Caso práctico sobre bases de datos relacionales

Pedro Cardona Vilaplana

PID_00201457

Universitat Oberta de Catalunya

Índice

Introducción

Las bases de datos son herramientas muy potentes que permiten 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 documento vamos a poner en práctica conceptos tratados en los tres primeros módulos didácticos de la asignatura de Bases de datos: "Sistemas de base de datos", "El modelo relacional y el álgebra relacional" y "El lenguaje SQL". Especialmente 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 repasando (de una manera muy práctica y con ejemplos) algunos conceptos clave estudiados en el módulo "El modelo relacional y el álgebra relacional" y, principalmente, con ejercicios para poner en práctica estos conceptos por medio de una base de datos de ejemplo que nosotros mismos construiremos.

Dado que queremos poner en práctica la teoría aprendida, es muy recomendable que desarrolléis los ejercicios propuestos aquí creando una base de datos en vuestro propio ordenador.

Para hacer los ejercicios de ejemplo e implementar la base de datos del caso práctico, utilizaremos un sistema gestor de bases de datos relacional de amplia difusión: Access. De todos modos, puesto que hemos utilizado ejemplos y sentencias SQL estándar, podréis hacer fácilmente los distintos ejercicios sobre cualquier otro sistema gestor de bases de datos relacional como, por ejemplo OpenOffice. Los resultados que obtendréis serán los mismos.

1. Conceptos necesarios

1.1. El Modelo entidad-interrelación

El Modelo entidad-interrelación (E/R, del inglés *entity-relationship*), propuesto por Peter Chen en el año 1976, también llamado *Modelo Conceptual de Datos*, es una técnica de representación de las relaciones que tienen los datos y que permite recrear la realidad que queremos modelizar en nuestra base de datos.

Ved también

El concepto de modelo de datos se describe en el apartado "Modelo de datos" del módulo "Sistemas de 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 (utilizando símbolos gráficos) del diseño de los datos, y no de lo que se pretende hacer con ellos.

1.2. Entidades y atributos

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

Una entidad (o tipo de entidad) está formada por un conjunto de ocurrencias de entidad del mismo tipo.

Ejemplo de entidad

En una empresa, la entidad formada por el conjunto de empleados se podría denominar EMPLEADO.

Ved también

Encontraréis definiciones y ejemplos de ocurrencias y de tipos de entidades en los apartados "Ocurrencia y tipos" y "Clasificación (e instanciación)" del módulo "Sistemas de base de datos".

Entidad tipo y ocurrencia

- Un tipo de entidad (o entidad tipo) es una categoría generalizada que define un conjunto de entidades más específicas con los mismos atributos. Normalmente, se abrevia y solo se utiliza el término *entidad*.
- Una ocurrencia de entidad es un ejemplar de un tipo de entidad que comparte atributos con otras ocurrencias, cada una de las cuales tiene su propio valor para cada atributo. También se usa el término *instancia*.

Ejemplos de ocurrencia de entidad

Juan y Pedro son ocurrencias del tipo de entidad PERSONA.
Enero y febrero son ocurrencias del tipo de entidad MES.

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

Ejemplo de atributos de una entidad

En la entidad EMPLEADO, algunos atributos posibles serían: *DNI*, *nombre*, *apellidos*, *fecha de nacimiento*, etc.

1.3. Interrelaciones

Una interrelación es una asociación[1] que se da entre diferentes entidades.

Ejemplo de interrelación

En una empresa, además de la entidad EMPLEADO también tendremos una denominada DEPARTAMENTO, que es la división organizativa o funcional de la empresa a la que está adscrito el empleado. Sobre estas entidades, podríamos definir una interrelación que asociara al empleado Pedro con el departamento de Contabilidad (esta interrelación se puede denominar EMP-DEP o PERTENECE).

El término *relación*

El término *relación* se refiere a una tabla en el modelo relacional, que representa una entidad o una interrelación en el modelo conceptual E/R. Lo utilizan algunos autores para referirse de manera poco ortodoxa al concepto de *interrelación* (en inglés, *relationship*) entre entidades, y algunos SGBD para hacer referencia al vínculo entre tablas.

Además, una interrelación puede tener también sus propios atributos.

Ejemplo de atributos de una interrelación

En el caso anterior, un atributo de la interrelación de Pedro con el departamento de Contabilidad podría ser la *fecha de adscripción* a este departamento.

1.4. Claves

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

Ejemplos de clave

En la entidad EMPLEADO, el campo *NIF* sería la clave puesto que no encontramos ningún conjunto menor que este para identificar de manera unívoca a cada uno de los empleados.

Si asumimos que un país no tiene ciudades con el mismo nombre, el conjunto de atributos de la entidad CIUDAD formado por *país* y *nombre de ciudad* sería la clave, ya que ningún conjunto menor puede identificar de manera unívoca cada una de las ciudades del mundo. Por ejemplo, el subconjunto formado solo por el atributo *nombre de ciudad* no puede ser clave, puesto que en todo el mundo hay ciudades que comparten el nombre. Es el caso de Barcelona, Guadalajara, Santiago, Sydney, etc.

Nomenclatura, ortografía y formato

El nombre de los objetos de la base de datos se escribe con caracteres alfanuméricos, habitualmente sin acentos. Puede incluir determinados caracteres especiales (como guión "-" o guión bajo "_"). Denotaremos las entidades y las interrelaciones en letra mayúscula y en singular; y los atributos, en letra minúscula y cursiva.

(1)El concepto de *asociación* en el modelo entidad-interrelación hace referencia a la interrelación entre entidades, y se describe en el apartado "Asociación (y disociación)" del módulo "Sistemas de base de datos".

El nombre de las interrelaciones

El nombre de las interrelaciones se puede formar separando con un guión la abreviación (tres letras) de las entidades asociadas. En ocasiones, se puede usar el verbo (en infinitivo o en tercera persona del singular) que describe la asociación entre las entidades asociadas.

En una entidad, es posible que haya más de una clave. Todas las claves posibles se denominan **claves candidatas**, mientras que la clave elegida por el diseñador de la base de datos para identificar cada entidad (o instancia) se denomina **clave primaria**.

Especificidad terminológica de Access

- La clave primaria se denomina *clave principal*.
- La interrelación, asociación o vínculo entre tablas se denomina *relación*. Insistimos en no confundir este término con el concepto de relación (tabla) en el modelo relacional.

La **clave ajena** es el conjunto de atributos de una entidad que, a su vez, es clave primaria de otra entidad con la que está interrelacionada.

1.5. Clasificación de las entidades en función de las claves

- **Entidades fuertes**: son aquellas que tienen una clave primaria. Tienen existencia por sí mismas.

- **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 cada uno de sus atributos mediante una interrelación. Sus ocurrencias son identificables solo por estar asociadas a otra entidad (entidad fuerte). Su existencia depende de la existencia de otra entidad.

Aunque una entidad débil no tiene clave primaria, se necesita conocer un medio para distinguir todas las entidades que dependen de una entidad fuerte particular. El conjunto mínimo de atributos que lo permite se denomina **discriminante** de una entidad. La clave primaria de una entidad débil está formada por el discriminante más la clave primaria de la entidad fuerte con la que está asociada.

Ejemplo de entidad fuerte, entidad débil y discriminante

Siguiendo con el caso de empleados de una empresa, tendríamos una entidad denominada NOMINA que tiene los atributos *numNomina*, *fecha* e *importe*. El atributo *numNomina* es el número de nómina que ha cobrado el empleado (el valor de *numNomina* para la primera nómina que cobra es 1; para la siguiente es 2, y así de manera sucesiva). La entidad NOMINA por sí misma no tiene una clave que identifique a una entidad de manera unívoca (podríamos tener a dos empleados que en la primera nómina, pagada el mismo día, hubieran cobrado lo mismo). En este caso, NOMINA depende de EMPLEADO para existir, por lo que el discriminante sería *numNomina* (para un mismo EMPLEADO sí que se puede identificar de manera unívoca cada nómina). Por lo tanto, la clave primaria de NOMINA sería *NIF* (clave primaria de la entidad fuerte con la que se asocia, EMPLEADO) más *numNomina* (discriminante, es decir, que permite la identificación unívoca de cada entidad NOMINA para cada empleado).

1.6. Diagramas entidad-interrelación

El Diagrama entidad-interrelación (E/R) permite representar gráficamente la estructura lógica de una base de datos mediante los siguientes elementos:

- Rectángulos: representan las entidades.
- Elipsis: representan los atributos de las entidades y de las interrelaciones.
- Rombos: representan las interrelaciones entre las entidades.
- Líneas: enlazan atributos a entidades, atributos a interrelaciones y entidades a interrelaciones. Nunca enlazan entidades con entidades.

Concepto relacional	Representación gráfica
Entidad	
Interrelación	
Atributo	

Ejemplo de diagrama E/R

Si queremos modelizar las notas de las asignaturas de una titulación que han obtenido los alumnos, definiremos las entidades siguientes.

- ALUMNO: contiene los datos, de los que conocemos *DNI*, *nombre* y *apellidos*.
- ASIGNATURA: contiene los datos de las asignaturas, de las que sabemos *identificador de asignatura*, *nombre de la asignatura* y *curso* al que pertenece.

El modelo entidad-interrelación correspondiente se representaría con el diagrama E/R siguiente:

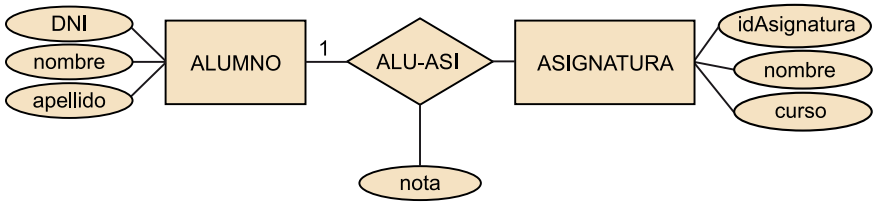

Figura 1. Diagrama E/R que representa la interrelación ALU-ASI entre las entidades ALUMNO y ASIGNATURA.

1.7. Cardinalidad de las interrelaciones

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 interrelación. Dicho de otro modo, expresa el número de entidades con las que puede asociarse una entidad.

De acuerdo con lo anterior, entre dos entidades A y B se pueden establecer las correspondencias siguientes.

- **Una a una (1:1)**: cada instancia u ocurrencia de la entidad A se asocia, 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 asocia con un número cualquiera de instancias u ocurrencias de la entidad B, mientras que una instancia u ocurrencia de la entidad B se asocia, 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 asocia, como máximo, con una instancia u ocurrencia de la entidad B, mientras que una instancia u ocurrencia de la entidad B se asocia 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 asocia con un número cualquiera de instancias u ocurrencias de la entidades B y viceversa.

La manera de representar la cardinalidad es la siguiente:

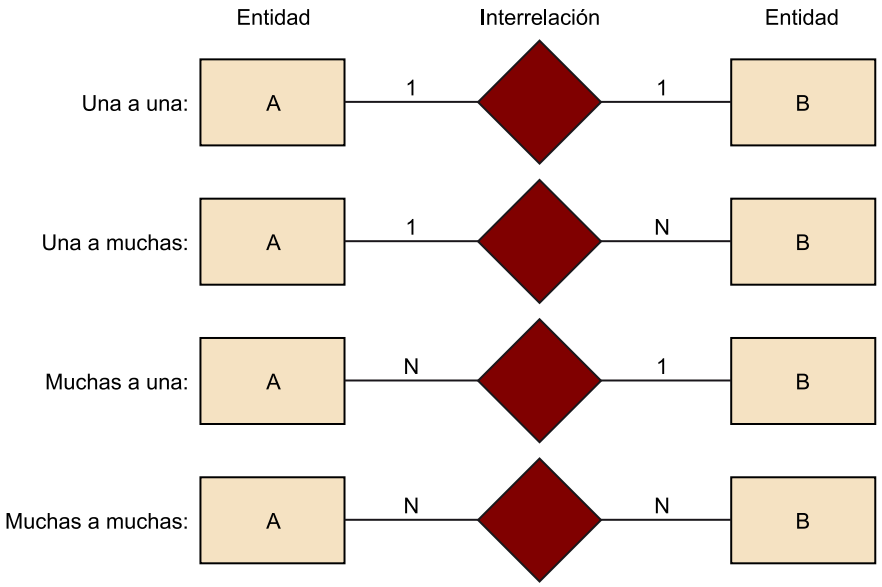

Figura 2. Representación de las diferentes correspondencias o razones de cardinalidad de las interrelaciones entre entidades.

Ejemplos de razón de cardinalidad

Si consideramos que un préstamo bancario puede ser, como máximo, de un cliente y, a su vez, que un mismo cliente puede tener varios préstamos, entonces la razón de cardinalidad de la interrelación entre las entidades CLIENTE y PRESTAMO es de una a muchas (1:*N*).

Si, como variación del ejemplo anterior, tenemos que un préstamo no solo puede pertenecer a un cliente sino también a más (los titulares del contrato de préstamo son varios), entonces estaremos ante una interrelación con razón de cardinalidad de muchas a muchas (*N:M*).

1.8. Generación del modelo relacional a partir del modelo entidad-interrelación

Una vez elaborado el modelo entidad-interrelación 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 modelo entidad-interrelación son las siguientes:

- Cada entidad fuerte se convierte en una tabla, y sus atributos en campos de la tabla.
- Cada entidad débil se transforma en una tabla cuyos campos son los atributos de la entidad débil más la clave primaria de la entidad fuerte con la que se asocia.
- Cada interrelación se transforma en una tabla, cuyos campos son las claves primarias de cada una de las entidades con las que se asocia más los atributos propios de la interrelación.
- Toda interrelación con razón de cardinalidad *N*:*M* entre dos entidades se transforma en una tabla intermedia, y se asocia con las tablas de las dos entidades anteriores mediante razones de cardinalidad 1:*N*.

El modelo relacional

El modelo relacional postulado por E. F. Codd en 1970 se basa en la lógica de predicados y la teoría de conjuntos. La idea fundamental es el uso de relaciones. Una relación se conceptualiza como si fuera una tabla compuesta por conjuntos de datos en **registros** (filas o tuplas) y campos (columnas).

Ved también

El concepto de tabla o relación se presenta en el apartado "Visión informal de una relación" del módulo "El modelo relacional y el álgebra relacional".

Recordad que los conceptos de *entidad* e *interrelación* en el modelo conceptual E/R se corresponden a un único concepto en el modelo relacional: el de *relación* (o *tabla*).

Tabla = relación

En el modelo relacional, el concepto de *relación* es sinónimo de *tabla*.

1.9. Ejemplo de conversión de un modelo entidad-interrelación 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 bancaria se identifica por un número de operación, una fecha de operación y un importe.
- Un cliente puede tener muchas cuentas.
- Una cuenta puede tener varios clientes.
- Una cuenta solo puede pertenecer a una sucursal.

Estas tres restricciones se pueden esquematizar de la manera siguiente:

- 1 cliente: N cuentas
- 1 cuenta: N clientes

→ N clientes: N cuentas

- 1 sucursal: N cuentas

A partir de las restricciones anteriores, identificamos las siguientes entidades con sus correspondientes atributos del modelo entidad-interrelación:

- CLIENTE (*NIF*, *nombre*, *apellido*)
- SUCURSAL (*numSuc*, *direccion*, *poblacion*, *telefono*)
- CUENTA (*numCta*, *tipo*, *interes*, *saldo*)
- OPERACION (*numOpe*, *fecha*, *importe*)

Este modelo E/R se puede representar de manera gráfica mediante el diagrama E/R siguiente:

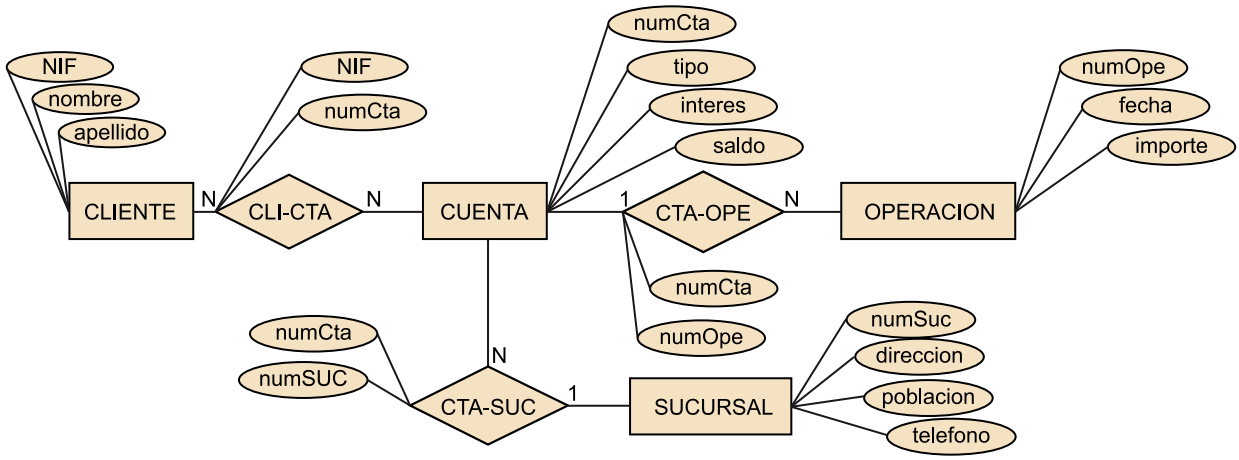

Figura 3. Modelo E/R de la base de datos de la entidad financiera del ejemplo.

Las entidades se asocian por medio de tres interrelaciones que también hemos representado.

Nombres de tablas y campos

De manera similar al convenio expuesto en el apartado 1.3 de este documento, escribimos las tablas en mayúscula y en singular y los campos, en minúscula y cursiva.

Si el nombre está formado por diferentes palabras, conviene escribirlo sin espacios, según la notación siguiente:

- **Tablas**: separando las palabras con un guión bajo (por ejemplo, PEDIDO_PENDIENTE) o, de manera inusual, siguiendo la sintaxis Pascal Case (letra inicial de cada palabra en mayúscula; por ejemplo: PedidoPendiente).
- **Campos**: según la sintaxis Camel Case (letra inicial de cada palabra en mayúscula, excepto la primera, en minúscula; por ejemplo, *codigoPostal*, *fechaDeNacimiento*). Ocasionalmente, también se usa la notación "Underscore separated"[2].

[(2)]Sintaxis para escribir nombres, que consiste en separar las palabras con un guión bajo.

Aplicando las reglas de transformación de entidades, interrelaciones y atributos, el diagrama E/R se convierte en la estructura de tablas siguiente (denominada esquema de la base de datos):

Tablas procedentes de entidades	Tablas procedentes de interrelaciones
CLIENTE (*NIF, nombre, apellido*)	CLI-CTA (*NIF, numCta*)
SUCURSAL (*numSuc, direccion, poblacion, telefono*)	CTA-OPE (*numCta, numOpe*)
CUENTA (*numCta, tipo, interes, saldo*)	CTA-SUC (*numCta, numSuc*)
OPERACION (*numOpe, fecha, importe*)	

Ved también

El concepto de esquema se describe en el apartado "Esquema de la base de datos" del módulo "Sistemas de base de datos". En los apartados "La arquitectura de tres niveles de los ANSI/SPARC" y "El nivel conceptual" del mismo módulo se explica la implicación del esquema conceptual de la base de datos en los SGBD.

Podemos ver lo siguiente:

- Las entidades y las interrelaciones se han transformado en tabla.
- Los campos de las tablas procedentes de entidades (CLIENTE, SUCURSAL, CUENTA y OPERACION) son los atributos de estas entidades.
- Los campos de las tablas procedentes de interrelaciones (CLI-CTA, CTA-SUC y CTA-OPE) son las claves primarias de cada una de las entidades que se asocian.
- La clave primaria se denota subrayando el nombre de los campos que la forman.

La estructura de tablas implementada con Access es la siguiente:

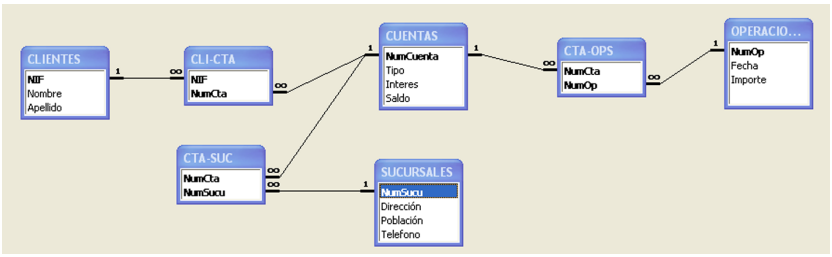

Figura 4. Estructura de tablas correspondiente al modelo E/R de la figura 3

Nota

La clave primaria de cada tabla se destaca en negrita. La razón de cardinalidad entre tablas se indica sobre la línea que las une mediante asociaciones 1 a ∞ (una a muchas).

1.10. Integridad referencial

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

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

Nota

Recordad que en Access la clave primaria se denomina *clave principal*.

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 ajena es la clave principal de la tabla de cardinalidad 1.

Por ello, a la hora de definir las interrelaciones, 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 las siguientes:

Nota

Recordad que en Access la asociación o vínculo entre tablas se denomina *relación*. Por lo tanto, en este SGBD, *relación* no es sinónimo de *tabla*.

1) Cuando llevamos a cabo una eliminación: 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 las eliminaciones en las tablas asociadas. Si eliminamos un registro de una tabla con cardinalidad 1 asociada mediante una razón de cardinalidad 1:*N* con otra, también se eliminarán los registros de la tabla asociada con cardinalidad *N*.

Ejemplo de eliminación de registros por propagación

Partiendo del caso que estamos abordando (la entidad financiera), si eliminamos al cliente con *NIF* 21.212.121-A (de la tabla CLIENTE, con cardinalidad 1) y exigimos integridad referencial, también se eliminarán los registros con este *NIF* de la tabla CLI-CTA (con cardinalidad *N*).

2) Cuando llevamos a cabo cambios en los campos que son clave de una tabla: de manera similar al caso anterior de las eliminaciones, es posible propagar los mismos cambios en los campos que son clave ajena de las tablas con las que se asocia la tabla original. Si modificamos un registro de un atributo clave de una tabla con cardinalidad 1 asociada mediante una razón de cardinalidad 1:*N* con otra, también se actualizarán al mismo valor los campos que son clave ajena de la tabla asociada con cardinalidad *N*.

Ejemplo de modificación de registros por propagación

Si tenemos un *NIF* de un cliente mal grabado y lo modificamos, cambiando el valor 20.200.200-B por 21.212.121-A, tenemos la actualización reflejada tanto en la tabla CLIENTE como en la tabla 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 la opción de exigir integridad referencial cuando definimos las interrelaciones entre tablas, tal y como muestra el cuadro de diálogo de la imagen siguiente:

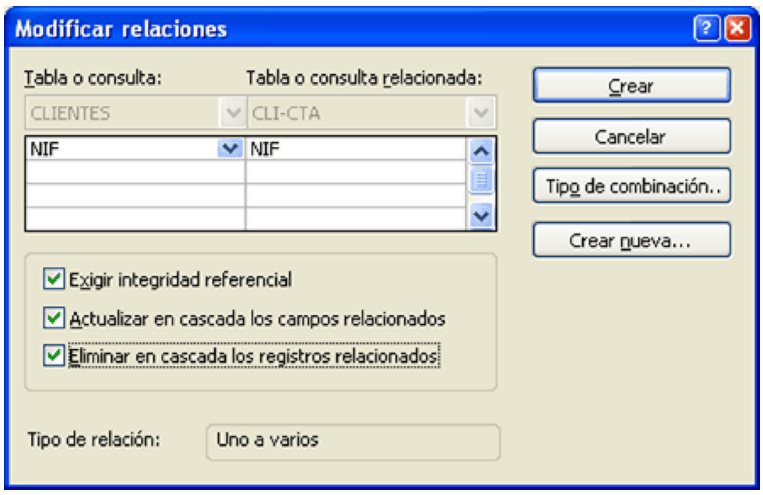

Figura 5. Cuadro de diálogo que muestra la exigencia de integridad referencial de una interrelación en Access.

1.11. Lenguaje SQL (*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 más ampliamente utilizado por los sistemas gestores de bases de datos. 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 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, para insertar datos, modificar datos en la base de datos, etc.

1.12. Contenido de la base de datos del ejemplo

Sobre la base de datos de cuentas y operaciones de los clientes de una entidad bancaria (expuesta en el apartado 1.9), vamos a considerar que tenemos cargada la siguiente información:

a) Sobre la entidad SUCURSAL:

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 SUCURSAL.

b) Sobre la entidad CLIENTE:

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 CLIENTE.

c) Sobre la entidad OPERACION:

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 OPERACION.

d) Sobre la entidad CUENTA:

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 CUENTA.

e) Sobre la interrelación CLI-CTA (entre las entidades CLIENTE y CUENTA):

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.

f) Sobre la interrelación CTA-OPE (entre las entidades CUENTA y OPERACION):

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-OPE.

g) Sobre la interrelación CTE-SUC (entre las entidades CUENTA y SUCURSAL):

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.

1.13. Método de consultas QBE (*Query By Example*)

Los sistemas gestores de bases de datos habitualmente permiten la generación de consultas utilizando ejemplos (*Query By Example*, QBE), 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 sistema QBE que resulta relativamente sencillo de manejar.

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

Para acceder al QBE de Access, debemos seleccionar la opción de crear una nueva consulta en la vista Diseño.

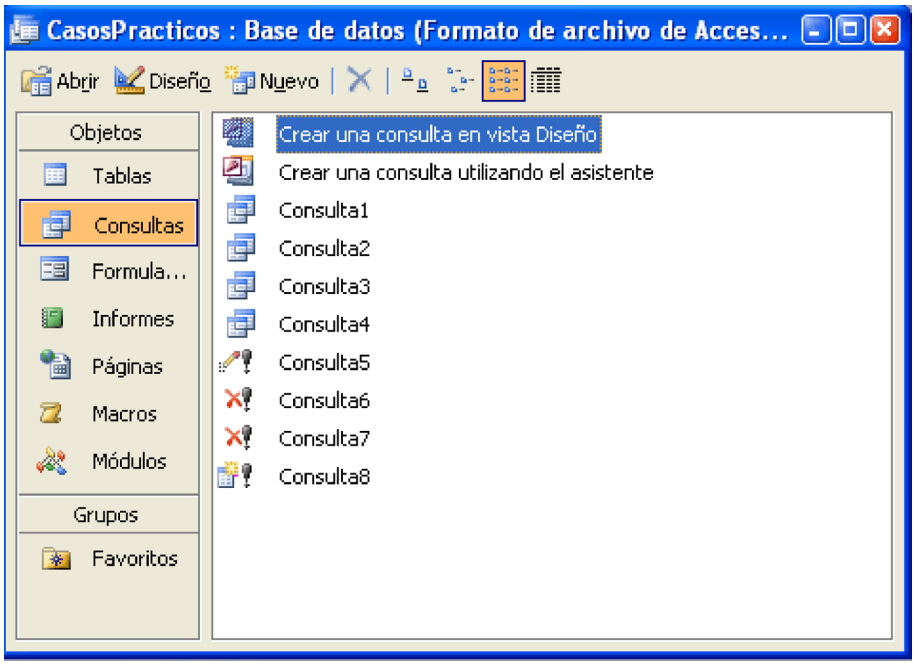

Figura 13. Ventana de aplicación con el detalle del acceso al QBE de Access.

Lo que aparece a continuación es un cuadro de diálogo emergente desde el que podemos seleccionar la tabla o tablas que vamos a utilizar para hacer nuestras consultas y una parrilla (en inglés, *grid*) sobre la que podremos arrastrar y soltar los campos que queramos de las tablas que hayamos seleccionado.

Grid o parrilla

Cuadrícula de filas y columnas con celdas sobre las cuales se pueden arrastrar los campos de las tablas de la base de datos.

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.

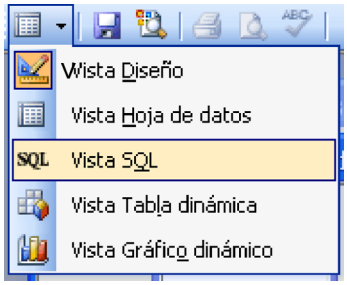

Figura 14. Menú de vistas con la opción de acceso a la redacción y visualización de sentencias SQL en Access.

Para simplificar la representación de los campos se utiliza, al igual que en SQL, el símbolo asterisco (*) 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" (es decir, el resultado de la consulta) calcula en el momento de lanzar la consulta que la utiliza. Es decir, trabajaremos con datos actualizados y no estáticos.

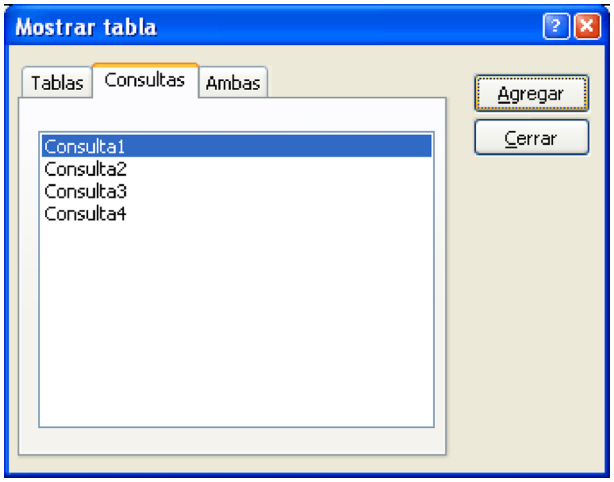

Figura 15. Cuadro de diálogo "Mostrar tabla" de Access que permite utilizar las consultas anteriores como si fueran tablas.

2. Caso práctico

Para el desarrollo de este caso práctico vamos a partir de la base de datos de la entidad financiera formada por las tablas CLIENTE, CUENTA, OPERACIONES y SUCURSAL que hemos creado en el ejemplo del apartado 1.9 de este documento.

Remisiones al material didáctico

Todas las referencias facilitadas en este caso práctico remiten a apartados del módulo "El lenguaje SQL" de la asignatura *Bases de datos*.

Sobre la base de datos, plantearemos una serie de consultas que tendremos que resolver mediante el lenguaje SQL y el editor QBE de Access.

2.1. Consulta 1: obtener los nombres de clientes ordenados alfabéticamente

La sentencia SQL que deberemos ejecutar es la siguiente:

Ved también

La instrucción SELECT FROM y la opción ORDER BY se describen respectivamente en los apartados "Consultas a una base de datos relacional" y "Ordenación de los datos obtenidos en respuestas a consultas" del módulo "El lenguaje SQL".

```
SELECT CLIENTE.nombre
FROM CLIENTE
ORDER BY CLIENTE.nombre
```

Hacerlo desde el editor QBE es igual de sencillo. Veamos los pasos a realizar:

1) En el cuadro de diálogo "Mostrar tabla", seleccionamos la tabla sobre la que queremos hacer la consulta; en este caso, CLIENTE. A continuación, cerramos este cuadro de diálogo.

2) Arrastramos el campo *nombre* de la tabla CLIENTE que aparece en la parte superior de la ventana y lo soltamos en la primera columna del *grid* (la parrilla que aparece en la parte inferior de la ventana). De este modo se seleccionan todos los clientes.

3) Como queremos que el resultado esté ordenado alfabéticamente, seleccionamos la opción "Ascendente" en la fila "Orden" del *grid*.

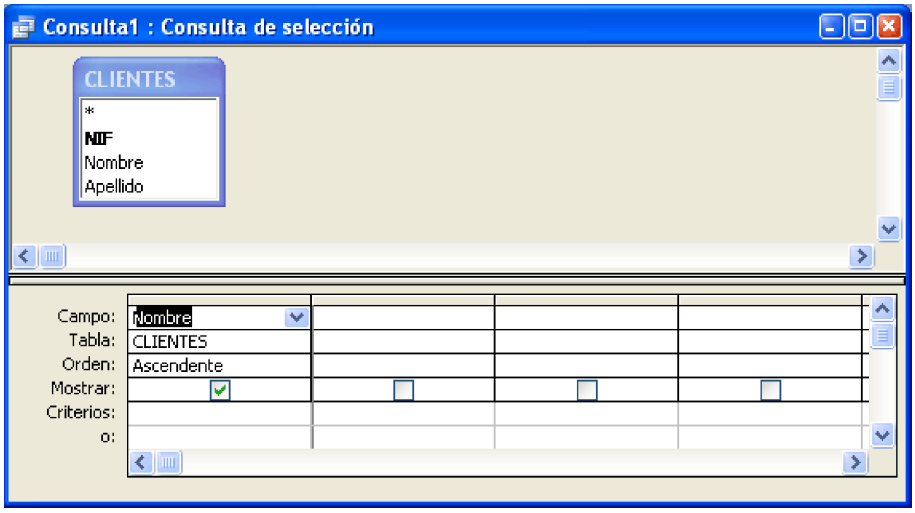

Figura 16. Ventana del editor QBE de Access que representa la consulta 1.

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

Para obtener la información de cuentas con un *NIF* determinado, debemos aplicar la operación de consulta SQL siguiente:

```
SELECT CUENTA.*, CLIENTE.NIF
   FROM CUENTA INNER JOIN (CLIENTE INNER JOIN
   [CLI-CTA] ON CLIENTE.NIF = [CLI-CTA].NIF) ON
   CUENTA.numCta = [CLI-CTA].numCta

WHERE (((CLIENTE.NIF) = 11111111-A));
```

La cláusula WHERE

La cláusula WHERE permite seleccionar las filas que cumplen una determinada condición.

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 (CLIENTE, CLI-CTA y CUENTA), 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 la siguiente:

Ved también

La instrucción de combinación interna INNER JOIN con la opción ON se describe en el apartado “Consultas a más de una tabla”.

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

La opción ON

La opción ON permite expresar las condiciones de combinación con operadores de comparación (=, <, >, <=, >=, o <>).

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

1) Desde el cuadro de diálogo "Mostrar tabla", seleccionamos las tablas que necesitaremos para esta consulta, que son CLIENTE, CUENTA, CLI-CTA. A continuación, cerramos el cuadro de diálogo.

2) Arrastramos todos los campos de la tabla CUENTA, es decir, arrastramos el símbolo asterisco (*) y lo soltamos en la primera columna del *grid* inferior.

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

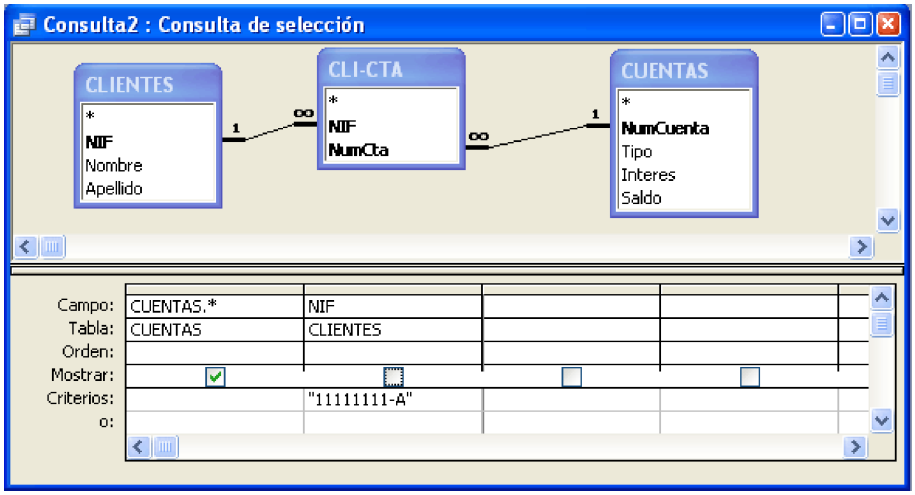

Figura 17. Ventana del editor QBE de Access que representa la consulta 2.

Hay una manera más sencilla de efectuar esta consulta: simplemente con dos tablas en vez de tres, utilizando exclusivamente las tablas CUENTA y CLI-CTA.

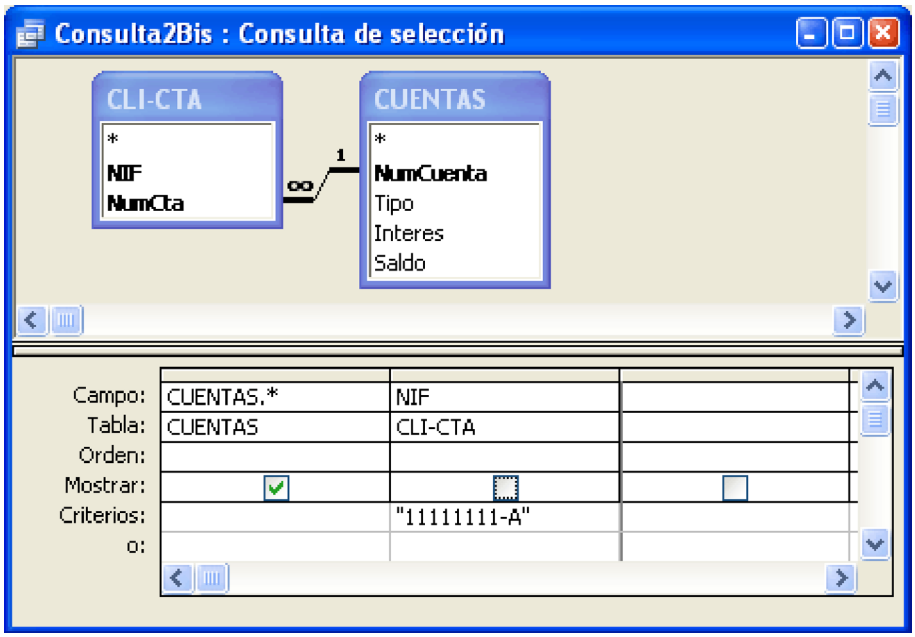

Figura 18. Ventana del editor QBE de Access que representa la consulta 2 con solo dos tablas.

2.3. Consulta 3: calcular el importe de los 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:

- El cálculo de intereses para todos los productos de la entidad se calcula multiplicando el saldo por el interés. Esto lo expresamos de la manera siguiente: *saldo* * *interes*.

- 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 CLIENTE.nombre, Sum(([interes]*[saldo])) AS subtotal
   FROM CUENTA INNER JOIN (CLIENTE INNER JOIN
   [CLI-CTA] ON CLIENTE.NIF = [CLI-CTA].NIF) ON
   CUENTA.numCta = [CLI-CTA].numCta
GROUP BY CLIENTE.nombre;
```

La opción AS permite la definición de alias, es decir, renombrar los campos que queremos seleccionar o las tablas que deseamos consultar. Observad que, en este caso, hemos dado un alias al campo calculado y lo hemos denominado *subtotal*.

Ved también

Podéis encontrar la sintaxis de la sentencia SELECT FROM con la opción AS en el apartado "Consultas a una base de datos relacional".

Por otro lado, la cláusula GROUP BY permite agrupar registros según el campo indicado. En este caso, agrupamos a los clientes por *nombre*.

Ved también

La sintaxis de la cláusula GROUP BY se indica en el apartado "Consultas con agrupación de filas de una tabla".

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 los siguientes:

1) Desde el cuadro de diálogo "Mostrar tabla", seleccionamos las tablas CLIENTE, CLI-CTA y CUENTA. Cerramos el cuadro de diálogo "Mostrar Tabla".

2) Desde la tabla CLIENTE 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 *interes* multiplicado por *saldo*. Dado que en esta consulta no hay campos que se denominen igual en la tabla CUENTA y en la tabla CLIENTE, podemos evitar escribir la tabla de la que provienen. Si no fuera así, las tendríamos que referenciar precediendo los campos con el nombre de la tabla y un punto.

Nota

Recordad que, en una consulta, no siempre hay que indicar para cada campo el nombre de la tabla separado con un punto. Esto solo es necesario cuando seleccionamos datos de tablas diferentes pero con el mismo nombre de campo.

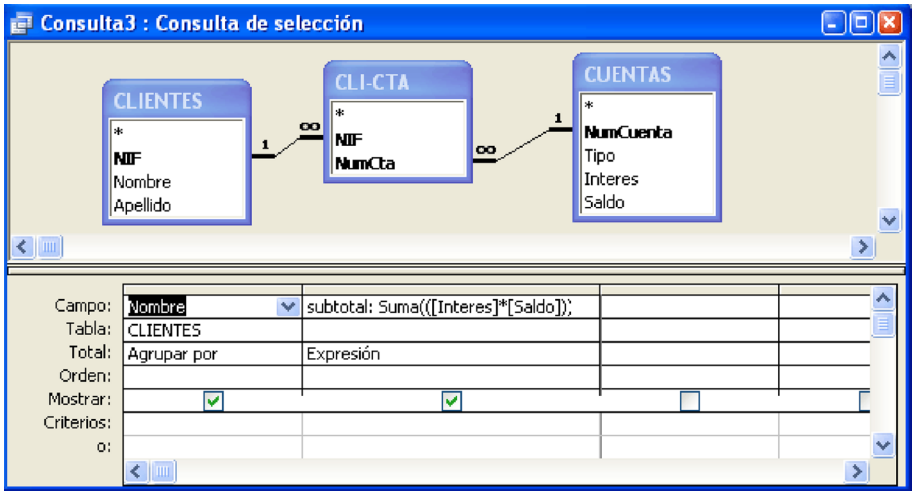

Figura 19. Ventana del editor QBE de Access que representa la consulta 3.

Veamos cual es la utilidad de asignar 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.

Actividad

Tablas diferentes con campos del mismo nombre

Con esta consulta 3, podéis intentar indicar el nombre de la tabla separada con un punto del nombre del campo. Esto es útil cuando hay que tratar con diferentes tablas que contengan algunos campos con nombre idéntico.

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 (CUENTA.saldo)
FROM CUENTA;
```

La función de agregación SUM suma los valores de una columna; en este caso, la columna saldo de la tabla CUENTA.

Ved también

Podéis encontrar un ejemplo de uso de la función SUM al final del apartado "Consultas con agrupación de filas de una tabla".

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

1) Como siempre, desde el cuadro de diálogo "Mostrar tabla" seleccionamos las tablas que necesitamos para nuestra consulta; en este caso, solo la tabla CUENTA.

2) Desde la barra de herramientas pinchamos sobre el botón con 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.

El símbolo Σ

El símbolo Σ es la letra sigma mayúscula del alfabeto griego y se utiliza con mucha frecuencia como notación de sumatorio (operador matemático que representa de manera compacta la suma de un conjunto de números).

3) Arrastramos el campo *saldo* desde la tabla CUENTA a la primera columna del *grid*.

4) En la fila "Total", aparece la opción "Agrupar por", y si desplegamos el menú veremos que hay varias opciones. Como es lógico, seleccionamos la opción "Suma".

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

Con frecuencia, resulta útil aprovechar este cálculo para otras consultas reutilizando la consulta como si fuera una tabla más. En este caso, sin embargo, y a diferencia del anterior, no hemos asignado un alias a este cálculo.

De nuevo, la *grid* también permite hacer este cálculo dando un alias. Simplemente, tenemos que desactivar el símbolo Σ y escribir la fórmula de cálculo en la primera fila de la *grid*. En este caso, el cálculo es la suma de saldos, que expresamos del modo siguiente: Suma (*saldo*).

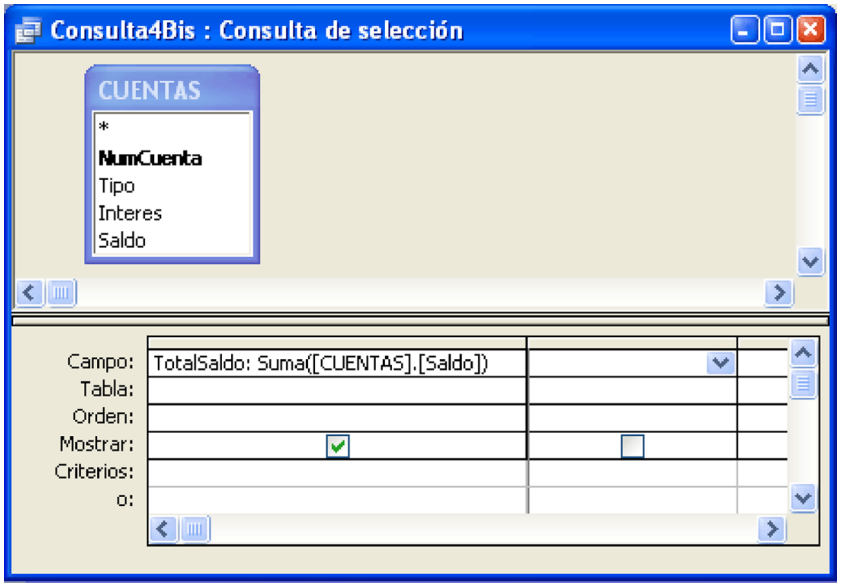

Figura 21. Ventana del editor QBE de Access que representa la consulta 4 sin utilizar un alias.

Para que sirva de guía a la hora de referenciar campos de las tablas, para este ejemplo se ha utilizado la ruta completa del campo, es decir, "[CUENTA].[saldo]", aunque simplemente con *saldo* hubiera sido suficiente, ya que no hay otro campo con el mismo nombre 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 CUENTA sume una unidad al valor de su campo *interes*.

Para modificar los valores de algunos registros de una tabla, hay que usar la sentencia UPDATE SET WHERE. La consulta en SQL que necesitamos sería la siguiente:

```
UPDATE CUENTA INNER JOIN (CLIENTE INNER JOIN [CLI-
   CTA] ON CLIENTE.NIF = [CLI-CTA].NIF) ON
   CUENTA.numCta = [CLI-CTA].numCta

SET CUENTA.interes = [interes]+1

WHERE (((CLIENTE.nombre)="Juan"));
```

Ved también

La sintaxis de la sentencia UPDATE SET WHERE se indica en el apartado “Modificación de filas de una tabla”.

Como ya sabéis, estamos ante una consulta 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 consulta debemos escoger la opción correspondiente (creación de tabla, anexión de datos, actualización, eliminación, etc.).

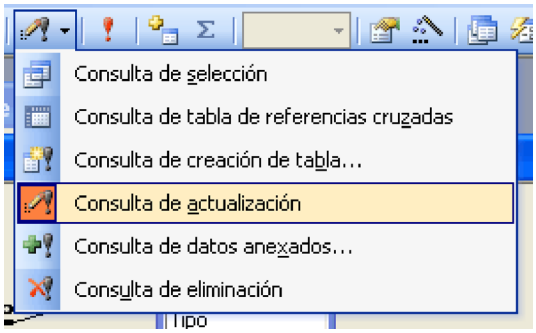

Figura 22. Menú en Access para la selección del tipo de consulta, seleccionando la opción correspondiente a una consulta de actualización.

En este caso, elegimos la opción correspondiente a una consulta de actualización. Cuando llevamos a cabo este cambio, observamos ciertas variaciones sobre los datos que solicita el QBE de Access en función del tipo de consulta de la que se trate.

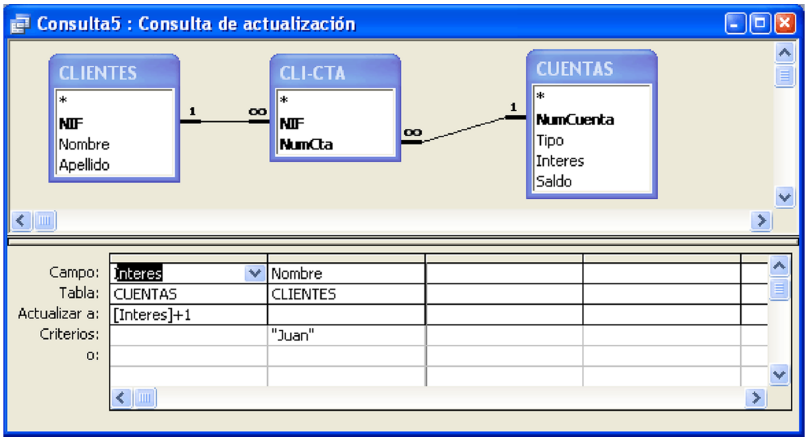

Figura 23. Ventana del editor QBE de Access que representa la consulta 5.

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 CLIENTE.

Para borrar valores de algunas filas de una tabla, disponemos de la sentencia DELETE FROM WHERE.

Ved también

El formato de la sentencia DELETE FROM WHERE se indica en el apartado “Borrado de filas de una tabla”.

La sentencia SQL que permite eliminar todos los datos de Juan es la siguiente:

```
DELETE CLIENTE.nombre
FROM CLIENTE
WHERE (((CLIENTE.nombre)="Juan"));
```

Notemos que es opcional la especificación de los campos a eliminar detrás 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 CLIENTE.NIF
FROM CLIENTE
WHERE (((CLIENTE.nombre)="Juan"));
```

O bien:

```
DELETE CLIENTE.apellido
FROM CLIENTE
WHERE (((CLIENTE.nombre)="Juan"));
```

Veamos qué ha pasado con la tabla CLI-CTA. Observemos que se han borrado también las ocurrencias que hubiera 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 CLIENTE y CLI-CTA tras la ejecución de la sentencia SQL son, respectivamente, los siguientes:

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 CLIENTE 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, puesto que tras su ejecución no tenemos posibilidad de recuperar la situación inmediatamente anterior. 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 relativo a la exigencia de integridad referencial, ya que esto implicará, como hemos visto, la actualización o el borrado de registros adicionales a los de la tabla sobre la que la estamos aplicando directamente. Este punto es especialmente importante si tenemos en cuenta que Access no advertierte de estas implicaciones, es decir, no avisa de los borrados o cambios que va a realizar sobre otras tablas de la base de datos.

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

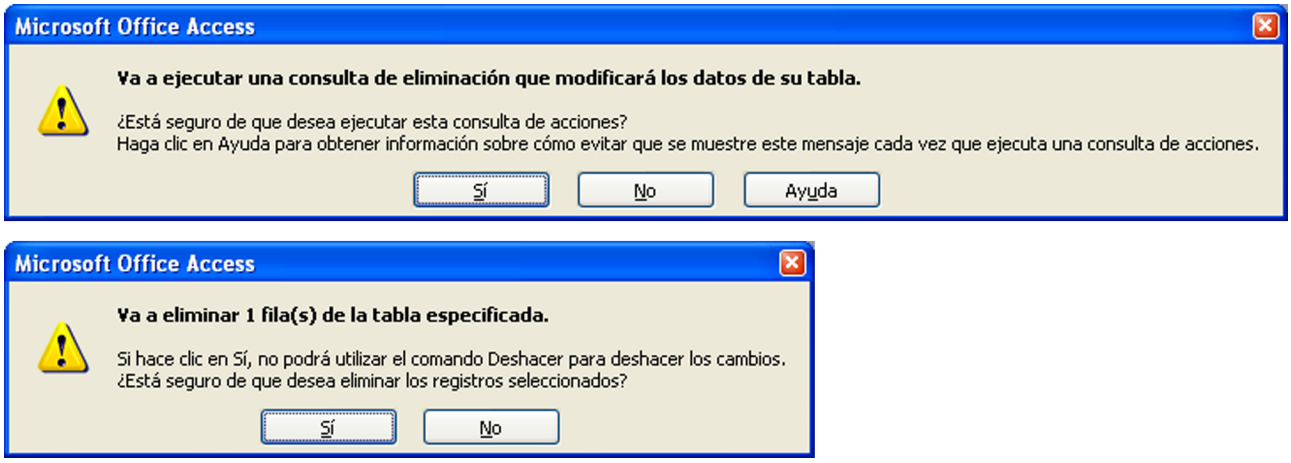

Figura 26. Cuadros de diálogo con mensajes de aviso de actualización y/o borrado de tablas de Access.

Como veis, Access sólo advierte del borrado de una fila (registro) de una tabla, cuando en realidad estaremos eliminando un registro de la tabla CLIENTE y dos de la tabla CLI-CTA (estos últimos por el hecho de haber exigido integridad referencial en la asociación entre CLIENTE y CLI-CTA).

2.7. Consulta 7: duplicar la tabla CLIENTE sobre una nueva llamada CLIENTE2

Si no hay una tabla con el nombre de la tabla destino de la consulta, Access la crea, lo que permite usar 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 evita tener que especificarlos.

SELECT-INTO

La instrucción SELECT con la cláusula INTO permite combinar datos de una o más tablas en una tabla nueva. Se suele utilizar para hacer una copia de seguridad de los datos de una tabla.

Por lo tanto, la sentencia que necesitamos es la siguiente:

```
SELECT CLIENTE * INTO CLIENTE2
FROM CLIENTE;
```

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

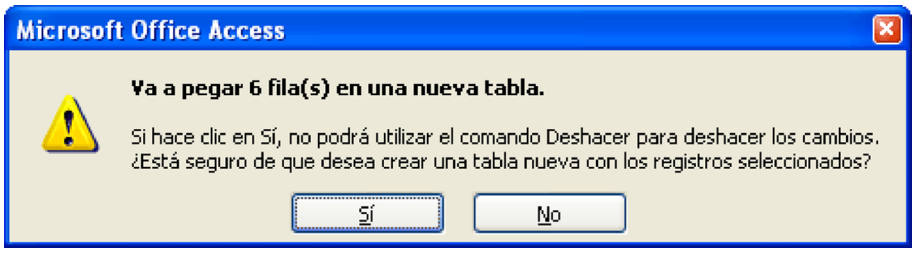

Figura 27. Cuadros de diálogo con mensaje de creación de una nueva tabla con incorporación de datos en Access.

Utilizando la interfaz QBE de Access deberíamos seguir los pasos siguientes:

1) Elegimos la opción de creación de nueva consulta en vista Diseño y añadimos la tabla CLIENTE del cuadro de diálogo "Mostrar tabla". Cerramos el cuadro de diálogo.

2) De los tipos de consulta, seleccionamos la consulta de creación de tabla (botón Crear tabla).

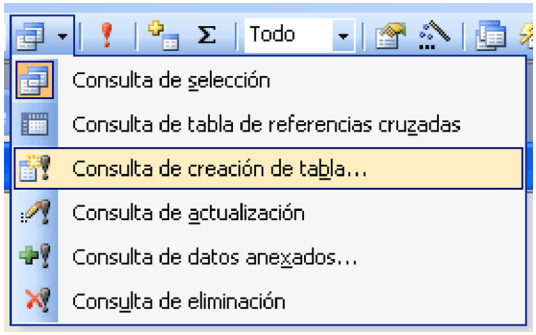

Figura 28. Menú para la selección de una consulta de creación de tabla en Access.

3) Especificamos el nombre de la tabla nueva que queremos crear. En este punto hay que tener en cuenta que Access presenta la posibilidad de incorporar los registros sobre tablas ya existentes que incluso pueden ser de otras bases de datos.

Actividad

Copiar tablas sobre otras bases de datos

Como variante de esta consulta 7, podéis intentar incorporar registros sobre tablas de otras bases de datos.

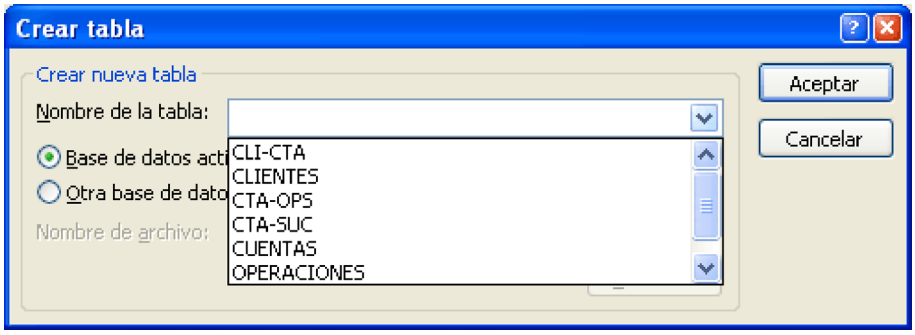

Figura 29. Cuadro de diálogo de creación de tabla con la selección de su nombre.

4) A continuación indicamos en la parrilla (*grid*) los campos de los que se nutrirá la información de la nueva tabla CLIENTE2. Por este motivo, indicaremos que son todos los campos de la tabla CLIENTE, lo que podemos hacer, como ya sabemos, mediante el símbolo asterisco (*).

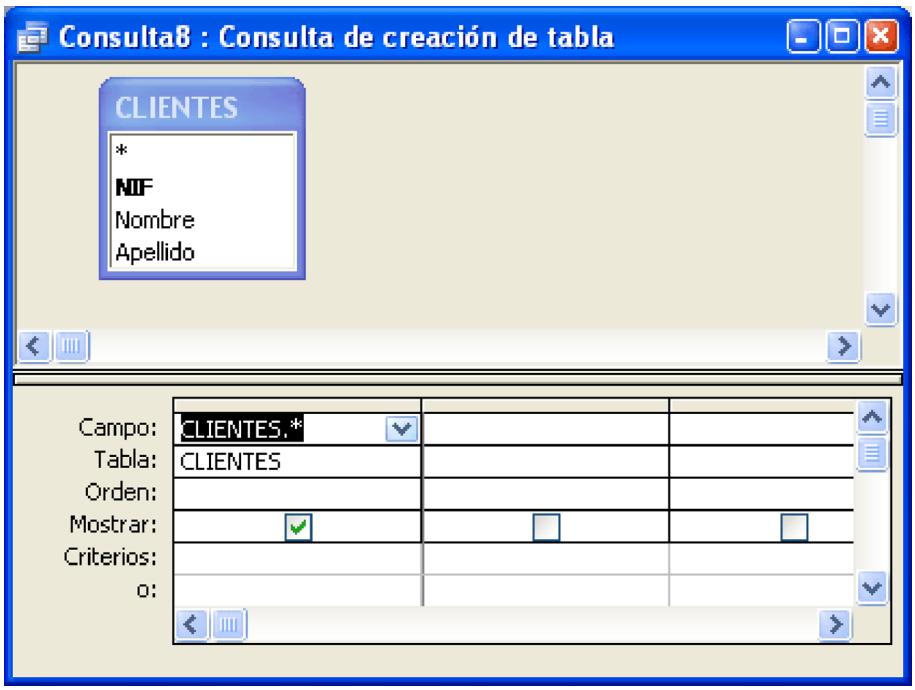

Figura 30. Ventana del editor QBE de Access que representa la consulta 7.

2.8. Consulta 8: incorporar un nuevo cliente a la tabla CLIENTE2

También es posible que queramos incorporar nuevos registros en las tablas. En este caso, incorporemos a la tabla CLIENTE2 (que hemos creado en la consulta anterior) un nuevo cliente llamado Mario García y con *NIF* 77777777-G.

Para introducir nuevas filas en una tabla con SQL, utilizaríamos la sentencia INSERT INTO VALUES de la manera siguiente:

```
INSERT INTO CLIENTE2
VALUES ("77777777-G","Mario","García");
```

Ved también

El formato de la sentencia INSERT INTO con la cláusula VALUES se especifica en el apartado "Inserción de filas en una tabla".

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

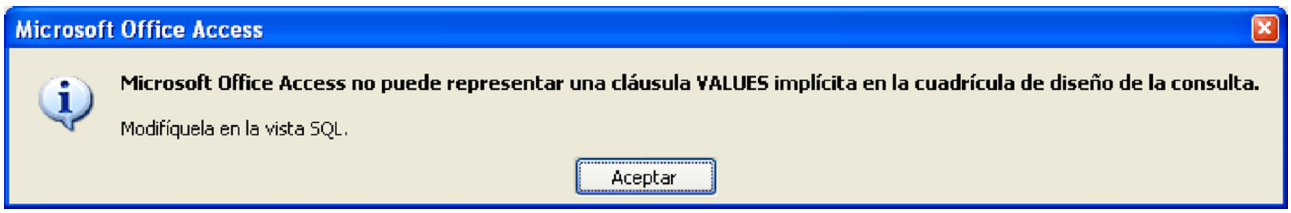

Figura 31. Cuadro de diálogo de aviso de error por haber utilizado VALUES en el editor QBE de Access.

Bibliografía

Beaulieu, A. (2010). *Aprende SQL* (2.ª ed.). Madrid: Anaya Multimedia.

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

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

Date, C. J.; Darwen, H. (1997). *A guide to the SQL Standard* (4.ª ed.). Reading, Massachussets: Addison-Wesley.

Melton, J.; Simon, A. R. (2002). *SQL: 1999. Understanding Relational Language Components*. San Francisco: Morgan Kaufmann.

Silberschatz, A.; Korth, H. F.; Sudarshan, S. (2006). *Fundamentos de bases de datos* (5.ª ed.). Madrid [etc.]: McGraw-Hill.

Database Language SQL. Document ANSI/X3.135. American National Standards Institute (ANSI).

Database Language SQL. Document ISO/IEC 9075. International Organization for Standardization (ISO).