Caso práctico sobre bases de datos relacionales

Pedro Cardona Vilaplana

PID_00170168

Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

Las bases de datos son unas herramientas muy potentes que nos van a permitir gestionar grandes cantidades de datos de manera ordenada, que se convierten en información mediante su tratamiento, así como en conocimiento mediante la difusión de ésta. En este módulo vamos a poner en práctica conceptos tratados en el módulo teórico *El modelo relacional y el álgebra relacional*, es decir, lo que tiene que ver con las bases de datos relacionales y el lenguaje SQL.

El enfoque con el que hemos planteado este caso práctico es rememorando (de una manera muy práctica y con ejemplos) algunos de los conceptos clave estudiados en el módulo y, principalmente, con ejercicios de cómo poner en práctica estos conceptos por medio de una base de datos de ejemplo que nosotros mismos construiremos.

Dado que se trata de poner en práctica la teoría aprendida con anterioridad, es del todo recomendable que desarrolléis los ejercicios que aquí proponemos en una base de datos de vuestro propio ordenador personal.

A pesar de que como base de datos soporte para los ejemplos y el caso práctico vamos a utilizar una base de datos de amplia difusión como es Access 2007, podréis, con facilidad, desarrollar los diferentes ejercicios sobre cualquier otra base de datos relacional, como por ejemplo OpenOffice, dado que hemos utilizado ejemplos y sentencias SQL estándares. Los resultados que obtendréis serán los mismos.

1. Conceptos necesarios

1.1. El Modelo Entidad-Relación

El Modelo Entidad-Relación (E-R), también llamado Modelo Conceptual de Datos, es una técnica de representación de las relaciones que tienen los datos de la base de datos y que permite recrear una representación de la realidad que tenemos como objetivo tratar en nuestra base de datos.

La realización de un modelo E-R es siempre un paso previo al diseño que finalmente se implementará en una base de datos y comprende exclusivamente una representación del diseño de los datos, y no lo que se pretende hacer con ellos.

1.2. Entidades y atributos

Una entidad es una cosa u objeto concreto o abstracto que existe y que puede diferenciarse de otros, como pueden ser personas, meses del año, etc. El primero de estos ejemplos corresponde con un objeto concreto y el segundo, con uno abstracto.

El conjunto de entidades corresponde con un grupo de entidades del mismo tipo.

> En una empresa, el conjunto de empleados podría denominarse "Empleados".

Una entidad siempre está representada por un conjunto de atributos que representan sus características.

> En la entidad "Empleados", algunos atributos posibles serían: DNI, Nombre, Apellidos, etc.

1.3. Relaciones

Una relación es una asociación que se da entre diferentes entidades.

> En una empresa, además de la entidad "Empleados" que hemos comentado también tendríamos una llamada "Direcciones", que hace referencia a la dirección organizativa (del organigrama de la empresa) a la que está adscrito el empleado. Sobre esto podríamos definir una relación que asociaría al empleado "Pedro" con la dirección "Contabilidad".

Además, una relación puede tener también sus propios atributos.

> En el caso anterior, el atributo de la relación de "Pedro" con la dirección "Contabilidad" podría ser la fecha de adscripción en la dirección.

1.4. Claves

Una clave es el conjunto mínimo compuesto por uno o más atributos que permite identificar de manera unívoca a una entidad dentro de un conjunto. Por lo tanto, ningún subconjunto de atributos podrá funcionar también como clave.

> En la entidad "Empleados", el NIF sería la clave, ya que ningún conjunto menor que éste podría identificar de manera unívoca a cada uno de los empleados.

Es posible que en un conjunto de entidades exista más de una clave. A todas las claves posibles se les denomina **claves candidatas**, mientras que a la clave elegida por el diseñador de la base de datos para identificar a cada una de las entidades se le denomina **clave primaria**.

1.5. Clasificación de las entidades en función de las claves

- **Entidades fuertes**: son aquellas que tienen una clave primaria.

- **Entidades débiles**: son las que no tienen entre sus atributos una clave primaria, por lo que dependen de una entidad fuerte que les permite identificar, mediante una relación, a cada uno de sus atributos.

Aunque un conjunto de entidades débiles no tienen clave primaria, se necesita conocer, no obstante, un medio para distinguir todas aquellas entidades de su conjunto que dependen de una entidad fuerte particular. El conjunto mínimo de uno o varios atributos que lo permite es el que llamamos **discriminante** de un conjunto de entidades. La clave primaria de una entidad débil está formada por el discriminante más la clave primaria de la relación fuerte con la que está asociada.

> Siguiendo con nuestro caso de empleados de una empresa, tendríamos una entidad llamada "Nominas" que tiene los atributos NumNomina, Fecha e Importe. NumNomina es el número de nómina que ha cobrado el empleado (la primera nómina que cobra tiene NumNomina = 1, la siguiente el valor 2 y así sucesivamente). "Nominas" por sí misma no tiene una clave que identifique una entidad de manera unívoca (podríamos tener dos empleados que en su primera nómina, pagada el mismo día, hubieran cobrado lo mismo). En este caso, "Nominas" depende de "Empleados" para su existencia, por lo que el discriminante sería NumNomina (para un mismo empleado, sí se puede identificar de manera unívoca cada nómina). Por lo tanto, la clave primaria de "Nominas" sería NIF (clave primaria de la entidad fuerte con la que se asocia, "Empleados") más NumNomina (discriminador, es decir, que permite la identificación unívoca de cada entidad "Nominas" para cada empleado).

Las claves ajenas o foráneas son el conjunto de atributos de una entidad que a su vez son claves primarias de otra entidad con la que está relacionada.

1.6. Diagramas Entidad-Relación

El Diagrama Entidad-Relación permite representar gráficamente la estructura lógica de una base de datos mediante los siguientes elementos:

- Rectángulos: representan las entidades.
- Elipses: representan los atributos de las entidades.
- Rombos: representan las relaciones entre las entidades.
- Líneas: enlazan atributos a entidades, atributos a relaciones y entidades a relaciones.

Supongamos que queremos modelizar las notas de las asignaturas de una titulación que han obtenido los alumnos:

- Alumnos de los que conocemos: DNI, Nombre y Apellidos.
- Asignaturas de las que sabemos: Identificador de Asignatura, Nombre de la Asignatura y Curso al que pertenece.

El modelo Entidad-Relación correspondiente sería:

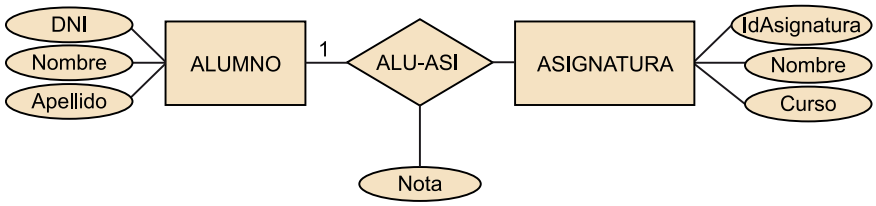

Figura 1. Modelo Entidad-Relación que representa la relación entre "Alumno" y "Asignatura".

1.7. Cardinalidad de las relaciones

La correspondencia de cardinalidades, o razón de cardinalidad, expresa el número de entidades a las que otra entidad puede estar asociada por medio de una relación o, dicho de otro modo, expresa el número de entidades con las que puede asociarse otra entidad.

A tenor de lo anterior, tenemos las siguientes relaciones posibles entre dos conjuntos de entidades A y B:

- **Una a Una (1:1)**: a cada instancia u ocurrencia de la entidad A se relaciona, como máximo, con una instancia u ocurrencia de la entidad B y viceversa.

- **Una a Muchas (1:N)**: una instancia u ocurrencia de la entidad A se relaciona con un número cualquiera de instancias u ocurrencias de la entidad B, mientras que una instancia u ocurrencia de la entidad B se relaciona, como máximo, con una instancia u ocurrencia de la entidad A.

- **Muchas a Una (N:1)**: una instancia u ocurrencia de la entidad A se relaciona, como máximo, con una instancia u ocurrencia de la entidad B, mientras que una instancia u ocurrencia de la entidad B se relaciona con un número cualquiera de instancias u ocurrencias de la entidad A.

- **Muchas a Muchas (N:M)**: una instancia u ocurrencia de la entidad A se relaciona con un número cualquiera de instancias u ocurrencias de la entidades B y viceversa.

La manera de representar la cardinalidad es:

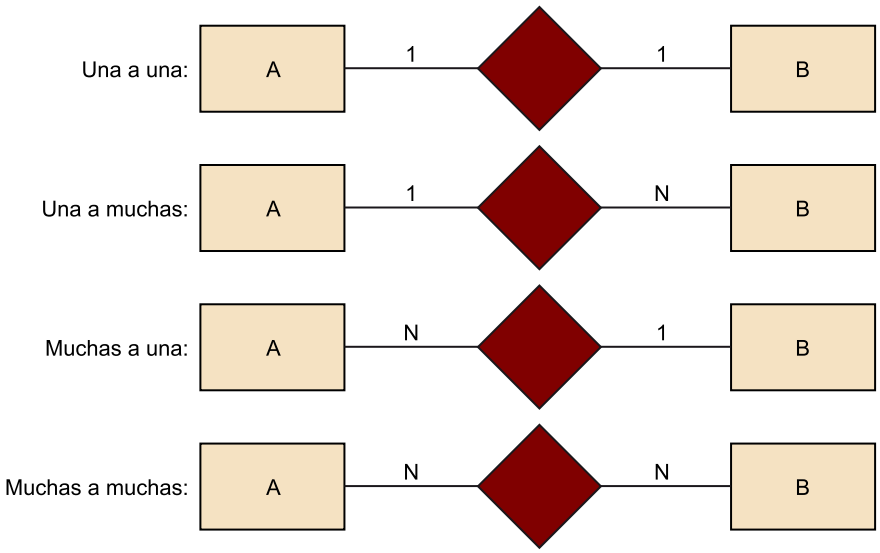

Figura 2. Representaciones de los tipos de cardinalidades

Ejemplos de cardinalidades

Un préstamo puede ser, como máximo, de un cliente y, a su vez, un mismo cliente puede tener varios préstamos. Con lo cual, la relación cliente a préstamo es de uno a varios.

Si, como variación al ejemplo anterior, tenemos que un préstamo no sólo puede pertenecer a un cliente sino a varios (son varios los titulares del contrato de préstamo), entonces estaríamos ante una relación de varios a varios.

1.8. Generación del modelo relacional a partir del Entidad-Relación

Una vez realizado el modelo Entidad-Relación de una base de datos, se genera el modelo relacional. A modo de resumen, las reglas de conversión para poder derivar el modelo relacional a partir del Entidad-Relación es:

- Cada conjunto de entidades fuertes del modelo Entidad-Relación se convierte en una tabla en el modelo relacional, siendo los atributos los campos de la tabla.

- Cada conjunto de entidades débiles se transforma en una tabla, siendo sus campos los atributos de la entidad débil más la clave primaria de la entidad fuerte con la que se relaciona.

- Cada conjunto de relaciones se transforma en una tabla cuyos campos son las claves principales de cada una de las entidades con las que se relaciona más los atributos propios de la relación.

- Toda relación N:M se transforma en una tabla intermedia, relacionándose con las dos anteriores mediante relaciones de cardinalidad 1:N.

1.9. Ejemplo de conversión de un modelo Entidad-Relación a un modelo relacional

Se quiere diseñar un sistema relacional para una entidad financiera que contenga información sobre los clientes, los contratos de los clientes y las operaciones realizadas sobre cada uno de éstos. Para ello hay que considerar las siguientes restricciones:

- Una operación se identifica por un número de operación, fecha de operación e importe.
- Un cliente puede tener muchas cuentas.
- Una cuenta puede ser de varios clientes.
- Una cuenta sólo puede estar en una sucursal.

A partir de lo anterior, identificamos las siguientes entidades con sus correspondientes atributos del modelo Entidad-Relación:

- "Clientes" (nif, nombre, apellido)
- "Sucursales" (numsucu, dirección, población, teléfono)
- "Cuentas" (numcta, tipo, interés, saldo)
- "Operaciones" (numop, fecha, importe)

Gráficamente se representaría de la siguiente manera:

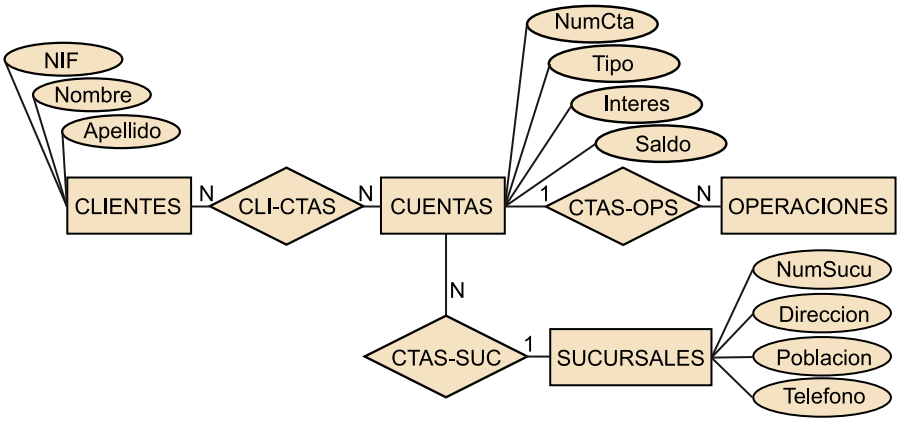

Figura 3. Modelo Entidad-Relación de la entidad financiera del ejemplo

Aplicando las reglas de transformación se convierte en la siguiente estructura de tablas que hemos realizado mediante Access:

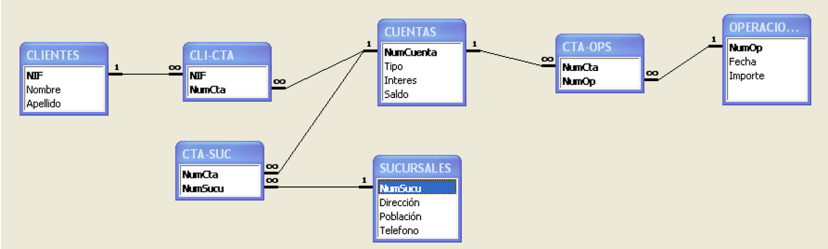

Figura 4. Estructura de tablas correspondiente al modelo Entidad-Relación de la figura 3

Podemos ver que las entidades y relaciones se han transformado en tablas, siendo los campos de las primeras los atributos de las entidades y, en el caso de las relaciones, las claves primarias de cada atributo con los que se relacionan.

1.10. Integridad referencial

A la hora de implementar en una base de datos nuestro modelo, debemos tener claro cómo queremos que se comporte ante modificaciones y borrados de claves con el objetivo de asegurar que se mantiene la integridad en las relaciones de nuestra base de datos.

No se mantendría la integridad cuando en una relación 1:N se elimina el registro en la tabla con cardinalidad 1, que es clave foránea de la tabla con cardinalidad N (quedarían registros en la tabla de cardinalidad N que tendrían como clave foránea una clave principal de la tabla con cardinalidad N que no existiría).

Tampoco se mantendría la integridad si hacemos cambios sobre un campo que sea clave principal de la tabla con cardinalidad 1 y éstos no se ven reflejados sobre los registros de la tabla de cardinalidad N cuya clave foránea es la clave principal de la tabla de cardinalidad 1.

Por ello, a la hora de definir las relaciones, algunos sistemas gestores de bases de datos, como por ejemplo Access, permiten establecer diferentes comportamientos del sistema en cuanto a la integridad referencial (si es que queremos que la tenga). Con caracter general, las dos posibilidades son:

1) Cuando llevamos a cabo un borrado: con el fin de mantener la integridad referencial, cabe la posibilidad de que el sistema gestor de bases de datos lleve a cabo la propagación de los borrados a las tablas relacionadas. Si borramos un registro de una tabla que se relaciona 1:N con otra, también se borrarían los registros de la tabla con cardinalidad N en la relación.

> Partiendo del caso que venimos tratando (la entidad financiera), si borramos al cliente con NIF "21.212.121-A" y exigimos integridad referencial, también se borrarían los registros con este NIF de la tabla CLI-CTA.

2) Cuando llevamos a cabo cambios en los campos que son clave de una tabla: de manera similar al caso anterior de los borrados, es posible propagar los mismo cambios a los campos clave foránea de las tablas con las que se

relaciona. Si modificamos un registro de un atributo clave de una tabla que se relaciona 1:N con otra, también se actualizarían al mismo valor los campos clave foránea de la tabla con cardinalidad N en la relación.

Si nos hemos dado cuenta de que tenemos un NIF de un cliente mal grabado y lo modificamos, y pasa de ser el "20.200.200-B" al valor "21.212.121-A", tenemos la actualización reflejada tanto en la tabla "Clientes" como en la relación CLI-CTA (las cuentas que antes se relacionaban con el NIF "20.200.200-B" ahora lo hacen con el "21.212.121-A").

En Access encontramos las posibilidades de integridad referencial cuando definimos las relaciones entre tablas:

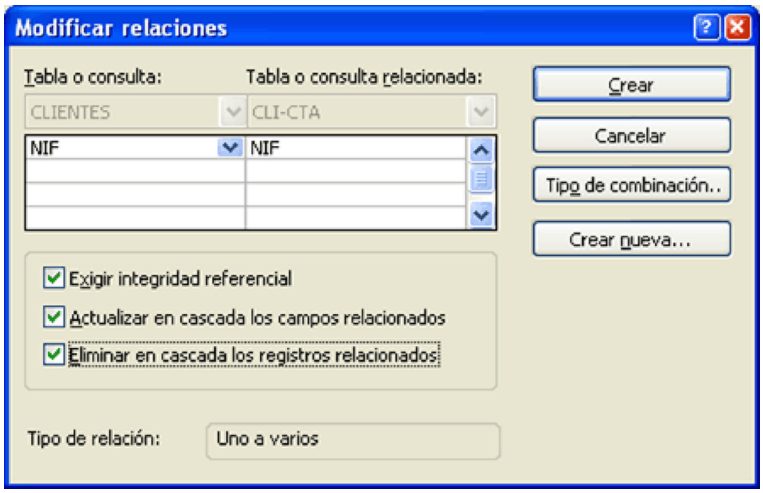

Figura 5. Muestra de la exigencia de integridad referencial en Access

1.11. *Structured Query Language*

SQL son las iniciales de *Structured Query Language* o Lenguaje Estructurado de Consulta. Se trata de un lenguaje no procedural inventado por IBM en los años setenta para implementar el modelo relacional definido por Codd.

Realmente SQL es una abreviatura que ha devenido con el tiempo, ya que su nombre inicial era SEQUEL (*Structured English Query Language*). Actualmente éste es el lenguaje que más ampliamente es utilizado por los sistemas gestores de bases de datos, ya que todos ellos presentan sus pequeñas adaptaciones de este lenguaje, de modo que encontramos algunas variaciones del lenguaje SQL en función del SGBD que utilicemos, aunque los comandos más habituales suelen ser los mismos en todos ellos.

Aunque el lenguaje SQL se considere un lenguaje de consultas, contiene muchas otras capacidades además de la consulta en bases de datos, incluyendo características para definir la estructura de los datos, inserción de datos, modificación de datos en la base de datos, etc.

Sobre nuestra base de datos de cuentas de una entidad bancaria, vamos a considerar que tenemos cargada la siguiente información:

Sobre la entidad "Sucursales":

SUCURSALES : Tabla

	NumSucu	Dirección	Población	Telefono
+	1	Av. Pearson, 123	Barcelona	933333333
+	2	Plaza Catalunya,	Barcelona	934444444
+	3	Rambla, 16	Barcelona	935555555
+	4	Plaza Neptuno, 1	Madrid	911111111
+	5	Gran Vía, 11	Madrid	916666666
+	6	Colon, 9	Valencia	963333333
	0			0

Figura 6. Detalle de los datos cargados en la tabla "Sucursales"

Sobre la entidad "Clientes":

CLIENTES : Tabla

	NIF	Nombre	Apellido
+	11111111-A	Juan	García
+	22222222-B	Pedro	Pérez
+	33333333-C	Helena	Martínez
+	44444444-D	Marta	González
+	55555555-E	Daniel	Fernández
+	66666666-F	Pablo	López

Figura 7. Detalle de los datos cargados en la tabla "Clientes"

Sobre la entidad "Operaciones":

OPERACIONES : Tabla

	NumOp	Fecha	Importe
+	1	10/02/2010	5
+	2	11/02/2010	7
+	3	11/02/2010	6
+	4	12/02/2010	6
+	5	13/02/2010	8
+	6	15/02/2010	9
+	7	16/02/2010	10
+	8	17/02/2010	12
+	9	20/02/2010	4
+	10	21/02/2010	5
+	11	23/02/2010	6
+	12	24/02/2010	11
	0		0

Figura 8. Detalle de los datos cargados en la tabla "Operaciones"

Sobre la entidad "Cuentas":

CUENTAS : Tabla

	NumCuenta	Tipo	Interes	Saldo
+	1	Cuenta corriente	1	100
+	2	Préstamo	5	50
+	3	Plazo Fijo	4	125
+	4	Cuenta corriente	1	154
+	5	Préstamo	4	95
+	6	Cuenta Corriente	2	67
	0		0	0

Figura 9. Detalle de los datos cargados en la tabla "Cuentas"

Sobre la entidad CLI-CTA (derivada de la relación "Clientes"-"Cuentas"):

CLI-CTA : Tabla

NIF	NumCta
11111111-A	1
11111111-A	2
22222222-B	3
22222222-B	5
33333333-C	1
33333333-C	4
44444444-D	1
55555555-E	5
66666666-F	6
	0

Figura 10. Detalle de los datos cargados en la tabla "CLI-CTA"

Sobre la entidad CTA-OPS (derivada de la relación "Cuentas"-"Operaciones"):

CTA-OPS : Tabla

NumCta	NumOp
1	1
1	2
2	3
2	4
3	5
3	6
4	7
5	8
5	9
5	10
6	11
6	12
0	0

Figura 11. Detalle de los datos cargados en la tabla "CTA-OPS"

Sobre la entidad CTA-SUC (derivada de la relación "Cuentas"-"Sucursales"):

CTA-SUC : Tabla

NumCta	NumSucu
1	1
2	2
3	3
4	4
5	5
6	6
0	0

Figura 12. Detalle de los datos cargados en la tabla "CTA-SUC"

1.12. *Query By Example*

Los sistemas gestores de bases de datos habitualmente permiten la generación de consultas utilizando ejemplos (QBE, *Query By Example*), lo que permite poder llevar a cabo consultas a la base de datos de una manera más intuitiva y gráfica. Concretamente Access también tiene un QBE que resulta relativamente sencillo de manejar.

Mediante QBE evitamos conocer la sintaxis concreta SQL que utiliza nuestro sistema gestor de bases de datos, lo que nos permite explotar la base de datos de una manera más sencilla.

Para acceder al QBE de Access, debemos seleccionar la entrada "Consultas" del menú vertical de la izquierda y, a continuación, seleccionar "Crear una nueva consulta en vista Diseño".

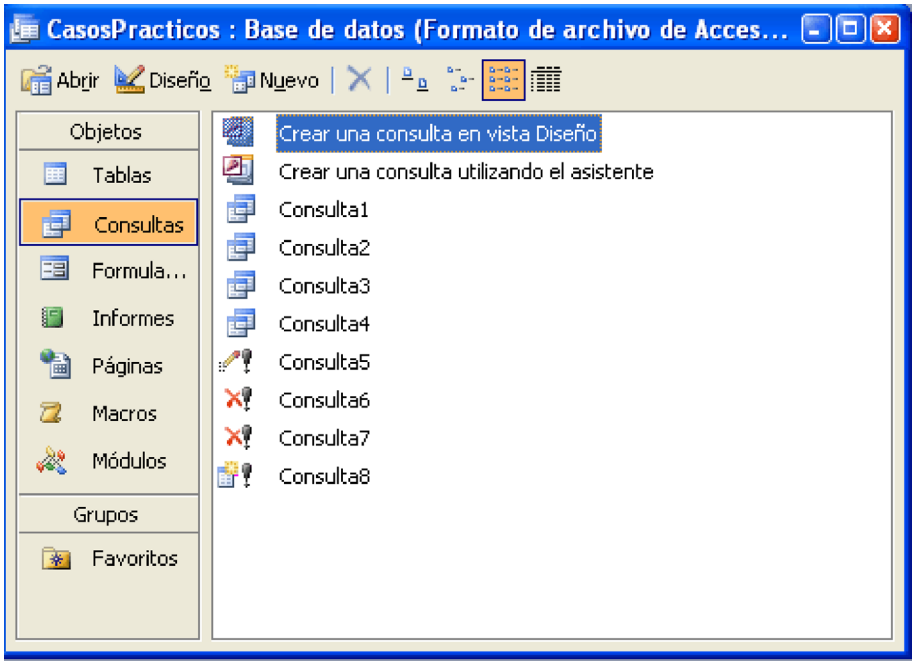

Figura 13. Detalle del acceso al QBE de Access

Lo que nos aparece a continuación es una ventana emergente desde la que podemos seleccionar la/s tabla/s que vamos a utilizar para hacer nuestras consultas y una rejilla (*grid*) sobre la que podremos "arrastrar y soltar" los campos que queramos de las tablas que hayamos seleccionado.

De esta manera, mediante acciones muy simples podremos construir sentencias SQL que, en algunos casos, son sensiblemente complejas. Posteriormente podemos visualizar la sentencia en formato SQL.

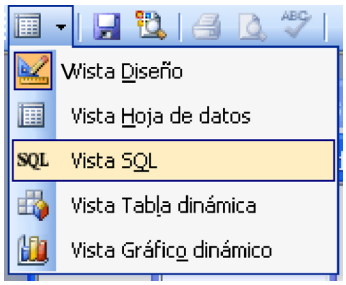

Figura 14. Acceso a la redacción y visualización de sentencias SQL en Access

Para simplificar la representación de los campos se utiliza, al igual que en SQL, el símbolo * para significar que queremos contemplar todos los atributos de la tabla desde la que "arrastramos y soltamos".

La consulta generada puede guardarse y, lo que es muy útil, puede ser utilizada posteriormente como punto de partida para otras consultas como si de una tabla se tratara. El contenido de esta "tabla" se calcula en el momento de lanzar la consulta que la utiliza, es decir, trabajaremos con datos actualizados y no estáticos.

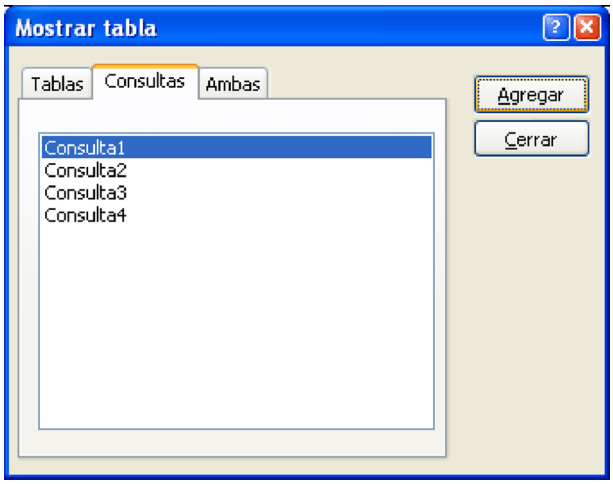

Figura 15. Muestra de la ventana Access que nos permite utilizar las consultas anteriores como si fueran tablas.

2. Caso práctico

Para el desarrollo de este caso práctico vamos a partir de la base de datos de Clientes, Cuentas, Operaciones y Sucursales que hemos venido utilizando como ejemplo en este documento.

Sobre ésta, vamos a plantear una serie de consultas que deberemos resolver mediante SQL y el editor QBE de Access.

2.1. Consulta 1: Obtener los nombres de nuestros clientes ordenados alfabéticamente

La sentencia SQL que deberemos ejecutar es la siguiente:

```
SELECT "Clientes".Nombre
FROM "Clientes"
ORDER BY "Clientes".Nombre
```

Hacerlo desde el editor QBE es igual de sencillo:

1) Debemos seleccionar en la ventana "Mostrar Tablas" la tabla sobre la que queremos realizar la consulta, en este caso, "Clientes". A continuación, cerramos esta ventana.

2) Arrastramos el campo "Nombre" de la tabla "Clientes" que aparece en la parte superior de la ventana y lo soltamos en la primera columna del *grid* (la rejilla que aparece en la parte inferior de la ventana).

3) Hasta aquí nos hubiera seleccionado todos los clientes, pero como queremos que esté ordenado alfabéticamente, debemos seleccionar en la fila "Orden" del *grid* Ascendente.

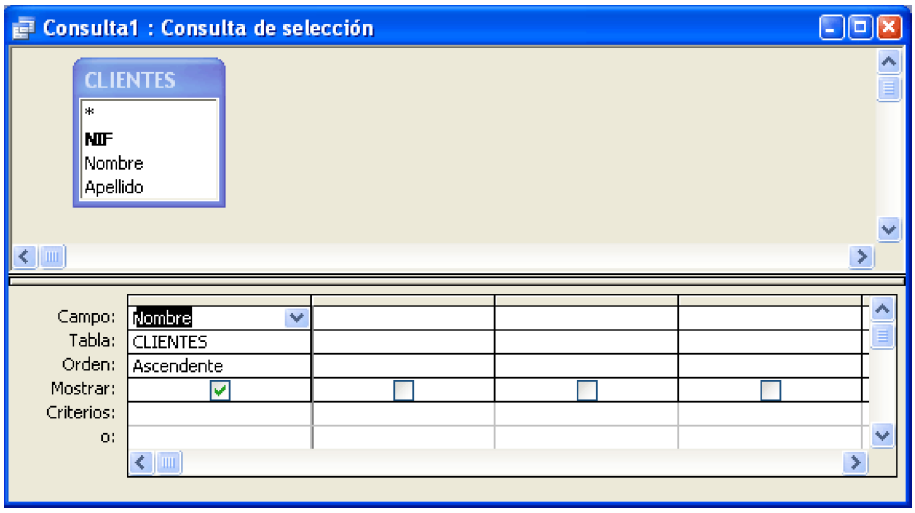

Figura 16. Ventana del editor QBE de Access que representa la consulta 1.

2.2. Consulta 2: Obtener la información de cuentas con NIF 11111111-A

Para obtener la información de cuentas con un NIF determinado, lo que hemos de aplicar es la operación de consulta SQL siguiente:

```
SELECT "Cuentas".*, "Clientes".NIF
FROM "Cuentas" INNER JOIN ("Clientes" INNER JOIN [CLI-CTA] ON "Clientes".NIF = [CLI-CTA].NIF) ON "Cuentas".NumCuenta = [CLI-CTA].NumCta
WHERE ((("Clientes".NIF)="11111111-A"));
```

La función de combinación interna INNER JOIN selecciona las filas que tienen valores idénticos en los campos sobre los que compara. En este caso debemos hacer la consulta sobre tres tablas ("Clientes", CLI-CTA y "Cuentas"), de modo que hemos de anidar dos INNER JOIN para poder obtener el equivalente a una tabla a partir de la combinación de las tres iniciales. La sintaxis general de anidación de sentencias INNER JOIN es:

```
FROM (...(tabla1 JOIN, tabla2 ON condicion1 JOIN, tabla3 ON condicion3) JOIN...)
```

Ved también

La función de combinación interna INNER JOIN se describe en el módulo *El modelo relacional y el álgebra relacional.*

Hacerlo desde el editor QBE es también muy sencillo, y en este caso tiene la ventaja de que nos podemos despreocupar de la sintaxis de las INNER JOIN anidadas que necesitamos. Los pasos que debemos seguir son:

1) Desde la ventana "Mostrar Tablas" debemos seleccionar las tablas que necesitaremos para esta consulta y que son: "Clientes", "Cuentas", CLI-CTA. A continuación, cerramos la ventana.

2) Arrastramos desde la tabla "Cuentas" todos sus campos, es decir, arrastramos el *, y lo soltamos en la primera columna del *grid* inferior.

3) Arrastramos desde la tabla "Clientes" el campo NIF a la segunda columna del *grid*, quitando la marca "Mostrar", ya que sólo queremos presentar la información de las cuentas.

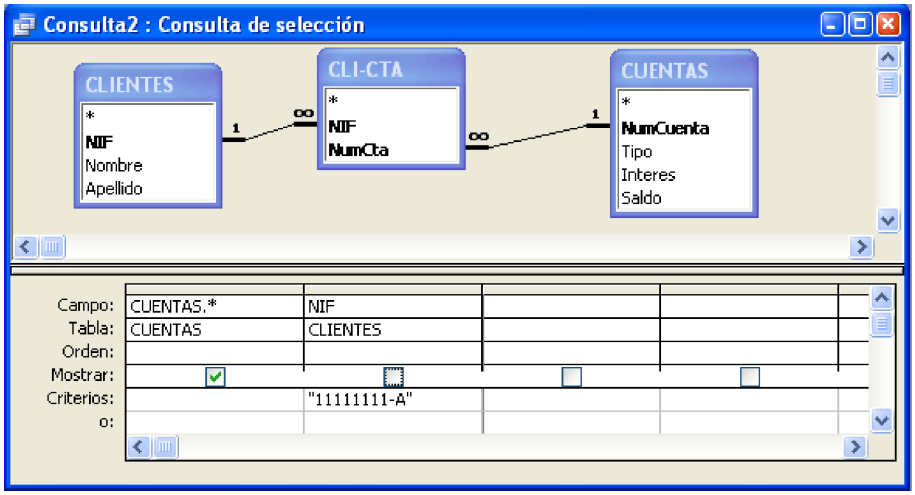

Figura 17. Ventana del editor QBE de Access que representa la consulta 2.

¿Tenemos una manera más sencilla de efectuar esta consulta? La respuesta es sí. Podemos hacerlo simplemente con dos tablas en vez de tres, utilizando exclusivamente las tablas "Cuentas" y CLI-CTA.

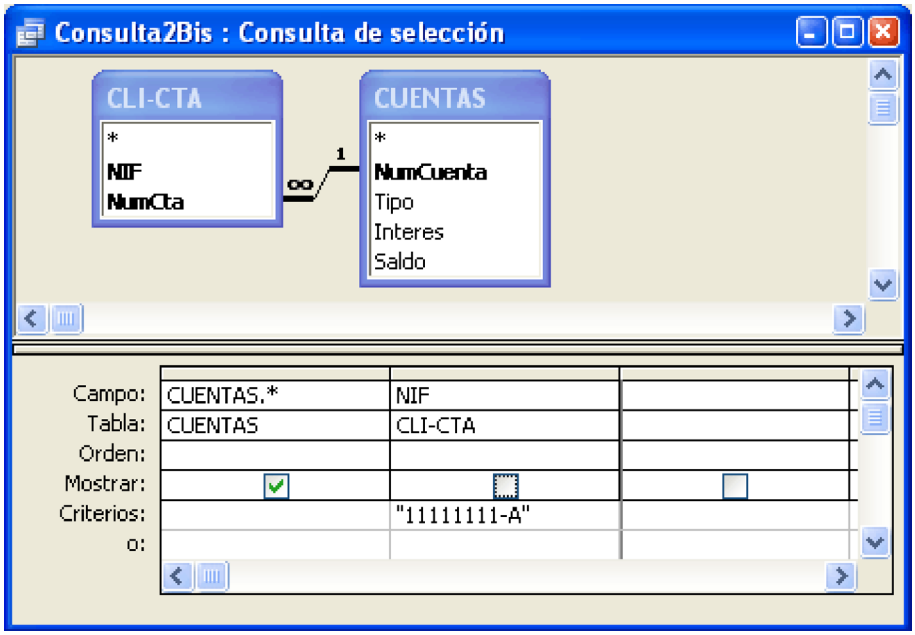

Figura 18. Ventana del editor QBE de Access que representa la consulta 2 con solo dos tablas.

2.3. Consulta 3: Calcular el importe de intereses de los clientes

Nos piden calcular el importe en concepto de intereses de cada uno de los clientes de la entidad bancaria de nuestro ejemplo, considerando lo siguiente:

- La fórmula del cálculo de intereses para todos los productos de la entidad es Saldo * Interés.

- No aplicamos tratamientos diferenciados por haber más de un cliente en una misma cuenta, es decir, los intereses se multiplican en función del número de clientes asociados a la cuenta.

El resultado es el siguiente:

```
SELECT "Clientes".Nombre, Sum(([Interes]*[Saldo])) AS subtotal
FROM "Cuentas" INNER JOIN ("Clientes" INNER JOIN [CLI-CTA] ON "Clientes".NIF = [CLI-CTA].NIF) ON "Cuentas".NumCuenta = [CLI-CTA].NumCta
GROUP BY "Clientes".Nombre;
```

Observad que al campo calculado le hemos dado un alias y lo hemos llamado "subtotal".

Como siempre, hacerlo desde el *grid* es intuitivo, aunque en este caso también hay que escribir un poco, no sólo es arrastrar y soltar. Los pasos que debemos seguir son:

1) Desde la ventana "Mostrar Tablas" seleccionamos las tablas "Clientes", CLI-CTA y "Cuentas". Cerramos la ventana "Mostrar Tablas".

2) Desde la tabla "Clientes" arrastramos el campo Nombre y lo soltamos en la primera columna del *grid*.

3) En la segunda columna del *grid* debemos escribir el alias que queremos darle al resultado del cálculo seguido de dos puntos y la fórmula que queremos aplicar, en este caso Interés multiplicado por Saldo. Dado que no hay campos que se llamen igual en más tablas que en "Cuentas", podemos evitar escribir la tabla de la que provienen; de lo contrario, deberíamos haberlas referenciado precediendo a los campos del nombre de la tabla y un punto. Os invito a realizar la prueba con este ejemplo, de modo que os resulte de utilidad cuando tengáis que tratar con diferentes tablas que contengan algunos campos con idéntico nombre.

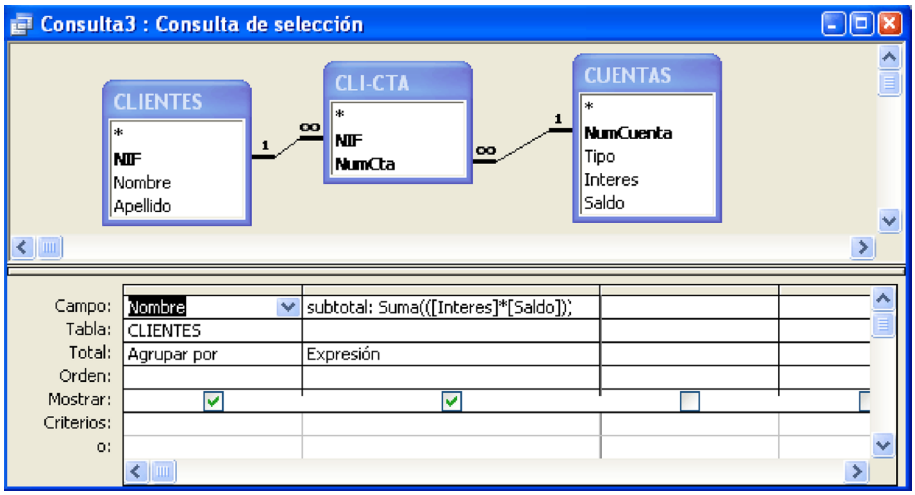

Figura 19. Ventana del editor QBE de Access que representa la consulta 3.

¿Para qué hemos querido darle un alias al resultado del cálculo? Recordad que hemos dicho que una utilidad bastante significativa de Access es que las consultas se pueden utilizar como si fueran tablas (para, por ejemplo, generar otras consultas a partir de éstas). En este caso, si posteriormente necesitamos hacer algo más con este cálculo, podemos hacerlo fácilmente, ya que al tener el alias, es como si se tratara de un campo cualquiera.

2.4. Consulta 4: Calcular la suma del total de los saldos de las cuentas

El resultado de la sentencia SQL que deberemos ejecutar es el siguiente:

```
SELECT SUM ("Cuentas".Saldo)
FROM "Cuentas";
```

El uso del QBE para este caso resulta también muy fácil. Veamos los pasos que debemos dar:

1) Como siempre, desde la ventana "Mostrar Tablas" seleccionamos las tablas que necesitamos para nuestra consulta, en este caso "Cuentas".

2) Desde la barra de herramientas pinchamos sobre el símbolo Σ, lo que provoca que en el *grid* aparezca una nueva fila llamada "Total". Veremos que el símbolo Σ queda con fondo de diferente color, lo que quiere decir que está activada la opción.

3) Arrastramos el campo Saldo desde la tabla "Cuentas" a la primera columna del *grid*.

4) En la fila "Total", por defecto nos aparece la opción "Agrupar por" y, si desplegamos el menú, veremos que hay múltiples opciones. Debemos seleccionar, como era de esperar, "Suma".

Figura 20. Ventana del editor QBE de Access que representa la consulta 4.

¿Es sencillo reutilizar este cálculo en otras consultas? La verdad es que sí, porque podemos reutilizar la consulta como si fuera una tabla más, si bien es cierto que, a diferencia del caso anterior, no le hemos dado un alias a este cálculo.

El *grid* también nos permite realizar este cálculo dándole un alias. Simplemente debemos desactivar el símbolo Σ y escribir el cálculo en la primera fila del *grid*. En este caso el cálculo es: Suma (Saldo).

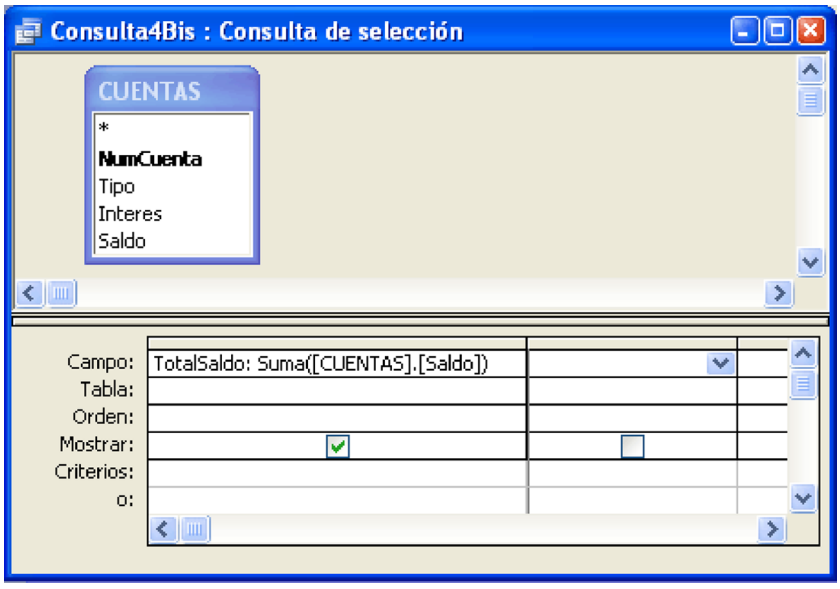

Figura 21. Ventana del editor QBE de Access que representa la consulta 4 sin utilizar un alias.

Para que sirva de guía a la hora de referenciar a campos de las tablas, para este ejemplo se ha utilizado la ruta completa del campo, es decir, "["Cuentas"].[Saldo]", aunque simplemente con "Saldo" hubiera sido suficiente, ya que no hay otro campo que se llame de la misma manera y menos en este caso, en el que estamos utilizando únicamente una tabla para la consulta.

2.5. Consulta 5: Incrementar el tipo de interés en un punto a los clientes que se llamen Juan

Para ello, debemos desarrollar una consulta para que a todos los registros de la tabla "Cuentas" sume al valor de su campo "Interés" una unidad.

La consulta en SQL sería:

```
UPDATE "Cuentas" INNER JOIN ("Clientes" INNER JOIN [CLI-CTA]
ON "Clientes".NIF = [CLI-CTA].NIF) ON "Cuentas".NumCuenta = [CLI-
CTA].NumCta
SET "Cuentas".Interes = [Interes]+1
WHERE ((("Clientes".Nombre)="Juan"));
```

Como ya sabéis, estamos ante una consulta de actualización.

Ved también

En el módulo 2 estudiaremos las consultas de actualización.

En Access por defecto las sentencias SQL son de selección, es decir, no alteran el contenido de la base de datos. Para llevar a cabo otro tipo de consultas (actualización, inserción, borrado, etc.) deberemos hacerlo mediante el menú de consultas.

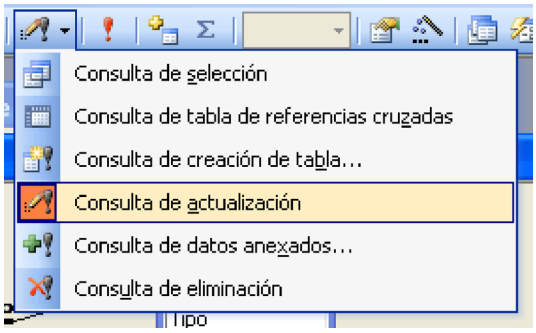

Figura 22. Menú en Access para la selección del tipo de consulta, seleccionando "consulta de actualización"

Cuando llevamos a cabo este cambio, observamos unas ciertas variaciones sobre los datos que nos solicita el QBE de Access en función del tipo de consulta de la que se trate.

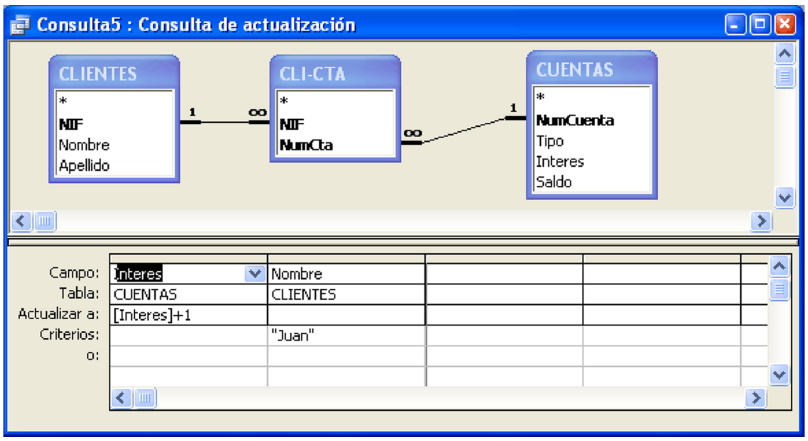

Figura 23. Ventana del editor QBE de Access que representa la consulta 5.

2.6. Consulta 6: Eliminar a Juan como cliente de la entidad bancaria

El cliente de nombre "Juan" ha decidido dejar de ser cliente de la entidad financiera, por lo que debemos eliminarlo de la tabla "Clientes".

La sentencia SQL que nos permitirá llevar a cabo esta acción es:

```
DELETE "Clientes".Nombre
FROM "Clientes"
WHERE ((("Clientes".Nombre)="Juan"));
```

Notemos que es opcional la especificación de los campos a eliminar a continuación de la palabra DELETE, ya que siempre se borran registros completos y nunca campos aislados. Por lo tanto hubieran tenido el mismo resultado las siguientes sentencias SQL:

```
DELETE "Clientes".NIF
FROM "Clientes"
WHERE ((("Clientes".Nombre)="Juan"));
```

O bien:

```
DELETE "Clientes".Apellido
FROM "Clientes"
WHERE ((("Clientes".Nombre)="Juan"));
```

¿Qué ha pasado con la tabla CLI-CTA? Observad que se han borrado también las ocurrencias que hubieran relativas al cliente "Juan", es decir, ya no aparecen registros en la tabla CLI-CTA que tengan "11111111-A" como valor del campo NIF (que es el NIF de Juan). Los contenidos de las tablas "Clientes" y CLI-CTA tras la ejecución de la sentencia SQL son, respectivamente:

CLIENTES : Tabla

	NIF	Nombre	Apellido
+	22222222-B	Pedro	Pérez
+	33333333-C	Helena	Martínez
+	44444444-D	Marta	González
+	55555555-E	Daniel	Fernández
+	66666666-F	Pablo	López
*			

Figura 24. Contenido de la tabla "Clientes" tras la ejecución de la consulta 6

CLI-CTA : Tabla

NIF	NumCta
22222222-B	3
22222222-B	5
33333333-C	1
33333333-C	4
44444444-D	1
55555555-E	5
66666666-F	6
*	0

Figura 25. Contenido de la tabla CLI-CTA tras la ejecución de la consulta 6

Hay que tener un cuidado especial con este tipo de sentencias que actualizan la base de datos Access, puesto que tras su ejecución no tenemos posibilidad de recuperar la situación inmediatamente anterior a la ejecución de ésta (no existe una opción "deshacer").

Por otra parte, también es necesario que entendamos el diseño que hemos hecho de la base de datos en lo que respecta a la exigencia de integridad referencial, ya que esto implicará, como hemos visto, actualización o borrado de registros adicionales a los de la tabla sobre la que los estamos haciendo directamente. Este punto es especialmente importante si tenemos en cuenta que Access no nos va a advertir de estas implicaciones, es decir, no nos va a avisar de los borrados o cambios que va a realizar sobre otras tablas de la base de datos.

Concretamente para esta consulta que hemos ejecutado los dos mensajes de aviso que aparecen antes a la ejecución de la sentencia son:

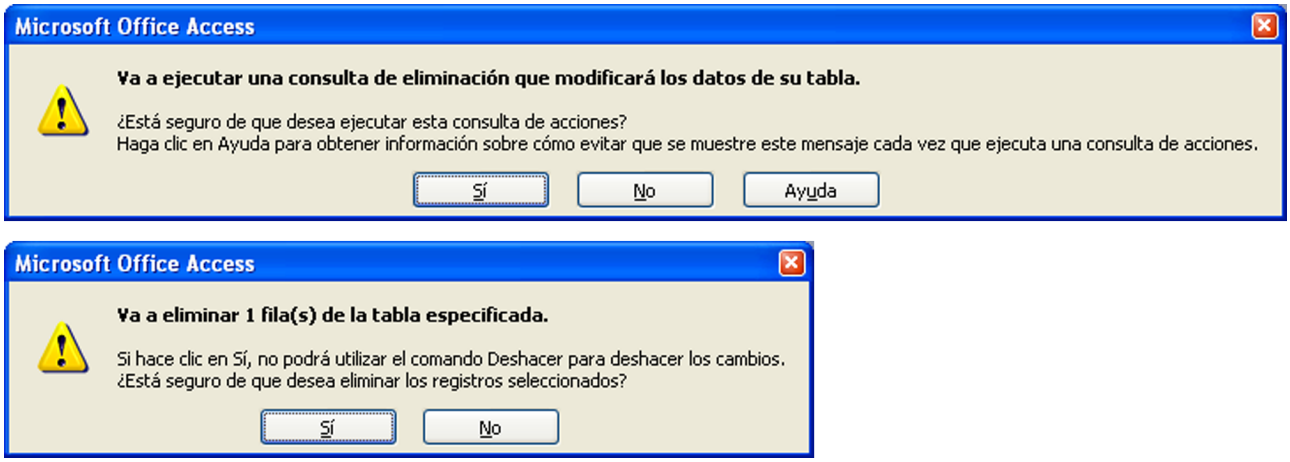

Figura 26. Ventanas de aviso de actualización de tablas de Access

Como podéis ver, sólo nos advierte del borrado de una fila (registro) de una tabla, cuando en realidad eliminaremos un registro de la tabla "Clientes" y dos de la tabla CLI–CTA (estos últimos por haber exigido integridad referencial en la relación "Clientes" y CLI–CTA).

2.7. Consulta 7: Duplicar la tabla "Clientes" sobre una nueva llamada CLIENTES2

Access, si no hay una tabla con el nombre de la tabla destino de la consulta, la crea, lo que nos permite hacer uso de la función SELECT INTO directamente sin necesidad de haber creado previamente la tabla, es decir, sin utilizar CREATE TABLE. Esto resulta muy cómodo cuando las tablas tienen muchos campos, ya que nos ahorramos tener que especificarlos.

Por lo tanto, la sentencia que necesitamos es:

```
SELECT "Clientes" * INTO CLIENTES2
FROM "Clientes";
```

Al igual que para el resto de sentencias de actualización, Access nos advierte de la imposibilidad de deshacer los cambios.

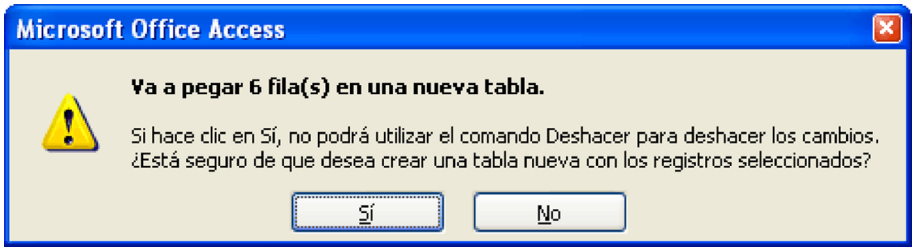

Figura 27. Ventanas de creación de una nueva tabla con incorporación de datos en Access

Utilizando la interfaz QBE de Access deberíamos seguir estos pasos:

1) Elegir la opción Nueva Consulta, Vista Diseño y seleccionar la tabla "Clientes" en la ventana Mostrar Tabla.

2) Seleccionar "Consulta de Creación de Tabla..." de los tipos de consulta.

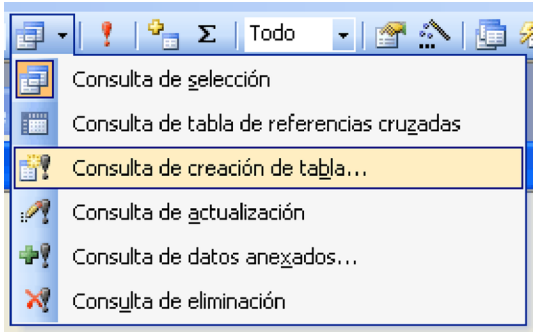

Figura 28. Menú para la selección de una consulta de creación de tabla en Access

3) Indicar la tabla nueva que queremos crear. En este punto hay que tener en cuenta que Access nos presenta la posibilidad de incorporar los registros sobre tablas ya existentes que incluso pueden ser de otras bases de datos (os animamos a que probéis estas posibilidades, que son muy interesantes de cara a realizar copias de tablas sobre otras bases de datos).

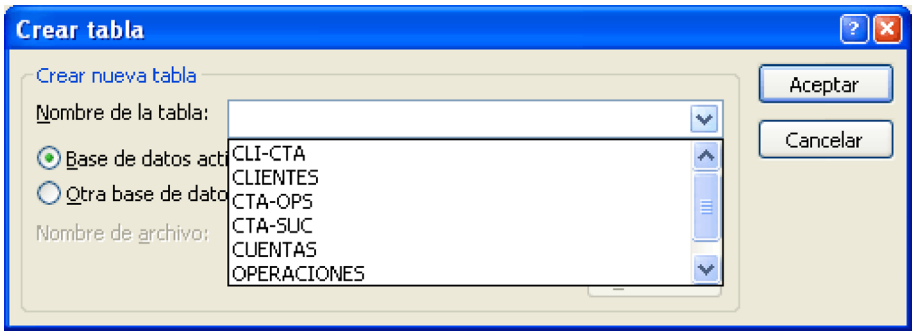

Figura 29. Ventana de selección del nombre de la nueva tabla que vamos a crear

4) A continuación debemos indicar en la rejilla *grid* los campos desde los que se nutrirá la información sobre la nueva tabla "Clientes2". Es decir, desde "Clientes", indicaremos que son todos los campos de "Clientes", lo que podemos hacer, como ya sabemos, mediante el *.

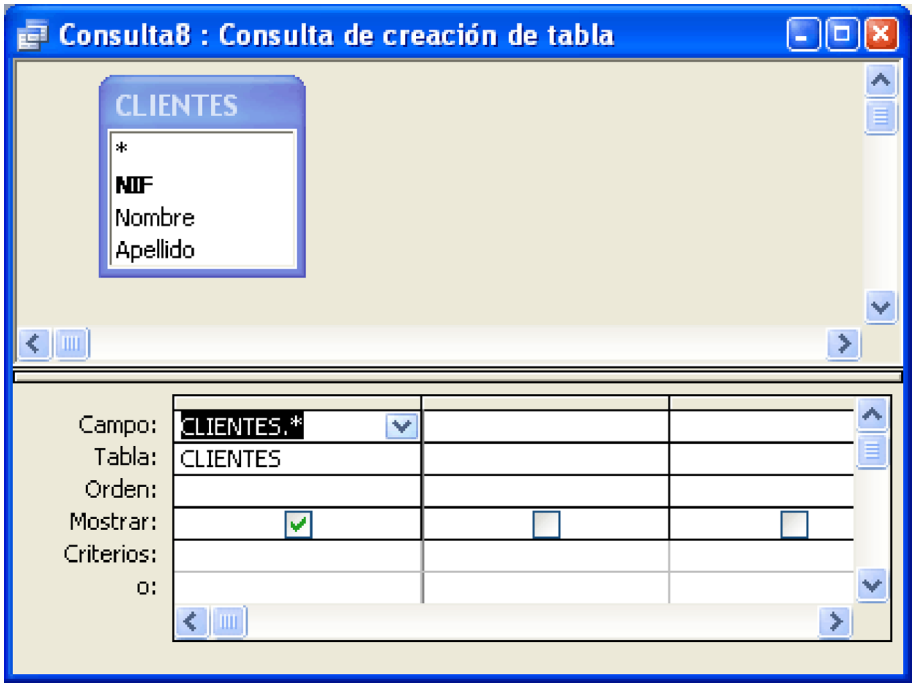

Figura 30. Ventana del editor QBE de Access que representa la consulta 7.

2.8. Consulta 8: Incorporar un nuevo cliente a la tabla CLIENTES2

También es posible que queramos incorporar nuevos registros sobre tablas. En este caso, se nos pide que incorporemos a la tabla CLIENTES2 (que acabamos de crear en el paso anterior) un nuevo cliente llamado Mario García y con NIF 77777777-G.

Utilizando el SQL, lo haríamos de la siguiente manera:

```
INSERT INTO Clientes2
VALUES ("77777777-G","Mario","García");
```

En este caso, el editor QBE de Access no permite hacer uso de esta funcionalidad, por lo que si intentamos mostrarlo, nos aparecerá el aviso:

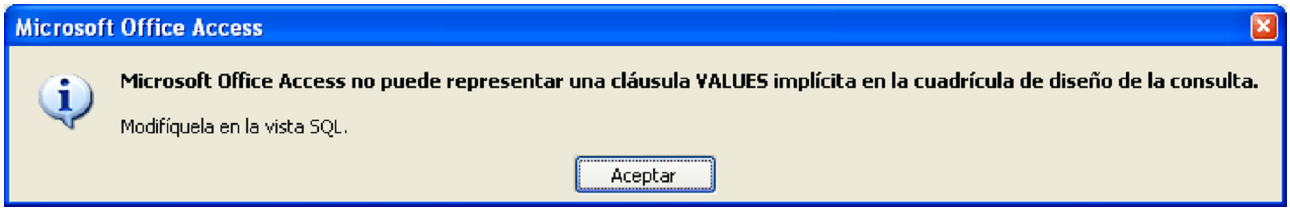

Figura 31. Aviso de error de Access por uso de sentencia VALUES en editor QBE

Bibliografía

Date, C. J. (2001). *Introducción a los sistemas de bases de datos* (7.ª ed.). Prentice Hall.

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (1998). *Fundamentos de bases de datos* (3.ª ed.). McGraw-Hill.

Celma, M.; Casamayor, J. C.; Mota, L. (2003). *Bases de datos relacionales*. Pearson, Prentice Hall.