Tema 3. Modelo relacional

Un modelo de datos permite crear una representación de la realidad. Uno de estos modelos es el modelo Entidad-Relación, que permite crear una representación abstracta de la realidad. Dado que la representación de la realidad que obtiene es una representación abstracta, necesitamos un modelo de datos que sea directamente implementable. Uno de estos modelos es el modelo relacional, el cual será el objeto de estudio de este tema. El modelo relacional, además de diferenciarse del modelo Entidad-Relación en que es un modelo de implementación, se diferencia en que es un modelo lógico basado en registros en lugar de ser un modelo lógico basado en objetos. Así, la percepción de la realidad se realiza modelando los conceptos de la realidad como registros. Estos registros son agrupados en tablas, denominadas *relaciones*, lo que da lugar al nombre de modelo relacional. En este tema estudiaremos el modelo relacional, así como los dos lenguajes formales desarrollados para este modelo, como son el álgebra relacional y el cálculo relacional.

3.1. Orígenes del modelo relacional

Desde el punto de vista histórico, el modelo relacional fue introducido por Codd en 1970 sobre una base teórica bastante sólida y utiliza una estructura de datos sencilla y uniforme: *la relación*.

El modelo de datos relacional es relativamente nuevo y se ha establecido como el principal modelo de datos para aplicaciones comerciales de procesamiento de datos, debido fundamentalmente a la existencia en el mercado de muchos SGBD relacionales comerciales. Otros modelos de datos son más expresivos, como es el caso del modelo orientado a objetos, pero los SGBD para estos modelos no están tan extendidos.

No obstante, y debido a que el modelo relacional presenta algunas carencias para el modelado de objetos complejos, se ha desarrollado un modelo de datos que combina el modelo relacional con algunas de las características del modelo orientado a objetos. Este modelo mixto es conocido como el *modelo objeto-relacional*, el cual parece ser el modelo de datos que domine el mercado en los próximos años. Sin embargo, debido a que la mayor parte de las aplicaciones que se siguen desarrollando son para SGBD relacionales, y a que los conceptos explicados en este tema son aplicables tanto al modelo relacional como al objeto-relacional, no trataremos este último en este curso.

3.2. La estructura del modelo relacional

El modelo relacional representa a una base de datos como una colección de relaciones, es decir, un conjunto de tablas formadas por filas y columnas. Cada fila representa un conjunto de datos relacionados entre sí. Dichos valores pueden referirse a un conjunto de hechos que describen a una entidad o bien a un vínculo entre entidades. Por ejemplo, podemos tener una fila de una tabla que incorpore datos de los empleados para cada uno de los empleados de una empresa. Las columnas representan las propiedades de cada una de las filas de la tabla. Por ejemplo, en una tabla que contenga información de empleados podemos tener columnas como *DNI* y *Nombre* para describir distintas características o propiedades de los empleados. El nombre de la tabla y los nombres de las columnas ayudan a interpretar el significado de los valores que están en cada una de las filas de la tabla. Por tanto, una base de datos relacional sería un conjunto de tablas con nombres únicos, en los que las filas representan hechos y las columnas representan propiedades.

En la terminología del modelo relacional, una fila se denomina *tupla*, una cabecera de columna es un *atributo* y una tabla es una *relación*.

Informático	Matemático
Tabla	Relación
Registro o fila	Tupla
Campo o columna	Atributo

3.2.1. Dominios, atributos tuplas y relaciones

Considérese la tabla (relación) *Clientes* de la Figura 3.1. Esta tabla tiene tres atributos: *nombreCli*, *dniCli*, y *domicilio*.

nombreCli	dniCli	Domicilio
Aranda	1	La Reina nº7
García	2	Fragata azul nº8
Hayes	3	Gibraltar español nº14
Turner	4	Gibraltar español nº17
Vilches	5	Diamante S/N
Lara	6	Gato negro nº13
Guerrero	7	Perro nº1

Figura 3.1. La relación *Clientes.*

Formalmente:

Un *dominio D* es un conjunto de valores atómicos (indivisibles). Puede especificarse con el tipo de datos al que pertenecen los valores, y también con un nombre significativo que ayude a interpretarlos, así como otra información adicional. Por ejemplo, el dominio Peso_personas es un dominio numérico, especificado en kilogramos.

Un *esquema de relación R* se denotará por $R(A_1, A_2, ..., A_n)$, y se compone de un nombre de relación *R* y una lista de atributos $A_1, A_2, ..., A_n$. Cada atributo A_i es el nombre del papel desempeñado por algún dominio *D* en el esquema de relación *R*. Se dice que *D* es el dominio de A_i, y se denota por $dom(A_i)$. Un esquema de relación sirve para describir una relación. El *grado de una relación* es el número de atributos *n* de su esquema de relación. Por ejemplo, aquí tenemos un esquema de relación para una relación de grado 3 que describe a los clientes de un banco:

Clientes (*nombreCli, dniCli, domicilio*)

Una *relación* del esquema de relación $R(A_1, A_2, ..., A_n)$, denotada también por $r(R)$, es un conjunto de *n*-tuplas $r = \{ t_1, t_2, ..., t_n \}$. Cada *n*-tupla *t* es una lista de *n* valores ordenados $t=<v_1,v_2, ..., v_n>$ donde cada v_i es un elemento de $dom(A_i)$, o un valor nulo especial. El *i*-ésimo valor de la tupla *t*, que se corresponde con el atributo A_i, se referencia como $t[A_i]$. La Figura 3.1 muestra un ejemplo de la relación *clientes*, en forma de tabla.

Una relación también puede plantearse como una *relación matemática* de grado *n* sobre los dominios $dom(A_1)$, $dom(A_2)$ ... $dom(A_n)$, que es un subconjunto del producto cartesiano de los dominios de *R*:

$r(R) \subseteq (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$

El producto cartesiano especifica todas las combinaciones posibles de valores de los dominios, y el estado actual de la relación refleja sólo las tuplas válidas en ese momento. Los atributos reflejan los papeles o interpretaciones de cada dominio, y puede haber varios atributos de una relación definidos sobre el mismo dominio.

3.2.2. Características de las relaciones

La definición de relación implica características que la distinguen de un fichero o tabla, que vamos a describir a continuación.

3.2.2.1. Orden de las tuplas en una relación

Como una relación es un conjunto de tuplas, y los elementos de un conjunto no están ordenados, las tuplas de una relación no tienen un orden específico. En cambio, los registros de un archivo se almacenan físicamente, de modo que siempre existe un orden entre ellos.

3.2.2.2. Orden de los valores dentro de una tupla y definición alternativa de relación

Según la definición de relación, cada tupla es una *lista ordenada* de *n* valores, por lo que el orden de los valores y el orden de los atributos en la definición de relación es importante. Sin embargo, a nivel lógico esto no es importante, siempre que se mantenga la correspondencia entre atributos y valores.

Hay una *definición alternativa* de relación que no necesita la ordenación de los valores de una tupla, que consiste en definir cada tupla como un conjunto de pares (<atributo><valor>). Esto hace que el orden de los atributos *no* sea importante, ya que el nombre del atributo aparece junto a su valor. En nuestro caso, usaremos la

primera definición de relación, donde los atributos y los valores de las tuplas están ordenados, ya que esto simplifica la notación.

3.2.2.3. Valores en las tuplas

Cada valor de una tupla es un valor indivisible en el modelo relacional básico, por lo que no se permiten atributos compuestos ni multivaluados. Sin embargo, recientemente se han intentado eliminar estas limitaciones, mediante el concepto de relaciones anidadas.

Cuando el valor de un atributo de una tupla es desconocido, o no se le puede aplicar, se usa un valor especial llamado **valor nulo**, aunque se debe intentar evitar su uso, ya que estos valores posteriormente son difíciles de manejar.

3.2.2.4. Interpretación de una relación

El esquema de una relación se puede interpretar como una *declaración*. Por ejemplo, el esquema de la relación *clientes* establece un cliente tiene *nombreCli*, *dniCli* y *ciudadCli*. Cada tupla de la relación se puede interpretar como un *hecho.*

Un esquema de relación también puede interpretarse como un *predicado*, y los valores de cada tuplas serán valores que *satisfacen* el predicado. Esta interpretación es útil en el contexto de la programación lógica.

3.2.3. Notación del modelo relacional

En este tema usaremos esta notación:

- Un esquema de relación R de grado n se denotará con $R(A_1, A_2, ..., A_n)$.
- Una n-tupla de una relación $r(R)$ se denotará con $t=<v_1,v_2, ..., v_n>$.
- Para representar el valor v_i de t para el atributo A_i usaremos $t[A_i]$ o $t.A_i$.
- $t[A_1, A_2, ..., A_z]$ es una subtupla de valores $<v_1, v_2, ..., v_z>$ de t que se corresponden con los atributos especificados en la lista.
- Las letras mayúsculas denotan nombres de relaciones.
- Las letras *q, r, s* denotan estados de relaciones.
- Las letras *t, u, v* denotan tuplas.
- Un atributo A puede especificarse con el nombre de la relación R a la que pertenece usando la notación $R.A$. Esto es útil cuando se usa el mismo nombre para dos atributos en relaciones distintas, ya que dentro de una relación todos los atributos deben tener nombres distintos.

3.3 Restricciones relacionales y esquemas de bases de datos relacionales

Vamos a describir algunos tipos de restricciones sobre los datos que se pueden expresar en el esquema de una base de datos relacional, como son las restricciones de dominio, de clave, de integridad de entidades y de integridad referencial.

3.3.1. Restricciones de dominio

Las restricciones de dominio especifican que el valor de cada atributo debe ser un valor atómico de su dominio. Los tipos de datos asociados a los dominios suelen incluir los tipos numéricos estándar para enteros y reales, así como caracteres y cadenas de longitud fija y variable, fecha, hora, fecha y hora y moneda. También pueden especificarse intervalos o conjuntos explícitos de valores permitidos.

3.3.2. Restricciones de clave y restricciones sobre nulos

Todas la relaciones que forman el esquema de una base de datos relacional tienen que tener definida una clave primaria. Los conceptos de superclave, clave candidata y clave primaria ya vistos para el modelo Entidad-Relación también se puede aplicar al modelo relacional.

Por ejemplo, en el esquema de las sucursales, {*nombreSuc*} y {*nombreSuc, ciudadSuc*} son superclaves, pero {*nombreSuc, ciudadSuc*} no es clave candidata, puesto que si le quitamos el atributo *ciudadSuc* sigue siendo superclave, o lo que es lo mismo {*nombreSuc*} $\subseteq$ {*nombreSuc, ciudadSuc*} y {*nombreSuc*} es en sí una superclave.

Por tanto, se tiene que cumplir que en una relación R, si t_1 y t_2 son dos tuplas distintas de R, y K es el conjunto de atributos que forman la superclave, $t_1[K] \neq t_2[K]$.

La clave de una relación se determina a partir del significado de los atributos, y es una propiedad que no varía con el tiempo. Usaremos como convenio subrayar los atributos que forman la clave primaria de un esquema de relación.

Otra de las restricciones sobre los atributos especifica si se permiten o no los valores nulos. Si toda tupla de una relación debe tener un valor válido y no nulo para un atributo, entonces tendrá la restricción de ser *no nulo*.

3.3.3. Bases de datos relacionales y esquemas de bases de datos

En esta sección vamos a definir una base de datos relacional y un esquema de base de datos relacional. Un *esquema de base de datos relacional* S es un conjunto de esquemas de relaciones $S = \{R_1, R_2, ..., R_n\}$ y un conjunto de restricciones de integridad *RI*. Un *estado* o *instancia* de una base de datos relacional *BD* de S es un conjunto de estados de relaciones $BD = \{r_1, r_2, ...,r_m\}$, en el que cada r_i es un estado de R_i y los estados de relaciones satisfacen las restricciones de integridad especificadas en *RI*.

De ahora en adelante, utilizaremos un ejemplo de una empresa bancaria, que tiene los siguientes esquemas de relaciones:

Sucursales = (nombreSuc, ciudadSuc, activo)

Empleados = (nombreEmp, dniEmp, telefono, nombreSuc)

Cuentas = (numeroCta, saldo, nombreSuc)

Clientes = (nombreCli, dniCli, domicilio)

EsquemaCtaCli = (numeroCta, dniCli)

EsquemaTransacciones = (numeroCta, numeroTrans, fecha, importe)

Este conjunto de esquemas formaría el esquema conceptual de la base de datos del banco, y podría haberse obtenido mediante el paso a tablas de un diagrama entidad-relación como el que se muestra en la Figura 4.

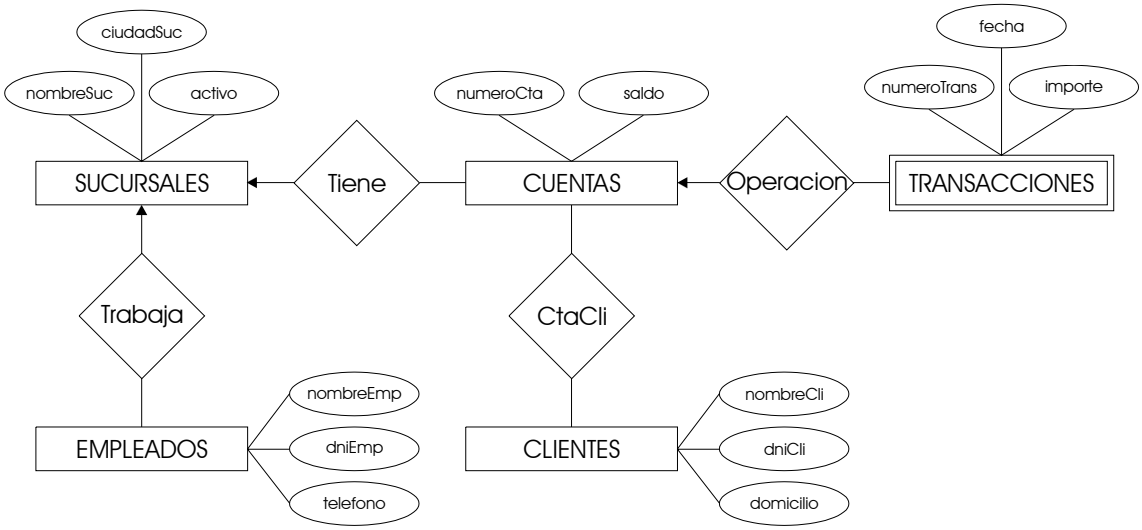

Figura 3.4. Diagrama E-R del sistema bancario

A continuación se muestra la instancia de la base de datos que vamos a utilizar para nuestros ejemplos.

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000
Ganivet	Granada	21000
Paseo	Almería	17000
Zapillo	Almería	4000
Ronda	Almería	80000
Aduana	Roquetas	3000
Los Pinos	Huercal	37000
Sol	Madrid	71000

La relación Sucursales

nombreEmp	dniEmp	telefono	NombreSuc
García	10	101010	Castellana
Torres	11	111111	Castellana
López	12	121212	Paseo
Villegas	13	131313	Paseo
Fernández	14	141414	Zapillo
Urrutia	15	151515	Sol

La relación Empleados

numeroCta	saldo	nombreSuc

nombreCli	dniCli	Domicilio

1	100	Castellana
2	200	Castellana
3	300	Paseo
4	400	Paseo
5	500	Zapillo
6	600	Sol

La relación Cuentas

Aranda	1	La Reina nº7
García	2	Fragata azul nº8
Hayes	3	Gibraltar español nº14
Turner	4	Gibraltar español nº17
Vilches	5	Diamante S/N
Lara	6	Gato negro nº13
Guerrero	7	Perro nº1

La relación Clientes

dniCli	numeroCta
1	1
1	2
2	3
3	4
4	5
5	5
6	5
7	6

La relación CtaCli

numeroCta	numeroTrans	fecha	importe
1	1	10-10	+100
2	1	10-10	+300
2	2	11-10	-200
3	1	12-10	+300
4	1	12-10	+400
5	1	13-10	+500
6	1	13-10	+600

La relación Transacciones

Los atributos que representan el mismo concepto del mundo real pueden tener o no el mismo nombre en relaciones distintas, y viceversa.

Las restricciones de integridad se especifican en el esquema de una base de datos, y además de las restricciones de dominio y clave, incluyen las resatricciones de integridad de entidades e integridad referencial.

3.3.4. Integridad de entidades, integridad referencial y claves externas.

La restricción de *integridad de entidades* establece que ningún valor de clave primaria puede ser nulo. Esto se debe a que el valor de la clave primaria sirve para identificar las tuplas de una relación, y si la clave primaria puede tener valores nulos, no podríamos identificar algunas tuplas.

La restricción de *integridad referencial* se especifica entre dos relaciones, y establece que en una tupla de una relación que haga referencia a otra relación, deberá referirse a una *tupla existente* en dicha relación. Por ejemplo, el atributo *nombreSuc* de *empleados* indica la sucursal en que trabaja un empleado, y su valor deberá coincidir con el valor de *nombreSuc* en alguna tupla de la relación *sucursales*.

Para completar la definición de integridad referencial, debemos definir el concepto de *clave externa*. Un conjunto de atributos CE del esquema de la relación R_1 es una clave externa de R_1 si satisface estas condiciones:

1. Los atributos de CE tienen el mismo dominio que los atributos de la clave primaria CP de otro esquema de relación R_2; se dice que los atributos CE hacen referencia o se refieren a la relación R_2.
2. Un valor de CE en una tupla t_1 del estado actual de R_1 es el valor de CP en alguna tupla t_2 del estado actual de R_2, o bién es nulo. Si no es nulo, diremos que la tupla t_1 hace referencia a la tupla t_2. R_1 será la *relación referenciante*, y R_2 la *relación referenciada*.

En una base de datos con muchas relaciones, suele haber muchas restricciones de integridad referencial, ya que surgen de las relaciones representadas entre las entidades representadas por los esquemas de relación. Una clave externa también puede hacer referencia a su propia relación (caso de supervisor y empleado).

Las restricciones de integridad referencial pueden representarse gráficamente trazando un arco desde la clave externa hacia la relación a la que hace referencia, con la flecha apuntando hacia el atributo o atributos referenciados. Deben expresarse en el esquema de la base de datos, para que se mantengan automáticamente.

Las restricciones vistas no incluyen otra clase de restricciones llamadas *restricciones de integridad semántica*, tales como "un alumno no puede estar matriculado de más de 80 créditos", que necesitan un lenguaje de especificación de restricciones de propósito general. Para ello, se usan *disparadores* o *triggers* y *aserciones*. También hay otras restricciones llamadas *restricciones de transición*, para tratar con cambios de estado de la base de datos ("el sueldo de un empleado sólo puede incrementarse"). Estas restricciones se especifican con reglas de actividad y disparadores.

3.4. Operaciones de actualización y tratamiento de la violación de restricciones

Las operaciones del modelo relacional pueden clasificarse en *recuperaciones* y *actualizaciones*. En esta sección, vamos a ver las operaciones de actualización básicas, que son tres: Insertar (insertar una o más tuplas nuevas en una relación), Eliminar (elimina tuplas en una relación) y Actualizar (modifica los valores de algunos atributos de tuplas existentes). Cuando se realizan estas operaciones, no se deben violar las restricciones de integridad.

3.4.1. Insertar

Para insertar datos en una relación, bien especificamos la lista de valores de la tupla que se va a insertar o escribimos una consulta cuyo resultado sea el conjunto de tuplas que se va a insertar.

Para insertar una nueva transacción en la cuenta 6 de 700 € realizada el 7-11-2003, escribiríamos lo siguiente:

Insertar < "6", “2”, "7-11-2003", 700> en *transacciones*.

Al insertar, puede violarse una restricción de dominio si se proporciona un valor de atributo que no pertenece al dominio correspondiente. También puede violarse una restricción de clave si la clave de la nueva tupla ya existe en la relación. La integridad de entidades puede violarse si la clave primaria de la nueva tupla es nula. Y la integridad referencial puede violarse si el valor de cualquier clave externa de la nueva tupla hace referencia a una tupla que no existe en la relación referenciada.

Cuando una inserción viola una o más restricciones, la opción predeterminada es *rechazar* la inserción.

3.4.2. Eliminar

La operación Eliminar sólo puede violar restricciones de integridad referencial, cuando las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se va a eliminar. Para especificar la información a eliminar, se expresa una condición en términos de los atributos de la relación correspondiente.

Por ejemplo, para eliminar todas las transacciones de la cuenta número 1 escribiríamos:

Eliminar en *transacciones* las tuplas con *numeroCta=”*1*”*

Cuando la eliminación provoca una violación de restricciones de integridad referencial, tenemos tres opciones. La primera es *rechazar* la eliminación. La segunda es la *eliminación en cascada*, eliminando también las tuplas que hacen referencia a la tupla o tuplas que se desean eliminar. Una tercera opción es *modificar* los valores del atributo de referencia que provocan la violación: estos valores se pondrían a nulo o se modificarían para hacer referencia a otra tupla válida. La opción seleccionada en cada caso debe poder especificarse también en el esquema de la base de datos.

3.4.3. Actualizar

La operacion Actualizar modifica los valores de uno o más atributos de una o más tuplas de una relación. Es necesario especificar una condición sobre los atributos de la relación para seleccionar la tupla o tuplas a modificar.

Por ejemplo, para cambiar el domicilio del cliente con dniCli 1 a C/Real nº 14, escribiríamos:

Actualizar *domicilio* de *clientes* con *dniCli* = "1"

La actualización de un atributo que no es clave primaria ni clave externa no suele dar problemas (excepto de dominio). Modificar un valor de la clave primaria es similar a eliminar una tupla e insertar otra en su lugar, por lo que pueden darse los problemas vistos para inserciones y eliminaciones. Si se modifica un atributo de una clave externa, hay que comprobar que el nuevo valor existe en la relación referenciada (o es nulo).

3.5. Lenguajes de consulta

Un *lenguaje de consulta* es un lenguaje en el que el usuario puede solicitar información de la base de datos. Estos lenguajes de consulta suelen ser de más alto nivel que los lenguajes de programación estándar, y pueden clasificarse en procedimentales y no procedimentales.

En un *lenguaje procedimental* el usuario da instrucciones al sistema para que realice una serie de operaciones sobre la base de datos con el fin de obtener el resultado deseado.

En un *lenguaje no procedimental* el usuario describe la información deseada sin dar un procedimiento concreto sobre cómo obtener esa información.

En este tema estudiaremos dos lenguajes de consulta formales (no comerciales), como son el *álgebra relacional*, que es un lenguaje de consulta procedimental, y el *cálculo relacional*, un lenguaje de consulta no procedimental.

3.6. Álgebra relacional

El álgebra relacional es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman una o dos relaciones como entrada y producen una nueva relación como salida.

Las operaciones fundamentales del álgebra relacional son:

- Selección
- Proyección
- Producto cartesiano
- Renombrar
- Unión
- Diferencia de conjuntos

3.6.1. Operaciones fundamentales del álgebra relacional

3.6.1.1. Selección

La operación de *selección* selecciona tuplas que satisfacen un predicado determinado. Usaremos la letra griega sigma (σ) para indicar la selección. El predicado aparece como subíndice de σ.

Por tanto, para seleccionar las tuplas de la relación *Sucursales* que representan sucursales que están en la ciudad "Madrid", escribimos:

$$\sigma_{ciudadSuc = "Madrid"}(sucursales)$$

y la relación resultante es la siguiente:

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000

Sol	Madrid	71000

Podemos encontrar todas las tuplas en las que el activo de la sucursal sea mayor que 50.000 € escribiendo:

$\sigma_{activo > 50000}$ *(sucursales)*

con lo que obtendríamos

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000
Ronda	Almería	80000
Sol	Madrid	71000

En general se permiten las comparaciones utilizando los operadores =, ≠, <, ≤, > y ≥ en el predicado de selección. Además, pueden combinarse predicados para formar un predicado compuesto utilizando los conectores lógicos *y* (∧) y *o* (∨). Por tanto, para encontrar las tuplas de la relación *sucursales* de las sucursales de "Madrid" con un activo superior a 80000, escribiríamos:

$\sigma_{ciudadSuc = "Madrid" \wedge activo > 80000}$ *(sucursales)*

lo que nos devolvería la relación

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000

3.6.1.2. Proyección

La operación de *proyección* es una operación unaria que devuelve su relación argumento con ciertas columnas omitidas. Puesto que la relación es un conjunto, devolvería todas las filas duplicadas. La proyección se denota por la letra griega pi (Π), y los atributos que se desea obtener en el resultado se ponen como subíndices de Π.

Para conocer el nombre y teléfono de los empleados del banco escribiríamos:

$\Pi_{nombreEmp, telefono}$ *(empleados)*

y obtendríamos la siguiente relación:

nombreEmp	telefono
García	101010
Torres	111111
López	121212
Villegas	131313
Fernández	141414
Urrutia	151515

3.6.1.3 Secuencias de operaciones

En general, es posible aplicar varias operaciones del álgebra relacional del mismo tipo una tras otra, pero también es posible agrupar varias operaciones diferentes en

una sola expresión del álgebra relacional, y esto se hace anidándolas. Esta característica se debe a que el resultado de aplicar a una relación una operación del álgebra relacional es otra relación.

Para conocer el nombre y el teléfono de los empleados que trabajan en la sucursal de "Castellana" escribiríamos:

$$\Pi_{\text{nombreEmp, telefono}} (\sigma_{\text{nombreSuc = "Castellana"}}(\text{sucursales}))$$

lo que daría lugar a la siguiente relación:

nombreEmp	telefono
García	101010
Torres	111111

Obsérvese como en lugar de pasar una relación "original" a la operación de proyección, se ha pasado una expresión, que se evalúa como relación.

3.6.1.4. Unión

La operación unión se limita a unir resultados de relaciones, y funciona de la misma forma que la unión de conjuntos, con la observación de que las relaciones que se unan han de tener los mismos atributos, y que se eliminan las tuplas repetidas.

Supongamos que queremos conocer todos los nombres de personas relacionadas con el sistema bancario, ya sea que trabajen en el banco, o que tengan una cuenta en el banco. Para esto necesitamos obtener información de la relación *empleados* y de la relación *clientes*.

Para conocer los nombres de los clientes, escribiríamos:

$\Pi_{nombreCli}(clientes)$

Para conocer los nombres de los empleados, escribiríamos:

$\Pi_{nombreEmp}(empleados)$

Luego para contestar a la consulta planteada de cuáles son los nombres de todas las personas relacionadas con el sistema bancario deberíamos unir las dos relaciones resultantes, mediante el operador unión, para que aparecieran los nombres de los clientes y los nombres de los empleados, por lo que tendríamos que escribir lo siguiente:

$\Pi_{nombreEmp}(empleados) \cup \Pi_{nombreCli}(clientes)$

El resultado de esta consulta sería el siguiente, teniendo en cuenta que las tuplas repetidas se eliminan.

NombreEmp
García
Torres
López
Villegas
Fernández

Urrutia
Aranda
García
Hayes
Turner
Vilches
Lara
Guerrero

En general, debemos asegurarnos que las uniones se realizan entre relaciones compatibles, es decir, que tengan el mismo número de atributos y que los dominios de los atributos sean iguales dos a dos.

3.6.1.5. Diferencia de conjuntos

La operación de *diferencia de conjuntos*, representada por el símbolo -, nos permite encontrar las tuplas que estén en una relación pero no estén en otra. La expresión *R - S* da como resultado la relación que contiene las tuplas que están en *R* pero que no están en *S*.

Como ejemplo, la consulta que nos daría el nombre de los clientes que no son empleados sería:

$\Pi_{nombreCli}(clientes) - \Pi_{nombreEmp}\ (empleados)$

El resultado de la consulta sería la siguiente relación:

nombreCli
Aranda
Hayes
Turner
Vilches
Lara
Guerrero

3.6.1.6 Producto cartesiano

Las operaciones de selección y proyección vistas hasta ahora sólo permiten extraer información de una relación a la vez. Pues bien, para poder combinar varias relaciones se puede utilizar la operación de *producto cartesiano*, cuyo símbolo es una equis (*x*). El resultado de la operación de producto cartesiano es una nueva relación que tiene tantas columnas como la suma del número de columnas de cada relación, y las tuplas de esta nueva relación se obtienen mendiante la combinación de todas las tuplas de las relaciones que participan en la operación de producto cartesiano.

A la hora de realizar la operación de producto cartesiano, hay que tener en cuenta que como se está haciendo el producto de las relaciones se puede generar un resultado bastante grande, por lo que siempre tendremos que tener claro, antes de escribir la operación, si existe un predicado que debamos considerar, especialmente

cuando lo que queramos realizar sea relacionar las tablas. Para hacer referencia a los nombres de las columnas de forma que se eviten posibles ambigüedades, se han de especificar los nombres de las tablas seguidos de un punto y el nombre de los atributos, para determinar con exactitud a qué atributo de qué relación estamos haciendo referencia. A continuación se ilustra la relación que resulta de hacer el producto cartesiano *Empleados x Sucursales*

Empleados.nombreEmp	Empleados.dniEmp	Empleados.telefono	Empleados.nombreSuc	Sucusales.nombreSuc	Sucursales.ciudadSuc	Sucursales.activo
García	10	101010	Castellana	Castellana	Madrid	90000
Torres	11	111111	Castellana	Castellana	Madrid	90000
López	12	121212	Paseo	Castellana	Madrid	90000
Villegas	13	131313	Paseo	Castellana	Madrid	90000
Fernández	14	141414	Zapillo	Castellana	Madrid	90000
Urrutia	15	151515	Sol	Castellana	Madrid	90000
García	10	101010	Castellana	Ganivet	Granada	21000
Torres	11	111111	Castellana	Ganivet	Granada	21000
López	12	121212	Paseo	Ganivet	Granada	21000
Villegas	13	131313	Paseo	Ganivet	Granada	21000
Fernández	14	141414	Zapillo	Ganivet	Granada	21000
Urrutia	15	151515	Sol	Ganivet	Granada	21000
García	10	101010	Castellana	Paseo	Almería	17000
Torres	11	111111	Castellana	Paseo	Almería	17000
López	12	121212	Paseo	Paseo	Almería	17000
Villegas	13	131313	Paseo	Paseo	Almería	17000
Fernández	14	141414	Zapillo	Paseo	Almería	17000
Urrutia	15	151515	Sol	Paseo	Almería	17000
García	10	101010	Castellana	Zapillo	Almería	4000
Torres	11	111111	Castellana	Zapillo	Almería	4000
López	12	121212	Paseo	Zapillo	Almería	4000
Villegas	13	131313	Paseo	Zapillo	Almería	4000
Fernández	14	141414	Zapillo	Zapillo	Almería	4000
Urrutia	15	151515	Sol	Zapillo	Almería	4000
García	10	101010	Castellana	Ronda	Almería	80000
Torres	11	111111	Castellana	Ronda	Almería	80000
López	12	121212	Paseo	Ronda	Almería	80000
Villegas	13	131313	Paseo	Ronda	Almería	80000
Fernández	14	141414	Zapillo	Ronda	Almería	80000
Urrutia	15	151515	Sol	Ronda	Almería	80000
García	10	101010	Castellana	Aduana	Roquetas	3000
Torres	11	111111	Castellana	Aduana	Roquetas	3000
López	12	121212	Paseo	Aduana	Roquetas	3000
Villegas	13	131313	Paseo	Aduana	Roquetas	3000
Fernández	14	141414	Zapillo	Aduana	Roquetas	3000
Urrutia	15	151515	Sol	Aduana	Roquetas	3000
García	10	101010	Castellana	Los Pinos	Huercal	37000
Torres	11	111111	Castellana	Los Pinos	Huercal	37000

Empleados.nombreEmp	Empleados.dniEmp	Empleados.telefono	Empleados.nombreSuc	Sucusales.nombreSuc	Sucursales.ciudadSuc	Sucursales.activo
López	12	121212	Paseo	Los Pinos	Huercal	37000
Villegas	13	131313	Paseo	Los Pinos	Huercal	37000
Fernández	14	141414	Zapillo	Los Pinos	Huercal	37000
Urrutia	15	151515	Sol	Los Pinos	Huercal	37000
García	10	101010	Castellana	Sol	Madrid	71000
Torres	11	111111	Castellana	Sol	Madrid	71000
López	12	121212	Paseo	Sol	Madrid	71000
Villegas	13	131313	Paseo	Sol	Madrid	71000
Fernández	14	141414	Zapillo	Sol	Madrid	71000
Urrutia	15	151515	Sol	Sol	Madrid	71000

Como se puede observar en la relación resultante del producto cartesiano anterior, el número de tuplas de la relación producto cartesiano se dispara, debido a que se realiza la combinación de todas las tuplas de una relación con todas las tuplas de las relaciones que también participen en el producto cartesiano. Sin embargo, no todas las tuplas que hay en dicha relación parecen tener un significado lógico. Este significado lógico se refiere a que cuando se quieren combinar varias relaciones, uno de los resultados buscados es el de obtener sólo las tuplas en las que coinciden los valores de los atributos que establecen la relación o relaciones. Para ello, deberíamos de aplicar una selección al resultado del producto cartesiano de forma que se seleccionasen únicamente esas tuplas. Ilustremos esto con un ejemplo.

Si en la relación anterior aplicamos el predicado que selecciona sólo las tuplas en las que coincide el nombre de sucursal, estaremos combinando para cada empleado los datos de la sucursal en la que trabaja dicho empleado, descartando las tuplas en las que se combinan los datos de los empleados con datos de las sucursales en las que no trabajan. Para ello, escribiríamos la siguiente expresión en álgebra relacional

$$\sigma_{empleados.nombreSuc = sucursales.nombreSuc}\ (sucursales\ x\ empleados)$$

dando lugar a este resultado.

Empleados.nombreEmp	Empleados.dniEmp	Empleados.telefono	Empleados.nombreSuc	Sucusales.nombreSuc	Sucursales.ciudadSuc	Sucursales.activo
García	10	101010	Castellana	Castellana	Madrid	90000
Torres	11	111111	Castellana	Castellana	Madrid	90000
López	12	121212	Paseo	Paseo	Almería	17000
Villegas	13	131313	Paseo	Paseo	Almería	17000
Fernández	14	141414	Zapillo	Zapillo	Almería	4000
Urrutia	15	151515	Sol	Sol	Madrid	71000

A este resultado podemos seguir aplicándole operaciones, como por ejemplo una proyección para obtener una nueva relación con un conjunto específico de columnas. Por ejemplo, para conocer los nombres de los empleados que trabajan en las sucursales de "Madrid" podríamos realizar el producto cartesiano de *empleados* y

sucursales, aplicarles el predicado de selección que filtre las tuplas en las que coinciden los nombres de sucursal, y realizar posteriormente una proyección sobre los nombres de empleado. La expresión correspondiente en álgebra relacional sería la siguiente:

$\Pi_{empleados.nombreEmp}\ (\sigma_{empleados.nombreSuc\ =\ sucursales.nombreSuc\ \wedge\ sucursales.ciudadSuc\ =\ \text{“Madrid”}}$ *(sucursales x empleados))*

No obstante, esto también se podría hacer alterando el orden en que se realizan estas operaciones, de forma que se seleccionasen las tuplas de la relación *sucursales* donde el atributo *ciudadSuc* sea "Madrid", se hiciera el producto cartesiano de este resultado parcial con la relación *cliente* cumpliendo el predicado de que el nombre de la sucursal sea el mismo, y finalmente realizar una proyección sobre el atributo *nombreEmp*. Así pues, para esta segunda alternativa tendríamos que escribir lo siguiente:

$\Pi_{empleados.nombreEmp}\ (\sigma_{sucursales.nombreSuc\ =\ empleados.nombreSuc}\ (\sigma_{sucursales.ciudadSuc\ =\ \text{"Madrid"}}$ *(sucursales x empleados)))*

En cualquier caso, el resultado de aplicar estas relaciones sería la siguiente relación

Empleados.nombreEmp
García
Torres
Urrutia

3.6.1.7. Renombrar

Las consultas en las que aparece varias veces una misma relación pueden originar ambigüedades, que necesitamos resolver de algún modo.

Imaginemos que deseamos conocer los nombres de los empleados que trabajan en la misma sucursal que *García*.

Para obtener el nombre de la sucursal donde trabaja García, escribiríamos

$\Pi_{nombreSuc}\ (\sigma_{nombreEmp\ =\ \text{"García"}}$*(empleados))*

Sin embargo, para encontrar otros empleados que trabajen en esa misma sucursal debemos volver a hacer referencia a la relación empleados

$\sigma_P\ x\ \Pi_{nombreSuc}\ (\sigma_{nombreEmp\ =\ \text{"García"}}$*(empleados))*

donde *P* es un predicado de selección que hace que los valores de *nombreSuc* sean iguales a los de García. Para especificar a qué valor de *nombreSuc* estamos haciendo referencia, no podemos utilizar *empleados.nombreSuc*, ya que se estarían tomando de la relación *empleados*, y no el del resultado parcial.

Esto se soluciona mediante el operador *renombrar* (ρ). La expresión

$\rho_X(R)$

devuelve la relación *R* con el nombre *X*. De esta forma podemos hacer referencia varias veces a la misma relación sin ambigüedad.

En la consulta que estamos intentando resolver, utilizaremos el nombre *empleados2* para renombrar la relación *empleados*. Por tanto la consulta quedaría de la forma siguiente:

$$\Pi_{empleados.nombreEmp}(\sigma_{empleados2.nombreSuc = empleados.nombreSuc}(empleados\ x$$

$$\rho_{empleados2}(\Pi_{nombreSuc}(\sigma_{nombreEmp = "García"}(empleados)))))$$

El resultado de la consulta sería

nombreEmp
García
Torres

3.6.2. Otras operaciones del álgebra relacional

3.6.2.1 Intersección de conjuntos

Se trata de una operación adicional del álgebra de conjuntos, puesto que la operación de *intersección* $\cap$, se puede expresar como dos diferencias de conjuntos de esta forma:

$$R \cap S = R - (R - S)$$

Para conocer el nombre de los empleados que tienen cuenta en alguna de las sucursales del banco, escribiríamos lo siguiente:

$$\Pi_{nombreCli}(clientes) \cap \Pi_{nombreEmp}(empleados)$$

El resultado de la consulta sería la siguiente relación:

nombreCli
García

3.6.2.2 Producto theta

Se trata de una operación para simplificar las consultas que utilizan la operación de producto cartesiano, y que permite especificar la condición de selección de registros como subíndice de dicha operación. Se representa mediante $x_{\Theta,}$ siendo Θ subíndice el predicado que selecciona las tuplas deseadas, que pueden ser las que tienen valores coincidentes de campos comunes o no.

Como ejemplo, supongamos que estamos interesados en obtener el nombre de los clientes que tienen una cuenta en la sucursal de "Castellana". Con lo que sabemos hasta ahora, haríamos el producto cartesiano de las relaciones *clientes, cuentas* y *ctacli*, luego seleccionaríamos de esta relación las tuplas en las que coincidiesen los DNI de los clientes y los números de cuenta, y por último haríamos una proyección para filtrar los atributos que deseamos. Todo esto lo podríamos escribir de la forma siguiente:

$\Pi_{nombreCli}\ (\sigma_{cuentas.numeroCta\ =\ ctacli.numeroCta\ \wedge\ ctacli.dniCli\ =\ clientes.dniCli}\ (\sigma_{cuentas.nombreSuc\ =\ "Castellana"}\ (clientes\ x\ ctacli\ x\ cuentas)))$

Pues bien, con la operación de producto theta, la consulta se escribiría de la forma siguiente:

$\Pi_{nombreCli}\ ((\sigma_{cuentas.nombreSuc\ =\ "Castellana"}\ (clientes\ x_{clientes.dniCliente=cuentacli.dniCliente}\ cuentacli\ x_{cuentacli.numeroCta=cuentas.numeroCta}\ cuentas)$

El resultado de la consulta sería el siguiente:

nombreCli
Aranda

3.6.2.3 Producto natural

Se trata de una operación para simplificar las consultas que utilizan la operación de producto cartesiano o producto theta, y que permite obviar la especificación de condición de selección de registros como subíndice de dicha operación. Se representa mediante |x|, y hace directamente el producto cartesiano y la selección de las tuplas deseadas, que serán las que tienen valores coincidentes de campos comunes.

Volvamos al ejemplo anterior, en el que deseamos obtener el nombre de los clientes que tienen una cuenta en la sucursal de "Castellana". Con lo que sabemos hasta ahora, haríamos el producto cartesiano de las relaciones *clientes, cuentas* y *ctacli*, y seleccionaríamos de esta relación las tuplas en las que coincidiesen los DNI de los clientes y los números de cuenta, y por último haríamos una proyección para filtrar los atributos que deseamos. Con el producto natural, lo podríamos escribir de la forma siguiente:

$\Pi_{nombreCli}\ ((\sigma_{cuentas.nombreSuc\ =\ "Castellana"}\ (clientes\ |x|_{clientes.dniCliente=cuentacli.dniCliente}\ cuentacli\ |x|_{cuentacli.numeroCta=cuentas.numeroCta}\ cuentas)$

El resultado de la consulta sería el siguiente:

nombreCli
Aranda

3.6.2.4. División

La operación de *división*, cuyo símbolo es ÷, es adecuada para consultas que incluyen la expresión "todos". Suponga que desea saber cuáles son los clientes que tienen cuentas en todas las sucursales de Granada. En primer lugar, podemos obtener los nombres de todas las sucursales de Granada, con la expresión

$\Pi_{nombreSuc}\ (\sigma_{ciudadSuc\ =\ "Granada"}\ (sucursales))$

A continuación, podemos obtener los nombres de las sucursales en las que tienen cuenta todos los clientes:

$\Pi_{nombreCli,nombreSuc}(clientes\ |x|\ cuentacli\ |x|\ cuentas)$

Por último, necesitamos ver los clientes que aparecen en la segunda relación con todos los nombres de sucursales de la primera.

Formalmente, sean *r(R)* y *s(S)* dos relaciones, tales que $S \subseteq R$ (todos los atributos del esquema *S* están incluidos en el esquema *R*). La relación $r \div s$ es una relación con el esquema $R - S$ (atributos de *R* que no están en *S*). Una tupla *t* pertenece a $r \div s$ si y sólo si se cumplen estas dos condiciones:

1. *t* pertenece a Π_{R-S} (r)
2. Para cada tupla t_S de *s* existe una tupla t_r de *r* que satisface que

$$t_r[S] = t_s[S] \quad \text{y} \quad t_r[R - S] = t$$

3.6.2.5. Asignación

La operación de asignación, representada por ←, funciona de forma similar a la asignación en un lenguaje de programación.

La evaluación de una asignación no da como resultado una relación que se presenta al usuario, sino que el resultado de la expresión que está a la derecha de ← se asigna a la variable de relación situada a la izquierda de ←.

3.6.2.6. Extensiones del producto natural

A continuación estudiaremos algunas extensiones del producto natural y de la unión, dada su creciente aparición en los SGBD comerciales.

Recordemos que en la operación de producto natural $R \mid x \mid S$, se creaba una relación con las tuplas de *R* que tienen tuplas coincidentes en *S* y viceversa. Por tanto, si una tupla de una relación no encuentra otra "tupla relacionada" en la otra relación, no aparece en la relación resultante.

Para evitar esta discriminación de tuplas existen unas operaciones denominadas genéricamente *uniones externas* (outer joins), y aquí estudiaremos la Unión externa izquierda y la Unión externa.

Unión externa izquierda (Left Join)

Se denota por el símbolo =x. R =x S conserva todas las tuplas de la primera relación (la relación de la izquierda), de forma que si no se encuentran tuplas coincidentes en *S* se rellenan con valores nulos.

Imaginemos que deseamos saber los nombres de sucursal de la ciudad de "Almería", junto con los datos de los empleados que trabajan en dichas sucursales. Los datos de las sucursales son necesarios aunque no existan datos sobre los empleados. Para ello, haríamos una combinación de una operación de proyección, selección y un left join de esta forma.

$$\Pi_{sucursales.nombreSuc,empleados.nombreEmp}($$

$$\sigma_{ciudadSuc="Almería"}(sucursales) =x_{\ sucursales.nombreSuc=empleados.nombreSuc}\ empleados)$$

Así pues, el resultado sería el siguiente:

Sucursales.nombreSuc	Empleados.nombreEmp
Paseo	López
Paseo	Villegas
Zapillo	Fernández
Ronda	nulo

Análogamente, existe una operación denominada Unión externa derecha, denotada por x=, que incluye en la relación *R* x= *S* todas las tuplas de *S* (la de la derecha) aunque no tengan tuplas con valores coincidentes en la relación de la izquierda.

Unión externa

Se trata de una unión que une dos relaciones que no sean compatibles, es decir, que no se les pueda aplicar el operador de unión, ya que sólo algunos de los atributos son compatibles con la unión.

En el sistema bancario imaginemos que deseamos comocer los nombres, DNI, teléfonos y dirección de todas las personas implicadas en el sistema bancario (clientes y empleados). Para ello, haríamos lo siguiente:

$$\Pi_{nombreEmp,\ dniEmp,\ telefono}\ (empleados) =x= \Pi_{nombreCli,\ dniCli,\ domicilio}\ (clientes)$$

y el resultado sería

nombreEmp	dniEmp	telefono	domicilio
García	10	101010	nulo
Torres	11	111111	nulo
López	12	121212	nulo
Villegas	13	131313	nulo
Fernández	14	141414	nulo
Urrutia	15	151515	nulo
Guerrero	7	777777	Perro nº1
Aranda	1	nulo	La Reina nº7
García	2	nulo	Fragata azul nº8
Hayes	3	nulo	Gibraltar español nº14
Turner	4	nulo	Gibraltar español nº17
Vilches	5	nulo	Diamante S/N
Lara	6	nulo	Gato negro nº13

3.7. Cálculo relacional de tuplas

Cuando escribimos una expresión en álgebra relacional, damos la secuencia de operaciones que generan la respuesta a la consulta. Sin embargo, el cálculo relacional es un lenguaje no procedimental, en el que se escribe una expresión declarativa para especificar una solicitud de recuperación de datos. Ambos lenguajes son equivalentes, en el sentido de que toda consulta expresada en uno de ellos puede

traducirse al otro. En esta sección veremos muy brevemente la creación de consultas simples en cálculo relacional de tuplas

3.7.1. Variables de tupla y relaciones de rango

El cálculo relacional de tuplas se basa en construir expresiones en las que se especifican una o más *variables de tupla*. Generalmente, cada variable se asocia con una relación de una base de datos, lo que implica que la variable puede tomar como valor cualquier tupla individual de dicha relación. Una consulta en el cálculo relacional de tuplas se expresa como

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

es decir, el conjunto de todas las tuplas t, en las que el predicado P en el que interviene t es verdadero. Siguiendo la notación utilizada anteriormente, utilizaremos $t.A$, para referirnos al atributo A de la tupla t. Antes de continuar con el cálculo relacional, veamos el aspecto de una expresión sencilla en cálculo relacional.

Para encontrar las tuplas de las sucursales de la ciudad de "Madrid" escribiríamos la siguiente expresión del cálculo relacional:

$\{s \mid s \in sucursal$ **and** $s.ciudadSuc = "Madrid"\}$

La condición $s \in sucursal$ especifica que la *relación de rango* de la variable de tupla t es *sucursal*.

Para las operaciones de proyección no pediremos tuplas completas, sino sólo la parte que nos interese, es decir, los atributos que necesitamos. Para encontrar los nombres de las sucursales de la ciudad de "Madrid" escribiríamos la siguiente expresión del cálculo relacional:

$\{s.nombreSuc \mid s \in sucursal$ **and** $s[ciudadSuc] = "Madrid"\}$

3.7.2. Expresiones y fórmulas en el cálculo relacional de tuplas

Una *expresión* general del cálculo relacional toma la forma

$$\{t_1.A_1, t_2.A_{2,\ldots,} t_n.A_n \mid P(t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m})\}$$

donde $t_1, t_2, \ldots, t_n, t_{n+1}, t_{n+2}, \ldots, t_{n+m}$ son variables de tupla, cada A_i es un atributo de la relación asociada a t_i, y P es un *predicado*, *condición* o *fórmula* del cálculo relacional de tuplas. Una fórmula consta de *átomos* del cálculo de predicados, que pueden ser los siguientes:

1. Un átomo $t_i \in R$, donde R es un nombre de relación y t_i una variable de tupla.
2. Un átomo $t_i.A$ ***op*** $t_j.B$, donde ***op*** es un operador de comparación del conjunto $\{=, \neq, <, \leq, >, \geq\}$, t_i y t_j son variables de tupla y A y B son atributos de las relaciones de rango correspondientes.
3. Un átomo $t_i.A$ ***op*** c, donde ***op*** es un operador de comparación del conjunto $\{=, \neq, <, \leq, >, \geq\}$, t_i es una variable de tupla, A es un atributo de la relación de rango de t_i y c es un valor constante.

Una fórmula consta de uno o más átomos conectados mediante los operadores lógicos **and**, **or** y **not**.

3.7.3. Uso de cuantificadores

Hay dos símbolos especiales, llamados *cuantificadores*, que pueden aparecer en las fórmulas; estos son el *cuantificador universal* ($\forall$) y el *cuantificador existencial* ($\exists$). El cuantificador ($\exists$) se llama cuantificador existencial porque una fórmula ($\exists t$) (F) es verdadera si *existe* alguna tupla t que hace que F sea verdadera. El cuantificador ($\forall$) se llama cuantificador universal porque una fórmula ($\forall t$) (F) es verdadera si *todas* las tuplas que pueden sustituir a t hacen que F sea verdadera. Diremos que una variable de tupla t está *ligada* si está cuantificada, es decir, si aparece en una cláusula ($\exists t$) o ($\forall t$). En caso contrario, diremos que la variable está *libre*.

Para ilustrar el uso del cuantificador existencial, supongamos que deseamos conocer los nombres de los empleados que trabajan en alguna sucursal de la ciudad de Madrid. La expresión de cálculo relacional siguiente da respuesta a esta consulta:

$\{e.nombreEmp \mid e \in empleados$ **and** $e.ciudadSuc = "Madrid"$ **and**

$(\exists s)\ (s \in sucursales$ **and** $s.nombreSuc = e.nombreSuc)\}$

Para ver el uso del cuantificador universal, supongamos que deseamos conocer los nombres de los clientes que tienen cuenta en todas las sucursales de Madrid. La siguiente expresión del cálculo relacional obtiene esta información:

$\{t \mid (\exists c)\ (c \in clientes$ **and** $c.nombreCli = t.nombreCli)$ **and** $((\forall u)\ (u \in sucursal$ **and** $u.ciudadSuc =$ "Madrid" **and** $((\exists s)\ (s \in clientes$ **and** $t.nombreCli = s.nombreCli$ **and** $((\exists u)\ (u \in ctacli$ **and** $s.dniCli = u.dniCli$ **and** $((\exists w)\ (w \in cuentas$ **and** $w.numeroCta = u.numeroCta)))))))\}$

La primera condición es necesaria para protegerse del caso de que no haya ninguna sucursal en Madrid, ya que en este caso cualquier valor de t sería válido (incluyendo valores que no son nombres de clientes de la relación *clientes*).

3.8. Cálculo relacional de dominios

En el cálculo relacional de dominios, las variables representan atributos en lugar de tuplas. Lo que en cálculo relacional de tuplas se denota como $t.A_1$ es ahora A_1. Por tanto, si queremos manejar una tupla tenemos que nombrar todas las variables correspondientes a los atributos de esa tupla ($< x_1, x_2, ..., x_n >$). Por ejemplo, una tupla de *cuentas* se expresaría de la siguiente forma: $< c, a, s >$ con:

$c = numeroCta$; $a = Saldo$; $s = nombreSuc$

En el cálculo relacional de dominios, la expresión general tendrá la forma:

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

Veamos como se llevarían a cabo las distintas operaciones en este lenguaje por medio de ejemplos:

Selección: Queremos los números de cuenta, saldos y sucursales de las cuentas cuyo saldo es superior a 600 €:

$\{ < c, a, s > | < c, a, s > \in \textit{cuentas}$ **and** $a > 600 \}$

Proyección: Utilizaremos el cuantificador $\exists$. Por ejemplo, si sólo queremos los números de cuenta y sucursales de cuentas con saldo superior a 600 €, usaríamos esta expresión

$\{ < c, s > | (\exists\, c, a, s\,) (< c, a, s > \in \textit{cuentas}$ **and** $a > 600) \}$