		P.	_x

CONDICION
_x ≥ 260.000
_x ≤ 300.000

Ejemplo 7: " Obtener todas las sucursales con activos superiores al activo de al menos una sucursal con sede en barcelona".

SUCURSAL	NOMBRE-SUCUR	CIUDAD	ACTIVO
	P._x	P._x Barcelona	_y _z

CONDICION
_y > z

Ejemplo 8: " Obtener el nombre de los clientes, el número de cuenta y el saldo de todas las cuentas de las sucursal de Navacerrada".

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO
	Navacerrada	_y	_ z

IMPOSITOR	NOM-CLI	NUM-CUENTA
	_x	_y

RESULTADO	NOM-CLI	NUM-CUENTA	SALDO
P.	_x	_y	_z

Creamos una tabla resultado, donde aparecerán los nombres de todos los atributos.

Ejemplo 9: " Obtener el saldo total de todas las cuentas de la sucursal Navacerrada".

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO
	Navacerrada		P.SUM.ALL

Con el operador ALL. evitamos que se eliminen los duplicados.

Ejemplo 10: " Obtener todos los clientes que tienen cuenta en cada una de las sucursales con sede en Barcelona".

IMPOSITOR	NOM-CLI	NUM-CUENTA
	P.G. _x	_y

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO
	CNT.UNQ.ALL._z	_y	

SUCURSAL	NOMBRE-SUCUR	CIUDAD	ACTIVO
	_z	Barcelona	
	_w	Barcelona	

CONDICION
CNT.UNQ.ALL._z = CNT.UNQ.ALL._w

La variable de dominio w puede tomar el valor de nombres de sucursales con sede en Barcelona. Así , CNT.UNQ.ALL._w es el número de sucursales distintas de Barcelona.

Ejemplo 11: " Borrar todos los prestamos con cantidades comprendidas entre 260.000 y 300.000 ptas".

PRESTAMO	**NOMBRE-SUCUR**	**NUM-PRESTAMO**	**IMPORTE**
D.		_y	_x

PRESTATARIO	**NOM-CLI**	**NUM-PRESTAMO**
D.		_y

CONDICION
_x =(≥ 260.000 **and** ≤ 300.000)

6. QBE vs SQL

En la tabla siguiente exponemos las principales diferencias que exiten entre estos 2 lenguajes de consulta:

QBE	SQL
Basado en el cálculo relacional de dominios	Basado en el álgebra relacional
Programación visual	Programación mediante lenguaje formal
Sintaxis bidimensional basado en tablas	Sintaxis unidimensional basado en un lenguaje formal
Consultas mediante ejemplos	Consultas mediante condiciones lógicas

A continuación exponemos algunos ejemplos que ilustran las diferencias:

- " Encontrar todos los números de prestamos de aquellos prestamos con una cantidad mayor a 140.000 ptas".

QBE

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE
		P.	> 140.000

SQL

SELECT NUM-PRESTAMO
FROM PRESTAMO
WHERE IMPORTE > 140.000

- " Borrar todos los prestamos con cantidades comprendidas entre 260.000 y 300.000 ptas".

QBE

PRESTAMO	NOMBRE-SUCUR	NUM-PRESTAMO	IMPORTE
D.		_y	_x

PRESTATARIO	NOM-CLI	NUM-PRESTAMO
D.		_y

CONDICION
_x =(≥ 260.000 **and** ≤ 300.000)

SQL

DELETE *
FROM PRESTAMO
WHERE IMPORTE **BETWEEN** 260.000 **AND** 300.000

- " Obtener el saldo total de todas las cuentas de la sucursal Navacerrada".

QBE

CUENTA	NOMBRE-SUCUR	NUM-CUENTA	SALDO
	Navacerrada		P.SUM.ALL

SQL

SELECT SUM(SALDO) AS SUMA
FROM CUENTA
WHERE NOMBRE-SUCUR = 'Navacerrada'
GROUP BY NOMBRE-SUCUR

BIBLIOGRAFÍA.

- Fernandez Baizán , C., " El modelo relacional de datos", Diaz de Santos, [1987].

- Delobel,C., Adiba, M. " Bases de datos y sistemas relacionales " , Omega [1987].

- Malpica, J., Vargas Villazón, A., " Introducción al los sistemas de Bases de Datos", Addison-Wesley [1975].

- Elmasri, R., Navathe, S., " Sistemas de Bases de Datos", Addison-Wesley [1997].

- Hursch , C.J., Hursch, J.L., " SQL. El lenguaje de consulta estructurado", Rama [1989].