CAPÍTULO

3 EL MODELO RELACIONAL

El modelo relacional se ha establecido actualmente como el principal modelo de datos para las aplicaciones de procesamiento de datos. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos anteriores como el de red y el jerárquico.

En este capítulo se estudia en primer lugar los fundamentos del modelo relacional, que proporciona una forma muy simple y potente de representar datos. A continuación se describen tres lenguajes formales de consulta; los lenguajes de consulta se usan para especificar las solicitudes de información. Los tres que se estudian en este capítulo no son cómodos de usar, pero a cambio sirven como base formal para lenguajes de consulta que sí lo son y que se estudiarán más adelante. El primer lenguaje de consulta, el álgebra relacional, se estudia en detalle. El álgebra relacional forma la base del lenguaje de consulta SQL ampliamente usado. A continuación se proporcionan visiones generales de otros dos lenguajes formales: el cálculo relacional de tuplas y el cálculo relacional de dominios, que son lenguajes declarativos de consulta basados en la lógica matemática. El cálculo relacional de dominios es la base del lenguaje QBE.

Existe una amplia base teórica de las bases de datos relacionales. En este capítulo se estudia la base teórica referida a las consultas. En el Capítulo 7 se examinarán aspectos de la teoría de bases de datos relacionales que ayudan en el diseño de esquemas de bases de datos relacionales, mientras que en los Capítulos 13 y 14 se estudian aspectos de la teoría que se refieren al procesamiento eficiente de consultas.

3.1. LA ESTRUCTURA DE LAS BASES DE DATOS RELACIONALES

Una base de datos relacional consiste en un conjunto de **tablas**, a cada una de las cuales se le asigna un nombre exclusivo. Cada tabla tiene una estructura parecida a la presentada en el Capítulo 2, donde se representaron las bases de datos E-R mediante tablas. Cada fila de la tabla representa una *relación* entre un conjunto de valores. Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de *tabla* y el concepto matemático de *relación*, del que toma su nombre el modelo de datos relacional. A continuación se introduce el concepto de relación.

En este capítulo se utilizarán varias relaciones diferentes para ilustrar los conceptos subyacentes al modelo de datos relacional. Estas relaciones representan parte de una entidad bancaria. Se diferencian ligeramente de las tablas que se utilizaron en el Capítulo 2, por lo que se puede simplificar la representación. En el Capítulo 7 se estudiarán los criterios sobre la adecuación de las estructuras relacionales.

3.1.1. Estructura básica

Considérese la tabla *cuenta* de la Figura 3.1. Tiene tres cabeceras de columna: *número-cuenta*, *nombre-sucursal* y *saldo*. Siguiendo la terminología del modelo relacional se puede hacer referencia a estas cabeceras como **atributos** (igual que se hizo en el modelo E-R en el Capítulo 2). Para cada atributo hay un conjunto de valores permitidos, llamado **dominio** de ese atributo. Para el atributo *nombre-sucursal*, por ejemplo, el dominio es el conjunto de los nombres de las sucursales. Supongamos que D_1 denota el conjunto de todos los números de cuenta, D_2 el conjunto de todos los nombres de sucursal y D_3 el conjunto de los saldos. Como se vio en el Capítulo 2 todas las filas de *cuenta* deben consistir en una tupla (v_1, v_2, v_3), donde v_1 es un número de cuenta (es decir, v_1 está en el dominio D_1), v_2 es un nombre de sucursal (es decir, v_2 está en el dominio D_2) y v_3 es un saldo (es decir, v_3 está en el dominio D_3). En general,

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

FIGURA 3.1. La relación *cuenta*.

cuenta sólo contendrá un subconjunto del conjunto de todas las filas posibles. Por tanto, *cuenta* es un subconjunto de

$$D_1 \times D_2 \times D_3$$

En general, una **tabla** de *n* atributos debe ser un subconjunto de

$$D_1 \times D_2 \times \ldots \times D_{n-1} \times D_n$$

Los matemáticos definen las **relaciones** como subconjuntos del producto cartesiano de la lista de dominios. Esta definición se corresponde de manera casi exacta con la definición de tabla dada anteriormente. La única diferencia es que aquí se han asignado nombres a los atributos, mientras que los matemáticos sólo utilizan «nombres» numéricos, utilizando el entero 1 para denotar el atributo cuyo dominio aparece en primer lugar en la lista de dominios, 2 para el atributo cuyo dominio aparece en segundo lugar, etcétera. Como las tablas son esencialmente relaciones, se utilizarán los términos matemáticos **relación** y **tupla** en lugar de los términos **tabla** y **fila**. Una **variable tupla** es una variable que representa a una tupla; en otras palabras, una tupla que representa al conjunto de todas las tuplas.

En la relación *cuenta* de la Figura 3.1 hay siete tuplas. Supóngase que la variable tupla *t* hace referencia a la primera tupla de la relación. Se utiliza la notación *t*[*número-cuenta*] para denotar el valor de *t* en el atributo *número-cuenta*. Por tanto, *t*[*número-cuenta*] = «C-101» y *t*[*nombre-sucursal*] = «Centro». De manera alternativa, se puede escribir *t*[1] para denotar el valor de la tupla *t* en el primer atributo (*número-cuenta*), *t*[2] para denotar *nombre-sucursal*, etcétera. Dado que las relaciones son conjuntos se utiliza la notación matemática $t \in r$ para denotar que la tupla *t* está en la relación *r*.

El orden en que aparecen las tuplas es irrelevante, dado que una relación es un *conjunto* de tuplas. Así, si las tuplas de una relación se muestran ordenadas como en la Figura 3.1, o desordenadas, como en la Figura 3.2, no importa; las relaciones de estas figuras son las mismas, ya que ambas contienen el mismo conjunto de tuplas.

Se exigirá que, para todas las relaciones *r*, los dominios de todos los atributos de *r* sean atómicos. Un dominio es **atómico** si los elementos del dominio se consideran unidades indivisibles. Por ejemplo, el conjunto de los enteros es un dominio atómico, pero el conjunto de todos los conjuntos de enteros es un dominio no atómico. La diferencia es que no se suele considerar que los enteros tengan subpartes, pero sí se considera que los conjuntos de enteros las tienen; por ejemplo, los enteros que forman cada conjunto. Lo importante no es lo que sea el propio dominio, sino la manera en que se utilizan los elementos del dominio en la base de datos. El dominio de todos los enteros sería no atómico si se considerase que cada entero fuera una lista ordenada de cifras. En todos los ejemplos se supondrá que los dominios son atómicos. En el Capítulo 9 se estudiarán extensiones al modelo de datos relacional para permitir dominios no atómicos.

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-215	Becerril	700
C-102	Navacerrada	400
C-305	Collado Mediano	350
C-201	Galapagar	900
C-222	Moralzarzal	700
C-217	Galapagar	750

FIGURA 3.2. La relación *cuenta* con las tuplas desordenadas.

Es posible que varios atributos tengan el mismo dominio. Por ejemplo, supóngase que se tiene una relación *cliente* que tiene los tres atributos *nombre-cliente*, *calle-cliente* y *ciudad-cliente* y una relación *empleado* que incluye el atributo *nombre-empleado*. Es posible que los atributos *nombre-cliente* y *nombre-empleado* tengan el mismo dominio, el conjunto de todos los nombres de personas, que en el nivel físico son cadenas de caracteres. Los dominios de *saldo* y *nombre-sucursal*, por otra parte, deberían ser distintos. Quizás es menos claro si *nombre-cliente* y *nombre-sucursal* deberían tener el mismo dominio. En el nivel físico, tanto los nombres de clientes como los nombres de sucursales son cadenas de caracteres. Sin embargo, en el nivel lógico puede que se desee que *nombre-cliente* y *nombre-sucursal* tengan dominios diferentes.

Un valor de dominio que es miembro de todos los dominios posibles es el valor **nulo**, que indica que el valor es desconocido o no existe. Por ejemplo, supóngase que se incluye el atributo *número-teléfono* en la relación *cliente*. Puede ocurrir que un cliente no tenga número de teléfono, o que su número de teléfono no figure en la guía. Entonces habrá que recurrir a los valores nulos para indicar que el valor es desconocido o que no existe. Más adelante se verá que los valores nulos crean algunas dificultades cuando se tiene acceso a la base de datos o cuando se actualiza y que, por tanto, deben eliminarse si es posible. Se asumirá inicialmente que no hay valores nulos y en el Apartado 3.3.4 se describirá el efecto de los valores nulos en las diferentes operaciones.

3.1.2. Esquema de la base de datos

Cuando se habla de bases de datos se debe diferenciar entre el **esquema de la base de datos**, o diseño lógico de la misma, y el **ejemplar de la base de datos**, que es una instantánea de los datos de la misma en un momento dado.

El concepto de relación se corresponde con el concepto de variable de los lenguajes de programación. El concepto de **esquema de la relación** se corresponde con el concepto de definición de tipos de los lenguajes de programación.

Resulta conveniente dar un nombre a los esquemas de las relaciones, igual que se dan nombres a las definiciones de tipos en los lenguajes de programación. Se adopta el convenio de utilizar nombres en minúsculas para las relaciones y nombres que comiencen por una letra mayúscula para los esquemas de las relaciones. Siguiendo esta notación se utilizará *Esquema-cuenta* para denotar el esquema de la relación de la relación *cuenta*. Por tanto,

Esquema-cuenta = (*número-cuenta, nombre-sucursal, saldo*)

Se denota el hecho de que *cuenta* es una relación de *Esquema-cuenta* mediante

cuenta (*Esquema-cuenta*)

En general, los esquemas de las relaciones incluyen una lista de los atributos y de sus dominios correspondientes. La definición exacta del dominio de cada atributo no será relevante hasta que se discuta el lenguaje SQL en el Capítulo 4.

El concepto de **ejemplar de relación** se corresponde con el concepto de valor de una variable en los lenguajes de programación. El valor de una variable dada puede cambiar con el tiempo; de manera parecida, el contenido del ejemplar de una relación puede cambiar con el tiempo cuando la relación se actualiza. Sin embargo, se suele decir simplemente «relación» cuando realmente se quiere decir «ejemplar de la relación».

Como ejemplo de ejemplar de una relación, considérese la relación *sucursal* de la Figura 3.3. El esquema de esa relación es

Esquema-relación = (*nombre-sucursal, ciudad-sucursal, activos*)

Obsérvese que el atributo *nombre de la sucursal* aparece tanto en *Esquema-sucursal* como en *Esquema-cuenta*. Esta duplicidad no es una coincidencia. Más bien, utilizar atributos comunes en los esquemas de las relaciones es una manera de relacionar las tuplas de relaciones diferentes. Por ejemplo, supóngase que se desea obtener información sobre todas las cuentas abiertas en sucursales ubicadas en Arganzuela. Primero se busca en la relación *sucursal* para encontrar los nombres de todas las sucursales sitas en Arganzuela. Luego, para cada una de ellas, se mira en la relación *cuenta* para encontrar la información sobre las cuentas abiertas en esa sucursal. Esto no es sorprendente: recuérdese que los atributos que forma la clave primaria de un conjunto de entidades fuertes aparecen en la tabla creada para representar el conjunto de entidades, así como en las tablas creadas para crear relaciones en las que participar el conjunto de entidades.

Continuemos con el ejemplo bancario. Se necesita una relación que describa la información sobre los clientes. El esquema de la relación es:

Esquema-cliente = (*nombre-cliente, calle-cliente, ciudad-cliente*)

En la Figura 3.4 se muestra un ejemplo de la relación *cliente* (*Esquema-cliente*). Obsérvese que se ha omitido el atributo *id-cliente*, que se usó en el Capítulo 2, porque no se desea tener esquemas de relación más pequeños en este ejemplo. Se asume que el nombre de cliente identifica unívocamente un cliente; obviamente, esto no es cierto en el mundo real, pero las suposiciones hechas en estos ejemplos los hacen más sencillos de entender.

En una base de datos del mundo real, *id-cliente* (que podría ser el número de la seguridad social o un identificador generado por el banco) serviría para identificar unívocamente a los clientes.

También se necesita una relación que describa la asociación entre los clientes y las cuentas. El esquema de la relación que describe esta asociación es:

Esquema-impositor = (*nombre-cliente, número-cuenta*)

En la Figura 3.5 se muestra un ejemplo de la relación *impositor* (*Esquema-impositor*).

Puede parecer que, para el presente ejemplo bancario, se podría tener sólo un esquema de relación, en vez de tener varios. Es decir, puede resultar más sencillo para el usuario pensar en términos de un esquema de

nombre de la sucursal	*ciudad de la sucursal*	*activos*
Galapagar	Arganzuela	7.500
Centro	Arganzuela	9.000.000
Becerril	Aluche	2.000
Segovia	Cerceda	3.700.000
Navacerrada	Aluche	1.700.000
Navas de la Asunción	Alcalá de Henares	1.500
Moralzarzal	La Granja	2.500
Collado Mediano	Aluche	8.000.000

FIGURA 3.3. La relación *sucursal*.

nombre-cliente	*calle-cliente*	*ciudad-cliente*
Abril	Preciados	Valsaín
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín
Fernández	Jazmín	León
Gómez	Carretas	Cerceda
González	Arenal	La Granja
López	Mayor	Peguerinos
Pérez	Carretas	Cerceda
Rodríguez	Yeserías	Cádiz
Rupérez	Ramblas	León
Santos	Mayor	Peguerinos
Valdivieso	Goya	Vigo

FIGURA 3.4. La relación *cliente*.

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

FIGURA 3.5. La relación *impositor*.

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

FIGURA 3.6. La relación *préstamo*.

relación, en lugar de en términos de varios esquemas. Supóngase que sólo se utilizara una relación para el ejemplo, con el esquema

(*nombre-sucursal, ciudad-sucursal, activos, nombre-cliente, calle-cliente, ciudad-cliente, número-cuenta, saldo*)

Obsérvese que si un cliente tiene varias cuentas hay que repetir su dirección una vez por cada cuenta. Es decir, hay que repetir varias veces parte de la información. Esta repetición supone un gasto inútil y se evita mediante el uso de varias relaciones, como en el ejemplo presente.

Además, si una sucursal no tiene ninguna cuenta (por ejemplo, una sucursal recién creada que todavía no tiene clientes), no se puede construir una tupla completa en la relación única anterior, dado que no hay todavía ningún dato disponible referente a *cliente* ni a *cuenta*. Para representar las tuplas incompletas hay que utilizar valores *nulos* que indiquen que ese valor es desconocido o no existe. Por tanto, en el ejemplo presente, los valores de *nombre-cliente*, *calle-cliente*, etcétera, deben quedar nulos. Utilizando varias relaciones se puede representar la información de las sucursales de un banco sin clientes sin utilizar valores nulos. Sencillamente, se utiliza una tupla en *Esquema-sucursal* para representar la información de la sucursal y sólo crear tuplas en los otros esquemas cuando esté disponible la información adecuada.

En el Capítulo 7 se estudiarán los criterios para decidir cuándo un conjunto de esquemas de relaciones es más apropiado que otro en términos de repetición de la información y de la existencia de valores nulos. Por ahora se supondrá que los esquemas de las relaciones vienen dados de antemano.

Se incluyen dos relaciones más para describir los datos de los préstamos concedidos en las diferentes sucursales del banco:

Esquema-préstamo = (*número-préstamo, nombre-sucursal, importe*)
Esquema-prestatario = (*nombre-cliente, número-préstamo*)

Las relaciones de ejemplo *préstamo* (*Esquema-préstamo*) y *prestatario* (*Esquema-prestatario*) se muestran en las Figuras 3.5 y 3.6, respectivamente.

La entidad bancaria que se ha descrito se deriva del diagrama E-R mostrado en la Figura 3.8. Los esquemas de las relaciones se corresponden con el conjunto de tablas que se podrían generar utilizando el método esbozado en el Apartado 2.9. Obsérvese que las tablas para *cuenta-sucursal* y *préstamo-sucursal* se han combinado en las tablas de *cuenta* y *préstamo* respectivamente. Esta combinación es posible dado que las relaciones son de varios a uno desde *cuenta* y *préstamo*, respectivamente, a *sucursal* y, además, la participación de *cuenta* y *préstamo* en las relaciones correspondientes es total, como indican las líneas dobles en la figura. Finalmente, obsérvese que la relación *cliente* puede contener información sobre clientes que ni tengan cuenta ni un préstamo en el banco.

La entidad bancaria aquí descrita servirá como ejemplo principal en este capítulo y en los siguientes. Cuando sea necesario, habrá que introducir más esquemas de relaciones para ilustrar casos concretos.

3.1.3. Claves

Los conceptos de *superclave*, de *clave candidata* y de *clave primaria*, tal y como se discute en el Capítulo 2, también son aplicables en el modelo relacional. Por ejemplo, en *Esquema-sucursal*, tanto {*nombre-sucursal*} como {*nombre-sucursal*, *ciudad-sucursal*} son superclaves. {*nombre-sucursal*, *ciudad-sucursal*} no es una clave candidata porque {*nombre-sucursal*} es un subconjunto de {*nombre-sucursal*, *ciudad-sucursal*} y {*nombre-sucursal*} es una superclave. Sin embargo, {*nombre-sucursal*} es una clave candidata, y servirá también como clave primaria para estos fines. El atributo *ciudad-sucursal* no es una superclave, dado que dos sucursales de la misma ciudad pueden tener nombres diferentes (y diferentes volúmenes de activos).

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

FIGURA 3.7. La relación *prestatario*.

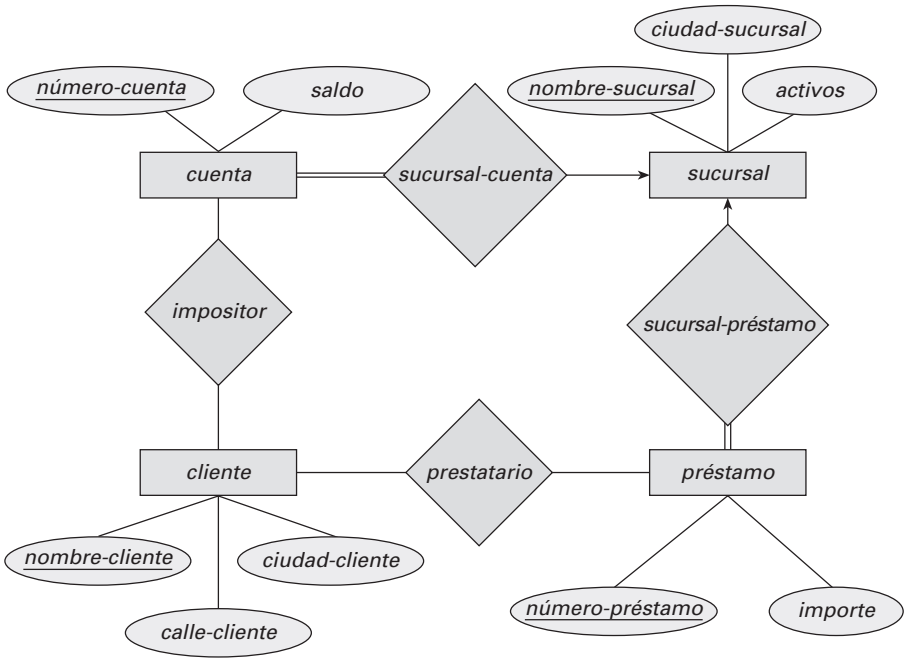

FIGURA 3.8. Diagrama E-R de la entidad bancaria.

Sea R el esquema de una relación. Si se dice que un subconjunto K de R es una *superclave* de R para las relaciones $r(R)$ en las que no hay dos tuplas diferentes que tengan los mismos valores en todos los atributos de K. Es decir, si t_1 y t_2 están en R y $t_1 \neq t_2$, entonces $t_1[K] \neq t_2[K]$.

Si el esquema de una base de datos relacional se basa en las tablas derivadas de un esquema E-R es posible determinar la clave primaria del esquema de una relación a partir de las claves primarias de los conjuntos de entidades o de relaciones de los que se deriva el esquema:

- **Conjunto de entidades fuertes.** La clave primaria del conjunto de entidades se convierte en la clave primaria de la relación.
- **Conjunto de entidades débiles.** La tabla y, por tanto, la relación correspondientes a un conjunto de entidades débiles incluyen
  - — Los atributos del conjunto de entidades débiles.
  - — La clave primaria del conjunto de entidades fuertes del que depende el conjunto de entidades débiles.

  La clave primaria de la relación consiste en la unión de la clave primaria del conjunto de entidades fuertes y el discriminante del conjunto de entidades débil.
- **Conjunto de relaciones.** La unión de las claves primarias de los conjuntos de entidades relacionados se transforma en una superclave de la relación. Si la relación es de varios a varios, esta superclave es también la clave primaria. En el Apartado 2.4.2 se describe la manera de determinar las claves primarias en otros casos. Recuérdese del Apartado 2.9.3 que no se genera ninguna tabla para los conjuntos de relaciones que vinculan un conjunto de entidades débiles con el conjunto de entidades fuertes correspondiente.
- **Tablas combinadas.** Recuérdese del Apartado 2.9.3 que un conjunto binario de relaciones de varios a uno entre A y B puede representarse mediante una tabla que consista en los atributos de A y en los atributos (si hay alguno) del conjunto de relaciones. La clave primaria de la entidad «varios» se transforma en la clave primaria de la relación (es decir, si el conjunto de relaciones es de varios a uno entre A y B, la clave primaria de A es la clave primaria de la relación). Para los conjuntos de relaciones de uno a uno la relación se construye igual que en el conjunto de relaciones de varios a uno. Sin embargo, cualquiera de las claves primarias del conjunto de entidades puede elegirse como clave primaria de la relación, dado que ambas son claves candidatas.
- **Atributos multivalorados.** Recuérdese del Apartado 2.9.4 que un atributo multivalorado M se representa mediante una tabla consistente en la clave primaria del conjunto de entidades o de relaciones del que M es atributo y en una columna C que guarda un valor concreto de M. La clave primaria del conjunto de entidades o de relaciones

junto con el atributo C se convierte en la clave primaria de la relación.

A partir de la lista precedente se puede ver que el esquema de una relación puede incluir entre sus atributos la clave primaria de otro esquema, digamos r_2. Este atributo es una **clave externa** de r_1 que hace referencia a r_2. La relación r_1 también se denomina la **relación referenciante** de la dependencia de clave externa, y r_2 se denomina la **relación referenciada** de la clave externa. Por ejemplo, el atributo *nombre-sucursal* de *Esquema-cuenta* es una clave externa de *Esquema-cuenta* referenciando a *Esquema-sucursal*, ya que *nombre-sucursal* es la clave primaria de *Esquema-sucursal*. En cualquier ejemplar de la base de datos, dada una tupla t_a de la relación *cuenta*, debe haber alguna tupla t_b en la relación *cuenta* tal que el valor del atributo *nombre-sucursal* de t_a sea el mismo que el valor de la clave primaria, *nombre-sucursal*, de t_b.

Es obligado listar los atributos que forman clave primaria de un esquema de relación antes que el resto; por ejemplo, el atributo *nombre-sucursal* de *Esquema-sucursal* se lista en primer lugar, ya que es la clave primaria.

3.1.4. Diagramas de esquema

Un esquema de bases de datos, junto con las dependencias de clave primaria y externa, se puede mostrar gráficamente mediante **diagramas de esquema**. La Figura 3.9 muestra el diagrama de esquema del ejemplo bancario. Cada relación aparece como un cuadro con los atributos listados dentro de él y el nombre de la relación sobre él. Si hay atributos clave primaria, una línea horizontal cruza el cuadro con los atributos clave primaria listados sobre ella. Las dependencias de clave externa aparecen como flechas desde los atributos clave externa de la relación referenciante a la clave primaria de la relación referenciada.

No hay que confundir un diagrama de esquema con un diagrama E-R. En particular, los diagramas E-R no muestran explícitamente los atributos clave externa, mientras que los diagramas de esquema sí.

Muchos sistemas de bases de datos proporcionan herramientas de diseño con una interfaz gráfica de usuario para la creación de diagramas de esquema.

3.1.5. Lenguajes de consulta

Un **lenguaje de consulta** es un lenguaje en el que un usuario solicita información de la base de datos. Estos lenguajes suelen ser de un nivel superior que el de los lenguajes de programación habituales. Los lenguajes de consulta pueden clasificarse como procedimentales o no procedimentales. En los **lenguajes procedimentales** el usuario instruye al sistema para que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado. En los **lenguajes no procedimentales** el usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.

La mayor parte de los sistemas comerciales de bases de datos relacionales ofrecen un lenguaje de consulta que incluye elementos de los enfoques procedimental y no procedimental. Se estudiarán varios lenguajes comerciales en el Capítulo 4. El Capítulo 5 trata los lenguajes QBE y Datalog, este último parecido a Prolog.

En este capítulo se examinarán los lenguajes «puros»: el álgebra relacional es procedimental, mientras que el cálculo relacional de tuplas y el de dominios son no procedimentales. Estos lenguajes de consulta son rígidos y formales, y carecen del «azúcar sintáctico» de los lenguajes comerciales, pero ilustran las técnicas fundamentales para la extracción de datos de las bases de datos.

Aunque inicialmente sólo se estudiarán las consultas, un lenguaje de manipulación de datos completo no sólo incluye un lenguaje de consulta, sino también un lenguaje para la modificación de las bases de datos. Estos lenguajes incluyen órdenes para insertar y borrar tuplas, así como órdenes para modificar partes de las tuplas existentes. Las modificaciones de las bases de datos se examinarán después de completar la discusión sobre las consultas.

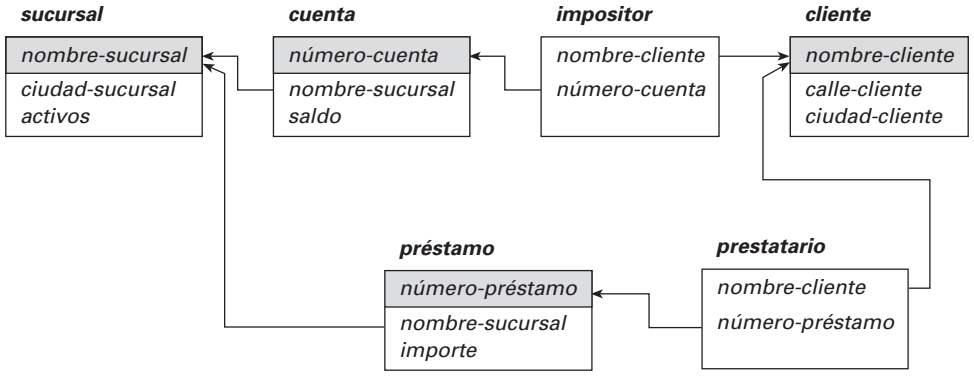

FIGURA 3.9. Diagrama de esquema para el banco.

3.2. EL ÁLGEBRA RELACIONAL

El álgebra relacional es un lenguaje de consulta *procedimental*. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación. Las operaciones fundamentales del álgebra relacional son *selección*, *proyección*, *unión*, *diferencia de conjuntos*, *producto cartesiano* y *renombramiento*. Además de las operaciones fundamentales hay otras operaciones, por ejemplo, intersección de conjuntos, reunión natural, división y asignación. Estas operaciones se definirán en términos de las operaciones fundamentales.

3.2.1. Operaciones fundamentales

Las operaciones selección, proyección y renombramiento se denominan operaciones *unarias* porque operan sobre una sola relación. Las otras tres operaciones operan sobre pares de relaciones y se denominan, por lo tanto, operaciones *binarias*.

3.2.1.1. La operación selección

La operación **selección** selecciona tuplas que satisfacen un predicado dado. Se utiliza la letra griega sigma minúscula (σ) para denotar la selección. El predicado aparece como subíndice de σ. La relación del argumento se da entre paréntesis a continuación de σ. Por tanto, para seleccionar las tuplas de la relación *préstamo* en que la sucursal es «Navacerrada» hay que escribir

$$\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»}} (préstamo)$$

Si la relación *préstamo* es como se muestra en la Figura 3.6, la relación que resulta de la consulta anterior es como se muestra en la Figura 3.10.

Se pueden buscar todas las tuplas en las que el importe prestado sea mayor que 1.200 € escribiendo

$$\sigma_{importe>1200} (préstamo)$$

En general, se permiten las comparaciones que utilizan $=, \neq, <, \leq, >$ o $\geq$ en el predicado de selección. Además, se pueden combinar varios predicados en uno mayor utilizando las conectivas *y* ($\wedge$) y *o* ($\vee$). Por tanto, para encontrar las tuplas correspondientes a préstamos de más de 1.200 € concedidos por la sucursal de Navacerrada, se escribe

$$\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»} \wedge importe>1200} (préstamo)$$

número-préstamo	*nombre-sucursal*	*importe*
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300

FIGURA 3.10. Resultado de $\sigma_{\text{nombre-sucursal = «Navacerrada»}}$ (préstamo).

El predicado de selección puede incluir comparaciones entre dos atributos. Para ilustrarlo, considérese la relación *responsable-préstamo*, que consta de tres atributos: *nombre-cliente*, *nombre-banquero* y *número-préstamo*, que especifica que un empleado concreto es el responsable del préstamo concedido a un cliente. Para hallar todos los clientes que se llaman igual que su responsable de préstamos se puede escribir

$$\sigma_{nombre\text{-}cliente = nombre\text{-}banquero} (responsable\text{-}préstamo)$$

Dado que el valor especial *nulo* indica «valor desconocido o inexistente», cualquier comparación que implique a un valor nulo se evalúa como **falsa**.

3.2.1.2. La operación proyección

Supóngase que se desea hacer una lista de todos los números de préstamo y del importe de los mismos, pero sin que aparezcan los nombres de las sucursales. La operación **proyección** permite producir esta relación. La operación proyección es una operación unaria que devuelve su relación de argumentos, excluyendo algunos argumentos. Dado que las relaciones son conjuntos, se eliminan todas las filas duplicadas. La proyección se denota por la letra griega mayúscula pi (Π). Se crea una lista de los atributos que se desea que aparezcan en el resultado como subíndice de Π. La relación de argumentos se escribe a continuación entre paréntesis. Por tanto, la consulta para crear una lista de todos los números de préstamo y del importe de los mismos puede escribirse como

$$\Pi_{número\text{-}préstamo,\ importe} (préstamo)$$

La relación que resulta de esta consulta se muestra en la Figura 3.11.

3.2.1.3. Composición de operaciones relacionales

Es importante el hecho de que el resultado de una operación relacional sea también una relación. Considérese la consulta más compleja «Encontrar los clientes que viven en Peguerinos». Hay que escribir:

número-préstamo	*importe*
P-11	900
P-14	1.500
P-15	1.500
P-16	1.300
P-17	1.000
P-23	2.000
P-93	500

FIGURA 3.11. Números de préstamo y sus importes.

$$\Pi_{nombre\text{-}cliente}\,(\sigma_{ciudad\text{-}cliente\,=\,\text{«Peguerinos»}}\,(cliente))$$

Téngase en cuenta que, en vez de dar en el argumento de la operación proyección el nombre de una relación, se da una expresión que se evalúa como una relación.

En general, dado que el resultado de una operación del álgebra relacional es del mismo tipo (relación) que los datos de entrada, las operaciones del álgebra relacional pueden componerse para formar una **expresión del álgebra relacional**. La composición de operaciones del álgebra relacional para formar expresiones del álgebra relacional es igual que la composición de operaciones aritméticas (como +, –, ∗ y □) para formar expresiones aritméticas. La definición formal de las expresiones de álgebra relacional se estudia en el Apartado 3.2.2.

3.2.1.4. La operación unión

Considérese una consulta para averiguar el nombre de todos los clientes del banco que tienen una cuenta, un préstamo o ambas cosas. Obsérvese que la relación *cliente* no contiene esa información, dado que los clientes no necesitan tener ni cuenta ni préstamo en el banco. Para contestar a esta consulta hace falta la información de la relación *impositor* (Figura 3.5) y la de la relación *prestatario* (Figura 3.7). Se conoce la manera de averiguar los nombres de todos los clientes con préstamos en el banco:

$$\Pi_{nombre\text{-}cliente}\,(prestatario)$$

También se conoce la manera de averiguar el nombre de los clientes con cuenta en el banco:

$$\Pi_{nombre\text{-}cliente}\,(impositor)$$

Para contestar a la consulta hace falta la **unión** de estos dos conjuntos; es decir, hacen falta todos los nombres de clientes que aparecen en alguna de las dos relaciones o en ambas. Estos datos se pueden averiguar mediante la operación binaria unión, denotada, como en la teoría de conjuntos, por ∪. Por tanto, la expresión buscada es

$$\Pi_{nombre\text{-}cliente}\,(prestatario) \cup \Pi_{nombre\text{-}cliente}\,(impositor)$$

La relación resultante de esta consulta aparece en la Figura 3.10. Téngase en cuenta que en el resultado hay diez tuplas, aunque hay siete prestatarios y seis impositores distintos. Esta discrepancia aparente se debe a que Gómez, Santos y López son a la vez prestatarios e impositores. Dado que las relaciones son conjuntos, se eliminan los valores duplicados.

Obsérvese que en este ejemplo se toma la unión de dos conjuntos, ambos consistentes en valores de *nombre-cliente*. En general, se debe asegurar que las uniones se realicen entre relaciones *compatibles*. Por ejemplo, no tendría sentido realizar la unión de las relaciones *préstamo* y *prestatario*. La primera es una relación con tres atributos, la segunda sólo tiene dos. Más aún, considérese la unión de un conjunto de nombres de clientes y de un conjunto de ciudades. Una unión así no tendría sentido en la mayor parte de los casos. Por tanto, para que una operación unión $r \cup s$ sea válida hay que exigir que se cumplan dos condiciones:

1. Las relaciones r y s deben ser de la misma aridad. Es decir, deben tener el mismo número de atributos.
2. Los dominios de los atributos i-ésimos de r y de s deben ser iguales para todo i.

Téngase en cuenta que r y s pueden ser, en general, relaciones temporales que sean resultado de expresiones del álgebra relacional.

nombre-cliente
Abril
Fernández
Gómez
González
López
Pérez
Rupérez
Santos
Sotoca
Valdivieso

FIGURA 3.12. Nombres de todos los clientes que tienen un préstamo o una cuenta.

3.2.1.5. La operación diferencia de conjuntos

La operación **diferencia de conjuntos**, denotada por –, permite buscar las tuplas que estén en una relación pero no en la otra. La expresión $r - s$ da como resultado una relación que contiene las tuplas que están en r pero no en s.

Se pueden buscar todos los clientes del banco que tienen abierta una cuenta pero no tienen concedido ningún préstamo escribiendo

$$\Pi_{nombre\text{-}cliente}\,(impositor) - \Pi_{nombre\text{-}cliente}\,(prestatario)$$

La relación resultante de esta consulta aparece en la Figura 3.13.

Como en el caso de la operación unión, hay que asegurarse de que las diferencias de conjuntos se realicen entre relaciones *compatibles*. Por tanto, para que una

nombre-cliente
Abril
González
Rupérez

FIGURA 3.13. Clientes con cuenta abierta pero sin préstamo concedido.

operación diferencia de conjuntos $r - s$ sea válida hay que exigir que las relaciones r y s sean de la misma aridad y que los dominios de los atributos i-ésimos de r y s sean iguales.

3.2.1.6. La operación producto cartesiano

La operación **producto cartesiano**, denotada por un aspa (×), permite combinar información de cualesquiera dos relaciones. El producto cartesiano de las relaciones r_1 y r_2 como $r_1 \times r_2$.

Recuérdese que las relaciones se definen como subconjuntos del producto cartesiano de un conjunto de dominios. A partir de esta definición ya se debe tener una intuición sobre la definición de la operación producto cartesiano. Sin embargo, dado que el mismo nombre de atributo puede aparecer tanto en r_1 como en r_2, hay que crear un esquema de denominaciones para distinguir entre ambos atributos. En este caso se logra adjuntando al atributo el nombre de la relación de la que proviene originalmente. Por ejemplo, el esquema de relación de $r = prestatario \times préstamo$ es

(*prestatario.nombre-cliente, prestatario.número-préstamo, préstamo.nombre-sucursal, préstamo.número-préstamo, préstamo.importe*)

Con este esquema se puede distinguir entre *prestatario.número-préstamo* y *préstamo.número-préstamo*. Para los atributos que sólo aparecen en uno de los dos esquemas se suele omitir el prefijo con el nombre de la relación. Esta simplificación no genera ambigüedad alguna. Por tanto, se puede escribir el esquema de relación de r como

(*nombre-cliente, prestatario.número-préstamo, nombre-sucursal, préstamo.número-préstamo, importe*)

El acuerdo de denominaciones precedente *exige* que las relaciones que sean argumentos de la operación producto cartesiano tengan nombres diferentes. Esta exigencia causa problemas en algunos casos, como cuando se desea calcular el producto cartesiano de una relación consigo misma. Se produce un problema similar si se utiliza el resultado de una expresión del álgebra relacional en un producto cartesiano, dado que hará falta un nombre para la relación para poder hacer referencia a sus atributos. En el Apartado 3.2.1.7 se verá la manera de evitar estos problemas utilizando una operación renombramiento.

Ahora que se conoce el esquema de relación de $r = prestatario \times préstamo$ hay que averiguar las tuplas que aparecerán en r. Como se podía imaginar, se crea una tupla de r a partir de cada par de tuplas posible: una de la relación *prestatario* y otra de la relación *préstamo*. Por tanto, r es una relación de gran tamaño, como se puede ver en la Figura 3.14, donde sólo se ha incluido una parte de las tuplas que forman parte de r.

Supóngase que se tienen n_1 tuplas en *prestatario* y n_2 tuplas en *préstamo*. Por tanto, hay $n_1 * n_2$ maneras de escoger un par de tuplas, una tupla de cada relación; por lo que hay $n_1 * n_2$ tuplas en r. En concreto, obsérvese que para algunas tuplas t de r puede ocurrir que t[*prestatario.número-préstamo*] ≠ t[*préstamo.número-préstamo*].

En general, si se tienen las relaciones r_1 (R_1) y r_2 (R_2), $r_1 \times r_2$ es una relación cuyo esquema es la concatenación de R_1 y de R_2. La relación R contiene todas las tuplas t para las que hay unas tuplas t_1 en r_1 y t_2 en r_2 para las que $t[R_1] = t_1[R_1]$ y $t[R_2] = t_2[R_2]$.

Supóngase que se desea averiguar los nombres de todos los clientes que tienen concedido un préstamo en la sucursal de Navacerrada. Se necesita para ello información de las relaciones *préstamo* y *prestatario*. Si se escribe

$$\sigma_{\text{nombre-sucursal} = \text{«Navacerrada»}} (prestatario \times préstamo)$$

entonces el resultado es la relación mostrada en la Figura 3.15. Se tiene una relación que sólo atañe a la sucursal de Navacerrada. Sin embargo, la columna *nombre-cliente* puede contener clientes que no tengan concedido ningún préstamo en la sucursal de Navacerrada. (Si no se ve el motivo por el que esto es cierto, recuérdese que el producto cartesiano toma todos los emparejamientos posibles de una tupla de *prestatario* con una tupla de *préstamo*.)

Dado que la operación producto cartesiano asocia *todas* las tuplas de *préstamo* con todas las tuplas de *prestatario*, se sabe que, si un cliente tiene concedido un préstamo en la sucursal de Navacerrada, hay alguna tupla de $prestatario \times préstamo$ que contiene su nombre y que *prestatario.número-préstamo = préstamo.número-préstamo*. Por tanto, si escribimos

$$\sigma_{prestatario.número\text{-}préstamo = préstamo.número\text{-}préstamo} (\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»}} (prestatario \times préstamo))$$

sólo se obtienen las tuplas de $prestatario \times préstamo$ que corresponden a los clientes que tienen concedido un préstamo en la sucursal de Navacerrada.

Finalmente, dado que sólo se desea obtener *nombre-cliente*, se realiza una proyección:

$$\Pi_{nombre\text{-}cliente} (\sigma_{prestatario.número\text{-}préstamo = préstamo.número\text{-}préstamo} (\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»}} (prestatario \times préstamo)))$$

El resultado de esta expresión se muestra en la Figura 3.16 y es la respuesta correcta a la consulta formulada.

3.2.1.7. La operación renombramiento

A diferencia de las relaciones de la base de datos, los resultados de las expresiones de álgebra relacional no tienen un nombre que se pueda utilizar para referirse a ellas. Resulta útil poder ponerles nombre; el operador

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-11	Collado Mediano	900
Santos	P-17	P-14	Centro	1.500
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Santos	P-17	P-17	Centro	1.000
Santos	P-17	P-23	Moralzarzal	2.000
Santos	P-17	P-93	Becerril	500
Gómez	P-23	P-11	Collado Mediano	900
Gómez	P-23	P-14	Centro	1.500
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
Gómez	P-23	P-17	Centro	1.000
Gómez	P-23	P-23	Moralzarzal	2.000
Gómez	P-23	P-93	Becerril	500
López	P-15	P-11	Collado Mediano	900
López	P-15	P-14	Centro	1.500
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
López	P-15	P-17	Centro	1.000
López	P-15	P-23	Moralzarzal	2.000
López	P-15	P-93	Becerril	500
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
Valdivieso	P-17	P-11	Collado Mediano	900
Valdivieso	P-17	P-14	Centro	1.500
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Valdivieso	P-17	P-17	Centro	1.000
Valdivieso	P-17	P-23	Moralzarzal	2.000
Valdivieso	P-17	P-93	Becerril	500
Fernández	P-16	P-11	Collado Mediano	900
Fernández	P-16	P-14	Centro	1.500
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300
Fernández	P-16	P-17	Centro	1.000
Fernández	P-16	P-23	Moralzarzal	2.000
Fernández	P-16	P-93	Becerril	500

FIGURA 3.14. Resultado de *prestatario* ⋈ *préstamo*.

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
Sotoca	P-14	P-15	Navacerrada	1.500
Sotoca	P-14	P-16	Navacerrada	1.300
Pérez	P-93	P-15	Navacerrada	1.500
Pérez	P-93	P-16	Navacerrada	1.300
Gómez	P-11	P-15	Navacerrada	1.500
Gómez	P-11	P-16	Navacerrada	1.300
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300

FIGURA 3.15. Resultado de $\sigma_{\text{nombre-sucursal = «Navacerrada»}}$ (*prestatario* ⋈ *préstamo*).

nombre-cliente
Férnandez
López

FIGURA 3.16. Resultado de $\Pi_{nombre\text{-}cliente}\,(\sigma_{prestatario.número\text{-}préstamo\,=\,préstamo.número\text{-}préstamo}\,(\sigma_{nombre\text{-}sucursal\,=\,\text{«Navacerrada»}}\,(prestatario \times préstamo)))$.

renombramiento, denotado por la letra griega rho minúscula (ρ), permite realizar esta tarea. Dada una expresión E del álgebra relacional, la expresión

$$\rho_x\,(E)$$

devuelve el resultado de la expresión E con el nombre x.

Las relaciones r por sí mismas se consideran expresiones (triviales) del álgebra relacional. Por tanto, también se puede aplicar la operación renombramiento a una relación r para obtener la misma relación con un nombre nuevo.

Otra forma de la operación renombramiento es la siguiente. Supóngase que una expresión del álgebra relacional E tiene aridad n. Por tanto, la expresión

$$\rho_{x\,(A_1,\,A_2,\,\ldots\,A_n)}\,(E)$$

devuelve el resultado de la expresión E con el nombre x y con los atributos con el nombre cambiado a A_1, A_2, …, A_n.

Para ilustrar el uso del renombramiento de las relaciones, considérese la consulta «Buscar el máximo saldo de cuenta del banco». La estrategia empleada para obtener el resultado es 1) calcular una relación intermedia consistente en los saldos que *no* son el máximo y 2) realizar la diferencia entre la relación $\Pi_{saldo}\,(cuenta)$ y la relación intermedia recién calculada.

Paso 1: Para calcular la relación intermedia hay que comparar los valores de los saldos de todas las cuentas. Esta comparación se puede hacer calculando el producto cartesiano $cuenta \times cuenta$ y formando una selección para comparar el valor de cualesquiera dos saldos que aparezcan en una tupla. En primer lugar hay que crear un mecanismo para distinguir entre los dos atributos *saldo*. Se utilizará la operación renombramiento para cambiar el nombre de una referencia a la relación cuenta; así, se puede hacer referencia dos veces a la relación sin ambigüedad alguna.

La relación temporal que se compone de los saldos que no son el máximo puede escribirse ahora como

$$\Pi_{cuenta.saldo}\,(\sigma_{cuenta.saldo\,<\,d.saldo}\,(cuenta \times \rho_d\,(cuenta)))$$

Esta expresión proporciona los saldos de la relación *cuenta* para los que aparece un saldo mayor en alguna parte de la relación *cuenta* (cuyo nombre se ha cambiado a *d*). El resultado contiene todos los saldos *salvo* el máximo. Esta relación se muestra en la Figura 3.17.

saldo
500
400
700
750
350

FIGURA 3.17. Resultado de la subexpresión $\Pi_{cuenta\text{-}saldo}\,(\sigma_{cuenta\text{-}saldo\,<\,d.saldo}\,(cuenta \times \rho_d\,(cuenta)))$.

Paso 2: La consulta para averiguar el máximo saldo de cuenta del banco puede escribirse de la manera siguiente:

$$\Pi_{saldo}\,(cuenta) - \Pi_{cuenta.saldo}\,(\sigma_{cuenta.saldo\,<\,d.saldo}\,(cuenta \times \rho_d\,(cuenta)))$$

En la Figura 3.18 se muestra el resultado de esta consulta.

Considérese la siguiente consulta como un nuevo ejemplo de la operación renombramiento: «Averiguar los nombres de todos los clientes que viven en la misma calle y en la misma ciudad que Gómez». Se puede obtener la calle y la ciudad en la que vive Gómez escribiendo

$$\Pi_{calle\text{-}cliente,\,ciudad\text{-}cliente}\,(\sigma_{nombre\text{-}cliente\,=\,\text{«Gómez»}}\,(cliente))$$

Sin embargo, para hallar a otros clientes que vivan en esa calle y en esa ciudad hay que hacer referencia por segunda vez a la relación *cliente*. En la consulta siguiente se utiliza la operación renombramiento sobre la expresión anterior para darle al resultado el nombre *dirección-Gómez* y para cambiar el nombre de los atributos a *calle* y *ciudad* en lugar de *calle-cliente* y *ciudad-cliente*:

$$\Pi_{cliente.nombre\text{-}cliente}\,(\sigma_{cliente.calle\text{-}cliente\,=\,dirección\text{-}Gómez\,\wedge\,cliente.ciudad\text{-}cliente\,=\,dirección\text{-}Gómez.\,ciudad}\,(cliente \times \rho_{dirección\text{-}Gómez\,(calle,\,ciudad)}\,(\Pi_{calle\text{-}cliente,\,ciudad\text{-}cliente}\,(\sigma_{nombre\text{-}cliente\,=\,\text{«Gómez»}}\,(cliente)))))$$

El resultado de esta consulta, cuando se aplica a la relación *cliente* de la Figura 3.4, se muestra en la Figura 3.19.

La operación renombramiento no es estrictamente necesaria, dado que es posible utilizar una notación posicional para los atributos. Se pueden nombrar los atributos de una relación de manera implícita utilizando una notación posicional, donde \$1, \$2, … hagan refe-

saldo
900

FIGURA 3.18. Saldo máximo de las cuentas del banco.

nombre-cliente
Gómez
Pérez

FIGURA 3.19. Los clientes que viven en la misma calle y en la misma ciudad que Gómez.

rencia al primer atributo, al segundo, etcétera. La notación posicional también se aplica a los resultados de las operaciones del álgebra relacional. La siguiente expresión del álgebra relacional ilustra el uso de la notación posicional con el operador unario σ:

$$\sigma_{\$2=\$3}(R \times R)$$

Si una operación binaria necesita distinguir entre las dos relaciones que son sus operandos, se puede utilizar una notación posicional parecida para los nombres de las relaciones. Por ejemplo, $\$R1$ puede hacer referencia al primer operando y $\$R2$, al segundo. Sin embargo, la notación posicional no resulta conveniente para las personas, dado que la posición del atributo es un número en vez de un nombre de atributo fácil de recordar. Por tanto, en este libro no se utiliza la notación posicional.

3.2.2. Definición formal del álgebra relacional

Las operaciones que se vieron en el Apartado 3.2.1 permiten dar una definición completa de las expresiones del álgebra relacional. Las expresiones fundamentales del álgebra relacional se componen de alguna de las siguientes:

- Una relación de la base de datos
- Una relación constante

Una relación constante se escribe listando sus tuplas entre llaves ({}), por ejemplo {(C-101,Centro,500) (C-215, Becerril, 700)}.

Las expresiones generales del álgebra relacional se construyen a partir de subexpresiones menores. Sean E_1 y E_2 expresiones de álgebra relacional. Todas las siguientes son expresiones del álgebra relacional:

- $E_1 \cup E_2$
- $E_1 - E_2$
- $E_1 \times E_2$
- $\sigma_P(E_1)$, donde P es un predicado de atributos de E_1
- $\Pi_S(E_1)$, donde S es una lista que se compone de algunos de los atributos de E_1
- $\rho_x(E_1)$, donde x es el nuevo nombre del resultado de E_1.

3.2.3. Otras operaciones

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta del álgebra relacional[1]. Sin embargo, si uno se limita únicamente a las operaciones fundamentales, algunas consultas habituales resultan de expresión intrincada. Por tanto, se definen otras operaciones que no añaden potencia al álgebra, pero que simplifican las consultas habituales. Para cada operación nueva se facilita una expresión equivalente utilizando sólo las operaciones fundamentales.

3.2.3.1. La operación intersección de conjuntos

La primera operación adicional del álgebra relacional que se definirá es la **intersección de conjuntos** ($\cap$). Supóngase que se desea averiguar todos los clientes que tienen un préstamo concedido y una cuenta abierta. Utilizando la intersección de conjuntos se puede escribir

$$\Pi_{nombre\text{-}cliente}(prestatario) \cap \Pi_{nombre\text{-}cliente}(impositor)$$

La relación resultante de esta consulta aparece en la Figura 3.20.

Obsérvese que se puede volver a escribir cualquier expresión del álgebra relacional utilizando la intersección de conjuntos sustituyendo la operación intersección por un par de operaciones de diferencia de conjuntos, de la manera siguiente:

$$r \cap s = r - (r - s)$$

Por tanto, la intersección de conjuntos no es una operación fundamental y no añade potencia al álgebra relacional. Sencillamente, es más conveniente escribir $r \cap s$ que $r - (r - s)$.

3.2.3.2. La operación reunión natural

Suele resultar deseable simplificar ciertas consultas que exigen un producto cartesiano. Generalmente, las consultas que implican un producto cartesiano incluyen un operador selección sobre el resultado del producto cartesiano. Considérese la consulta «Hallar los nombres de todos los clientes que tienen concedido un préstamo en el banco y averiguar el importe del mismo». En primer lugar se calcula el producto cartesiano de las relaciones *prestatario* y *préstamo*. Luego, se seleccionan las tuplas que sólo atañen al mismo *número-préstamo*, seguidas por la proyección de *nombre-cliente*, *número-préstamo* e *importe* resultantes:

$$\Pi_{nombre\text{-}cliente,\ préstamo.número\text{-}préstamo,\ importe}(\sigma_{prestatario.número\text{-}préstamo\ =\ préstamo.número\text{-}préstamo}(prestatario \times préstamo))$$

[1] En el Apartado 3.3 se introducen las operaciones que extienden la potencia del álgebra relacional al tratamiento de los valores nulos y los valores de agregación.

nombre-cliente
Gómez
Pérez
Santos

FIGURA 3.20. Clientes con una cuenta abierta y un préstamo en el banco

La *reunión natural* es una operación binaria que permite combinar ciertas selecciones y un producto cartesiano en una sola operación. Se denota por el símbolo de la «reunión» ⋈ . La operación reunión natural forma un producto cartesiano de sus dos argumentos, realiza una selección forzando la igualdad de los atributos que aparecen en ambos esquemas de relación y, finalmente, elimina los atributos duplicados.

Aunque la definición de la reunión natural es compleja, la operación es sencilla de aplicar. Como ilustración, considérese nuevamente el ejemplo «Averiguar los nombres de todos los clientes que tienen concedido un préstamo en el banco y averiguar su importe». Esta consulta puede expresarse utilizando la reunión natural de la manera siguiente:

$$\Pi_{\textit{nombre-cliente, número-préstamo, importe}} (\textit{prestatario} \bowtie \textit{préstamo})$$

Dado que los esquemas de *prestatario* y de *préstamo* (es decir, *Esquema-prestatario* y *Esquema-préstamo*) tienen en común el atributo *número-préstamo*, la operación reunión natural sólo considera los pares de tuplas que tienen el mismo valor de *número-préstamo*. Esta operación combina cada uno de estos pares en una sola tupla en la unión de los dos esquemas (es decir, *nombre-cliente*, *nombre-sucursal*, *número-préstamo*, *importe*). Después de realizar la proyección, se obtiene la relación mostrada en la Figura 3.21.

Considérense dos esquemas de relación R y S que son, por supuesto, listas de nombres de atributos. Si se consideran los esquemas como *conjuntos*, en vez de como listas, se pueden denotar los nombres de los atributos que aparecen tanto en R como en S mediante $R \cap S$, y los nombres de los atributos que aparecen en R, en S o en ambos mediante $R \cup S$. De manera parecida, los nombres de los atributos que aparecen en R pero no en S se denotan por $R - S$, mientras que $S - R$ denota los nombres de los atributos que aparecen en S pero no en R. Obsérvese que las operaciones unión, intersección y diferencia aquí operan sobre conjuntos de atributos, y no sobre relaciones.

Ahora se está preparado para una definición formal de la reunión natural. Considérense dos relaciones $r(R)$ y $s(S)$. La **reunión natural** de r y de s, denotada por $r \bowtie s$ es una relación del esquema $R \cup S$ definida formalmente de la manera siguiente:

$$r \bowtie s = \Pi_{R \cup S} (\sigma_{r.A_1 = s.A_1 \wedge r.A_2 = s.A_2 \wedge \ldots \wedge r.A_n = s.A_n} r \times s)$$

donde $R \cap S = \{A_1, A_2, \ldots, A_n\}$.

Como la reunión natural es fundamental para gran parte de la teoría y de la práctica de las bases de datos relacionales, se ofrecen varios ejemplos de su uso.

- Hallar los nombres de todas las sucursales con clientes que tienen una cuenta abierta en el banco y que viven en Peguerinos.

  $$\Pi_{\textit{nombre-sucursal}} (\sigma_{\textit{ciudad-cliente} = \text{«Peguerinos»}} (\textit{cliente} \bowtie \textit{cuenta} \bowtie \textit{impositor}))$$

  La relación resultante de esta consulta aparece en la Figura 3.22.

  Obsérvese que se escribió *cliente* ⋈ *cuenta* ⋈ *impositor* sin añadir paréntesis para especificar el orden en que se deben ejecutar las operaciones reunión natural de las tres relaciones. En el caso anterior hay dos posibilidades:

  — (*cliente* ⋈ *cuenta*) ⋈ *impositor*
  — *cliente* ⋈ (*cuenta* ⋈ *impositor*)

  No se especificó la expresión deseada porque las dos son equivalentes. Es decir, la reunión natural es **asociativa**.

- Hallar todos los clientes que tienen una cuenta abierta *y* un préstamo concedido en el banco.

  $$\Pi_{\textit{nombre-cliente}} (\textit{prestatario} \bowtie \textit{impositor})$$

  Obsérvese que en el Apartado 3.2.3.1 se escribió una expresión para esta consulta utilizando la intersección de conjuntos. Aquí se repite esa expresión.

$$\Pi_{\textit{nombre-cliente}} (\textit{prestatario}) \cap \Pi_{\textit{nombre-cliente}} (\textit{impositor})$$

nombre-cliente	*número-préstamo*	*importe*
Fernández	P-16	1.300
Gómez	P-23	2.000
Gómez	P-11	900
López	P-15	1.500
Pérez	P-93	500
Santos	P-17	1.000
Sotoca	P-14	1.500
Valdivieso	P-17	1.000

FIGURA 3.21. Resultado de $\Pi_{\textit{nombre-cliente, número-préstamo, importe}} (\textit{prestatario} \bowtie \textit{préstamo})$.

nombre-sucursal
Galapagar
Navacerrada

FIGURA 3.22. Resultado de $\Pi_{\textit{nombre-sucursal}} (\sigma_{\textit{ciudad-cliente} = \text{«Peguerinos»}} (\textit{cliente} \bowtie \textit{cuenta} \bowtie \textit{impositor}))$.

La relación resultante de esta consulta se mostró anteriormente en la Figura 3.20. Este ejemplo ilustra una realidad común del álgebra relacional: se pueden escribir varias expresiones del álgebra relacional equivalentes que sean bastante diferentes entre sí.

- Sean $r(R)$ y $s(S)$ relaciones sin atributos en común; es decir, $R \cap S = \emptyset$. ($\emptyset$ denota el conjunto vacío.) Por tanto, $r \bowtie s = r \times s$.

La operación *reunión zeta* es una extensión de la operación reunión natural que permite combinar una selección y un producto cartesiano en una sola operación. Considérense las relaciones $r(R)$ y $s(S)$, y sea θ un predicado de los atributos del esquema $R \cup S$. La operación **reunión zeta** $r \bowtie_\theta s$ se define de la manera siguiente:

$$r \bowtie_\theta s = \sigma_\theta (r \times s)$$

3.2.3.3. La operación división

La operación **división**, denotada por $\div$, resulta adecuada para las consultas que incluyen la expresión «para todos». Supóngase que se desea hallar a todos los clientes que tengan abierta una cuenta en *todas* las sucursales ubicadas en Arganzuela. Se pueden obtener todas las sucursales de Arganzuela mediante la expresión

$$r_1 = \Pi_{nombre\text{-}sucursal} (\sigma_{ciudad\text{-}sucursal = \text{«Arganzuela»}} (sucursal))$$

La relación resultante de esta expresión aparece en la Figura 3.23.

Se pueden encontrar todos los pares (*nombre-cliente*, *nombre-sucursal*) para los que el cliente tiene una cuenta en una sucursal escribiendo

$$r_2 = \Pi_{nombre\text{-}cliente,\ nombre\text{-}sucursal} (impositor \bowtie cuenta)$$

La Figura 3.24 muestra la relación resultante de esta expresión.

Ahora hay que hallar los clientes que aparecen en r_2 con los nombres de *todas* las sucursales de r_1. La operación que proporciona exactamente esos clientes es la operación división. La consulta se formula escribiendo

$$\Pi_{nombre\text{-}cliente,\ nombre\text{-}sucursal} (impositor \bowtie cuenta) \div \Pi_{nombre\text{-}sucursal} (\sigma_{ciudad\text{-}sucursal = \text{«Arganzuela»}} (sucursal))$$

El resultado de esta expresión es una relación que tiene el esquema (*nombre-cliente*) y que contiene la tupla (González).

nombre-sucursal
Centro
Galapagar

FIGURA 3.22. Resultado de $\Pi_{nombre\text{-}sucursal} (\sigma_{ciudad\text{-}sucursal = \text{«Arganzuela»}} (sucursal))$.

nombre-cliente	*nombre-sucursal*
Abril	Collado Mediano
Gómez	Becerril
González	Centro
González	Galapagar
López	Navacerrada
Rupérez	Moralzarzal
Santos	Galapagar
Valdivieso	Navacerrada

FIGURA 3.24. Resultado de $\Pi_{nombre\text{-}cliente,\ nombre\text{-}sucursal} (impositor \bowtie cuenta)$.

Formalmente, sean $r(R)$ y $s(S)$ relaciones y $S \subseteq R$; es decir, todos los atributos del esquema S están también en el esquema R. La relación $r \div s$ es una relación del esquema $R - S$ (es decir, del esquema que contiene todos los atributos del esquema R que no están en el esquema S). Una tupla t está en $r \div s$ si y sólo si se cumplen estas dos condiciones:

1. t está en $\Pi_{R-S} (r)$
2. Para cada tupla t_S de s hay una tupla t_r de r que cumple las dos condiciones siguientes:
   a. $t_r[S] = t_s[S]$
   b. $t_r[R - S] = t$

Puede resultar sorprendente descubrir que, dados una operación división y los esquemas de las relaciones, se puede, de hecho, definir la operación división en términos de las operaciones fundamentales. Sean $r(R)$ y $s(S)$ dadas, con $S \subseteq R$:

$$r \div s = \Pi_{R-S} (r) - \Pi_{R-S} ((\Pi_{R-S} (r) \times s) - \Pi_{R-S,S} (r))$$

Para comprobar que esta expresión es verdadera, obsérvese que $\Pi_{R-S} (r)$ da todas las tuplas t que cumplen la primera condición de la definición de la división. La expresión del lado derecho del operador diferencia de conjuntos,

$$\Pi_{R-S} ((\Pi_{R-S} (r) \times s) - \Pi_{R-S,S} (r)),$$

sirve para borrar esas tuplas que no cumplen la segunda condición de la definición de la división. Esto se logra de la manera siguiente. Considérese $\Pi_{R-S} (r) \times s$. Esta relación está en el esquema R y empareja cada tupla de $\Pi_{R-S} (r)$ con cada tupla de s. La expresión $\Pi_{R-S,S} (r)$ sólo reordena los atributos de r.

Por tanto, $(\Pi_{R-S} (r) \times s) - \Pi_{R-S,S} (r)$ genera los pares de tuplas de $\Pi_{R-S} (r)$ y de s que no aparecen en r. Si una tupla t_j está en

$$\Pi_{R-S} ((\Pi_{R-S} (r) \times s) - \Pi_{R-S,S} (r)),$$

hay alguna tupla t_s de s que no se combina con la tupla t_j para formar una tupla de r. Por tanto, t_j guarda un valor de los atributos $R - S$ que no aparece en $r \div s$. Estos valores son los que se eliminan de $\Pi_{R-S} (r)$.

3.2.3.4. La operación asignación

En ocasiones resulta conveniente escribir una expresión del álgebra relacional por partes utilizando la asignación a una variable de relación temporal. La operación **asignación**, denotada por ←, actúa de manera parecida a la asignación de los lenguajes de programación. Para ilustrar esta operación, considérese la definición de la división dada en el Apartado 3.2.3.3. Se puede escribir $r \div s$ como

$$temp1 \leftarrow \Pi_{R-S}(r)$$
$$temp2 \leftarrow \Pi_{R-S}((temp1 \times s) - \Pi_{R-S,S}(r))$$
$$resultado = temp1 - temp2$$

La evaluación de una asignación no hace que se muestre ninguna relación al usuario. Por el contrario, el resultado de la expresión a la derecha de ← se asigna a la variable relación a la izquierda de ←. Esta variable relación puede utilizarse en expresiones posteriores.

Con la operación asignación se pueden escribir las consultas como programas secuenciales consistentes en una serie de asignaciones seguida de una expresión cuyo valor se muestra como resultado de la consulta. En las consultas del álgebra relacional la asignación siempre debe hacerse a una variable de relación intermedia. Las asignaciones a relaciones permanentes constituyen una modificación de la base de datos. Este asunto se discutirá en el Apartado 3.4. Obsérvese que la operación asignación no añade potencia alguna al álgebra. Resulta, sin embargo, una manera conveniente de expresar las consultas complejas.

3.3. OPERACIONES DEL ÁLGEBRA RELACIONAL EXTENDIDA

Las operaciones básicas del álgebra relacional se han ampliado de varias maneras. Una ampliación sencilla es permitir operaciones aritméticas como parte de la proyección. Una ampliación importante es permitir *operaciones de agregación*, como el cálculo de la suma de los elementos de un conjunto, o su media. Otra ampliación importante es la operación *reunión externa*, que permite a las expresiones del álgebra relacional trabajar con los valores nulos que modelan la información que falta.

3.3.1. Proyección generalizada

La operación **proyección generalizada** amplía la operación proyección permitiendo que se utilicen funciones aritméticas en la lista de proyección. La operación proyección generalizada tiene la forma

$$\Pi_{F_1, F_2, \ldots, F_n}(E)$$

donde E es cualquier expresión del álgebra relacional y $F_1, F_2, \ldots, F_n$ son expresiones aritméticas que incluyen constantes y atributos en el esquema de E. Como caso especial la expresión aritmética puede ser simplemente un atributo o una constante.

Por ejemplo, supóngase que se dispone de una relación *información-crédito*, como se muestra en la Figura 3.25, que da el límite de crédito y el importe dispuesto hasta el momento presente (el *saldo-crédito* de la cuenta). Si se desea averiguar el importe disponible por cada persona, se puede escribir la expresión siguiente:

$$\Pi_{nombre\text{-}cliente,\ límite\ -\ saldo\text{-}crédito}(información\text{-}crédito)$$

El atributo resultante de la expresión *límite – saldo-crédito* no tiene un nombre. Se puede aplicar la operación renombramiento al resultado de la proyección generalizada para darle un nombre. Como conveniencia notacional, el renombramiento de atributos se puede combinar con la proyección generalizada como se ilustra a continuación:

$$\Pi_{nombre\text{-}cliente,\ (límite\ -\ saldo\text{-}crédito)\ \textbf{as}\ crédito\text{-}disponible}(información\text{-}crédito)$$

Al segundo atributo de esta proyección generalizada se le ha dado el nombre *crédito-disponible*. En la Figura 3.26 se muestra el resultado de aplicar esta expresión a la relación de la Figura 3.25.

3.3.2. Funciones de agregación

Las **funciones de agregación** son funciones que toman una colección de valores y devuelven como resultado un único valor. Por ejemplo, la función de agregación

nombre-cliente	*límite*	*saldo-crédito*
Gómez	2.000	400
López	1.500	1.500
Pérez	2.000	1.750
Santos	6.000	700

FIGURA 3.25. La relación *información-crédito*.

nombre-cliente	*crédito-disponible*
Gómez	1.600
López	0
Pérez	250
Santos	5.300

FIGURA 3.26. Resultado de $\Pi_{nombre\text{-}cliente,\ (límite\ -\ saldo\text{-}crédito)\ \textbf{as}\ crédito\text{-}disponible}(información\text{-}crédito)$.

sum toma un conjunto de valores y devuelve la suma de los mismos. Por tanto, la función **sum** aplicada a la colección

$$\{1, 1, 3, 4, 4, 11\}$$

devuelve el valor 24. La función de agregación **avg** devuelve la media de los valores. Cuando se aplica al conjunto anterior devuelve el valor 4. La función de agregación **count** devuelve el número de elementos del conjunto, y devolvería 6 en el caso anterior. Otras funciones de agregación habituales son **min** y **max**, que devuelven el valor mínimo y el máximo de la colección; en el ejemplo anterior devuelven 1 y 11, respectivamente.

Las colecciones en las que operan las funciones de agregación pueden tener valores repetidos; el orden en el que aparezcan los valores no tiene importancia. Estas colecciones se denominan **multiconjuntos**. Los conjuntos son un caso especial de los multiconjuntos, en los que sólo hay una copia de cada elemento.

Para ilustrar el concepto de agregación se utilizará la relación *trabajo-por-horas* descrita en la Figura 3.27, que muestra los empleados a tiempo parcial. Supóngase que se desea averiguar la suma total de los sueldos de los empleados del banco a tiempo parcial. La expresión del álgebra relacional para esta consulta es:

$$\mathcal{G}_{\textbf{sum}(sueldo)}\,(trabajo\text{-}por\text{-}horas)$$

El símbolo $\mathcal{G}$ es la letra G en el tipo de letra caligráfico; se lee «G caligráfica». La operación del álgebra relacional $\mathcal{G}$ significa que se debe aplicar agregación, y el subíndice indica la operación de agregación a aplicar. El resultado de la expresión anterior es una relación con un único atributo, que contiene una sola fila con un valor correspondiente a la suma de los sueldos de todos los trabajadores que trabajan en el banco a tiempo parcial.

Hay casos en los que se deben borrar los valores repetidos antes de calcular una función de agregación. Si se desean borrar los valores repetidos hay que utilizar los mismos nombres de funciones que antes, con la cadena de texto «**distinct**» precedida de un guión añadida al final del nombre de la función (por ejemplo, **count-distinct**). Un ejemplo se da en la consulta «Averiguar el número de sucursales que aparecen en la relación *trabajo-por-horas*». En este caso, el nombre de cada sucursal sólo se cuenta una vez, independientemente del número de empleados que trabajen en la misma. Esta consulta se escribe de la manera siguiente:

$$\mathcal{G}_{\textbf{count-distinct}(\text{nombre-}sucursal)}\,(trabajo\text{-}por\text{-}horas)$$

Para la relación mostrada en la Figura 3.27 el resultado de esta consulta es el valor 3.

Supóngase que se desea hallar la suma total de sueldos de todos los empleados a tiempo parcial en cada sucursal del banco por separado, en lugar de hallar la suma de sueldos de todo el banco. Para ello hay que dividir la relación *trabajo-por-horas* en **grupos** basados en la sucursal y aplicar la función de agregación a cada grupo.

La expresión siguiente obtiene el resultado deseado utilizando el operador de agregación $\mathcal{G}$:

$$_{nombre\text{-}sucursal}\mathcal{G}_{\textbf{sum}(sueldo)}\,(trabajo\text{-}por\text{-}horas)$$

El atributo *nombre-sucursal* subíndice a la izquierda de *G* indica que la relación de entrada *trabajo-por-horas* debe dividirse en grupos de acuerdo con el valor de *nombre-sucursal*. Los grupos resultantes se muestran en la Figura 3.28. La expresión **sum**(*sueldo*) en el subíndice derecho de $\mathcal{G}$ indica que, para cada grupo de tuplas (es decir, para cada sucursal) hay que aplicar la función de agregación **sum** al conjunto de valores del atributo *sueldo*. La relación resultante consiste en las tuplas con el nombre de la sucursal y la suma de los sueldos de la sucursal, como se muestra en la Figura 3.29.

La forma general de la **operación de agregación** $\mathcal{G}$ es la siguiente:

$$_{G_1, G_2, \ldots, G_n}\mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_m(A_m)}\,(E)$$

donde E es cualquier expresión del álgebra relacional; $G_1, G_2, \ldots, G_n$ constituye una lista de atributos que indican cómo se realiza la agrupación, cada F_i es una función de agregación y cada A_i es el nombre de un atributo. El significado de la operación se define de la manera siguiente. Las tuplas en el resultado de la expresión E se dividen en grupos tales que

nombre-empleado	*nombre-sucursal*	*sueldo*
González	Centro	1.500
Díaz	Centro	1.300
Jiménez	Centro	2.500
Catalán	Leganés	1.600
Cana	Leganés	1.500
Cascallar	Navacerrada	5.300
Fernández	Navacerrada	1.500
Ribera	Navacerrada	1.300

FIGURA 3.27. La relación *trabajo-por-horas*.

nombre-empleado	*nombre-sucursal*	*sueldo*
González	Centro	1.500
Díaz	Centro	1.300
Jiménez	Centro	2.500
Catalán	Leganés	1.600
Cana	Leganés	1.500
Cascallar	Navacerrada	5.300
Fernández	Navacerrada	1.500
Ribera	Navacerrada	1.300

FIGURA 3.28. La relación *trabajo-por-horas* después de la agrupación.

nombre-sucursal	*suma de sueldos*
Centro	5.300
Leganés	3.100
Navacerrada	8.100

FIGURA 3.29. Resultado de $_{nombre\text{-}sucursal}\mathcal{G}_{\mathbf{sum}(sueldo)}$ (*trabajo-por-horas*).

1. Todas las tuplas del grupo tienen los mismos valores para $G_1, G_2, \ldots, G_n$.
2. Las tuplas de grupos diferentes tienen valores diferentes para $G_1, G_2, \ldots, G_n$.

Por tanto, los grupos pueden identificarse por el valor de los atributos $G_1, G_2, \ldots, G_n$. Para cada grupo $(g_1, g_2, \ldots, g_n)$ el resultado tiene una tupla $(g_1, g_2, \ldots, g_n, a_1, a_2, \ldots, a_m)$ donde, para cada i, a_i es el resultado de aplicar la función de agregación F_i al multiconjunto de valores del atributo A_i en el grupo.

Como caso especial de la operación de agregación, la lista de atributos $G_1, G_2, \ldots, G_n$ puede estar vacía, en cuyo caso sólo hay un grupo que contiene todas las tuplas de la relación. Esto corresponde a la agregación sin agrupación.

Volviendo al ejemplo anterior, si se deseara averiguar el sueldo máximo de los empleados a tiempo parcial de cada oficina, además de la suma de los sueldos, habría que escribir la expresión

$$_{nombre\text{-}sucursal}\mathcal{G}_{\mathbf{sum}(sueldo),\ \mathbf{max}(sueldo)}(trabajo\text{-}por\text{-}horas)$$

Como en la proyección generalizada, el resultado de una operación de agregación no tiene nombre. Se puede aplicar la operación renombramiento al resultado para darle un nombre. Como conveniencia notacional, los atributos de una operación de agregación se pueden renombrar como se indica a continuación:

$$_{nombre\text{-}sucursal}\mathcal{G}_{\mathbf{sum}(sueldo)\ \mathbf{as}\ suma\text{-}sueldo,\mathbf{max}(sueldo)\ \mathbf{as}\ sueldo\text{-}máximo}(trabajo\text{-}por\text{-}horas)$$

El resultado de la expresión se muestra en la Figura 3.30.

3.3.3. Reunión externa

La operación **reunión externa** es una ampliación de la operación reunión para trabajar con la información que falta. Supóngase que se dispone de relaciones con los siguientes esquemas, que contienen datos de empleados a tiempo completo:

empleado (*nombre-empleado, calle, ciudad*)
trabajo-a-tiempo-completo (*nombre-empleado, nombre-sucursal, sueldo*)

Considérense las relaciones *empleado* y *trabajo-a-tiempo-completo* mostradas en la Figura 3.31. Supóngase que se desea generar una única relación con toda la información (calle, ciudad, nombre de la sucursal y sueldo) de los empleados a tiempo completo. Un posible enfoque sería utilizar la operación reunión natural de la manera siguiente:

$$empleado \bowtie trabajo\text{-}a\text{-}tiempo\text{-}completo$$

El resultado de esta expresión se muestra en la Figura 3.32. Obsérvese que se ha perdido la información sobre la calle y la ciudad de residencia de Gómez, dado que la tupla que describe a Gómez no está presente en la relación *trabajo-a-tiempo-completo*; de manera parecida, se ha perdido la información sobre el nombre de la sucursal y sobre el sueldo de Barea, dado que la tupla que describe a Barea no está presente en la relación *empleado*.

Se puede utilizar la operación reunión externa para evitar esta pérdida de información. En realidad, esta operación tiene tres formas diferentes: *reunión externa por la izquierda*, denotada por $\mathbin{⟕}$; *reunión externa por la derecha*, denotada por $\mathbin{⟖}$ y *reunión externa completa*, denotada por $\mathbin{⟗}$. Las tres formas de la reunión externa calculan la reunión y añaden tuplas adicionales al resultado de la misma. El resultado de las expresiones *empleado* $\mathbin{⟕}$ *trabajo-a-tiempo-completo*, *empleado* $\mathbin{⟖}$ *trabajo-a-tiempo-completo* y *empleado* $\mathbin{⟗}$ *trabajo-a-tiempo-completo* se muestra en las Figuras 3.33, 3.34 y 3.35, respectivamente.

La **reunión externa por la izquierda** ($\mathbin{⟕}$) toma todas las tuplas de la relación de la izquierda que no coincidan con ninguna tupla de la relación de la derecha, las rellena con valores nulos en todos los demás atributos de la relación de la derecha y las añade al resul-

nombre-sucursal	*suma-sueldo*	*sueldo-máximo*
Centro	5.300	2.500
Leganés	3.100	1.600
Navacerrada	8.100	5.300

FIGURA 3.30. Resultado de $_{nombre\text{-}sucursal}\mathcal{G}_{\mathbf{sum}(sueldo)\ \mathbf{as}\ suma\text{-}sueldo\ \mathbf{max}(sueldo)\ \mathbf{as}\ sueldo\text{-}máximo}$ (*trabajo-por-horas*).

nombre-empleado	*calle*	*ciudad*
Segura	Tebeo	La Loma
Domínguez	Viaducto	Villaconejos
Gómez	Bailén	Alcorcón
Valdivieso	Fuencarral	Móstoles

nombre-empleado	*nombre-sucursal*	*sueldo*
Segura	Majadahonda	1.500
Domínguez	Majadahonda	1.300
Barea	Fuenlabrada	5.300
Valdivieso	Fuenlabrada	1.500

FIGURA 3.31. Las relaciones *empleado* y *trabajo-a-tiempo-completo.*

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500

FIGURA 3.32. La relación *empleado* ⋈ *trabajo-a-tiempo-completo.*

tado de la reunión natural. En la Figura 3.33 la tupla (Gómez, Bailén, Alcorcón, *nulo*, *nulo*) es una tupla de este tipo. Toda la información de la relación de la izquierda se halla presente en el resultado de la reunión externa por la izquierda.

La **reunión externa por la derecha** (⟖) es simétrica de la reunión externa por la izquierda. Las tuplas de la relación de la derecha que no coincidan con ninguna tupla de la relación de la izquierda se rellenan con valores nulos y se añaden al resultado de la reunión natural. En la Figura 3.34 la tupla (Barea, *nulo*, *nulo*, Fuenlabrada, 5.300) es una tupla de este tipo. Por tanto, toda la información de la relación de la derecha se halla presente en el resultado de la reunión externa por la derecha.

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*

FIGURA 3.33. Resultado de *empleado* ⟕ *trabajo-a-tiempo-completo.*

La **reunión externa completa** (⟗) realiza estas dos operaciones, rellenando las tuplas de la relación de la izquierda que no coincidan con ninguna tupla de la relación de la derecha y las tuplas de la relación de la derecha que no coincidan con ninguna tupla de la relación de la izquierda, y añadiéndolas al resultado de la reunión. En la Figura 3.35 se muestra el resultado de una reunión externa completa.

Puesto que las operaciones de reunión pueden generar resultados que contengan nulos, es necesario especificar cómo deben manejar estos valores las operaciones del álgebra relacional. El Apartado 3.3.4 aborda este aspecto.

Es interesante observar que las operaciones de reunión externa pueden expresar mediante las operaciones básicas del álgebra relacional. Por ejemplo, la opera-

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

FIGURA 3.34. Resultado de *empleado* ⟖ *trabajo-a-tiempo-completo.*

ción de reunión externa por la izquierda $r ⟕ s$ se puede expresar como:

$$(r \bowtie s) \cup (r - \Pi_R(r \bowtie s)) \times \{(nulo, \ldots, nulo)\}$$

donde la relación constante $\{(nulo, \ldots, nulo)\}$ se encuentra en el esquema $S - R$.

3.3.4. Valores nulos**

En este apartado se define la forma en que las diferentes operaciones del álgebra relacional tratan los valores nulos y las complicaciones que surgen cuando los valores nulos participan en las operaciones aritméticas o en

nombre-empleado	*calle*	*ciudad*	*nombre-sucursal*	*sueldo*
Segura	Tebeo	La Loma	Majadahonda	1.500
Domínguez	Viaducto	Villaconejos	Majadahonda	1.300
Valdivieso	Fuencarral	Móstoles	Fuenlabrada	1.500
Gómez	Bailén	Alcorcón	*nulo*	*nulo*
Barea	*nulo*	*nulo*	Fuenlabrada	5.300

FIGURA 3.35. Resultado de *empleado* ⟗ *trabajo-a-tiempo-completo.*

las comparaciones. Como se verá, a menudo hay varias formas de tratar los valores nulos y, como resultado, las siguientes definiciones pueden ser a veces arbitrarias. Las operaciones y las comparaciones con valores nulos se deberían evitar siempre que sea posible.

Dado que el valor especial *nulo* indica «valor desconocido o no existente», cualquier operación aritmética (como +, –, ∗ y /) que incluya valores nulos debe devolver un valor nulo.

De manera similar, cualquier comparación (como <, <=, >, >= y ≠) que incluya un valor nulo se evalúa al valor especial **desconocido**; no se puede decir si el resultado de la comparación es cierto o falso, así que se dice que el resultado es el nuevo valor lógico *desconocido*.

Las comparaciones que incluyan nulos pueden aparecer dentro de expresiones booleanas que incluyan las operaciones y (conjunción), o (disyunción) y no (negación). Se debe definir la forma en que estas operaciones tratan el valor lógico *desconocido*.

- **y**: (*cierto* **y** *desconocido*) = *desconocido*; (*falso* **y** *desconocido*) = *falso*; (*desconocido* **y** *desconocido*) = *desconocido*.
- **o**: (*cierto* **o** *desconocido*) = *cierto*; (*falso* **o** *desconocido*) = *desconocido*; (*desconocido* **o** *desconocido*) = *desconocido*.
- **no**: (**no** *desconocido*) = *desconocido*.

Ahora es posible describir la forma en que las diferentes operaciones del álgebra relacional tratan los valores nulos. Nuestras definiciones siguen las usadas en el lenguaje SQL.

- **select**: la operación selección evalúa el predicado P en $\sigma_P(E)$ sobre cada tupla de E. Si el predicado devuelve el valor *cierto*, se añade t al resultado. En caso contrario, si el predicado devuelve *desconocido* o *falso*, t no se añade al resultado.
- **reunión**: las reuniones se pueden expresar como un producto cartesiano seguido de una selección. Por tanto, la definición de la forma en que la selección trata los nulos también define la forma en que la operación reunión trata los nulos.

  En una reunión natural $r \bowtie s$ se puede observar de la definición anterior que si dos tuplas, $t_r \in r$ y $t_s \in s$, tienen un valor nulo en un atributo común, entonces las tuplas no casan.
- **proyección**: la operación proyección trata los nulos como cualquier otro valor al eliminar duplicados. Así, si dos tuplas del resultado de la proyección son exactamente iguales, y ambos tienen nulos en los mismos campos, se tratan como duplicados.

  La decisión es un tanto arbitraria porque sin saber cuál es el valor real no se sabe si los dos valores nulos son duplicados o no.
- **unión, intersección, diferencia**: estas operaciones tratan los valores nulos al igual que la operación proyección; tratan las tuplas que tienen los mismos valores en todos los campos como duplicados incluso si algunos de los campos tienen valores nulos en ambas tuplas.

  El comportamiento es un tanto arbitrario, especialmente en el caso de la intersección y la diferencia, dado que no se sabe si los valores reales (si existen) representados por los nulos son los mismos.
- **proyección generalizada**: se describió la manera en que se tratan los nulos en las expresiones al principio del Apartado 3.3.4. Las tuplas duplicadas que contienen valores nulos se tratan como en la operación proyección.

  Cuando hay nulos en los atributos agregados, la operación borra los valores nulos del resultado antes de aplicar la agregación. Si el multiconjunto resultante está vacío, el resultado agregado es nulo.

  Obsérvese que el tratamiento de los valores nulos aquí es diferente que en las expresiones aritméticas ordinarias; se podría haber definido el resultado de una operación de agregación como nulo si incluso sólo uno de los valores agregados es nulo. Sin embargo, esto significaría que un único valor desconocido en un gran grupo podría hacer que el resultado agregado sobre el grupo fuese nulo, y se perdería una gran cantidad de información útil.
- **reunión externa**: las operaciones de reunión externa se comportan como las operaciones reunión, excepto sobre las tuplas que no aparecen en el resultado. Estas tuplas se pueden añadir al resultado (dependiendo de si la operación es $\bowtie$, $\ltimes$ o $\rtimes$) añadiendo nulos.

3.4. MODIFICACIÓN DE LA BASE DE DATOS

Hasta ahora hemos centrado la atención en la extracción de información de la base de datos. En este apartado se abordará la manera de insertar, borrar o modificar información de la base de datos.

Las modificaciones de la base de datos se expresan utilizando la operación asignación. Las asignaciones a las relaciones reales de la base de datos se realizan utilizando la misma notación que se describió para la asignación en el Apartado 3.2.3.

3.4.1. Borrado

Las solicitudes de borrado se expresan básicamente igual que las consultas. Sin embargo, en lugar de mostrar las

tuplas al usuario, se eliminan de la base de datos las tuplas seleccionadas. Sólo se pueden borrar tuplas enteras; no se pueden borrar valores de atributos concretos. En el álgebra relacional los borrados se expresan mediante

$$r \leftarrow r - E$$

donde r es una relación y E es una consulta del álgebra relacional.

He aquí varios ejemplos de solicitudes de borrado del álgebra relacional:

- Borrar todas las cuentas de Gómez.

$$impositor \leftarrow impositor - \sigma_{nombre\text{-}cliente = \text{«Gómez»}}(impositor)$$

- Borrar todos los préstamos con importes entre 0 y 50.

$$préstamo \leftarrow préstamo - \sigma_{importe \geq 0 \text{ and } importe \leq 50}(préstamo)$$

- Borrar todas las cuentas de las sucursales sitas en Getafe.

$$r_1 \leftarrow \sigma_{ciudad\text{-}sucursal = \text{«Getafe»}}(cuenta \bowtie sucursal)$$
$$r_2 \leftarrow \Pi_{nombre\text{-}sucursal,\ número\text{-}cuenta,\ saldo}(r_1)$$
$$cuenta \leftarrow cuenta - r_2$$

  Obsérvese que en el último ejemplo se simplificó la expresión utilizando la asignación a las relaciones temporales (r_1 y r_2).

3.4.2. Inserción

Para insertar datos en una relación hay que especificar la tupla que se va a insertar o escribir una consulta cuyo resultado sea un conjunto de tuplas que vayan a insertarse. Evidentemente, el valor de los atributos de las tuplas insertadas deben ser miembros del dominio de cada atributo. De manera parecida, las tuplas insertadas deben ser de la aridad correcta. En el álgebra relacional las inserciones se expresan mediante

$$r \leftarrow r \cup E$$

donde r es una relación y E es una expresión del álgebra relacional. La inserción de una sola tupla se expresa haciendo que E sea una relación constante que contiene una tupla.

Supóngase que se desea insertar el hecho de que Gómez tiene 1.200 € en la cuenta C-973 en la sucursal de Navacerrada. Hay que escribir

$$cuenta \leftarrow cuenta \cup \{(\text{C-973}, \text{«Navacerrada»}, 1200)\}$$
$$impositor \leftarrow impositor \cup \{(\text{«Gómez»}, \text{C-973})\}$$

De forma más general, puede que se desee insertar tuplas de acuerdo con el resultado de una consulta. Supóngase que se desea ofrecer una nueva cuenta de ahorro con 200 € como regalo a todos los clientes con préstamos concedidos en la sucursal de Navacerrada. Sea el número de préstamo el que se utilice como número de cuenta de esta cuenta de ahorro. Hay que escribir

$$r_1 \leftarrow (\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»}}(prestatario \bowtie préstamo))$$
$$r_2 \leftarrow \Pi_{nombre\text{-}sucursal,\ número\text{-}préstamo}(r_1)$$
$$cuenta \leftarrow cuenta \cup (r_2 \times \{(200)\})$$
$$impositor \leftarrow impositor \cup \Pi_{nombre\text{-}cliente,\ número\text{-}préstamo}(r_1)$$

En lugar de especificar las tuplas como se hizo anteriormente, se especifica un conjunto de tuplas que se insertan en las relaciones *cuenta* e *impositor*. Cada tupla de la relación *cuenta* tiene el *nombre-sucursal* (Navacerrada), un *número-cuenta* (que es igual que el número de préstamo) y el saldo inicial de la nueva cuenta (200 €). Cada tupla de la relación *impositor* tiene como *nombre-cliente* el nombre del prestatario al que se le da la nueva cuenta y el mismo número de cuenta que la correspondiente tupla de *cuenta*.

3.4.3. Actualización

Puede que, en algunas situaciones, se desee modificar un valor de una tupla sin modificar *todos* los valores de la tupla. Se puede utilizar el operador proyección generalizada para realizar esta tarea:

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n}(r)$$

donde cada F_i es el i-ésimo atributo de r, si el i-ésimo atributo no está actualizado, o, si hay que actualizar el atributo, una expresión, que sólo implica constantes y los atributos de r, que da el nuevo valor del atributo.

Si se desea seleccionar varias tuplas de r y sólo actualizar esas mismas tuplas, se puede utilizar la expresión siguiente, donde P denota la condición de selección que escoge las tuplas que hay que actualizar:

$$r \leftarrow \Pi_{F_1, F_2, \ldots, F_n}(\sigma_P(r)) \cup (r - \sigma_P(r))$$

Para ilustrar el uso de la operación actualización supóngase que se realiza el pago de los intereses y que hay que aumentar todos los saldos en un 5 por ciento. Hay que escribir

$$cuenta \leftarrow \Pi_{nombre\text{-}sucursal,\ número\text{-}cuenta,\ saldo,\ saldo * 1.05}(cuenta)$$

Supóngase ahora que las cuentas con saldos superiores a 10.000 € reciben un interés del 6 por ciento, mientras que los demás reciben un 5 por ciento. Hay que escribir

$$cuenta \leftarrow \Pi_{NS,\ NC,\ saldo * 1.06}(\sigma_{saldo > 10000}(cuenta)) \cup$$
$$cuenta \leftarrow \Pi_{NS,\ NC,\ saldo * 1.05}(\sigma_{saldo \leq 10000}(cuenta))$$

donde las abreviaturas *NS* y *NC* sustituyen a *nombre-sucursal* y a *número-cuenta*, respectivamente.

3.5. VISTAS

En los ejemplos propuestos hasta ahora se ha operado en el nivel del modelo lógico. Es decir, se ha asumido que el conjunto de relaciones que se da son las relaciones reales guardadas en la base de datos.

No es deseable que todos los usuarios puedan ver la totalidad del modelo lógico. Las consideraciones sobre la seguridad pueden exigir que algunos datos queden ocultos para los usuarios. Considérese una persona que necesita saber el número de préstamo de un cliente pero que no necesita ver el importe del préstamo. Esta persona debería ver una relación descrita en el álgebra relacional mediante

$$\Pi_{\textit{nombre-cliente, número-préstamo, nombre-sucursal}} (\textit{prestatario} \bowtie \textit{préstamo})$$

Aparte de las consideraciones sobre la seguridad puede que se desee crear un conjunto personalizado de relaciones que se adapte mejor que el modelo lógico a la intuición de un usuario concreto. Por ejemplo, puede que un empleado del departamento de publicidad quiera ver una relación que conste de los clientes que tengan abierta una cuenta o concedido un préstamo en el banco y de las sucursales con las que trabajan. La relación que se crearía para ese empleado es

$$\Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{impositor} \bowtie \textit{cuenta}) \cup \Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{prestatario} \bowtie \textit{préstamo})$$

Las relaciones que no forman parte del modelo lógico pero se hacen visibles a los usuarios como relaciones virtuales se denominan **vistas**. Se puede trabajar con gran número de vistas sobre cualquier conjunto dado de relaciones reales.

3.5.1. Definición de vistas

Las vistas se definen utilizando la instrucción **create view**. Para definir una vista hay que darle un nombre e indicar la consulta que la va a calcular. La forma de la instrucción **create view** es

create view *v* **as** <expresión de consulta>

donde <expresión de consulta> es cualquier expresión legal de consulta del álgebra relacional. El nombre de la vista se representa mediante *v*.

Como ejemplo considérese la vista consistente en las sucursales y sus clientes. Supóngase que se desea que esta vista se denomine *todos-los-clientes*. Esta vista se define de la manera siguiente:

create view *todos-los-clientes* **as**
$\Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{impositor} \bowtie \textit{cuenta}) \cup \Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{prestatario} \bowtie \textit{préstamo})$

Una vez se ha definido una vista se puede utilizar el nombre de la vista para hacer referencia a la relación virtual que genera la vista. Utilizando la vista *todos-los-clientes* se puede averiguar el nombre de todos los clientes de la sucursal de Navacerrada escribiendo

$$\Pi_{\textit{nombre-cliente}} (\sigma_{\textit{nombre-sucursal}} = \text{«Navacerrada»} (\textit{todos-los-clientes}))$$

Recuérdese que en el Apartado 3.2.1 se escribió la misma consulta sin utilizar vistas.

Los nombres de las vistas pueden aparecer en cualquier lugar en el que pueda encontrarse el nombre de una relación, siempre y cuando no se ejecuten sobre las vistas operaciones de actualización. El asunto de las operaciones de actualización de las vistas se estudia en el Apartado 3.5.2.

La definición de las vistas se diferencia de la operación asignación del álgebra relacional. Supóngase que se define la relación $r1$ de la manera siguiente:

$$r1 \leftarrow \Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{impositor} \bowtie \textit{cuenta}) \cup \Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{prestatario} \bowtie \textit{préstamo})$$

La operación asignación se evalúa una vez, y $r1$ no cambiará cuando se actualicen las relaciones *impositor*, *cuenta*, *préstamo* o *prestatario*. En cambio, si hay alguna modificación en estas relaciones, el conjunto de tuplas de la vista *todos-los-clientes* también cambia. De manera intuitiva, en cualquier momento dado, el conjunto de tuplas de la relación de vistas se define como el resultado de la evaluación de la expresión de consulta que define en ese momento la vista.

Por tanto, si una relación de vistas se calcula y se guarda, puede quedar desfasada si las relaciones utilizadas para definirla se modifican. En vez de eso, las vistas suelen implementarse de la manera siguiente. Cuando se define una vista, el sistema de la base de datos guarda la definición de la propia vista, en vez del resultado de la evaluación de la expresión del álgebra relacional que la define. Siempre que se utiliza una relación de vistas en una consulta, se sustituye por la expresión de consulta guardada. Por tanto, la relación de vistas vuelve a calcularse siempre que se evalúa la consulta.

Algunos sistemas de bases de datos permiten que se guarden las relaciones de vistas, pero se aseguran de que, si las relaciones reales utilizadas en la definición de la vista cambian, la vista se mantenga actualizada. Estas vistas se denominan **vistas materializadas**. El proceso de mantener actualizada la vista se denomina **mantenimiento de vistas**, que se trata en el Apartado 14.5. Las aplicaciones en las que se utiliza frecuentemente una vista se benefician del uso de vistas materializadas, igual que las aplicaciones que demandan una rápida respuesta a ciertas consultas basadas en las vistas. Las ventajas de

la materialización de una vista para las consultas deben sopesarse frente a los costes de almacenamiento y la sobrecarga añadida por las actualizaciones.

3.5.2. Actualizaciones mediante vistas y valores nulos

Aunque las vistas son una herramienta útil para las consultas, plantean problemas significativos si con ellas se expresan las actualizaciones, las inserciones o los borrados. La dificultad radica en que las modificaciones de la base de datos expresadas en términos de vistas deben traducirse en modificaciones de las relaciones reales en el modelo lógico de la base de datos.

Para ilustrar el problema considérese un empleado que necesita ver todos los datos de préstamos de la relación *préstamo* salvo *importe*. Sea *préstamo-sucursal* la vista dada al empleado. Se define esta vista como

create view *préstamo-sucursal* **as**
$\Pi_{\textit{nombre-sucursal, número-préstamo}}$ (*préstamo*)

Dado que se permite que los nombres de las vistas aparezcan en cualquier parte en la que estén permitidos los nombres de relaciones, el empleado puede escribir:

préstamo-sucursal ← *préstamo-sucursal*
∪ {(P-37, «Navacerrada»)}

Esta inserción debe representarse mediante una inserción en la relación *préstamo*, dado que *préstamo* es la relación real a partir de la cual se genera la vista *préstamo-sucursal*. Sin embargo, para insertar una tupla en *préstamo* hay que tener algún valor para *importe*. Hay dos enfoques razonables para trabajar con esta inserción:

- Rechazar la inserción y devolver al usuario un mensaje de error.
- Insertar una tupla (P-37, «Navacerrada», *nulo*) en la relación *préstamo*.

Otro problema resultante de la modificación de la base de datos mediante las vistas aparece en una vista como la siguiente:

create view *información-crédito* **as**
$\Pi_{\textit{nombre-cliente, importe}}$(*prestatario* ⋈ *préstamo*)

Esta vista da una lista del importe de cada préstamo que tenga concedido cualquier cliente del banco. Considérese la inserción siguiente realizada mediante esta vista:

información-crédito ← *información-crédito*
∪ {(«González», 1900)}

El único método posible de insertar tuplas en las relaciones *prestatario* y *préstamo* es insertar («González», *nulo*) en *prestatario* y (*nulo*, *nulo*, 1900) en *préstamo*. Así, se obtienen las relaciones mostradas en la Figura 3.36. Sin embargo, esta actualización no tiene el efecto deseado, dado que la relación de vistas *información-crédito* sigue *sin* incluir la tupla («González», 1900). Por tanto, no existe manera de actualizar las relaciones *prestatario* y *préstamo* utilizando valores nulos para obtener la actualización deseada de *información-crédito*.

Debido a este tipo de problemas generalmente no se permiten las modificaciones en las relaciones de vistas excepto en casos limitados. Los diferentes sistemas de bases de datos especifican diferentes condiciones bajo las que se permiten actualizaciones sobre las vistas; véanse los manuales de cada sistema de bases de datos en particular para consultar más detalles. El problema general de la modificación de las bases de datos mediante las vistas ha sido objeto de numerosas investigaciones. Las notas bibliográficas hacen mención de trabajos recientes sobre este asunto.

3.5.3. Vistas definidas utilizando otras vistas

En el Apartado 3.5.1 se mencionó que las relaciones de vistas pueden aparecer en cualquier lugar en que pueda hacerlo el nombre de una relación, salvo las restricciones en el uso de vistas en expresiones para la actualización. Por tanto, se pueden utilizar vistas en la expresión que define otra vista. Por ejemplo, se puede definir la vista *cliente-navacerrada* de la manera siguiente:

create view *cliente-navacerrada* **as**
$\Pi_{\textit{nombre-cliente}}$ ($\sigma_{\textit{nombre-sucursal}}$ = «Navacerrada»
(*todos-los-clientes*))

donde *todos-los-clientes* es, a su vez, una relación de vistas.

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500
nulo	*nulo*	1.900

nombre-cliente	*número-préstamo*
Fernández	P-16
Gómez	P-23
Gómez	P-11
López	P-15
Pérez	P-93
Santos	P-17
Sotoca	P-14
Valdivieso	P-17
González	nulo

FIGURA 3.36. Tuplas insertadas en *préstamo* y en *prestatario*.

La **expansión de vistas** es una manera de definir el significado de las vistas definidas en términos de otras vistas. El procedimiento asume que las definiciones de vistas no son **recursivas**; es decir, ninguna vista se usa en su propia definición, bien directa o indirectamente a través de otras definiciones de vistas. Por ejemplo, si v_1 se usa en la definición de v_2, se usa en la definición de v_3, y v_3 se usa en la definición de v_1, entonces v_1, v_2 y v_3 son recursivas. Las definiciones de vistas recursivas son útiles en algunos casos, y se volverá a ellas en el contexto del lenguaje Datalog, en el Apartado 5.2.

Sea la vista v_1 definida mediante una expresión e_1 que puede contener a su vez relaciones de vistas. Las relaciones de vistas representan a las expresiones que definen las vistas y, por tanto, se pueden sustituir por las expresiones que las definen. Si se modifica una expresión sustituyendo una relación de vistas por su definición, la expresión resultante puede seguir conteniendo otras relaciones de vistas. Por tanto, la expansión de vistas de una expresión repite la etapa de sustitución de la manera siguiente:

repeat
  Buscar todas las relaciones de vistas v_i de e_1
  Sustituir la relación de vistas v_i por la expresión que define v_i
until no queden más relaciones de vistas en e_1

Mientras las definiciones de las vistas no sean recursivas el bucle concluirá. Por tanto, una expresión e que contenga relaciones de vistas puede entenderse como la expresión resultante de la expansión de vistas de e, que no contiene ninguna relación de vistas.

Como ilustración de la expansión de vistas considérese la expresión siguiente:

$$\sigma_{\textit{nombre-cliente} = \text{«Martín»}} (\textit{cliente-navacerrada})$$

El procedimiento de expansión de vistas produce inicialmente

$$\sigma_{\textit{nombre-cliente} = \text{«Martín»}} (\Pi_{\textit{nombre-cliente}} (\sigma_{\textit{nombre-sucursal} = \text{«Navacerrada»}} (\textit{todos-los-clientes})))$$

luego produce

$$\sigma_{\textit{nombre-cliente} = \text{«Martín»}} (\Pi_{\textit{nombre-cliente}} (\sigma_{\textit{nombre-sucursal} = \text{«Navacerrada»}} (\Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{impositor} \bowtie \textit{cuenta}) \cup \Pi_{\textit{nombre-sucursal, nombre-cliente}} (\textit{prestatario} \bowtie \textit{préstamo}))))$$

No hay más usos de las relaciones de vistas y concluye la expansión de vistas.

3.6. EL CÁLCULO RELACIONAL DE TUPLAS

Cuando escribimos una expresión del álgebra relacional proporcionamos una serie de procedimientos que generan la respuesta a la consulta. El cálculo relacional de tuplas, en cambio, es un lenguaje de consulta **no procedimental**. Describe la información deseada sin dar un procedimiento específico para obtenerla.

Las consultas se expresan en el cálculo relacional de tuplas como

$$\{t \mid P(t)\}$$

es decir, son el conjunto de todas las tuplas tales que el predicado P es cierto para t. Siguiendo la notación utilizada previamente, se utiliza $t[A]$ para denotar el valor de la tupla t en el atributo A y $t \in r$ para denotar que la tupla t está en la relación r.

Antes de dar una definición formal del cálculo relacional de tuplas se volverán a examinar algunas de las consultas para las que se escribieron expresiones de álgebra relacional en el Apartado 3.2.

3.6.1. Consultas de ejemplo

Supóngase que se desea averiguar *nombre-sucursal*, *número-préstamo* e *importe* de los préstamos superiores a 1.200 €:

$$\{t \mid t \in \textit{préstamo} \wedge t[\textit{importe}] > 1200\}$$

Supóngase que sólo se desea obtener el atributo *número-préstamo*, en vez de todos los atributos de la relación *préstamo*. Para escribir esta consulta en el cálculo relacional de tuplas hay que escribir una expresión para una relación del esquema (*número-préstamo*). Se necesitan las tuplas de (*número-préstamo*) tales que hay una tupla en *préstamo* con el atributo *importe* > 1200. Para expresar esta solicitud hay que utilizar el constructor «existe» de la lógica matemática. La notación

$$\exists\, t \in r\, (Q\,(t))$$

significa «existe una tupla t en la relación r tal que el predicado $Q(t)$ es verdadero».

Utilizando esta notación se puede escribir la consulta «Averiguar el número de préstamo de todos los préstamos por importe superior a 1.200 €» como

$$\{t \mid \exists\, s \in \textit{préstamo}\ (t[\textit{número-préstamo}] = s[\textit{número-préstamo}] \wedge s[\textit{importe}] > 1200)\}$$

En español la expresión anterior se lee «el conjunto de todas las tuplas t tales que existe una tupla s en la relación *préstamo* para la que los valores de t y de s para el

atributo *número-préstamo* son iguales y el valor de *s* para el atributo *importe* es mayor que 1.200 €».

La variable tupla *t* sólo se define para el atributo *número-préstamo*, dado que es el único atributo para el que se especifica una condición para *t*. Por tanto, el resultado es una relación de (*número-préstamo*).

Considérese la consulta «Averiguar el nombre de todos los clientes que tienen concedido un préstamo en la sucursal de Navacerrada». Esta consulta es un poco más compleja que las anteriores, dado que implica a dos relaciones: *prestatario* y *préstamo*. Como se verá, sin embargo, todo lo que necesita es que tengamos dos instrucciones «existe» en la expresión de cálculo relacional de tuplas, relacionadas mediante *y* ($\wedge$). La consulta se escribe de la manera siguiente:

$$\{t \mid \exists\, s \in \textit{prestatario}\ (t[\textit{número-préstamo}] = s[\textit{número-préstamo}] \wedge \exists\, u \in \textit{préstamo}\ (u[\textit{número-préstamo}] = s[\textit{número-préstamo}] \wedge u[\textit{nombre-sucursal}] = \text{«Navacerrada»}))\}$$

En español esta expresión es «el conjunto de todas las tuplas (*nombre-cliente*) para las que el cliente tiene un préstamo concedido en la sucursal de Navacerrada». La variable tupla *u* asegura que el cliente es prestatario de la sucursal de Navacerrada. La variable tupla *s* está restringida para que corresponda al mismo número de préstamo que *s*. El resultado de esta consulta se muestra en la Figura 3.37.

Para averiguar todos los clientes del banco que tienen concedido un préstamo, una cuenta abierta, o ambas cosas, se utilizó la operación unión del álgebra relacional. En el cálculo relacional de tuplas harán falta dos instrucciones «existe» relacionadas por *o* ($\vee$):

$$\{t \mid \exists\, s \in \textit{prestatario}\ (t[\textit{nombre-cliente}] = s[\textit{nombre-cliente}]) \vee \exists\, u \in \textit{impositor}\ (t[\textit{nombre-cliente}] = u[\textit{nombre-cliente}])\}$$

Esta expresión da el conjunto de todas las tuplas de *nombre-cliente* tales que se cumple al menos una de las condiciones siguientes:

- *nombre-cliente* aparece en alguna tupla de la relación *prestatario* como prestatario del banco.
- *nombre-cliente* aparece en alguna tupla de la relación *impositor* como impositor del banco.

Si algún cliente tiene concedido un préstamo y una cuenta abierta en el banco, ese cliente sólo aparece una vez en el resultado, ya que la definición matemática de conjunto no permite elementos duplicados. El resultado de esta consulta se mostró previamente en la Figura 3.12.

nombre-cliente
Fernández
López

FIGURA 3.37. Nombre de todos los clientes que tienen concedido un préstamo en la sucursal de Navacerrada.

Si *sólo* queremos conocer los clientes que tienen en el banco una cuenta *y* un préstamo, todo lo que hay que hacer es cambiar en la expresión anterior la *o* ($\vee$) por una *y* ($\wedge$).

$$\{t \mid \exists\, s \in \textit{prestatario}\ (t[\textit{nombre-cliente}] = s[\textit{nombre-cliente}]) \wedge \exists\, u \in \textit{impositor}\ (t[\textit{nombre-cliente}] = u[\textit{nombre-cliente}])\}$$

El resultado de esta consulta se mostró en la Figura 3.20.

Considérese ahora la consulta «Averiguar todos los clientes que tienen una cuenta abierta en el banco pero no tienen concedido ningún préstamo». La expresión del cálculo relacional de tuplas para esta consulta es parecida a las expresiones que se acaban de ver, salvo el uso del símbolo *no* ($\neg$):

$$\{t \mid \exists\, u \in \textit{impositor}\ (t[\textit{nombre-cliente}] = u[\textit{nombre-cliente}]) \wedge \neg\, \exists\, s \in \textit{prestatario}\ (t[\textit{nombre-cliente}] = s[\textit{nombre-cliente}])\}$$

La expresión del cálculo relacional de tuplas anterior utiliza la instrucción $\exists\, u \in \textit{impositor}\ (\ldots)$ para exigir que el cliente tenga una cuenta abierta en el banco, y utiliza la instrucción $\neg\, \exists\, s \in \textit{prestatario}\ (\ldots)$ para borrar a aquellos clientes que aparecen en alguna tupla de la relación *prestatario* por tener un préstamo del banco. El resultado de esta consulta apareció en la Figura 3.13.

La consulta que se tomará ahora en consideración utiliza la implicación, denotada por $\Rightarrow$. La fórmula $P \Rightarrow Q$ es lógicamente equivalente a $\neg P \vee Q$. El uso de la implicación en lugar de *no* y *o* suele sugerir una interpretación más intuitiva de la consulta en español.

Considérese la consulta que se utilizó en el Apartado 3.2.3 para ilustrar la operación división: «Averiguar todos los clientes que tienen una cuenta en todas las sucursales sitas en Arganzuela». Para escribir esta consulta en el cálculo relacional de tuplas se introduce el constructor «para todo», denotado por $\forall$. La notación

$$\forall\, t \in r\ (Q\ (t))$$

significa «Q es verdadera para todas las tuplas t de la relación r».

La expresión para la consulta se escribe de la manera siguiente:

$$\{t \mid \exists\, r \in \textit{cliente}\ (r[\textit{nombre-cliente}] = t[\textit{nombre-cliente}] \wedge (\forall\, u \in \textit{sucursal}\ (u[\textit{ciudad-sucursal}] = \text{«Arganzuela»} \Rightarrow \exists\, s \in \textit{impositor}\ (t[\textit{nombre-cliente}] = s[\textit{nombre-cliente}] \wedge \exists\, w \in \textit{cuenta}\ (w[\textit{número-cuenta}] = s[\textit{número-cuenta}] \wedge w[\textit{nombre-sucursal}] = u[\textit{nombre-sucursal}])))))\}$$

En español esta expresión se interpreta como «el conjunto de todos los clientes (es decir, las tuplas t (*nombre-cliente*)) tales que, para todas las tuplas u de la relación *sucursal*, si el valor de u en el atributo *ciudad-sucursal* es Arganzuela, el cliente tiene una cuenta en la sucursal cuyo nombre aparece en el atributo *nombre-sucursal* de u».

Nótese que hay una sutileza en la consulta anterior: si no hay ninguna sucursal en Arganzuela, todos los nombres de cliente satisfacen la condición. La primera línea de la expresión de consulta es crítica en este caso: sin la condición

$$\exists\, r \in \textit{cliente}\ (r[\textit{nombre-cliente}] = t[\textit{nombre-cliente}]$$

si no hay sucursal en Arganzuela, cualquier valor de t (incluyendo los valores que no son nombres de cliente en la relación *cliente*) valdría.

3.6.2. Definición formal

Ahora se tiene la preparación necesaria para una definición formal. Las expresiones del cálculo relacional de tuplas son de la forma

$$\{t \mid P\,(t)\}$$

donde P es una *fórmula*. En una fórmula pueden aparecer varias variables tupla. Se dice que una variable tupla es una *variable libre* a menos que esté cuantificada mediante $\exists$ o $\forall$. Por tanto, en

$$t \in \textit{préstamo} \wedge \exists\, s \in \textit{cliente}\ (t[\textit{nombre-sucursal}] = s[\textit{nombre-sucursal}])$$

t es una variable libre. La variable tupla s se denomina variable *ligada*.

Las fórmulas de cálculo relacional de tuplas se construyen con *átomos*. Los átomos tienen una de las formas siguientes:

- $s \in r$, donde s es una variable tupla y r es una relación (no se permite el uso del operador $\notin$)
- $s[x]\ \Theta\ u[y]$, donde s y u son variables tuplas, x es un atributo en el que está definida s, y es un atributo en el que está definida u y Θ es un operador de comparación ($<, \leq, =, \neq, >, \geq$); es necesario que los atributos x e y tengan dominios cuyos miembros puedan compararse mediante Θ
- $s[x]\ \Theta\ c$, donde s es una variable tupla, x es un atributo en el que está definida s, Θ es un operador de comparación y c es una constante en el dominio del atributo x

Las fórmulas se construyen a partir de los átomos utilizando las reglas siguientes:

- Un átomo es una fórmula.
- Si P_1 es una fórmula, también lo son $\neg P_1$ y (P_1).
- Si P_1 y P_2 son fórmulas, también lo son $P_1 \vee P_2$, $P_1 \wedge P_2$ y $P_1 \Rightarrow P_2$.
- Si $P_1(s)$ es una fórmula que contiene una variable tupla libre s, y r es una relación,

$$\exists\, s \in r\,(P_1(s))\ \text{ y }\ \forall\, s \in r\,(P_1(s))$$

también son fórmulas

Igual que en el álgebra relacional, se pueden escribir expresiones equivalentes que no sean idénticas en apariencia. En el cálculo relacional de tuplas estas equivalencias incluyen las tres reglas siguientes:

1. $P_1 \wedge P_2$ es equivalente a $\neg\,(\neg\,(P_1) \vee \neg\,(P_2))$.
2. $\forall\, t \in r\,(P_1(t))$ es equivalente a $\neg \exists\, t \in r\,(\neg\, P_1(t))$.
3. $P_1 \Rightarrow P_2$ es equivalente a $\neg\,(P_1) \vee P_2$.

3.6.3. Seguridad de las expresiones

Queda un último asunto por tratar. Las expresiones del cálculo relacional de tuplas pueden generar relaciones infinitas. Supóngase que se escribió la expresión

$$\{t \mid \neg\,(t \in \textit{préstamo})\}$$

Hay infinitas tuplas que no están en *préstamo*. La mayor parte de estas tuplas contienen valores que ni siquiera aparecen en la base de datos. Resulta evidente que no se desea permitir ese tipo de expresiones.

Para ayudar a definir las restricciones del cálculo relacional de tuplas se introduce el concepto de **dominio** de una fórmula relacional de tuplas, P. De manera intuitiva, el dominio de P, denotado por $dom(P)$, es el conjunto de todos los valores a los que P hace referencia. Esto incluye a los valores mencionados en la propia P, así como a los valores que aparezcan explícitamente en P o en una o en varias relaciones cuyos nombres aparezcan en P. Así, el dominio de P es el conjunto de todos los valores que aparecen explícitamente en una o más relación cuyos nombres aparecen en P. Por ejemplo, $dom(t \in \textit{préstamo} \wedge t[\textit{importe}] > 1200)$ es el conjunto que contiene a 1200 y el conjunto de todos los valores que aparecen en *préstamo*. Además, $dom(\neg\,(t \in \textit{préstamo}))$ es el conjunto de todos los valores que aparecen en *préstamo*, dado que la relación *préstamo* se menciona en la expresión.

Se dice que una expresión $\{t \mid P(t)\}$ es *segura* si todos los valores que aparecen en el resultado son valores de $dom(P)$. La expresión $\{t \mid \neg\,(t \in \textit{préstamo})\}$ no es segura. Obsérvese que $dom(\neg\,(t \in \textit{préstamo}))$ es el conjunto de todos los valores que aparecen en *préstamo*. Sin embargo, es posible tener una tupla t que no esté en *préstamo* que contenga valores que no aparezcan en *préstamo*. El resto de ejemplos de expresiones del cálculo relacional de tuplas que se han escrito en este apartado son seguros.

3.6.4. Potencia expresiva de los lenguajes

El cálculo relacional de tuplas restringido a expresiones seguras es equivalente en potencia expresiva al álgebra relacional básica (con los operadores $\cup$, $-$, $\times$, σ y ρ, pero sin los operadores relacionales extendidos tales como la proyección generalizada G y las operaciones de reunión externa). Por tanto, para cada expresión del álgebra relacional hay una expresión equivalente del cálculo relacional de tuplas, y para cada expresión del cálculo relacional de tuplas hay una expresión equivalente del álgebra relacional. No se probará aquí esta afirmación; las notas bibliográficas contienen referencias a la demostración. Algunas partes de la misma se incluyen en los ejercicios. El cálculo relacional de tuplas no tiene ningún equivalente de la operación agregación, pero se puede extender para contenerla. La extensión del cálculo relacional de tuplas para manejar las expresiones aritméticas es sencilla.

3.7. EL CÁLCULO RELACIONAL DE DOMINIOS**

Hay una segunda forma de cálculo relacional denominada **cálculo relacional de dominios**. Esta forma utiliza variables de *dominio* que toman sus valores del dominio de un atributo, en vez de tomarlos de una tupla completa. El cálculo relacional de dominios, sin embargo, se halla estrechamente relacionado con el cálculo relacional de tuplas.

3.7.1. Definición formal

Las expresiones del cálculo relacional de dominios son de la forma

$$\{< x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$$

donde $x_1, x_2, \ldots, x_n$ representan las variables de dominio, P representa una fórmula compuesta de átomos, como era el caso en el cálculo relacional de tuplas. Los átomos del cálculo relacional de dominios tienen una de las formas siguientes:

- $< x_1, x_2, \ldots, x_n > \in r$, donde r es una relación con n atributos y $x_1, x_2, \ldots, x_n$ son variables de dominio o constantes de dominio.
- $x \; \Theta \; y$, donde x e y son variables de dominio y Θ es un operador de comparación ($<, \leq, =, \neq, >, \geq$). Se exige que los atributos x e y tengan dominios que puedan compararse mediante Θ.
- $x \; \Theta \; c$, donde x es una variable de dominio, Θ es un operador de comparación y c es una constante del dominio del atributo para el que x es una variable de dominio.

Las fórmulas se construyen a partir de los átomos utilizando las reglas siguientes:

- Un átomo es una fórmula.
- Si P_1 es una fórmula, también lo son $\neg P_1$ y (P_1).
- Si P_1 y P_2 son fórmulas, también lo son $P_1 \vee P_2$, $P_1 \wedge P_2$ y $P_1 \Rightarrow P_2$.
- Si $P_1(x)$ es una fórmula en x, donde x es una variable de dominio,

  $$\exists \, x < (P_1(x)) \quad \text{y} \quad \forall \, x \, (P_1(x))$$

  también son fórmulas

Como notación abreviada se escribe

$$\exists \, a, b, c \, (P(a, b, c))$$

en lugar de

$$\exists \, a \, (\exists \, b \, (\exists \, c \, (P(a, b, c))))$$

3.7.2. Consultas de ejemplo

Ahora se van a aportar consultas del cálculo relacional de dominios para los ejemplos considerados anteriormente. Obsérvese la similitud de estas expresiones con las expresiones correspondientes del cálculo relacional de tuplas

- Averiguar el nombre de la sucursal, el número de préstamo y el importe de los préstamos superiores a 1.200 €:

  $$\{< p, s, i > \mid < p, s, i > \in \textit{préstamo} \wedge i > 1200\}$$

- Averiguar todos los números de préstamo de los préstamos por importe superior a 1.200 €:

  $$\{< p > \mid \exists \, s, i \, (< p, s, i > \in \textit{préstamo} \wedge i > 1200)\}$$

Aunque la segunda consulta tenga un aspecto muy parecido al de la que se escribió para el cálculo relacional de tuplas, hay una diferencia importante. En el cálculo de tuplas, cuando se escribe $\exists \, s$ para alguna variable tupla s, se vincula inmediatamente con una relación escribiendo $\exists \, s \in r$. Sin embargo, cuando se escribe $\exists \, s$ en el cálculo de dominios, s no se refiere a una tupla, sino a un valor de dominio. Por tanto, el dominio de la variable s no está restringido hasta que la subfórmula $p, s, i \in \textit{préstamo}$ restringe s a los nom-

bres de sucursal que aparecen en la relación *préstamo*. Por ejemplo:

- Averiguar el nombre de todos los clientes que tienen concedido un préstamo en la sucursal de Navacerrada y averiguar el importe del préstamo:

  $\{< n, c > \mid \exists\, l\, (< n, p > \in \textit{prestatario} \wedge \exists\, s\, (< p, s, i > \in \textit{préstamo} \wedge s = \text{«Navacerrada»}))\}$

- Averiguar el nombre de todos los clientes que tienen concedido un préstamo, una cuenta abierta, o ambas cosas, en la sucursal de Navacerrada:

  $\{< n > \mid \exists\, p\, (< n, p > \in \textit{prestatario} \wedge \exists\, s, i\ (< p, s, i > \in \textit{préstamo} \wedge s = \text{«Navacerrada»})) \vee \exists\, c\, (< n, c > \in \textit{impositor} \wedge \exists\, s, i\ (< c, s, i > \in \textit{cuenta} \wedge s = \text{«Navacerrada»}))\}$

- Averiguar el nombre de todos los clientes que tienen una cuenta abierta en todas las sucursales sitas en Arganzuela:

  $\{< c > \mid \exists\, s, t\, (< c, s, t > \in \textit{cliente}) \wedge \forall\, x, y, z\ (< x, y, z > \in \textit{sucursal}) \wedge y = \text{«Arganzuela»} \Rightarrow \exists\, a, b\, (< x, a, b > \in \textit{cuenta} \wedge (< c, a > \in \textit{impositor}))\}$

  En español la expresión anterior se interpreta como «el conjunto de todas las tuplas *c* (*nombre-cliente*) tales que, para todas las tuplas *x*, *y*, *z* (*nombre-sucursal*, *ciudad-sucursal*, *activos*), si la ciudad de la sucursal es Arganzuela, las siguientes afirmaciones son verdaderas»:

  — Existe una tupla de la relación *cuenta* con número de cuenta *a* y nombre de sucursal *x*

  — Existe una tupla de la relación *impositor* con cliente *c* y número de cuenta *a*

3.7.3. Seguridad de las expresiones

Ya se observó que en el cálculo relacional de tuplas es posible escribir expresiones que pueden generar relaciones infinitas. Esto llevó a definir la *seguridad* de las expresiones de cálculo relacional de tuplas. Se produce una situación parecida en el cálculo relacional de dominios. Las expresiones como

$$\{< p, s, i > \mid \neg\, (< p, s, i > \in \textit{préstamo})\}$$

no son seguras porque permiten valores del resultado que no están en el dominio de la expresión.

En el cálculo relacional de dominios también hay que tener en cuenta la forma de las fórmulas dentro de las instrucciones «existe» y «para todo». Considérese la expresión

$$\{< x > \mid \exists\, y\, (< x, y \in r) \wedge \exists\, z\, (\neg\, (< x, z > \in r) \wedge P(x, z))\}$$

donde *P* es una fórmula que implica a *x* y a *z*. Se puede probar la primera parte de la fórmula, $\exists\, y\, (< x, y \in r)$, tomando en consideración sólo los valores de *r*. Sin embargo, para probar la segunda parte de la fórmula, $\exists\, z\, (\neg\, (< x, z > \in r) \wedge P(x, z))$, hay que tomar en consideración valores de *z* que no aparecen en *r*. Dado que todas las relaciones son finitas, no aparece en *r* un número infinito de valores. Por tanto, no resulta posible en general probar la segunda parte de la fórmula $\exists\, z\, (\neg\, (< x, z > \in r) \wedge P(x, z))$, hay que tomar en consideración valores de *z* que no aparecen en *r*. Dado que todas las relaciones son finitas, no aparece en *r* un número infinito de valores. Por tanto, no es posible en general probar la segunda parte de la fórmula sin tomar en consideración un número infinito de valores de *z*. En vez de eso, se añaden restricciones para prohibir expresiones como la anterior.

En el cálculo relacional de tuplas se restringió cualquier variable cuantificada existencialmente a variar sobre una relación concreta. Dado que no se hizo así en el cálculo de dominios, hay que añadir reglas a la definición de seguridad para tratar los casos parecidos a los del ejemplo. Se dice que la expresión

$$\{< x_1, x_2, \ldots, x_n > \mid P(x_1, x_2, \ldots, x_n)\}$$

es segura si se cumplen todas las condiciones siguientes:

1. Todos los valores que aparecen en las tuplas de la expresión son valores de $dom(P)$.
2. Para cada subfórmula «existe» de la forma $\exists\, x\, (P_1(x))$, la subfórmula es cierta si y sólo si hay un valor *x* en $dom(P_1)$ tal que $P_1(x)$ es verdadero.
3. Para cada subfórmula «para todo» de la forma $\forall\, x\, (P_1(x))$, la subfórmula es verdadera si y sólo si $P_1(x)$ es verdadero para todos los valores *x* de $dom(P_1)$.

El propósito de las reglas adicionales es asegurar que se puedan probar las subfórmulas «para todo» y «existe» sin tener que probar infinitas posibilidades. Considérese la segunda regla de la definición de seguridad. Para que $\exists\, x\, (P_1(x))$ sea verdadero sólo hay que encontrar una *x* para la que $P_1\ (x)$ lo sea. En general, habría que probar infinitos valores. Sin embargo, si la expresión es segura, se sabe que se puede restringir la atención a los valores de $dom(P_1)$. Esta restricción reduce las tuplas que hay que tomar en consideración a un número finito.

La situación de las subfórmulas de la forma $\forall\, x\, (P_1(x))$ es parecida. Para asegurar que $\forall\, x\, (P_1(x))$ es verdadero hay que probar en general todos los valores posibles, por lo que hay que examinar infinitos valores. Como antes, si se sabe que la expresión es segura, basta con probar $P_1(x)$ para los valores tomados de $dom(P_1)$.

Todas las expresiones del cálculo relacional de dominios que se han incluido en las consultas de ejemplo de este apartado son seguras.

3.7.4. Potencia expresiva de los lenguajes

Cuando el cálculo relacional de dominios se restringe a expresiones seguras es equivalente en potencia expresiva al cálculo relacional de tuplas restringido a expresiones seguras. Dado que se observó anteriormente que el cálculo relacional de tuplas restringido es equivalente al álgebra relacional, los tres lenguajes siguientes son equivalentes:

- El álgebra relacional básica (sin las operaciones extendidas)
- El cálculo relacional de tuplas restringido a expresiones seguras
- El cálculo relacional de dominios restringido a expresiones seguras

El cálculo relacional de dominios tampoco tiene equivalente para la operación agregación, pero se puede extender para contenerla, y su extensión para el tratamiento de expresiones aritméticas es sencilla.

3.8. RESUMEN

- El **modelo de datos relacional** se basa en un conjunto de tablas. El usuario del sistema de bases de datos puede consultar esas tablas, insertar nuevas tuplas, borrar tuplas y actualizar (modificar) las tuplas. Hay varios lenguajes para expresar estas operaciones.
- El **álgebra relacional** define un conjunto de operaciones algebraicas que operan sobre tablas y devuelven tablas como resultado. Estas operaciones se pueden combinar para obtener expresiones que expresan las consultas deseadas. El álgebra define las operaciones básicas usadas en los lenguajes de consulta relacionales.
- Las operaciones del álgebra relacional se pueden dividir en :
  - — Operaciones básicas
  - — Operaciones adicionales que se pueden expresar en términos de las operaciones básicas
  - — Operaciones extendidas, algunas de las cuales añaden mayor poder expresivo al álgebra relacional
- Las bases de datos se pueden modificar con la **inserción**, el **borrado** y la **actualización** de tuplas. Se usó el álgebra relacional con el **operador de asignación** para expresar estas modificaciones.
- Los diferentes usuarios de una base de datos compartida pueden aprovecharse de vistas individualizadas de la base de datos. Las vistas son «relaciones virtuales» definidas mediante expresiones de consulta.
- Las vistas son mecanismos útiles para simplificar las consultas a la base de datos, pero la modificación de la base de datos mediante las vistas puede tener consecuencias potencialmente desventajosas. Por tanto, los sistemas de bases de datos restringen estrictamente las actualizaciones mediante vistas.
- Por razones de eficiencia del procesamiento de las consultas, una vista puede estar **materializada**, es decir, la consulta se evalúa y el resultado se almacena físicamente. Cuando las relaciones de la base de datos se actualizan, la vista materializada se debe actualizar correspondientemente.
- El **cálculo relacional de tuplas** y el **cálculo relacional de dominios** son lenguajes no procedimentales que representan la potencia básica necesaria en un lenguaje de consultas relacionales. El álgebra relacional básica es un lenguaje procedimental que es equivalente en potencia a ambas formas del cálculo relacional cuando se restringen a las expresiones seguras.
- El álgebra relacional y los cálculos relacionales son lenguajes rígidos, formales, que no resultan adecuados para los usuarios ocasionales de los sistemas de bases de datos. Los sistemas comerciales de bases de datos, por tanto, utilizan lenguajes con más «azúcar sintáctico». En los Capítulos 4 y 5 se tomarán en consideración los tres lenguajes comerciales más influyentes: **SQL**, que está basado en el álgebra relacional, **QBE** y **Datalog**, que están basados en el cálculo relacional de dominios.

TÉRMINOS DE REPASO

- Agrupación
- Álgebra relacional
- Cálculo relacional de dominios
- Cálculo relacional de tuplas
- Clave externa
  - Relación referenciada
  - Relación referenciante
- Claves
- Definición de vistas
- Dominio atómico
- Diagrama de esquema
- Ejemplar de la base de datos
- Ejemplar de la relación
- Esquema de la base de datos
- Esquema de la relación
- Expansión de vistas
- Lenguaje de consulta
- Lenguaje procedimental
- Lenguaje no procedimental
- Modificación de la base de datos
  - Actualización
  - Borrado
  - Inserción
- Multiconjuntos
- Operaciones adicionales
  - División /
  - Intersección de conjuntos $\cap$
  - Reunión natural $\bowtie$
- Operaciones del álgebra relacional
  - Diferencia de conjuntos –
  - Producto cartesiano $\times$
  - Proyección Π
  - Renombramiento ρ
  - Selección σ
  - Unión $\cup$
- Operaciones del álgebra relacional extendida
  - Agregación G
  - Proyección generalizada Π
  - Reunión externa
    - Reunión externa completa $\overset{\bowtie}{}$
    - Reunión externa por la derecha $\overset{\bowtie}{}$
    - Reunión externa por la izquierda $\overset{\bowtie}{}$
- Operación asignación
- Potencia expresiva de los lenguajes
- Relación
- Seguridad de las expresiones
- Tabla
- Valor nulo
- Valores nulos
- Variable tupla
- Vistas
- Vistas recursivas

EJERCICIOS

3.1. Diséñese una base de datos relacional para la oficina de registro de una universidad. La oficina conserva datos sobre cada curso, incluyendo el profesor, el número de estudiantes matriculados y la hora y el lugar de las clases. Por cada pareja estudiante-curso se guarda una calificación.

3.2. Descríbanse las diferencias de significado entre los términos *relación* y *esquema de la relación*. Ilústrese la respuesta haciendo referencia a la solución propuesta para el Ejercicio 3.1.

3.3. Diséñese una base de datos relacional correspondiente al diagrama E-R de la Figura 3.38.

3.4. En el Capítulo 2 se mostró la manera de representar los conjuntos de relaciones de varios a varios, de varios a uno, de uno a varios y de uno a uno. Explíquese la manera en que las claves primarias ayudan a representar estos conjuntos de relaciones en el modelo relacional.

3.5. Considérese la base de datos relacional de la Figura 3.39. Dese una expresión del álgebra relacional, otra del cálculo relacional de tuplas y una tercera del cálculo relacional de dominios para cada una de las consultas siguientes:

a. Averiguar los nombres de todos los empleados que trabajan para el Banco Importante.

b. Averiguar el nombre y la ciudad de residencia de todos los empleados que trabajan para el Banco Importante.

c. Averiguar el nombre, la calle y la ciudad de residencia de todos los empleados que trabajan para el Banco Importante y ganan más de 2.000.000 de pesetas anuales.

d. Averiguar el nombre de todos los empleados de esta base de datos que viven en la misma ciudad que la compañía para la que trabajan.

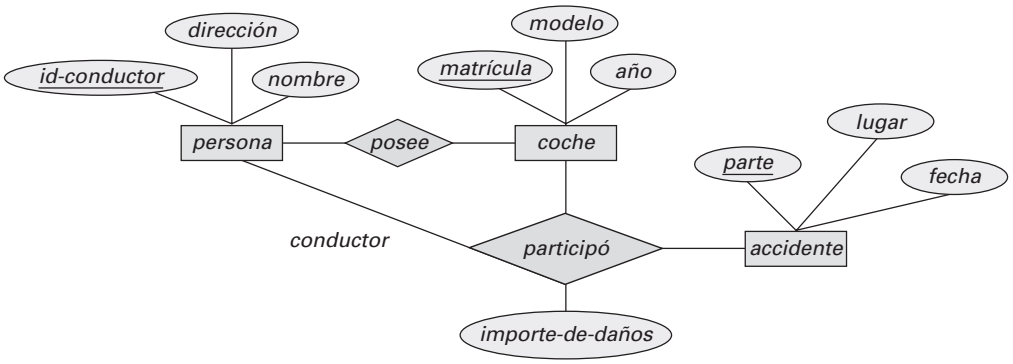

FIGURA 3.38. Diagrama E-R.

e. Averiguar el nombre de todos los empleados que viven en la misma ciudad y en la misma calle que sus jefes.

f. Averiguar el nombre de todos los empleados de esta base de datos que no trabajan para el Banco Importante.

g. Averiguar el nombre de todos los empleados que ganan más que cualquier empleado del Banco Pequeño.

h. Supóngase que las compañías pueden estar instaladas en ciudades pequeñas. Hállense todas las compañías instaladas en cada ciudad en la que está instalado el Banco Pequeño.

3.6. Considérese la relación de la Figura 3.21, que muestra el resultado de la consulta «Averígüese el nombre de todos los clientes que tienen concedido un préstamo en el banco». Vuélvase a escribir la consulta para incluir no sólo el nombre, sino también la ciudad de residencia de cada cliente. Obsérvese que ahora el cliente Sotoca ya no aparece en el resultado, aunque en realidad tiene un préstamo concedido por el banco.

a. Explíquese el motivo de que Sotoca no aparezca en el resultado.

b. Supóngase que se desea que Sotoca aparezca en el resultado. ¿Cómo habría que modificar la base de datos para conseguirlo?

c. Una vez más, supóngase que se desea que Sotoca aparezca en el resultado. Escríbase una consulta utilizando una reunión externa que cumpla esta condición sin que haya que modificar la base de datos.

3.7. Las operaciones de reunión externa amplían la operación reunión natural de manera que las tuplas de las relaciones participantes no se pierdan en el resultado de la reunión. Descríbase la manera en que la operación reunión zeta puede ampliarse para que las tuplas de la relación de la izquierda, las de la relación de la derecha o las de ambas relaciones no se pierdan en el resultado de una reunión zeta.

empleado (*nombre-empleado*, *calle*, *ciudad*)
trabaja (*nombre-empleado*, *nombre-empresa*, *sueldo*)
empresa (*nombre-empresa*, *ciudad*)
jefe (*nombre-empleado*, *nombre-jefe*)

FIGURA 3.39. Base de datos relacional para los Ejercicios 3.5 y 3.10.

3.8. Considérese la base de datos regional de la Figura 3.39. Dese una expresión del álgebra relacional para cada petición:

a. Modificar la base de datos de manera que Santos viva ahora en Tres Cantos.

b. Dar a todos los empleados del Banco Importante un aumento de sueldo del 10 por ciento.

c. Dar a todos los jefes de la base de datos un aumento de sueldo del 10 por ciento.

d. Dar a todos los jefes de la base de datos un aumento de sueldo del 10 por ciento, a menos que el sueldo resultante sea mayor que 100.000 €. En este caso, dar sólo un aumento del 3 por ciento.

e. Borrar todas las tuplas de los empleados de Banco Pequeño de la relación *trabajo*.

3.9. Utilizando el ejemplo bancario, escríbanse consultas del álgebra relacional para averiguar las cuentas abiertas por más de dos clientes:

a. utilizando una función de agregación.

b. sin utilizar funciones de agregación.

3.10. Considérese la base de datos relacional de la Figura 3.38. Dese una expresión del álgebra relacional para cada una de las consultas siguientes:

a. Averiguar la compañía con mayor número de empleados.

b. Averiguar la compañía con la nómina (suma de sueldos de sus empleados) más reducida.

c. Averiguar las compañías cuyos empleados ganen un sueldo más elevado, en media, que el sueldo medio del Banco Importante.

3.11. Dense dos motivos por los que se puede decidir definir una vista.

3.12. Cítense dos problemas importantes del procesamiento de la operación actualización expresadas en términos de vistas.

3.13. Sean los siguientes esquemas de relaciones:

$$R = (A, B, C)$$
$$S = (D, E, F)$$

Sean las relaciones $r(R)$ y $s(S)$. Dese una expresión del cálculo relacional de tuplas que sea equivalente a cada una de las expresiones siguientes:

a. $\Pi_A(R)$
b. $\sigma_{B=17}(r)$
c. $r \square s$
d. $\Pi_{A,F}(\sigma_{C=D}(r \square s))$

3.14. Sea $R = (A, B, C)$ y sean r_1 y r_2 relaciones del esquema R. Dese una expresión del cálculo relacional de dominios que sea equivalente a las expresiones siguientes:

a. $\Pi_A(r_1)$
b. $\sigma_{B=17}(r_1)$
c. $r_1 \cup r_2$
d. $r_1 \cap r_2$
e. $r_1 - r_2$
f. $\Pi_{A,B}(r_1) \bowtie \Pi_{B,C}(r_2)$

3.15. Repítase el Ejercicio 3.5 usando el cálculo relacional de tuplas y el de dominios.

3.16. Sean $R = (A, B)$ y $S = (A, C)$ y sean $r(R)$ y $s(S)$ relaciones. Escríbanse expresiones del álgebra relacional equivalentes a las expresiones siguientes del cálculo relacional de dominios:

a. $\{< a > \mid \exists\, b\, (< a, b > \in r \wedge b = 17)\}$
b. $\{< a, b, c > \mid < a, b > \in r \wedge < a, c > \in s\}$
c. $\{< a > \mid \exists\, b\, (< a, b > \in r) \vee \forall\, c\, (\exists\, d\, (< d, c > \in s) \Rightarrow < a, c > \in s)\}$
d. $\{< a > \mid \exists\, c\, (< a, c > \in s) \wedge \exists\, b_1, b_2\, (< a, b_1 > \in r \wedge < c, b_2 > \in r \wedge b_1 > b_2))\}$

3.17. Sea $R = (A, B)$ y $S = (A, C)$ y sean $r(R)$ y $s(S)$ relaciones. Utilizando la constante especial *nulo*, escríbanse expresiones del cálculo relacional de tuplas equivalentes a cada una de las expresiones siguientes:

a. $r \bowtie\!\!\!\!\square\, s$
b. $r \rtimes\!\!\!\!\square\, s$
c. $r \rtimes s$

3.18. Dense dos motivos por los que se puedan introducir valores nulos en la base de datos.

3.19. Algunos sistemas permiten los valores nulos *marcados*. Un valor nulo marcado $\perp_i$ es igual a sí mismo, pero si $i \neq j$, $\perp_i \neq \perp_j$. Una aplicación de valores nulos marcados debe permitir ciertas actualizaciones mediante el uso de vistas. Considérese la vista *información-crédito* (Apartado 3.5). Muéstrese la manera en que se pueden utilizar los valores nulos marcados para permitir la inserción de la tupla («González», 1900) mediante *información-crédito*.

NOTAS BIBLIOGRÁFICAS

El modelo relacional fue propuesto por E. F. Codd del Laboratorio de investigación de San José de IBM a finales de los años sesenta [Codd, 1970]. Este trabajo motivó la concesión a Codd del prestigioso Premio Turing de la ACM en 1981 (Codd [1982]).

Siguiendo el trabajo original de Codd se constituyeron varios proyectos de investigación con el objetivo de crear sistemas de bases de datos relacionales prácticos, incluyendo System R del Laboratorio de investigación de San José de IBM, Ingres en la Universidad de California en Berkeley, Query-by-Example en el Centro de investigación T. J. Watson de IBM (*IBM T. J. Watson Research Center*) y el vehículo de prueba relacional (*Peterlee Relational Test Vehicle*, PRTV) del Centro científico de IBM (*IBM Scientific Center*) en Peterlee, Reino Unido. System R se discute en Astrahan et al. [1976], Astrahan et al. [1979] y en Chamberlin et al. [1981]. Ingres se discute en Stonebraker [1980], Stonebraker [1986b] y en Stonebraker et al. [1976]. Query-by-Example se describe en Zloof [1977]. PRTV se describe en Todd [1976].

Actualmente están disponibles comercialmente numerosos productos de bases de datos relacionales. Ejemplos de ello son DB2 de IBM, Ingres, Oracle, Sybase, Informix y Microsoft SQL Server. Ejemplos de productos de bases de datos para las computadoras personales son Microsoft Access, dBase y FoxPro. La información sobre estos productos puede hallarse en sus manuales respectivos.

En la mayor parte de los textos sobre bases de datos se incluye una discusión general del modelo relacional de datos. Atzeni y De Antonellis [1993] y Maier [1983] son textos dedicados exclusivamente al modelo relacional de datos. La definición original del álgebra relacional está en Codd [1970]; la del cálculo relacional de tuplas en Codd [1972]. En Codd [1972] se encuentra una prueba formal de la equivalencia del cálculo relacional de tuplas y del álgebra relacional.

Se han propuesto varias ampliaciones del cálculo relacional. Klug [1982] y Escobar-Molano et al. [1993] describen ampliaciones para funciones de agregación escalares. En Codd [1979] se presentan ampliaciones del modelo relacional y discusiones sobre la incorporación de los valores nulos al álgebra relacional (el modelo RM/T), así como las la de las reuniones externas. Codd [1990] es un compendio de los trabajos de E. F. Codd sobre el modelo relacional. Las reuniones externas también se discuten en Date [1993b]. El problema de la actualización de las bases de datos relacionales mediante vistas se aborda en Bancilhon y Spyratos [1981], Cosmadakis y Papadimitriou [1984], Dayal y Bernstein [1978, 1982] y Langerak [1990]. El Apartado 14.5 trata el mantenimiento de las vistas materializadas, y las referencias a la literatura sobre ello se pueden encontrar al final de ese capítulo.