TEMA 39

LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS DE BASES DE DATOS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

ÍNDICE

1. INTRODUCCIÓN

Los lenguajes de consulta formal son determinantes a la hora de especificar las posibles consultas que se pueden realizar sobre una base de datos relacional, ya que éstas se pueden resolver mediante una secuencia de operaciones relacionales perfectamente coordinadas, que una vez aplicadas a la Base de Datos producen la respuesta deseada, o mediante la obtención de los predicados deseados aplicando operaciones sobre los datos que se disponen.

Para poder realizar consultas a una Base de Datos se precisa de un lenguaje interrogativo de alto nivel que actúe a modo de interfaz entre el usuario de la Base de Datos y dicha Base de Datos, de modo que el usuario, a través del lenguaje indique que pasos hay que seguir para resolver una determinada consulta/actualización, pero no cómo deben efectuarse dichos pasos.

Existen varios lenguajes interrogativos que son capaces de actuar de interfaz entre la Base de Datos y el usuario, estos lenguajes emplean como soporte de acceso a los datos un lenguaje de interrogación formal.

La publicación de los artículos de Codd en los que se introdujo el modelo relacional y los lenguajes relacionales provocó un gran impacto en las comunidades comerciales y de investigación para desarrollar versiones implementadas de los lenguajes relacionales. Los tres lenguajes más importantes que resultaron de este esfuerzo son SQL (Structured Query Language), QBE (Query-By-Example) y QUEL (Query Language). SQL y QBE fueron creados por IBM en los años setenta y realizan funciones muy similares, aunque SQL es un lenguaje textual y QBE es gráfico. QUEL es el lenguaje original de INGRES, un sistema de gestión de base de datos relacional desarrollado en los setenta por la Universidad de Berkeley, en California.

Cada uno de estos lenguajes cuales representa distintos estilos:

- **SQL** está basado en el cálculo relacional de dominios.
- **QUEL** está basado en el cálculo relacional de tuplas.
- **QBE** está basado en el álgebra relacional y construcciones del cálculo relacional.

Estos tres lenguajes no sólo permiten consultar las Bases de Datos, sino que también pueden realizar otras tareas, como definir y modificar la estructura de los datos, así como establecer los criterios de seguridad de los datos.

2. LENGUAJE SQL

Dado que en el mercado existe una clara tendencia a las Bases de Datos Relacionales, vamos a ver el lenguaje SQL, que es el lenguaje comercial más conocido.

En 1986 fue aprobado el primer estándar ANSI (American National Standars Institute) para SQL. Este estándar fue revisado modestamente en 1989 y significativamente en 1992. SQL ha quedado como el único lenguaje relacional de base de datos que es ANSI estándar. Desde 1980 numerosos fabricantes han lanzado implementaciones de SQL.

Aunque el nombre SQL sugiere que éste es un lenguaje de “consulta”, además de las facilidades de consulta, incluye la definición de tablas, la actualización de la base de datos, definición de vistas y otorgamiento de privilegios. Consiste en un conjunto de instrucciones para definir, manejar y controlar los datos en una Base de Datos Relacional. El software relacional está específicamente diseñado para facilitar la manipulación de los datos en forma de tablas. Es similar al lenguaje inglés. Puede actuar como un lenguaje autocontenido (de consulta interactiva) y se puede incorporar a lenguajes convencionales como Cobol. Es un lenguaje no procedimental, es decir, **los usuarios especifican el qué, no el cómo**.

Actúa sobre un conjunto de registros. La “navegación” es automática y es relacionalmente completo. En SQL las peticiones de acceso a los datos se expresan mediante comandos que se escriben según sus reglas sintácticas y semánticas.

2.1. Componentes de SQL

Este lenguaje consta de los siguientes elementos:

Diccionario de Datos.
Lenguaje de Descripción de Datos (LDD).
Lenguaje de Manipulación de Datos (LMD).
Lenguaje de Control de Datos (LCD).

El **Diccionario de Datos**, se llama también catálogo o repositorio de datos. Es un fichero que contiene "metadatos", es decir, datos acerca de los datos. Este fichero se consulta antes de leer o modificar los datos reales en un Sistema de Bases de Datos. Cada vez que se añada, modifique o borre información sobre la Base de Datos, el diccionario se actualizará para reflejar esos cambios.

El **Lenguaje de Descripción de Datos** (**LDD** o DDL), se usa para crear y definir las tablas de las Bases de Datos, así como los nombres y los tipos de las columnas de las tablas. También se puede usar para crear y definir índices sobre una tabla, y tablas virtuales o vistas.

Por *ejemplo*, una definición de una Base de Datos sería de la siguiente forma:

```
CREATE TABLE Instituto
(Código-instituto Códigos, Nombre CHAR(30) NOT NULL, Dirección Lugares, .... )
PRIMARY KEY IS Código-instituto;
```

Algunos *comandos* de LDD son:

CREATE TABLE
CREATE VIEW
CREATE INDEX
DROP TABLE
DROP VIEW
DROP INDEX
ALTER TABLE

Con esta parte del SQL se crean, modifican y borran las definiciones de tablas de la Base de Datos.

También permite la creación y borrado de vistas e índices.

La creación de tablas debe preceder lógicamente a la inserción y manipulación de los datos contenidos en ellas. Normalmente, la definición de los objetos que forman la Base de Datos es tarea del administrador de la Base de Datos.

Para **definir un esquema** de base de datos sólo se necesita identificar el comienzo de la definición con una instrucción CREATE SCHEMA y una cláusula adicional AUTHORIZATION y a continuación definir cada dominio, tabla, vista, etc. (SQL-92).

Una **definición de dominio** es un tipo de datos especializado definido dentro de un esquema y usado en las definiciones de las columnas.

Las tablas se definen en tres pasos:

1. Dar el nombre de la tabla
2. Definir cada columna
3. Definir las restricciones de la tabla

Ejemplo. A continuación se da parte de una definición de esquema para cierta base de datos:

```
CREATE SCHEMA CONSTRUCTORA_PREMIER
    AUTHORIZATION TONY_MELTON

CREATE DOMAIN IDENTIFICADOR NUMERIC (4) DEFAULT 0
    CHECK (VALUE IS NOT NULL)

CREATE TABLE TRABAJADOR (
    ID_TRABAJADOR       IDENTIFICADOR       PRIMARY KEY,
    NOM_TRABAJADOR      CHARACTER (12),
    TARIFA_HR           NUMERIC (5, 2),
    OFICIO              CHARACTER (8),
    ID_SUPV             NUMERIC (4),
FOREIGN KEY ID_SUPV REFERENCES TRABAJADOR
                                ON DELETE SET NULL)
```

La instrucción CREATE TABLE identifica el nombre de la tabla, que debe ser única dentro del esquema. Después de CREATE TABLE van encerradas entre paréntesis y separadas por coma las instrucciones de definir columnas y de definir restricciones sobre la tabla.

Cada columna se define dando su nombre, su tipo de dato, cuál es su valor por defecto y cuándo se aplican restricciones específicas (por ejemplo, NOT NULL, PRIMARY KEY y restricciones CHECK).

En las tablas también se pueden aplicar restricciones. La restricción de clave externa de la tabla TRABAJADOR:

```
FOREIGN KEY ID_SUPV REFERENCES TRABAJADOR
                                ON DELETE SET NULL
```

indica que ID_SUPV es una clave externa recursiva (clave externa que referencia a su propia relación).

Junto con la instrucción CREATE TABLE que define una nueva tabla, SQL ofrece otras instrucciones para cambiar las definiciones de las tablas (ALTER TABLE) o para borrar las tablas del esquema (DROP TABLE). ALTER TABLE puede usarse para añadir una columna a una tabla, cambiar la definición de una columna existente o eliminar una columna de una tabla. DROP TABLE borrará todas las filas de la tabla y quitará del esquema la definición completa de la tabla. Un esquema completo puede eliminarse mediante la instrucción DROP SCHEMA.

El **Lenguaje de Manipulación de Datos** (**LMD** o DML), es el aspecto más potente del SQL, puesto que las sentencias del LMD operan sobre conjuntos enteros de filas, no de una en una. Esto es algo que los lenguajes de programación de más alto nivel son incapaces de hacer. SQL contiene una gran variedad de capacidades de manipulación de datos, tanto para consulta como para actualización de una base de datos. Estas capacidades dependen sólo de la estructura lógica de la base de datos, no de su estructura física, consistente con los requisitos del modelo relacional.

La clave del SQL, es pues la diferencia entre cómo recuperar la información y qué datos queremos recuperar. En vez de especificar el cómo, como en otros lenguajes de Bases de Datos, en SQL, se especifica qué datos queremos recuperar, y el Sistema de Gestión de Bases de Datos relacional, hace el resto. El usuario pues, navega por medio de las órdenes dadas por la Base de Datos “física”, puesto que el sistema los realizará automáticamente

Algunos *comandos* de LMD son:

INSERT
UPDATE
DELETE
SELECT, etc.

Con esta parte se realizan las siguientes *operaciones*:

Consultas: SELECT
Actualización: Inserción, Borrado y Modificación: INSERT, DELETE, UPDATE.

Consultas simples

Una consulta simple es la que afecta a una sola tabla de la base de datos. Por ejemplo:

Consulta: ¿Qué profesores dan Informática?

```
SELECT Nombre_profesor
FROM Instituto
WHERE Especialidad = “Informática”;
```

Esta consulta ilustra las tres cláusulas más usadas del SQL:

Select. La **cláusula SELECT** indica las *columnas* que se desean en el resultado de la consulta.

From. La **cláusula FROM** lista una o más *tablas* que van a ser referidas en la consulta.

Where. La **cláusula WHERE** contiene una *condición* para seleccionar las filas de las tablas que se dan en la cláusula FROM.

Consultas multi-tablas

La capacidad para conectar datos sobre las fronteras de las tablas es esencial en cualquier lenguaje de base de datos. En el álgebra relacional esto se lleva a cabo con la reunión (join). Aunque una gran parte del SQL se ha modelado después del cálculo relacional, éste conecta los datos entre tablas de la misma manera que la reunión (join) del álgebra relacional. *Ejemplo*:

Consulta: ¿Cuáles son los oficios de los trabajadores asignados al edificio 435?

Los datos necesarios para responder a esta consulta se encuentran en dos relaciones: TRABAJADOR y ASIGNACIÓN. La solución SQL requiere poner ambas relaciones en la cláusula FROM junto con un tipo particular de condición de cláusula WHERE:

```
SELECT OFICIO
FROM TRABAJADOR, ASIGNACIÓN
WHERE TRABAJADOR.ID_TRABAJADOR = ASIGNACIÓN.ID_TRABAJADOR
        AND ID_EDIFICIO = 435
```

La primera de las condiciones es la reunión (join).

Subconsultas

Una subconsulta, o una consulta dentro de una consulta, puede ponerse dentro de la cláusula WHERE de una consulta. Esto produce una expansión de las capacidades de una cláusula WHERE. *Ejemplo*:

```
SELECT OFICIO
FROM TRABAJADOR
WHERE ID_TRABAJADOR IN
        (SELECT ID_TRABAJADOR
        FROM ASIGNACIÓN
        WHERE ID_EDIFICIO = 435)
```

EXITS y NOT EXITS

Los operadores EXITS y NOT EXITS siempre preceden a una subconsulta. **EXITS** evalúa verdadero si el conjunto resultante de la subconsulta no es vacío. Si no da valor falso. El operador **NOT EXITS** trabaja de modo opuesto. Este evalúa verdadero si el conjunto resultante es vacío y falso en caso contrario.

Funciones integradas

Son funciones estadísticas que operan sobre un conjunto de filas: **SUM**, **AVG**, **COUNT**, **MAX** y **MIN**.

Operaciones del álgebra relacional

Algunas de las que ha implementado SQL-92 son las siguientes:

- **UNION**. Crea el conjunto unión de dos relaciones.
- **INTERSECT**. Crea el conjunto intersección de dos relaciones.
- **EXCEPT**. Crea el conjunto diferencia entre dos relaciones.
- **NATURAL JOIN**. Conecta las relaciones cuando las columnas comunes tienen valores iguales.

Operaciones de modificación de la base de datos

INSERT. Permite insertar en una relación una fila mediante la especificación de los valores de cada una de las columnas de la fila, o insertar un grupo de filas especificando una consulta que nos dará el grupo de filas a insertar. *Ejemplo*:

```
INSERT INTO ASIGNACIÓN (ID_TRABAJADOR, ID_EDIFICIO, FECHA_INICIO)
VALUES (1284, 485, 13/05)
```

UPDATE. Se aplica a todas las filas que satisfacen la cláusula WHERE de la instrucción UPDATE. *Ejemplo*, para incrementar un 5% el salario de todos los trabajadores del supervisor 1520, se necesitaría la siguiente instrucción:

```
UPDATE TRABAJADOR
SET TARIFA_HR = 1.05 * TARIFA_HR
WHERE ID_SUPV = 1250
```

DELETE. Se aplican también a todas las filas que satisfacen la cláusula WHERE en la instrucción DELETE. Si no hay cláusula WHERE, se borran todas las filas de la relación. Ejemplo:

```
DELETE FROM TRABAJADOR
WHERE ID_SUPV = 1250
```

Usar SQL con lenguajes de procesamiento de datos

El **SQL empotrado** (embedded SQL) es un conjunto de instrucciones que permite que SQL sea utilizado con lenguajes de programación tradicionales como COBOL, C y Pascal (que se denominan lenguajes anfitrión). Estas instrucciones notifican al preprocesador que lo que sigue se debe reemplazar con llamadas a las rutinas del SGBD. Consta de: todos los comandos de SQL y comandos de Control que integran los comandos del SQL con el lenguaje de programación procedimental.

Definición de vistas

Una **vista** es como una *ventana* a una porción de la base de datos. Las vistas son útiles para mantener la confidencialidad al restringir el acceso a partes seleccionadas de la base de datos y para simplificar tipos de consultas que sean utilizados con frecuencia. Por *ejemplo*, para preservar la confidencialidad, podemos querer crear una vista que muestre la información sobre los trabajadores, excepto su tarifa por horas:

```
CREATE VIEW B_TRABAJADOR
AS SELECT ID_TRABAJADOR, NOMB_TRABAJADOR, OFICIO, ID_SUPV
      FROM TRABAJADOR
```

El **Lenguaje de Control de Datos** (**LCD** o DCL) contiene sentencias que tienen que ver con el control de datos, es decir, sobre el acceso y la utilización de la Base de Datos. Normalmente es el administrador de la Base de Datos el encargado de conceder y revocar autorizaciones de acceso a los datos. Por *ejemplo*:

```
LOCK TABLE Instituto
IN EXCLUSIVE MODE; bloquear la tabla Instituto.
```

Algunos comandos de LCD son:

```
COMMIT
ROLLBACK
GRANT
REVOKE
LOCK
UNLOCK, etc.
```

El acceso a la Base de Datos se controla mediante los comandos GRANT y REVOKE que permiten decidir quién puede acceder a una tabla y cómo puede hacerlo. La integridad de los datos se controla mediante los comandos COMMIT, ROLLBACK, LOCK y UNLOCK.

El lenguaje de Descripción de Datos (LDD) lo tenemos que situar entre el nivel conceptual y el nivel interno, puesto que su función es gestionar los ficheros existentes y proceder a presentarlos conforme a las limitaciones puestas por nosotros como usuarios. Podemos considerar al LDD como un tipo de sublenguaje dentro del lenguaje que tratamos en sí.

Cada **vista externa** se define por medio de un esquema externo, el cual se compone en esencia de las definiciones de cada uno de los diversos tipos de registros externos de esa vista externa. Cualquier cantidad de vistas externas pueden existir al mismo tiempo; cualquier número de usuarios puede compartir una vista externa específica. Algunos sistemas permiten que la definición de una vista externa se exprese en términos de otras, en vez de requerir siempre una definición explícita.

El **esquema externo** se describe usando la parte de LDD del sublenguaje de datos. Ese LDD, por tanto, se llamará en algunas ocasiones **LDD externo**. Además, debe haber una *correspondencia* entre el esquema externo y el esquema, o nivel, conceptual subyacente.

La **vista conceptual** se define por medio del **esquema conceptual**, el cual incluye definiciones de cada uno de los distintos tipos de registros conceptuales. El esquema conceptual se describe utilizando otro lenguaje de definición de datos; el **LDD conceptual**.

Si se desea lograr independencia de los datos, estas definiciones no deben incluir ninguna consideración sobre la estructura de almacenamiento, ni sobre la estrategia de acceso. Deben ser definiciones de contenido de información únicamente.

El tercer nivel de la arquitectura es el interno. La **vista interna** es una representación de muy bajo nivel de la Base de Datos. Se compone de múltiples ocurrencias de múltiples tipos de registros internos,

que es el término que el comité ANSI/SPARC utiliza para la construcción de un llamado *registro de almacenado*.

La vista interna supone en esencia, un espacio direccionado lineal infinito. Los detalles de la manera en que este espacio de direcciones corresponde al almacenamiento físico, son muy específicos de realización, por lo que no se describen de modo explícito en la arquitectura. La vista interna se describe por medio del **esquema interno**, el cual no sólo define los diversos tipos de registros almacenados, sino que también especifica qué índices existen, de qué manera se representan los campos almacenados, en qué secuencia física se hallan los registros almacenados, etc. El esquema interno se describe usando otro lenguaje de definición de datos, el interno, o **LDD interno**.

Todo lo descrito anteriormente, nos plantea la obligación de que existan dos niveles de correspondencia, uno entre los niveles externo y conceptual de la Base de Datos, y otro entre los niveles conceptuales e interno de la misma.

La correspondencia conceptual/interna está definida entre una vista externa específica y la vista conceptual. En general puede existir la misma clase de diferencias entre estos niveles, y entre la vista conceptual y la Base de Datos almacenada.

Al hablar de SQL no hay que olvidar su utilización del álgebra relacional, la cual se caracteriza por:

- Es un conjunto de operaciones que se realiza sobre las relaciones o tablas.
- El resultado de una operación de álgebra relacional es otra relación o tabla (propiedad de cierre).
- El álgebra relacional es la base de todas las manipulaciones de la Base de Datos. Sin embargo, SQL utilizará su propia sintaxis para efectuar esas operaciones.
- Sirve como punto de referencia en la comparación de lenguajes relacionales.

Las *operaciones* más importantes a realizar serían:

Restricción: Selecciona algunas o todas las filas de una tabla existente.
Proyección: Selecciona algunas o todas columnas de una tabla existente.
Unión: Combina alguna o todas las columnas y filas de dos más tablas.

3. LENGUAJE QUEL

QUEL es un lenguaje de consulta comercial que permite dos modos de trabajo:

Modo interactivo.

Modo inmerso: Las sentencias de QUEL se encuentran intercaladas con las del lenguaje anfitrión (Host programming language).

3.1. Definición y almacenamiento de datos

La estructura básica manejada por QUEL es la **Relación Base**. Una Relación Base es una relación caracterizada porque las tuplas que la constituyen están físicamente almacenadas en la Base de Datos, cada una de estas tuplas está formada por un número finito de atributos.

Existen diferentes *tipos de datos* que se pueden asignar a un atributo:

Datos numéricos.
Cadenas de caracteres.
Tipo DATE.
Tipo MONEY.

Para *definir una relación base y los atributos que la constituyen* se utiliza el comando **Create**:

CREATE Nomb_Rel_Bas (Nom_atr1 tipo, ... Nom_atrN tipo);

El comando empleado para *especificar el primer nivel de indexación de una relación base* es **Index**. Para *modificar la estructura de almacenamiento de una relación base* se utiliza el comando **Modify**. Las posibles estructuras de almacenamiento, que pueden presentar los registros de un fichero son:

Heap: Ficheros no ordenados. Los nuevos elementos se insertan al final del fichero.
Hash: Ficheros de acceso muy rápido. Almacenamiento basado en la existencia de una función hash que se aplica sobre un determinado campo del registro para obtener su dirección de almacenamiento.
Ficheros indexados: con las siguientes estructuras de almacenamiento: ISAM y BTREE.

El comando **Destroy** permite *eliminar una estructura de datos*.

3.2. Consultas en QUEL

La consulta en QUEL presenta la siguiente sintaxis:

RETRIEVE lista atributos
WHERE condición;

Los atributos deben estar cualificados, ya sea por el nombre de la relación base a la que pertenecen o por la tupla variable que ha sido definida sobre dicha relación base.

Las tuplas variables deben declararse previamente mediante la sentencia **Range**, está permite *definir una tupla variable e indicar la relación base* a partir de la cual se le asigna el rango.

3.3. Funciones en QUEL

Las funciones que se pueden emplear son: **SUM**, **MAX**, **MIN**, **AVG** y **COUNT**. Estas funciones se pueden emplear tanto con la sentencia retrieve como con la sentencia where. Cuando se usa una función en la sentencia retrieve es necesario asignarle un nuevo nombre al valor que devuelve.

Si se desea que las funciones actúen eliminando tuplas duplicadas se emplean las funciones: MAXU, COUNTU, MINU, AVGU y SUMU.

QUEL permite incluir sentencias where como constituyente del parámetro que se pasa a la función.

Existe la posibilidad de agrupar tuplas (BY). El cualificador BY se puede usar en cada función para especificar el grupo de tuplas sobre la que se desea ésta sea aplicada. Cada función empleada puede presentar sus propias agrupaciones.

QUEL permite pasar como parámetro de una función el conjunto de resultados obtenidos al aplicar otra función.

3.4. Manipulación de datos

Las operaciones de manipulación de datos que ofrece QUEL son las mismas que para SQL: inserción y eliminación de tuplas y actualización de columnas.

Inserción de tuplas

APPEND TO Nomb_Rel_Bas (Nomb_atr1 = valorl ... Nomb_AtrN = valorN);

Los valores asignados a un atributo pueden ser constantes o pueden obtenerse a partir de la aplicación de una función sobre otros atributos de la Base de Datos.

Eliminación de tuplas

La sintaxis del comando es:

DELETE Nomb_Rel_Bas
WHERE condición;

Actualización de datos

El nuevo valor que se le asigna a una determinada columna de una relación base puede ser una constante o puede obtenerse mediante la aplicación de funciones sobre otros atributos de la Base de Datos o puede ser el resultado de la aplicación de una operación aritmética:

REPLACE Nomb_Rel_Bas (Nomb_atrl = valorl ... Nomb_atrN = valorN)
WHERE condición;

4. LENGUAJE QBE

QBE (Query By Example) es un lenguaje relacional desarrollado por IBM Research. Está basado en el cálculo relacional de dominios.

4.1. Consultas en QBE

Las consultas en QBE se efectúan mediante la inserción de parámetros en determinadas columnas de la representación gráfica de una relación. Estos parámetros pueden ser:

P: Determina un dominio variable, es decir, determina la columna que se desea conocer.

Constante: Determina el valor que se impone a la columna sobre la que se aplica.

-Identif: Determina el nombre de la variable que se le asigna a una determinada columna.

"**-**": Prefijo que determina que la cadena de caracteres que le sigue es el nombre de una variable.

Identif: Nombre de la variable que se le asigna a la columna.

Ejemplo. Determinar el nombre del departamento número 5:

Departamento	Num_D	Nomb_D	NSS_Jef	Fech_Jef
	5	P.		

La interfaz ofrecida por QBE permite al usuario la posibilidad de seleccionar las relaciones necesarias para solucionar una consulta de entre la lista de todas las relaciones que constituyen la Base de Datos. Una vez seleccionadas las relaciones éstas son representadas. El usuario puede ir moviéndose de una columna a otra para ir configurando la consulta.

Condiciones de selección

Las condiciones de selección más sencillas de especificar son aquellas que se construyen a partir de los operadores de comparación {=, <, < .... etc.}.

Ejemplo: Determinar el número de seguridad social de los empleados que trabajan más de 10 horas en algún proyecto.

Trabaja_en	NSS	Num_P	Horas
	P.		> 10

Para especificar condiciones más complejas se puede emplear una *Caja de Condición*, ésta se puede crear mediante una de las facilidades ofrecidas por la interfaz de QBE, en esta caja el usuario puede especificar las condiciones que desea cumplan cada uno de los atributos pertenecientes a la relación, para poder hacer referencia a dichos atributos es preciso asignar a las columnas referenciadas un identificador.

Ejemplo: Determinar el número de la seguridad social de aquellos empleados que trabajan más de 10 horas en el proyecto número 8 o en el número 7:

Para solucionar esta cuestión mediante el empleo de parámetros definidos sobre la propia representación de la relación, la expresión QBE queda:

Trabaja_en	NSS	Num_P	Horas
	P.	7	> 10
	P.	8	> 10

Empleando la Caja de Condición queda:

Trabaja_en	NSS	Num_P	Horas
	P.	_NP	-HO

Condición:

-HO > 10 AND (_NP = 7 OR _NP = 8)

Búsquedas ordenadas

Para especificar que se desea obtener la información resultante de la consulta de forma ordenada, es necesario añadir uno de los siguientes prefijos a la columna que se toma como referencia para la ordenación:

- **AO** (Ascendent Order): si se desea que el criterio de ordenación sea ascendente.
- **DO** (Descendent Order): si el criterio de ordenación es descendente.

Si se emplea más de un dominio como criterio de ordenación es necesario indicar la prioridad que se desea dar a cada uno de los dominios: AO, DO.

Operación JOIN

La operación **JOIN** (producto cartesiano + "condición de salto") se determina en QBE mediante el empleo del mismo nombre de atributo variable en las diferentes columnas sobre las que se quiere realizar la operación.

Cuando se aplica la operación JOIN, se puede definir una *Tabla de Resultados* para poder visualizar los resultados obtenidos de una forma unificada. En caso de no definir esta tabla de resultados el sistema proporciona los resultados en cada una de las columnas de las tablas correspondientes.

El operador **UNQ** es un operador que proporciona OBE para eliminar las tuplas duplicadas resultantes de una consulta.

4.2. Operador G. y sus funciones

El operador G. puede ser aplicado a una columna para determinar sobre qué atributo quieren efectuarse agrupaciones de tuplas.

Este operador aparece de forma conjunta con algunas de las siguientes funciones:

- AVG: Determina el valor medio de los diferentes valores existentes en la columna sobre la que se especifica.
- MAX: Devuelve el valor máximo de los diferentes valores presentes en la columna.
- MIN: Obtiene el mínimo valor de los existentes en una columna.
- SUM: Realiza la suma de los diferentes valores presentes en una columna.
- CNT: Cuenta los distintos valores que aparecen en la columna especificada.

Las funciones AVG, SUM y CNT actúan únicamente sobre las tuplas que presentan valores diferentes en la columna sobre la que se aplican (ignoran los valores duplicados). Si se desea que dichas funciones se apliquen sobre los valores que presentan todas las tuplas, estando éstos duplicados o no, es preciso indicarlo explícitamente mediante el sufijo ALL.

Al igual que en SQL y QUEL si estas funciones se emplean en combinación con la opción de agrupación, se aplican de forma individual sobre cada una de las diferentes particiones resultantes de la agrupación.

Ejemplo: Determinar el número de empleados y los diferentes salarios que pueden ganar:

Empleado	NSS	Nomb	Salario	Num_D
	P.CNT.ALL		P.CNT.	

Mediante P.CNT.ALL se cuenta el número total de empleados, estén o no repetidos, mientras que a través de P.CNT. se cuentan únicamente aquellos salarios diferentes, eliminando todos aquellos valores que se encuentren repetidos.

4.3. Manipulación de datos

QBE ofrece tres operadores para realizar modificaciones en una Base de Datos:

I. Inserción de nuevas tuplas.
D. Eliminación de tuplas.
U. Actualización de valores.

La inserción y eliminación de tuplas se expresa directamente sobre las representaciones de las relaciones que se quieren modificar, mientras que la actualización se especifica sobre cada una de las columnas sobre las que se quiere actuar.

Inserción de tuplas

Para insertar una tupla en una relación es necesario indicar el operador I. y especificar los valores que se desean insertar en cada una de las columnas de la relación:

Departamento	Num_D	Nomb_D	NSS_Jef	Fech_Jef
I.	5	Investigación	12587923	30-12-92

Eliminación de tuplas

A través del operador D. y mediante la especificación de condiciones, se puede determinar las tuplas que se desean eliminar de una determinada relación.

QBE permite eliminar tuplas pertenecientes a diferentes relaciones mediante una única expresión, aplicando el operador D. a las relaciones apropiadas e imponiendo las condiciones necesarias en todas aquellas relaciones que se hayan implicadas.

Operación de actualización

Para actualizar unas tuplas, se emplea el operador U. en el atributo que se desea modificar, seguido por el nuevo valor que se desea asignar a esa columna.

Si se desean actualizar varias tuplas a un mismo tiempo se puede indicar cuál es la condición que deben verificar cada una de ellas.

4.4. Creación de estructuras

QBE permite la definición de las tablas que constituyen la Base de Datos de forma interactiva, igualmente permite la eliminación, alteración y renombrado de dichas tablas.

Para crear una relación se selecciona el menú correspondiente, se da un nombre a la relación y se van definiendo uno a uno los diferentes atributos que la constituyen, asignando a cada uno de ellos un nombre, un tipo de dato y el conjunto de restricciones que debe verificar.

QBE también ofrece la posibilidad de definir índices y vistas.

5. BIBLIOGRAFÍA

Gary W. Hansen
Diseño y Administración de Bases de Datos
Prentice Hall, 1998

Alberto Prieto
Introducción a la Informática
Mc Graw-Hill, 2ª edición, 1997

Alfonso Ureña López
Fundamentos de Informática
Ra-ma, 1997