Bases de Datos I
29/09/2020
Resumen Cap 5 y 6 29/09
IS-501
José Inestroza

Capítulo 5 Fundamentals
El modelo de datos relacionales y las restricciones de las bases de datos relacionales

Este capítulo abre la Parte 3 del libro, que cubre las bases de datos relacionales. El modelo de datos relacionales fue introducido por primera vez por Ted Codd de IBM Research en 1970 en un artículo clásico (Codd, 1970), y atrajo atención inmediata debido a su simplicidad y base matemática. El modelo utiliza el concepto de relación matemática, que se parece un poco a una tabla de valores, como su bloque de construcción básico y tiene su base teórica en la teoría de conjuntos y la lógica de predicados de primer orden. En este capítulo discutimos las características básicas del modelo y sus limitaciones.
Las primeras implementaciones comerciales del modelo relacional estuvieron disponibles a principios de la década de 1980, como el sistema SQL / DS en el sistema operativo MVS de IBM y Oracle DBMS. Desde entonces, el modelo se ha implementado en una gran cantidad de sistemas comerciales, así como en varios sistemas de código abierto. Los DBMS relacionales (RDBMS) comerciales populares actuales incluyen DB2 (de IBM), Oracle (de Oracle), Sybase DBMS (ahora de SAP) y SQLServer y Microsoft Access (de Microsoft). Además, se encuentran disponibles varios sistemas de código abierto, como MySQL y PostgreSQL.
Debido a la importancia del modelo relacional, toda la Parte 2 está dedicada a este modelo y algunos de los lenguajes asociados con él. En los capítulos 6 y 7, describimos algunos aspectos de SQL, que es un modelo y lenguaje integrales que es el estándar para los DBMS relacionales comerciales. (Los aspectos adicionales de SQL se tratarán en otros capítulos.) El capítulo 8 cubre las operaciones del álgebra relacional e introduce el cálculo relacional; estos son dos lenguajes formales asociados con el modelo relacional. Se considera que el cálculo relacional es la base del lenguaje SQL, y el álgebra relacional se usa en los aspectos internos de muchas implementaciones de bases de datos para el procesamiento y la optimización de consultas (consulte la Parte 8 del libro).
Otras características del modelo relacional se presentan en partes posteriores del libro. El Capítulo 9 relaciona las estructuras de datos del modelo relacional con las construcciones de los modelos ER y EER (presentados en los Capítulos 3 y 4), y presenta algoritmos para diseñar un esquema de base de datos relacional mapeando un esquema conceptual en el modelo ER o EER en una representación relacional.
Estas asignaciones se incorporan en muchas herramientas de diseño de bases de datos y CASE1. Los capítulos 10 y 11 de la parte 4 tratan las técnicas de programación utilizadas para acceder a los sistemas de bases de datos y la noción de conectarse a bases de datos relacionales a través de los protocolos estándar ODBC y JDBC. También presentamos el tema de la programación de bases de datos web en el Capítulo 11. Los Capítulos 14 y 15 de la Parte 6 presentan otro aspecto del modelo relacional, a saber, las restricciones formales de las dependencias funcionales y multivalor; estas dependencias se utilizan para desarrollar una teoría de diseño de bases de datos relacionales basada en el concepto conocido como normalización.
En este capítulo, nos concentramos en describir los principios básicos del modelo relacional de datos. Comenzamos definiendo los conceptos de modelado y la notación del modelo relacional en la Sección 5.1. La sección 5.2 está dedicada a una discusión de las restricciones relacionales que se consideran una parte importante del modelo relacional y se aplican automáticamente en la mayoría de los DBMS relacionales. La sección 5.3

define las operaciones de actualización del modelo relacional, analiza cómo se manejan las violaciones de las restricciones de integridad e introduce el concepto de transacción. La sección 5.4 resume el capítulo.
Este capítulo y el capítulo 8 se centran en los fundamentos formales del modelo relacional, mientras que los capítulos 6 y 7 se centran en el modelo relacional práctico de SQL, que es la base de la mayoría de los DBMS relacionales comerciales y de código abierto. Muchos conceptos son comunes entre los modelos formales y prácticos, pero existen algunas diferencias que señalaremos.

5.1 Conceptos del modelo relacional

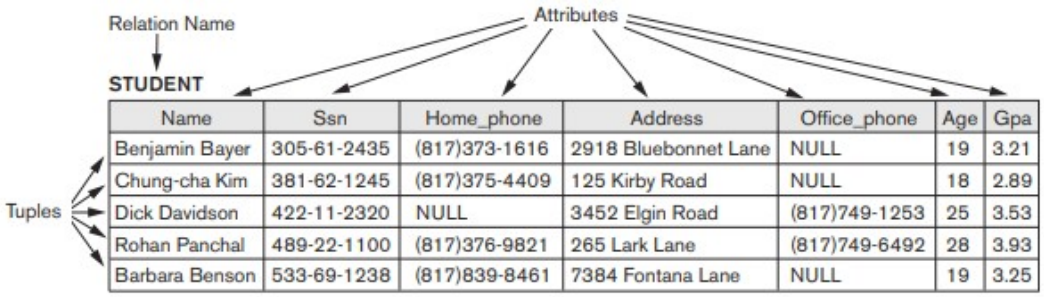
El modelo relacional representa la base de datos como una colección de relaciones. De manera informal, cada relación se asemeja a una tabla de valores o, hasta cierto punto, a un archivo plano de registros. Se llama archivo plano porque cada registro tiene una estructura lineal o plana simple. Por ejemplo, la base de datos de archivos que se muestra en la Figura 1.2 es similar a la representación del modelo relacional básico. Sin embargo, existen importantes diferencias entre relaciones y archivos, como veremos pronto.
Cuando se piensa en una relación como una tabla de valores, cada fila de la tabla representa una colección de valores de datos relacionados. Una fila representa un hecho que normalmente corresponde a una entidad o relación del mundo real. El nombre de la tabla y los nombres de las columnas se utilizan para ayudar a interpretar el significado de los valores en cada fila. Por ejemplo, la primera tabla de la Figura 1.2 se llama ESTUDIANTE porque cada fila representa hechos sobre una entidad estudiantil en particular
Los nombres de las columnas: Nombre, Número de alumno, Clase y Mayor especifican cómo interpretar los valores de los datos en cada fila, según la columna en la que se encuentra cada valor. Todos los valores de una columna son del mismo tipo de datos. En la terminología del modelo relacional formal, una fila se llama tupla, un encabezado de columna se llama atributo y la tabla se llama relación. El tipo de datos que describe los tipos de valores que pueden aparecer en cada columna está representado por un dominio de valores posibles. Ahora definimos estos términos —dominio, tupla, atributo y relación— formalmente.

5.1.1 Dominios, atributos, tuplas y relaciones

Un dominio D es un conjunto de valores atómicos. Por atómico queremos decir que cada valor en el dominio es indivisible en lo que respecta al modelo relacional formal. Un método común de especificar un dominio es especificar un tipo de datos del cual se extraen los valores de datos que forman el dominio. También es útil especificar un nombre para el dominio, para ayudar a interpretar sus valores. A continuación, se muestran algunos ejemplos de dominios:
■ **Números de teléfono de Estados Unidos**. El conjunto de números de teléfono de diez dígitos válidos en los Estados Unidos.
■ **Números de teléfono locales**. El conjunto de números de teléfono de siete dígitos válidos dentro de un código de área particular en los Estados Unidos. El uso de números de teléfono locales se está volviendo obsoleto rápidamente, siendo reemplazado por números estándar de diez dígitos.
■ **Números de seguridad social**. El conjunto de números de seguro social válidos de nueve dígitos. (Este es un identificador único asignado a cada persona en los Estados Unidos con fines de empleo, impuestos y beneficios).
■ **Nombres:** el conjunto de cadenas de caracteres que representan nombres de personas.
■ **Promedios de calificaciones**. Posibles valores de promedios de calificaciones calculados; cada uno debe ser un número real (punto flotante) entre 0 y 4.

■ **Edades de los empleados**. Posibles edades de los empleados de una empresa; cada uno debe ser un valor entero entre 15 y 80.
■ **Nombres de departamentos académicos.** El conjunto de nombres de departamentos académicos de una universidad, como Informática, Economía y Física.
■ **Códigos de departamento académico.** El conjunto de códigos de departamento académico, como "CS", "ECON" y "PHYS".
Las anteriores se denominan definiciones lógicas de dominios. También se especifica un tipo o formato de datos para cada dominio. Por ejemplo, el tipo de datos para el dominio Usa_phone_numbers puede declararse como una cadena de caracteres de la forma (ddd) ddd-dddd, donde cada d es un dígito numérico (decimal) y los primeros tres dígitos forman un código de área telefónico válido. El tipo de datos para Employee_ages es un número entero entre 15 y 80. Para Academic_department_names, el tipo de datos es el conjunto de todas las cadenas de caracteres que representan nombres de departamento válidos. Por lo tanto, un dominio recibe un nombre, tipo de datos y formato. También se puede proporcionar información adicional para interpretar los valores de un dominio; por ejemplo, un dominio numérico como Person_weights debe tener las unidades de medida, como libras o kilogramos.
Un esquema de relación2 R, denotado por R (A1, A2…, An), se compone de un nombre de relación R y una lista de atributos, A1, A2,…, An. Cada atributo Ai es el nombre de un papel desempeñado por algún dominio D en el esquema de relación R. D se denomina dominio de Ai y se denota por dom (Ai). Un esquema de relación se usa para describir una relación; R se llama el nombre de esta relación. El grado (o aridad) de una relación es el número de atributos n de su esquema de relación. Una relación de grado siete, que almacena información sobre estudiantes universitarios, contendría siete atributos que describen a cada estudiante de la siguiente manera: ESTUDIANTE (Nombre, Ssn, Home_phone, Address, Office_phone, Age, Gpa) Usando el tipo de datos de cada atributo, la definición a veces se escribe como:
ESTUDIANTE (Nombre: string, Ssn: string, Home_phone: string, Dirección: string, Office_phone: string, Edad: entero, Gpa: real)
Para este esquema de relación, ESTUDIANTE es el nombre de la relación, que tiene siete atributos. En la definición anterior, mostramos la asignación de tipos genéricos como cadena o entero a los atributos. Más precisamente, podemos especificar los siguientes dominios previamente definidos para algunos de los atributos de la relación ESTUDIANTE:
dom (Nombre) = Nombres; dom (Ssn) = números_seguridad_social; dom (HomePhone) = USA_phone_numbers3, dom (Office_phone) = USA_phone_numbers, y dom (Gpa) = Grade_point_average. También es posible referirse a atributos de un esquema de relación por su posición dentro de la relación; así, el segundo atributo de la relación ESTUDIANTE es Ssn, mientras que el cuarto atributo es Dirección.
Una relación (o estado de relación) 4 r del esquema de relación R (A1, A2,…, An), también denotado por r (R), es un conjunto de n-tuplas r = {t1, t2,…, tm} . Cada n-tupla t es una lista ordenada de n valores t = <v1, v2,…, vn>, donde cada valor vi, $1 \leq i \leq n$, es un elemento de dom (Ai) o es un valor NULL especial. (Los valores NULL se discuten más adelante y en la Sección 5.1.2.)
El i-ésimo valor de la tupla t, que corresponde al atributo Ai, se denomina t [Ai] o t.Ai (o t [i] si usamos la notación posicional). Los términos intensión de relación para el esquema R y extensión de relación para un estado de relación r (R) también se utilizan comúnmente.

La Figura 5.1 muestra un ejemplo de una relación ESTUDIANTE, que corresponde al esquema ESTUDIANTE que se acaba de especificar. Cada tupla de la relación representa una entidad (u objeto) de estudiante en particular. Mostramos la relación como una tabla, donde cada tupla se muestra como una fila y cada atributo corresponde a un encabezado de columna que indica un rol o interpretación de los valores en esa columna. Los valores NULL representan atributos cuyos valores son desconocidos o no existen para alguna



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

tupla de ESTUDIANTE individual.
La definición anterior de una relación se puede reformular de manera más formal utilizando los conceptos de la teoría de conjuntos de la siguiente manera. Una relación (o estado de relación) r (R) es una relación matemática de grado n en los dominios dom (A1), dom (A2),…, dom (An), que es un subconjunto del producto cartesiano (denotado por ×) de los dominios que definen R:

$$r(R) \subseteq (dom(A1) \times dom(A2) \times \ldots \times (dom(An))$$

El producto cartesiano especifica todas las posibles combinaciones de valores de los dominios subyacentes. Por tanto, si denotamos el número total de valores, o cardinalidad, en un dominio D por | D | (asumiendo que todos los dominios son finitos), el número total de tuplas en el producto cartesiano es

$$|dom(A1)| \times |dom(A2)| \times \ldots \times |dom(An)|$$

Este producto de cardinalidades de todos los dominios representa el número total de instancias o tuplas posibles que pueden existir en cualquier estado de relación r (R). De todas estas combinaciones posibles, un estado de relación en un momento dado (el estado de relación actual) refleja solo las tuplas válidas que representan un estado particular del mundo real. En general, a medida que cambia el estado del mundo real, también cambia el estado de relación, transformándose en otro estado de relación. Sin embargo, el esquema R es relativamente estático y cambia con muy poca frecuencia, por ejemplo, como resultado de agregar un atributo para representar nueva información que no se almacenó originalmente en la relación.
Es posible que varios atributos tengan el mismo dominio. Los nombres de los atributos indican diferentes roles o interpretaciones para el dominio. Por ejemplo, en la relación ESTUDIANTE, el mismo dominio USA_phone_numbers juega el papel de Home_phone, refiriéndose al teléfono de casa de un estudiante, y el papel de Office_phone, refiriéndose al teléfono de la oficina del estudiante. Un tercer atributo posible (no mostrado) con el mismo dominio podría ser Mobile_phone.

5.1.2 Características de las relaciones

La definición anterior de relaciones implica ciertas características que hacen que una relación sea diferente de un archivo o una tabla. Ahora discutimos algunas de estas características.
Ordenamiento de tuplas en una relación. Una relación se define como un conjunto de tuplas. Matemáticamente, los elementos de un conjunto no tienen orden entre ellos; por

tanto, las tuplas de una relación no tienen ningún orden en particular. En otras palabras, una relación no es sensible al orden de las tuplas. Sin embargo, en un archivo, los registros se almacenan físicamente en el disco (o en la memoria), por lo que siempre hay un orden entre los registros. Este orden indica el primer, segundo, iésimo y último registro del archivo. De manera similar, cuando mostramos una relación como una tabla, las filas se muestran en un orden determinado.

El orden de tuplas no es parte de la definición de una relación porque una relación intenta representar hechos a un nivel lógico o abstracto. Se pueden especificar muchos órdenes de tupla en la misma relación. Por ejemplo, las tuplas en la relación ESTUDIANTE en la Figura 5.1 podrían ordenarse por valores de Nombre, Ssn, Edad o algún otro atributo. La definición de relación no especifica ningún orden: no hay preferencia por un orden sobre otro. Por lo tanto, la relación que se muestra en la Figura 5.2 se considera idéntica a la que se muestra en la Figura 5.1. Cuando una relación se implementa como un archivo o se muestra como una tabla, se puede especificar un orden particular en los registros del archivo o en las filas de la tabla.

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

De acuerdo con la definición anterior de una relación, una n-tupla es una lista ordenada de n valores, por lo que el orden de los valores en una tupla y, por lo tanto, de los atributos en un esquema de relación es importante. Sin embargo, a un nivel más abstracto, el orden de los atributos y sus valores no es tan importante siempre que se mantenga la correspondencia entre atributos y valores. Se puede dar una definición alternativa de una relación, haciendo innecesario el ordenamiento de valores en una tupla. En esta definición, un esquema de relación R = {A1, A2,…, An} es un conjunto de atributos (en lugar de una lista ordenada de atributos), y un estado de relación r (R) es un conjunto finito de asignaciones r = {t1, t2,…, tm}, donde cada tupla ti es una asignación de R a D, y D es la unión (indicada por ∪) de los dominios de atributos; es decir, D = dom (A1) ∪ dom (A2) ∪… ∪ dom (An). En esta definición, t [Ai] debe estar en dom (Ai) para 1 ≤ i ≤ n para cada mapeo t en r. Cada mapeo ti se llama tupla.

De acuerdo con esta definición de tupla como mapeo, una tupla se puede considerar como un conjunto de pares (<attribute>, <valor>), donde cada par da el valor del mapeo de un atributo Ai a un valor vi de dom (Ai). El orden de los atributos no es importante, porque el nombre del atributo aparece con su valor. Según esta definición, las dos tuplas que se muestran en la Figura 5.3 son idénticas. Esto tiene sentido en un nivel abstracto, ya que realmente no hay razón para preferir que un valor de atributo aparezca antes que otro en una tupla. Cuando el nombre del atributo y el valor se incluyen juntos en una tupla, se conoce como datos autodescriptivos, porque la descripción de cada valor (nombre del atributo) se incluye en la tupla. Utilizaremos principalmente la primera definición de relación, donde los atributos están ordenados en el esquema de relación y los valores dentro de las tuplas están ordenados de manera similar, porque simplifica gran parte de la notación. Sin embargo, la definición alternativa dada aquí es más general

$t = <$ (Name, Dick Davidson),(Ssn, 422-11-2320),(Home_phone, NULL),(Address, 3452 Elgin Road),
(Office_phone, (817)749-1253),(Age, 25),(Gpa, 3.53)>

$t = <$ (Address, 3452 Elgin Road),(Name, Dick Davidson),(Ssn, 422-11-2320),(Age, 25),
(Office_phone, (817)749-1253),(Gpa, 3.53),(Home_phone, NULL)>

Figure 5.3
Two identical tuples when the order of attributes and values is not part of relation definition.

Figure 5.2
The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Valores y NULL en las tuplas. Cada valor de una tupla es un valor atómico; es decir, no es divisible en componentes dentro del marco del modelo relacional básico. Por lo tanto, los atributos compuestos y multivalor (consulte el Capítulo 3) no están permitidos. Este modelo a veces se denomina modelo relacional plano. Gran parte de la teoría detrás del modelo relacional se desarrolló teniendo en cuenta este supuesto, que se denomina el primer supuesto de forma normal. Por lo tanto, los atributos multivalor deben representarse mediante relaciones separadas y los atributos compuestos se representan solo por sus atributos componentes simples en el modelo relacional básico. Un concepto importante es el de los valores NULL, que se utilizan para representar los valores de atributos que pueden ser desconocidos o que pueden no aplicarse a una tupla. Un valor especial, llamado NULL, se utiliza en estos casos. Por ejemplo, en la Figura 5.1, algunas tuplas ESTUDIANTES tienen NULL para sus teléfonos de oficina porque no tienen una oficina (es decir, el teléfono de la oficina no se aplica a estos estudiantes). Otro estudiante tiene un NULL para el teléfono de casa, presumiblemente porque o no tiene un teléfono de casa o tiene uno, pero no lo sabemos (se desconoce el valor). En general, podemos tener varios significados para los valores NULL, como} valor desconocido, el valor existe pero no está disponible o el atributo no se aplica a esta tupla (también conocido como valor indefinido). Un ejemplo del último tipo de NULL ocurrirá si agregamos un atributo Visa_status a la relación ESTUDIANTE que se aplica solo a tuplas que representan estudiantes extranjeros. Es posible diseñar diferentes códigos para diferentes significados de valores NULL. La incorporación de diferentes tipos de valores NULL en las operaciones del modelo relacional ha resultado difícil y está fuera del alcance de nuestra presentación. El significado exacto de un valor NULL gobierna cómo le va durante las agregaciones aritméticas o comparaciones con otros valores. Por ejemplo, una comparación de dos valores NULL conduce a ambigüedades; si tanto el Cliente A como el B tienen direcciones NULL, no significa que tengan la misma dirección. Durante el diseño de la base de datos, es mejor evitar los valores NULL tanto como sea posible. Discutiremos esto más a fondo en los Capítulos 7 y 8 en el contexto de operaciones y consultas, y en el Capítulo 14 en el contexto del diseño y normalización de bases de datos.

Interpretación (significado) de una relación. El esquema de relación se puede interpretar como una declaración o un tipo de aserción. Por ejemplo, el esquema de la relación ESTUDIANTE de la Figura 5.1 afirma que, en general, una entidad de estudiante tiene un Nombre, Ssn, Teléfono de casa, Dirección, Teléfono de oficina, Edad y Gpa. Cada tupla de la relación se puede interpretar como un hecho o una instancia particular de la aserción. Por ejemplo, la primera tupla en la Figura 5.1 afirma el hecho de que hay un ESTUDIANTE cuyo Nombre es Benjamin Bayer, Ssn es 305-61-2435, Edad es 19, y así sucesivamente. Observe que algunas relaciones pueden representar hechos sobre entidades, mientras que otras relaciones pueden representar hechos sobre relaciones. Por ejemplo, un esquema de relación MAJORS (Student_ssn, Department_code) afirma que los estudiantes se especializan en disciplinas académicas. Una tupla en esta relación

relaciona a un estudiante con su disciplina principal. Por tanto, el modelo relacional representa hechos sobre entidades y relaciones de manera uniforme como relaciones. Esto a veces compromete la comprensibilidad porque uno tiene que adivinar si una relación representa un tipo de entidad o un tipo de relación.
Introdujimos el modelo entidad-relación (ER) en detalle en el Capítulo 3, donde los conceptos de entidad y relación se describieron en detalle. Los procedimientos de mapeo en el Capítulo 9 muestran cómo diferentes construcciones de los modelos de datos conceptuales ER / EER (ver Parte 2) se convierten en relaciones. Una interpretación alternativa de un esquema de relación es como predicado; en este caso, los valores de cada tupla se interpretan como valores que satisfacen el predicado. Por ejemplo, el predicado ESTUDIANTE (Nombre, Ssn,…) es verdadero para las cinco tuplas en relación ESTUDIANTE
de la Figura 5.1. Estas tuplas representan cinco proposiciones o hechos diferentes en el mundo real. Esta interpretación es bastante útil en el contexto de los lenguajes de programación lógica, como Prolog, porque permite que el modelo relacional se utilice dentro de estos lenguajes (consulte la Sección 26.5). Una suposición llamada la suposición del mundo cerrado establece que los únicos hechos verdaderos en el universo son aquellos presentes dentro de la extensión (estado) de la (s) relación (es). Cualquier otra combinación de valores hace que el predicado sea falso. Esta interpretación es útil cuando consideramos consultas sobre relaciones basadas en el cálculo relacional en la Sección 8.6.

5.1.3 Notación de modelo relacional

Usaremos la siguiente notación en nuestra presentación:
■ Un esquema de relación R de grado n se denota por R (A1, A2,…, An).
■ Las letras mayúsculas Q, R, S denotan nombres de relación.
■ Las letras minúsculas q, r, s denotan estados de relación.
■ Las letras t, u, v denotan tuplas.
■ En general, el nombre de un esquema de relación como ESTUDIANTE también indica el conjunto actual de tuplas en esa relación — el estado de relación actual — mientras que ESTUDIANTE (Nombre, Ssn,…) se refiere sólo al esquema de relación.
■ Un atributo A puede calificarse con el nombre de relación R a la que pertenece utilizando la notación de puntos R.A, por ejemplo, ESTUDIANTE.Nombre o ESTUDIANTE.Edad. Esto se debe a que se puede usar el mismo nombre para dos atributos en diferentes relaciones. Sin embargo, todos los nombres de atributo en una relación particular deben ser distintos.
■ Una n-tupla t en una relación r (R) se denota por t = <v1, v2,…, vn>, donde vi es el valor correspondiente al atributo Ai. La siguiente notación se refiere a los valores de los componentes de las tuplas:
Tanto t [Ai] como t.Ai (ya veces t [i]) se refieren al valor vi en t para el atributo Ai. Tanto t [Au, Aw,…, Az] como t. (Au, Aw,…, Az), donde Au, Aw,…, Az es una lista de atributos de R, se refieren al subtuple de valores <vu, vw ,…, Vz> de t correspondiente a los atributos especificados en la lista. Como ejemplo, considere la tupla t = <'Barbara Benson', '533-69-1238', '(817) 839-8461', '7384 Fontana Lane', NULL, 19, 3.25> de la relación ESTUDIANTE en la Figura 5,1; tenemos t [Nombre] = <‘Barbara Benson’>, y t [Ssn, Gpa, Edad] = <‘533-69-1238’,3,25, 19>.

5.2 Restricciones del modelo relacional y esquemas de bases de datos relacionales

Hasta ahora, hemos discutido las características de las relaciones simples. En una base de datos relacional, normalmente habrá muchas relaciones, y las tuplas de esas relaciones suelen estar relacionadas de varias formas. El estado de toda la base de datos corresponderá a los estados de todas sus relaciones en un momento determinado.

Generalmente, existen muchas restricciones o restricciones sobre los valores reales en el estado de una base de datos. Estas restricciones se derivan de las reglas del mini mundo que representa la base de datos, como discutimos en la Sección 1.6.8.
En esta sección, discutimos las diversas restricciones sobre los datos que se pueden especificar en una base de datos relacional en forma de restricciones. Las restricciones en las bases de datos generalmente se pueden dividir en tres categorías principales:
1. Restricciones que son inherentes al modelo de datos. Llamamos a estas restricciones inherentes basadas en modelos o restricciones implícitas.
2. Restricciones que se pueden expresar directamente en los esquemas del modelo de datos, normalmente especificándolas en el DDL (lenguaje de definición de datos, consulte la Sección 2.3.1).
Llamamos a estas restricciones basadas en esquemas o restricciones explícitas.
3. Restricciones que no pueden expresarse directamente en los esquemas del modelo de datos y, por lo tanto, deben ser expresadas y aplicadas por los programas de aplicación o de alguna otra manera. Las denominamos restricciones semánticas o basadas en aplicaciones o reglas comerciales.
Las características de las relaciones que discutimos en la Sección 5.1.2 son las restricciones inherentes del modelo relacional y pertenecen a la primera categoría. Por ejemplo, la restricción de que una relación no puede tener tuplas duplicadas es una restricción inherente. Las restricciones que discutimos en esta sección son de la segunda categoría, es decir, restricciones que pueden expresarse en el esquema del modelo relacional a través del DDL. Las restricciones en la tercera categoría son más generales, se relacionan tanto con el significado como con el comportamiento de los atributos y son difíciles de expresar y hacer cumplir dentro del modelo de datos, por lo que generalmente se verifican dentro de los programas de aplicación que realizan actualizaciones de la base de datos. En algunos casos, estas restricciones se pueden especificar como aserciones en SQL (consulte el Capítulo 7). Otra categoría importante de restricciones son las dependencias de datos, que incluyen dependencias funcionales y dependencias multivalor. Se utilizan principalmente para probar la "bondad" del diseño de una base de datos relacional y se utilizan en un proceso llamado normalización, que se analiza en los capítulos 14 y 15. Las restricciones basadas en el esquema incluyen restricciones de dominio, restricciones clave, restricciones en NULL. , restricciones de integridad de la entidad y restricciones de integridad referencial.

5.2.1 Restricciones de dominio

Las restricciones de dominio especifican que, dentro de cada tupla, el valor de cada atributo A debe ser un valor atómico del dominio dom (A). Ya hemos discutido las formas en que los dominios se pueden especificar en la Sección 5.1.1. Los tipos de datos asociados con dominios generalmente incluyen tipos de datos numéricos estándar para enteros (como entero corto, entero y entero largo) y números reales (flotante y flotante de doble precisión). También están disponibles caracteres, valores booleanos, cadenas de longitud fija y cadenas de longitud variable, al igual que la fecha, la hora, la marca de tiempo y otros tipos de datos especiales. Los dominios también se pueden describir mediante un subrango de valores de un tipo de datos o como un tipo de datos enumerados en el que se enumeran explícitamente todos los valores posibles. En lugar de describirlos en detalle aquí, discutimos los tipos de datos que ofrece el estándar relacional SQL en la Sección 6.1.

5.2.2 Restricciones y restricciones clave en valores NULL

En el modelo relacional formal, una relación se define como un conjunto de tuplas. Por definición, todos los elementos de un conjunto son distintos; por tanto, todas las tuplas de una relación también deben ser distintas. Esto significa que dos tuplas no pueden tener la misma combinación de valores para todos sus atributos. Por lo general, hay otros subconjuntos de atributos de un esquema de relación R con la propiedad de que no hay

dos tuplas en ningún estado de relación r de R que tengan la misma combinación de valores para estos atributos. Suponga que denotamos uno de esos subconjuntos de atributos por SK; entonces, para dos tuplas distintas t1 y t2 en un estado de relación r de R, tenemos la restricción de que:

$$t1[SK] \neq t2[SK]$$

Cualquier conjunto de atributos SK de este tipo se denomina superclave del esquema de relación R. Una superclave SK especifica una restricción de unicidad de que no hay dos tuplas distintas en ningún estado r de R que puedan tener el mismo valor para SK. Cada relación tiene al menos una superclave predeterminada: el conjunto de todos sus atributos. Sin embargo, una superclave puede tener atributos redundantes, por lo que El concepto más útil es el de una clave, que no tiene redundancia. Una clave k de un esquema de relación R es una superclave de R con la propiedad adicional de que eliminar cualquier atributo A de K deja un conjunto de atributos K ′ que ya no es una superclave de R. Por tanto, una clave satisface dos propiedades:

1. Dos tuplas distintas en cualquier estado de la relación no pueden tener valores idénticos para (todos) los atributos de la clave. Esta propiedad de singularidad también se aplica a una superclave.
2. Es una superclave mínima, es decir, una superclave de la que no podemos eliminar ningún atributo y aún se mantiene la restricción de unicidad. Esta propiedad de minimidad es necesaria para una clave, pero es opcional para una superclave. Por lo tanto, una clave es una superclave, pero no al revés. Una superclave puede ser una clave (si es mínima) o puede no ser una clave (si no es mínima). Considere la relación ESTUDIANTE de la Figura 5.1. El conjunto de atributos {Ssn} es una clave de ESTUDIANTE porque no hay dos tuplas de estudiantes que puedan tener el mismo valor para Ssn. Cualquier conjunto de atributos que incluya Ssn, para ejemplo, {Ssn, Name, Age} —es una superclave. Sin embargo, la superclave {Ssn, Nombre, Edad} no es una clave de ESTUDIANTE porque eliminar el Nombre o la Edad o ambos del conjunto todavía nos deja con una superclave. En general, cualquier superclave formada a partir de un solo atributo también es una clave. Una clave con múltiples atributos debe requerir todos sus atributos juntos para tiene la propiedad de unicidad. El valor de un atributo clave se puede utilizar para identificar de forma única cada tupla en la relación. Por ejemplo, el valor Ssn 305-61-2435 identifica de forma única la tupla correspondiente a Benjamin Bayer en la relación ESTUDIANTE. Observe que un conjunto de atributos que constituyen una clave es una propiedad del esquema de relación; es una restricción que debe mantenerse en cada estado de relación válido del esquema. Una clave se determina a partir del significado de los atributos, y la propiedad es invariante en el tiempo: debe continuar manteniéndose cuando insertamos nuevas tuplas en la relación. Por ejemplo, no podemos ni debemos designar el atributo Nombre de la relación ESTUDIANTE en la Figura 5.1 como una clave porque es posible que dos estudiantes con nombres idénticos existan en algún momento en un estado válido.9

En general, un esquema de relación puede tener más de una clave. En este caso, cada una de las claves se denomina clave candidata. Por ejemplo, la relación CAR en la Figura 5.4 tiene dos claves candidatas: License_number y Engine_serial_number. Es común designar una de las claves candidatas como clave principal de la relación. Esta es la clave candidata cuyos valores se utilizan para identificar tuplas en la relación. Usamos la convención de que los atributos que forman la clave principal de un esquema de relación están subrayados, como se muestra en la Figura 5.4. Observe que cuando un esquema de relación tiene varias claves candidatas, la elección de una para convertirse en la clave principal es algo arbitraria; sin embargo, generalmente es mejor elegir una clave primaria

con un solo atributo o una pequeña cantidad de atributos. Las otras claves candidatas se designan como claves únicas y no están subrayadas.
Otra restricción sobre los atributos especifica si los valores NULL están permitidos o no. Por ejemplo, si cada tupla de ESTUDIANTE debe tener un valor válido, no NULO para el atributo Nombre, entonces el Nombre del ESTUDIANTE está restringido a NO NULO.

5.2.3 Bases de datos relacionales y relacionales

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4
The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Esquemas de bases de datos Las definiciones y restricciones que hemos discutido hasta ahora se aplican a relaciones simples y sus atributos. Una base de datos relacional generalmente contiene muchas relaciones, con tuplas en relaciones que están relacionadas de varias formas. En esta sección, definimos 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 relación S = {R1, R2,…, Rm} y un conjunto de restricciones de integridad IC. Un estado de base de datos relacional10 DB de S es un conjunto de estados de relación DB = {r1, r2,…, rm} tal que cada ri es un estado de Ri y tal que los estados de relación ri satisfacen las restricciones de integridad especificadas en IC. La Figura 5.5 muestra un esquema de base de datos relacional que llamamos EMPRESA = {EMPLEADO, DEPARTAMENTO, LOCALIZACIONES_DEPARTAMENTO, PROYECTO, TRABAJOS_ON, DEPENDIENTE}. En cada esquema de relación, el atributo subrayado representa la clave primaria. La Figura 5.6 muestra el estado de una base de datos relacional correspondiente al esquema EMPRESA. Usaremos este esquema y el estado de la base de datos en este capítulo y en los Capítulos 4 al 6 para desarrollar consultas de muestra en diferentes lenguajes relacionales. (Los datos que se muestran aquí son expandido y disponible para cargar como una base de datos poblada desde el sitio web complementario para el texto, y se puede usar para los ejercicios prácticos del proyecto al final de los capítulos). Cuando nos referimos a una base de datos relacional, implícitamente incluimos tanto su esquema y su estado actual. Un estado de base de datos que no obedece a todas las restricciones de integridad se denomina no válido, y un estado que satisface todas las restricciones del conjunto definido de restricciones de integridad IC se denomina estado válido.
En la Figura 5.5, el atributo Dnumber tanto en DEPARTMENT como DEPT_LOCATIONS representa el mismo concepto del mundo real: el número dado a un departamento. Ese

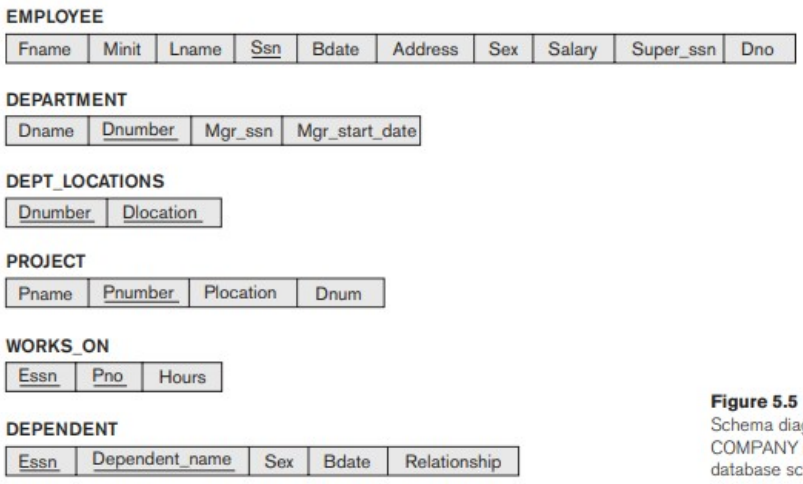


Figure 5.5
Schema diagram for the COMPANY relational database schema.

mismo concepto se llama Dno en EMPLOYEE y Dnum en PROJECT. Los atributos que representan el mismo concepto del mundo real pueden tener o no nombres idénticos en relaciones diferentes. Alternativamente, los atributos que representan diferentes conceptos pueden tener el mismo nombre en diferentes relaciones. Por ejemplo, podríamos haber utilizado el nombre de atributo Name para Pname of PROJECT y Dname of DEPARTMENT; en este caso, tendríamos dos atributos que comparten el mismo nombre, pero representan diferentes conceptos del mundo real: nombres de proyectos y nombres de departamentos.

En algunas versiones tempranas del modelo relacional, se asumió que el mismo concepto del mundo real, cuando se representa por un atributo, tendría nombres de atributo idénticos en todas las relaciones. Esto crea problemas cuando el mismo concepto del mundo real se usa en diferentes roles (significados) en la misma relación. Por ejemplo, el concepto de número de Seguro Social aparece dos veces en la relación EMPLEADO de la Figura 5.5: una vez en el rol del SSN del empleado y una vez en el rol del SSN del supervisor. Estamos obligados a darles nombres de atributos distintos (Ssn y Super_ssn, respectivamente) porque aparecen en la misma relación y para distinguir su significado.

Cada DBMS relacional debe tener un lenguaje de definición de datos (DDL) para definir un esquema de base de datos relacional. Los DBMS relacionales actuales utilizan principalmente SQL para este propósito. Presentamos el DDL SQL en las Secciones 6.1 y 6.2.

5.2.4 Integridad de la entidad, integridad referencial y claves externas

La restricción de integridad de la entidad establece que ningún valor de clave principal puede ser NULO. Esto se debe a que el valor de la clave principal se utiliza para identificar tuplas individuales en una relación. Tener valores NULL para la clave principal implica que no podemos identificar algunas tuplas. Por ejemplo, si dos o más tuplas tenían NULL para sus claves primarias, es posible que no podamos distinguirlas si intentamos hacer referencia a ellas de otras relaciones.

Las restricciones clave y las restricciones de integridad de la entidad se especifican en relaciones individuales. La restricción de integridad referencial se especifica entre dos relaciones y se utiliza para mantener la coherencia entre tuplas en las dos relaciones. De manera informal, la restricción de integridad referencial establece que una tupla en una relación que se refiere a otra relación debe hacer referencia a una tupla existente en esa relación. Por ejemplo, en la Figura 5.6, el atributo Dno de EMPLEADO da el número de departamento para el que trabaja cada empleado; por lo tanto, su valor en cada tupla

Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

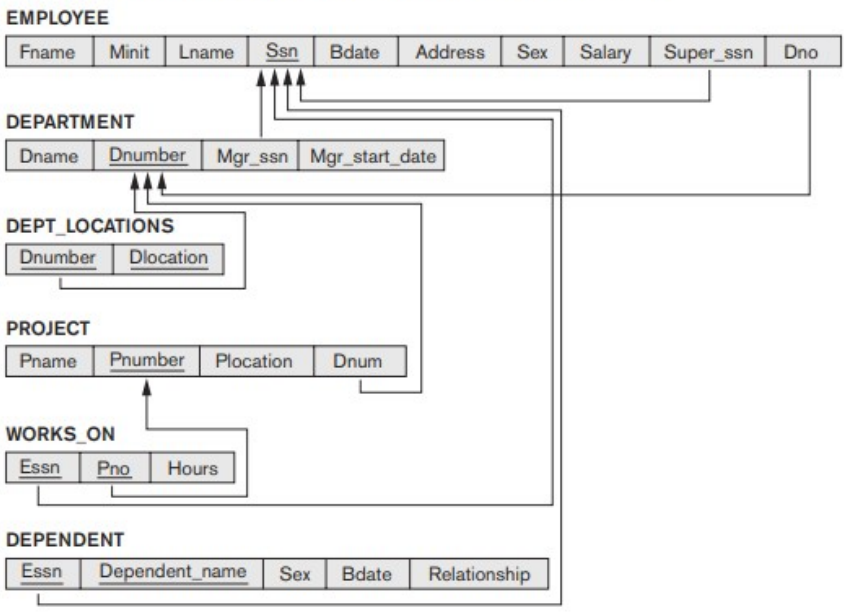
EMPLOYEE debe coincidir con el valor Dnumber de alguna tupla en la relación DEPARTMENT. Para definir la integridad referencial de manera más formal, primero definimos el concepto de clave externa. Las condiciones para una clave externa, que se indican a continuación, especifican una restricción de integridad referencial entre los dos esquemas de relación R1 y R2. Un conjunto de atributos FK en el esquema de relación R1 es una clave externa de R1 que hace referencia a la relación R2 si satisface las siguientes reglas:

1. Los atributos en FK tienen el mismo dominio (s) que los atributos de clave primaria PK de R2; se dice que los atributos FK hacen referencia o se refieren a la relación R2.
2. Un valor de FK en una tupla t1 del estado actual r1 (R1) ocurre como un valor de PK para alguna tupla t2 en el estado actual r2 (R2) o es NULL. En el primer caso, tenemos t1 [FK] = t2 [PK], y decimos que la tupla t1 hace referencia o se refiere a la tupla t2. En esta definición, R1 se denomina relación de referencia y R2 es la relación de referencia. Si se cumplen estas dos condiciones, se dice que se cumple una restricción de integridad referencial de R1 a R2.

En una base de datos de muchas relaciones, generalmente hay muchas restricciones de integridad referencial. Para especificar estas restricciones, primero debemos tener una comprensión clara del significado o papel que juega cada atributo o conjunto de atributos en los diversos esquemas de relación de la base de datos. Las restricciones de integridad referencial surgen típicamente de las relaciones entre las entidades representadas por los esquemas de relación.

Por ejemplo, considere la base de datos que se muestra en la Figura 5.6. En la relación EMPLEADO, el atributo Dno se refiere al departamento para el que trabaja un empleado; por lo tanto, designamos Dno para que sea una clave externa de EMPLEADO que hace referencia a la relación DEPARTAMENTO. Esto significa que un valor de Dno en cualquier tupla t1 de la relación EMPLEADO debe coincidir con un valor de la clave principal de DEPARTAMENTO (el atributo Dnumber) en alguna tupla t2 de la relación DEPARTAMENTO, o el valor de Dno puede ser NULO si el empleado no pertenece a un departamento o será asignado a un departamento más adelante. Por ejemplo, en la Figura 5.6, la tupla del empleado "John Smith" hace referencia a la tupla del departamento de "Investigación", lo que indica que "John Smith" trabaja para este departamento. Observe que una clave externa puede hacer referencia a su propia relación. Por ejemplo, el atributo Super_ssn en EMPLOYEE se refiere al supervisor de un empleado; este es otro empleado, representado por una tupla en la relación EMPLEADO. Por lo tanto, Super_ssn es una clave externa que hace referencia a la propia relación EMPLOYEE. En la Figura 5.6, la tupla del empleado "John Smith" hace referencia a la tupla del empleado

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



"Franklin Wong", lo que indica que "Franklin Wong" es el supervisor de "John Smith". Podemos mostrar en forma de diagrama las restricciones de integridad referencial dibujando un arco dirigido desde cada clave externa a la relación a la que hace referencia. Para mayor claridad, la punta de flecha puede apuntar a la clave primaria de la relación referenciada. La Figura 5.7 muestra el esquema de la Figura 5.5 con las restricciones de integridad referencial mostradas de esta manera. Todas las restricciones de integridad deben especificarse en el esquema de la base de datos relacional (es decir, como parte de su definición) si queremos que el DBMS haga cumplir estas restricciones en los estados de la base de datos. Por lo tanto, el DDL incluye disposiciones para especificar los diversos tipos de restricciones para que el DBMS pueda hacerlas cumplir automáticamente. En SQL, la declaración CREATE TABLE del DDL de SQL permite la definición de la clave principal, la clave única, NOT NULL, la integridad de la entidad y las restricciones de integridad referencial, entre otras restricciones (véanse las Secciones 6.1 y 6.2).

5.2.5 Otros tipos de restricciones

Las restricciones de integridad anteriores se incluyen en el lenguaje de definición de datos porque ocurren en la mayoría de las aplicaciones de bases de datos. Otra clase de restricciones generales, a veces llamadas restricciones de integridad semántica, no forman parte del DDL y deben especificarse y aplicarse de una manera diferente. Ejemplos de tales restricciones son el salario de un empleado no debe exceder el salario del empleado
supervisor y el número máximo de horas que un empleado puede trabajar en todos los proyectos por semana es 56. Tales restricciones se pueden especificar y hacer cumplir dentro de los programas de aplicación que actualizan la base de datos, o mediante el uso de un lenguaje de especificación de restricciones de propósito general. Los mecanismos denominados disparadores y aserciones se pueden usar en SQL, a través de las sentencias CREATE ASSERTION y CREATE TRIGGER, para especificar algunas de estas restricciones (consulte el Capítulo 7). Es más común verificar estos tipos de restricciones dentro de los programas de aplicación que usar lenguajes de especificación de restricciones porque estos últimos a veces son difíciles y complejos de usar, como discutimos en la Sección 26.1. Los tipos de restricciones que discutimos hasta ahora se pueden llamar restricciones de estado porque definen las restricciones que debe satisfacer un estado válido de la base de datos. Otro tipo de restricción, llamada restricciones de transición, se puede definir para hacer frente a los cambios de estado en la base de datos.11 Un ejemplo de una restricción de transición es: "el salario de un empleado sólo puede aumentar". Tales restricciones normalmente son impuestas por los programas de aplicación o especificadas usando reglas activas y disparadores, como discutimos en la Sección 26.1.

5.3 Actualizar operaciones, transacciones, y hacer frente a las infracciones de las restricciones

Las operaciones del modelo relacional se pueden clasificar en recuperaciones y actualizaciones. Las operaciones de álgebra relacional, que pueden usarse para especificar recuperaciones, se analizan en detalle en el capítulo 8. Una expresión de álgebra relacional forma una nueva relación después de aplicar varios operadores algebraicos a un conjunto existente de relaciones; su uso principal es consultar una base de datos para recuperar información. El usuario formula una consulta que especifica los datos de interés y se forma una nueva relación aplicando operadores relacionales para recuperar estos datos. La relación de resultado se convierte en la respuesta (o el resultado de) la consulta del usuario. El capítulo 8 también presenta el idioma llamado cálculo relacional, que se utiliza para definir una consulta de forma declarativa sin dar un

orden específico de operaciones. En esta sección, nos concentramos en las operaciones de modificación o actualización de la base de datos. Hay tres operaciones básicas que pueden cambiar los estados de las relaciones en la base de datos: Insertar, Eliminar y Actualizar (o Modificar). Insertan datos nuevos, eliminan datos antiguos o modifican registros de datos existentes, respectivamente. Insertar se usa para insertar una o más tuplas nuevas en una relación, Eliminar se usa para eliminar tuplas y Actualizar (o Modificar) se usa para cambiar los valores de algunos atributos en tuplas existentes. Siempre que se apliquen estas operaciones, no se deben violar las restricciones de integridad especificadas en el esquema de la base de datos relacional. En esta sección discutimos los tipos de restricciones que pueden ser violadas por cada una de estas operaciones y los tipos de acciones que pueden tomarse si una operación causa una violación. Usamos la base de datos que se muestra en la Figura 5.6 para ejemplos y discutimos solo las restricciones de dominio, restricciones clave, restricciones de integridad de entidad y las restricciones de integridad referencial que se muestran en la Figura 5.7. Para cada tipo de operación, damos algunos ejemplos y discutimos las restricciones que cada operación puede violar.

5.3.1 La operación de inserción

La operación Insertar proporciona una lista de valores de atributo para una nueva tupla t que se insertará en una relación R. Insertar puede violar cualquiera de los cuatro tipos de restricciones. Las restricciones de dominio se pueden violar si se proporciona un valor de atributo que no aparece en el dominio correspondiente o no es del tipo de datos apropiado. Las restricciones de clave se pueden violar si un valor clave en la nueva tupla t ya existe en otra tupla en la relación r (R). La integridad de la entidad se puede violar si cualquier parte de la clave primaria de la nueva tupla t es NULL. La integridad referencial se puede violar si el valor de cualquier clave externa en t se refiere a una tupla que no existe en la relación referenciada. A continuación, se muestran algunos ejemplos para ilustrar esta discusión.

- Operación: Inserte <'Cecilia', 'F', 'Kolonsky', NULL, '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> en EMPLOYEE.Resultado: esta inserción viola la restricción de integridad de la entidad (NULL para la clave principal Ssn), por lo que se rechaza.
- Operación: Inserte <'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, '987654321', 4> en EMPLEADO. Resultado: esta inserción viola la restricción de clave porque ya existe otra tupla con el mismo valor de Ssn en la relación EMPLEADO, por lo que se rechaza.
- Operación: Inserte <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 7> en EMPLOYEE. Resultado: esta inserción viola la restricción de integridad referencial especificada en Dno en EMPLOYEE porque no existe una tupla referenciada correspondiente en DEPARTMENT con Dnumber = 7.
- Operación: Inserte <'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windy Lane, Katy, TX', F, 28000, NULL, 4> en EMPLOYEE. Resultado: esta inserción satisface todas las restricciones, por lo que es aceptable.

Si una inserción viola una o más restricciones, la opción predeterminada es rechazar la inserción. En este caso, sería útil si el DBMS pudiera proporcionar una razón al usuario de por qué se rechazó la inserción. Otra opción es intentar corregir el motivo del rechazo de la inserción, pero normalmente no se usa para infracciones causadas por Insert; más bien, se utiliza con más frecuencia para corregir infracciones de Eliminar y Actualizar. En la primera operación, el DBMS podría pedirle al usuario que proporcione un valor para Ssn, y luego podría aceptar la inserción si se proporciona un valor Ssn válido. En la operación 3, el DBMS podría pedirle al usuario que cambie el valor de Dno a algún valor

válido (o establecerlo en NULL), o podría pedirle al usuario que inserte una tupla DEPARTMENT con Dnumber = 7 y podría aceptar la inserción original sólo después de que se aceptó tal operación. Observe que en el último caso, la infracción de inserción puede volver en cascada a la relación EMPLEADO si el usuario intenta insertar una tupla para el departamento 7 con un valor para Mgr_ssn que no existe en la relación EMPLEADO.

5.3.2 La operación de eliminación

La operación Eliminar solo puede violar la integridad referencial. Esto ocurre si las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se está eliminando. Para especificar la eliminación, una condición en los atributos de la relación selecciona la tupla (o tuplas) que se eliminarán. Aquí hay unos ejemplos.

■ Operación: elimine la tupla WORKS_ON con Essn = '999887777' y Pno = 10.
Resultado: esta eliminación es aceptable y elimina exactamente una tupla.

■ Operación: Elimine la tupla EMPLOYEE con Ssn = '999887777'.
Resultado: esta eliminación no es aceptable, porque hay tuplas en WORKS_ON que se refieren a esta tupla. Por lo tanto, si se elimina la tupla en EMPLEADO, se producirán violaciones de la integridad referencial.

■ Operación: Elimine la tupla EMPLOYEE con Ssn = "333445555".
Resultado: esta eliminación resultará en violaciones de integridad referencial aún peores, porque la tupla involucrada es referenciada por tuplas de las relaciones EMPLOYEE, DEPARTMENT, WORKS_ON y DEPENDENT.

Hay varias opciones disponibles si una operación de eliminación provoca una infracción. La primera opción, llamada restringir, es rechazar la eliminación. La segunda opción, denominada cascada, es intentar poner en cascada (o propagar) la eliminación eliminando tuplas que hacen referencia a la tupla que se está eliminando. Por ejemplo, en la operación 2, el DBMS podría eliminar automáticamente las tuplas ofensivas de WORKS_ON con Essn = '999887777'. Una tercera opción, llamada set null o set default, es modificar los valores de los atributos de referencia que causan la infracción; cada uno de estos valores se establece en NULL o se cambia a hacer referencia a otra tupla válida predeterminada. Tenga en cuenta que, si un atributo de referencia que causa una infracción es parte de la clave principal, no se puede establecer en NULL; de lo contrario, violaría la integridad de la entidad.

También son posibles combinaciones de estas tres opciones. Por ejemplo, para evitar que la operación 3 cause una infracción, el DBMS puede eliminar automáticamente todas las tuplas de WORKS_ON y DEPENDENT con Essn = "333445555". Las tuplas en EMPLOYEE con Super_ssn = "333445555" y la tupla en DEPARTMENT con Mgr_ssn = "333445555" pueden tener sus valores Super_ssn y Mgr_ssn cambiados a otros valores válidos o a NULL. Aunque puede tener sentido eliminar automáticamente las tuplas WORKS_ON y DEPENDENT que hacen referencia a una tupla EMPLOYEE, puede que no tenga sentido eliminar otras tuplas EMPLOYEE o una tupla DEPARTMENT. En general, cuando se especifica una restricción de integridad referencial en el DDL, la DBMS permitirá que el diseñador de la base de datos especifique cuál de las opciones se aplica en caso de una violación de la restricción. Discutimos cómo especificar estas opciones en SQL DDL en el Capítulo 6.

5.3.3 La operación de actualización

La operación Actualizar (o Modificar) se usa para cambiar los valores de uno o más atributos en una tupla (o tuplas) de alguna relación R. Es necesario especificar una condición sobre los atributos de la relación para seleccionar la tupla (o tuplas)) ser modificado. Aquí hay unos ejemplos.

■ Operación:
Actualice el salario de la tupla EMPLEADO con Ssn = '999887777' a 28000.
Resultado: aceptable.

■ Operación:
Actualice el Dno de la tupla EMPLOYEE con Ssn = '999887777' a 1.
Resultado: aceptable.
■ Operación:
Actualice el Dno de la tupla EMPLOYEE con Ssn = '999887777' a 7.
Resultado: Inaceptable, porque viola la integridad referencial.
■ Operación:
Actualice el Ssn de la tupla EMPLOYEE con Ssn = "999887777" a "987654321".
Resultado: Inaceptable, porque viola la restricción de clave primaria al repetir
Un valor que ya existe como clave principal en otra tupla; viola las restricciones de integridad referencial porque hay otras relaciones que se refieren al valor existente de Ssn.
La actualización de un atributo que no es parte de una clave primaria ni de una clave externa generalmente no causa problemas; el DBMS solo necesita verificar para confirmar que el nuevo valor es del tipo de datos y dominio correctos. La modificación de un valor de clave principal es similar a eliminar una tupla e insertar otra en su lugar porque usamos la clave principal para identificar tuplas. Por lo tanto, entran en juego los problemas discutidos anteriormente en las Secciones 5.3.1 (Insertar) y 5.3.2 (Eliminar). Si se modifica un atributo de clave externa, el DBMS debe asegurarse de que el nuevo valor se refiera a una tupla existente en la relación referenciada (o se establezca en NULL). Existen opciones similares para tratar las violaciones de la integridad referencial causadas por Update como las opciones discutidas para la operación Delete. De hecho, cuando se especifica una restricción de integridad referencial en el DDL, el DBMS permitirá al usuario elegir opciones separadas para lidiar con una violación causada por Eliminar y una violación causada por Actualización (consulte la Sección 6.2).

5.3.4 El concepto de transacción

Un programa de aplicación de base de datos que se ejecuta en una base de datos relacional normalmente ejecuta una o más transacciones. Una transacción es un programa en ejecución que incluye algunas operaciones de la base de datos, como leer de la base de datos o aplicar inserciones, eliminaciones o actualizaciones a la base de datos. Al final de la transacción, debe dejar la base de datos en un estado válido o consistente que satisfaga todas las restricciones especificadas en el esquema de la base de datos. Una sola transacción puede involucrar cualquier número de operaciones de recuperación (que se discutirán como parte de álgebra relacional y cálculo en el Capítulo 8, y como parte del lenguaje SQL en los Capítulos 6 y 7) y cualquier número de operaciones de actualización. Estas recuperaciones y actualizaciones formarán juntas una unidad atómica de trabajo contra la base de datos. Por ejemplo, una transacción para aplicar un retiro bancario generalmente leerá el registro de la cuenta del usuario, verificará si hay un saldo suficiente y luego actualizará el registro por el monto del retiro. Una gran cantidad de aplicaciones comerciales que se ejecutan en bases de datos relacionales en sistemas de procesamiento de transacciones en línea (OLTP) ejecutan transacciones a velocidades que alcanzan varios cientos por segundo. Los conceptos de procesamiento de transacciones, la ejecución concurrente de transacciones y la recuperación de fallas se discutirán en los Capítulos 20 al 22.

5.4 Resumen

En este capítulo presentamos los conceptos de modelado, las estructuras de datos y las restricciones proporcionadas por el modelo relacional de datos. Comenzamos presentando los conceptos de dominios, atributos y tuplas. Luego, definimos un esquema de relación como una lista de atributos que describen la estructura de una relación. Una relación, o estado de relación, es un conjunto de tuplas que se ajusta al esquema. Varias características diferencian las relaciones de las tablas o archivos ordinarios. La primera es

que una relación no es sensible al orden de las tuplas. El segundo implica el ordenamiento de atributos en un esquema de relación y el correspondiente ordenamiento de valores dentro de una tupla. Dimos una definición alternativa de relación que no requiere el ordenamiento de atributos, pero continuamos usando la primera definición, que requiere que los atributos y valores de tupla estén ordenados, por conveniencia. Luego, discutimos los valores en tuplas e introdujimos valores NULL para representar información que faltaba o desconocida. Enfatizamos que los valores NULL deben evitarse tanto como sea posible. Clasificamos las restricciones de la base de datos en restricciones inherentes basadas en modelos, restricciones explícitas basadas en esquemas y restricciones semánticas o reglas comerciales. Luego, discutimos las restricciones del esquema relacionadas con el modelo relacional, comenzando con las restricciones de dominio, luego las restricciones clave (incluidos los conceptos de superclave, clave y clave primaria) y la restricción NOT NULL en los atributos. Definimos bases de datos relacionales y esquemas de bases de datos relacionales. Las restricciones relacionales adicionales incluyen la restricción de integridad de la entidad, que prohíbe que los atributos de clave primaria sean NULL. Describimos la restricción de integridad referencial de interrelación, que se utiliza para mantener la coherencia de las referencias entre tuplas de varias relaciones.
Las operaciones de modificación en el modelo relacional son Insertar, Eliminar y Actualizar. Cada operación puede violar ciertos tipos de restricciones (consulte la Sección 5.3). Siempre que se aplica una operación, el estado de la base de datos resultante debe ser un estado válido. Finalmente, presentamos el concepto de transacción, que es importante en los DBMS relacionales porque permite agrupar varias operaciones de la base de datos en una sola acción atómica en la base de datos.

Capítulo 06 Fundamentals

SQL básico
El lenguaje SQL puede considerarse una de las principales razones del éxito comercial de las bases de datos relacionales. Debido a que se convirtió en un estándar para las bases de datos relacionales, los usuarios estaban menos preocupados por migrar sus aplicaciones de bases de datos desde otros tipos de sistemas de bases de datos, por ejemplo, redes más antiguas o sistemas jerárquicos, a sistemas relacionales. Esto se debe a que incluso si los usuarios no estaban satisfechos con el producto DBMS relacional particular que estaban usando, no se esperaba que la conversión a otro producto DBMS relacional fuera demasiado costosa y consumiera mucho tiempo porque ambos sistemas seguían los mismos estándares de lenguaje. En la práctica, por supuesto, existen diferencias entre varios paquetes de DBMS relacionales comerciales. Sin embargo, si el usuario es diligente en el uso de solo aquellas características que son parte del estándar, y si dos DBMS relacionales apoyan fielmente el estándar, entonces la conversión entre dos sistemas debería simplificarse. Otra ventaja de tener tal estándar es que los usuarios pueden escribir declaraciones en un programa de aplicación de base de datos que puede acceder a los datos almacenados en dos o más DBMS relacionales sin tener que cambiar el sub lenguaje de la base de datos (SQL), siempre que ambos / todos los DBMS relacionales admite SQL estándar. Este capítulo presenta el modelo relacional práctico, que se basa en el estándar SQL para DBMS relacionales comerciales, mientras que el Capítulo 5 presenta los conceptos más importantes que subyacen al modelo formal de datos relacionales. En el Capítulo 8 (Secciones 8.1 a 8.5), discutiremos las operaciones del álgebra relacional, que son muy importantes para comprender los tipos de solicitudes que pueden especificarse en una base de datos relacional. También son importantes para el procesamiento y la optimización de consultas en un DBMS relacional, como veremos en los capítulos 18 y 19. Sin embargo, las operaciones de álgebra

relacional son de un nivel demasiado bajo para la mayoría de los usuarios de DBMS comerciales porque una consulta en álgebra relacional se escribe como secuencia de operaciones que, cuando se ejecutan, producen el resultado requerido. Por lo tanto, el usuario debe especificar cómo, es decir, en qué orden, ejecutar las operaciones de consulta. Por otro lado, el lenguaje SQL proporciona una interfaz de lenguaje declarativo de nivel superior, por lo que el usuario solo especifica cuál será el resultado, dejando la optimización real y las decisiones sobre cómo ejecutar la consulta al DBMS. Aunque SQL incluye algunas características del álgebra relacional, se basa en mayor medida en el cálculo relacional de tuplas, que describimos en la sección 8.6. Sin embargo, la sintaxis SQL es más fácil de usar que cualquiera de los dos lenguajes formales. El nombre SQL se expande actualmente como lenguaje de consulta estructurado. Originalmente, SQL se llamaba SEQUEL (Structured English QUEry Language) y fue diseñado e implementado en IBM Research como la interfaz para un sistema de base de datos relacional experimental llamado SYSTEM R. SQL es ahora el lenguaje estándar para DBMS relacionales comerciales. La estandarización de SQL es un esfuerzo conjunto del Instituto Nacional Estadounidense de Estándares (ANSI) y la Organización Internacional de Estándares (ISO), y el primer estándar SQL se llama SQL-86 o SQL1. Posteriormente se desarrolló un estándar revisado y muy ampliado llamado SQL-92 (también conocido como SQL2). El siguiente estándar que es bien reconocido es SQL: 1999, que comenzó como SQL3. Las actualizaciones adicionales al estándar son SQL: 2003 y SQL: 2006, que agregaron características XML (consulte el Capítulo 13) entre otras actualizaciones del idioma. Otra actualización en 2008 incorporó más características de base de datos de objetos en SQL (consulte el Capítulo 12), y una actualización adicional es SQL: 2011. Intentaremos cubrir la última versión de SQL tanto como sea posible, pero algunas de las características más nuevas se comentan en capítulos posteriores. Tampoco es posible cubrir el idioma en su totalidad en este texto. Es importante tener en cuenta que cuando se agregan nuevas funciones a SQL, por lo general, algunas de estas funciones tardan algunos años en incorporarse a los DBMS SQL comerciales.

SQL es un lenguaje de base de datos completo: tiene declaraciones para definiciones de datos, consultas y actualizaciones. Por lo tanto, es un DDL y un DML. Además, tiene facilidades para definir vistas en la base de datos, o especificar seguridad y autorización, para definir restricciones de integridad y para especificar controles de transacciones. También tiene reglas para incrustar sentencias SQL en un lenguaje de programación de propósito general como Java o C / C ++. 1 los estándares SQL posteriores (que comienzan con SQL: 1999) se dividen en una especificación central más extensiones especializadas. Se supone que el núcleo debe ser implementado por todos los proveedores de RDBMS que cumplen con SQL. Las extensiones se pueden implementar como módulos opcionales para comprar de forma independiente para aplicaciones de bases de datos específicas como minería de datos, datos espaciales, datos temporales, almacenamiento de datos, procesamiento analítico en línea (OLAP), datos multimedia, etc. Debido a que el tema de SQL es importante y extenso, dedicamos dos capítulos a sus características básicas. En este capítulo, la Sección 6.1 describe los comandos SQL DDL para crear esquemas y tablas, y brinda una descripción general de los tipos de datos básicos en SQL. La sección 6.2 presenta cómo se especifican las restricciones básicas como la clave y la integridad referencial. La sección 6.3 describe las construcciones SQL básicas para especificando consultas de recuperación, y la Sección 6.4 describe los comandos SQL para la inserción, eliminación y actualización. En el Capítulo 7, describiremos consultas de recuperación SQL más complejas, así como los comandos ALTER para cambiar el esquema. También describiremos la instrucción CREATE ASSERTION, que permite la especificación de restricciones más generales en la base de datos, y el concepto de disparadores, que se presenta con más detalle en el Capítulo 26. En el Capítulo 26 discutiremos la función SQL para definir vistas en la base de datos 7.

Las vistas también se denominan tablas virtuales o derivadas porque presentan al usuario lo que parecen ser tablas; sin embargo, la información de esas tablas se deriva de tablas definidas previamente. La sección 6.5 enumera algunas características de SQL que se presentan en otros capítulos del libro; Estos incluyen características orientadas a objetos en el Capítulo 12, XML en el Capítulo 13, control de transacciones en el Capítulo 20, bases de datos activas (disparadores) en el Capítulo 26, características de procesamiento analítico en línea (OLAP) en el Capítulo 29 y seguridad / autorización en el Capítulo 30. Sección 6.6 resume el capítulo. Los capítulos 10 y 11 tratan las diversas técnicas de programación de bases de datos para programar con SQL.

6.1 Definición de datos SQL y tipos de datos

SQL utiliza los términos tabla, fila y columna para los términos relación, tupla y atributo del modelo relacional formal, respectivamente. Usaremos los términos correspondientes indistintamente. El comando SQL principal para la definición de datos es la instrucción CREATE, que se puede usar para crear esquemas, tablas (relaciones), tipos y dominios, así como otras construcciones como vistas, afirmaciones y disparadores. Antes de describir las declaraciones CREATE relevantes, discutimos los conceptos de esquema y catálogo en la Sección 6.1.1 para colocar nuestra discusión en perspectiva. La Sección 6.1.2 describe cómo se crean las tablas y la Sección 6.1.3 describe los tipos de datos más importantes disponibles para la especificación de atributos. Debido a que la especificación SQL es muy grande, damos una descripción de las características más importantes. Se pueden encontrar más detalles en los diversos documentos de estándares SQL (ver notas bibliográficas al final del capítulo).

6.1.1 Conceptos de esquema y catálogo en SQL

Las primeras versiones de SQL no incluían el concepto de un esquema de base de datos relacional; todas las tablas (relaciones) se consideraron parte del mismo esquema. El concepto de un esquema SQL se incorporó a partir de SQL2 para agrupar tablas y otras construcciones que pertenecen a la misma aplicación de base de datos (en algunos sistemas, un esquema se llama base de datos). Un esquema SQL se identifica mediante un nombre de esquema e incluye un identificador de autorización para indicar el usuario o la cuenta que posee el esquema, así como descriptores para cada elemento del esquema. Los elementos del esquema incluyen tablas, tipos, restricciones, vistas, dominios y otras construcciones (como concesiones de autorización) que describen el esquema. Se crea un esquema mediante la instrucción CREATE SCHEMA, que puede incluir todas las definiciones de los elementos del esquema. Alternativamente, al esquema se le puede asignar un nombre y un identificador de autorización, y los elementos se pueden definir más adelante. Por ejemplo, la siguiente declaración crea un esquema llamado EMPRESA propiedad del usuario con el identificador de autorización "Jsmith". Tenga en cuenta que cada instrucción en SQL termina con un punto y coma.

CREATE SCHEMA COMPANY **AUTHORIZATION** ‘Jsmith’;

En general, no todos los usuarios están autorizados a crear esquemas y elementos de esquema. El administrador del sistema o DBA debe otorgar explícitamente el privilegio de crear esquemas, tablas y otras construcciones a las cuentas de usuario relevantes. Además del concepto de esquema, SQL utiliza el concepto de catálogo: una colección de esquemas con nombre.2 Las instalaciones de bases de datos suelen tener un entorno y un esquema predeterminados, por lo que cuando un usuario se conecta e inicia sesión en la instalación de esa base de datos, el usuario puede hacen referencia directamente a tablas y otras construcciones dentro de ese esquema sin tener que especificar un nombre de esquema en particular. Un catálogo siempre contiene un esquema especial llamado INFORMATION_SCHEMA, que proporciona información sobre todos los esquemas en el catálogo y todos los descriptores de elementos en estos esquemas. Las restricciones de integridad, como la integridad referencial, se pueden definir entre relaciones solo si existen en esquemas dentro del mismo catálogo. Los esquemas dentro del mismo

catálogo también pueden compartir ciertos elementos, como las definiciones de tipo y dominio.

6.1.2 El comando CREATE TABLE en SQL

l comando CREATE TABLE se usa para especificar una nueva relación dándole un nombre y especificando sus atributos y restricciones iniciales. Los atributos se especifican primero, y a cada atributo se le da un nombre, un tipo de datos para especificar su dominio de valores y posiblemente restricciones de atributos, como NOT NULL. La clave, la integridad de la entidad y las restricciones de integridad eferencial se pueden especificar dentro de la declaración CREATE TABLE después de declarar los atributos, o se pueden agregar más tarde usando el comando ALTER TABLE (ver Capítulo 7). La Figura 6.1 muestra ejemplos de declaraciones de definición de datos en SQL para el esquema de base de datos relacional de la COMPAÑÍA que se muestra en la Figura 3.7. Normalmente, el esquema SQL en el que se declaran las relaciones se especifica implícitamente en el entorno en el que se ejecutan las sentencias CREATE TABLE. Alternativamente, podemos adjuntar explícitamente el nombre del esquema al nombre de la relación, separados por un punto. Por ejemplo, al escribir CREATE TABLE COMPANY.EMPLOYEE en lugar de CREATE TABLE EMPLOYEE como en la Figura 6.1, podemos hacer explícitamente (en lugar de implícitamente) hacer que la tabla EMPLOYEE forme parte del esquema COMPANY. Las relaciones declaradas mediante declaraciones CREATE TABLE se denominan tablas base (o relaciones base); esto significa que la tabla y sus filas se crean realmente y almacenado como un archivo por el DBMS. Las relaciones base se distinguen de las relaciones virtuales, creadas a través de la instrucción CREATE VIEW (consulte el Capítulo 7), que pueden corresponder o no a un archivo físico real. En SQL, se considera que los atributos de una tabla base están ordenados en la secuencia en la que se especifican en la instrucción CREATE TABLE. Sin embargo, las filas (tuplas) no se consideran ordenadas dentro de una tabla (relación). Es

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Figure 6.1 SQL CREATE TABLE data definition statements for defining the COMPANY schema from Figure 5.7.

importante notar que en la Figura 6.1, hay algunas claves foráneas que pueden causar errores porque se especifican a través de referencias circulares o porque hacen referencia a una tabla que aún no se ha creado. Por ejemplo, la clave externa Super_ssn en la tabla EMPLOYEE es una referencia circular porque se refiere a la tabla EMPLOYEE en sí. La clave externa Dno en la tabla EMPLOYEE se refiere a la tabla DEPARTMENT, que aún no se ha creado. Para hacer frente a este tipo de problema, estas restricciones se pueden dejar fuera de la instrucción CREATE TABLE inicial y luego agregarlas más tarde utilizando la instrucción ALTER TABLE (ver Capítulo 7). Mostramos todas las claves externas en la Figura 6.1 para mostrar el esquema COMPAÑÍA completo en un solo lugar.

6.1.3 Tipos de datos de atributos y dominios en SQL

Los tipos de datos básicos disponibles para los atributos incluyen numérico, cadena de caracteres, cadena de bits, booleano, fecha y hora.

■ Los tipos de datos numéricos incluyen números enteros de varios tamaños (INTEGER o INT y SMALLINT) y números de punto flotante (reales) de diversa precisión (FLOAT o REAL, y DOBLE PRECISIÓN). Los números formateados se pueden declarar usando DECIMAL (i, j), o DEC (i, j) o NUMERIC (i, j), donde i, la precisión, es el número total de dígitos decimales yj, la escala, es el número de dígitos después del punto decimal. El valor predeterminado para la escala es cero y el valor predeterminado para la precisión está definido por la implementación.

■ Los tipos de datos de cadena de caracteres son de longitud fija: CHAR (n) o CHARACTER (n), donde n es el número de caracteres, o de longitud variable, VARCHAR (n) o CHAR VARYING (n) o CHARACTER VARYING (n), donde n es el número máximo de caracteres. Al especificar un valor de cadena literal, se coloca entre comillas simples (apóstrofos) y distingue entre mayúsculas y minúsculas (se hace una distinción entre mayúsculas y minúsculas). Para cadenas de longitud fija, una cadena más corta se rellena con caracteres en blanco a la derecha. Por ejemplo, si el valor "Smith" es para un atributo de tipo CHAR (10), se rellena con cinco caracteres en blanco para convertirse en "Smith" si es necesario. Los espacios en blanco acolchados generalmente se ignoran cuando se comparan cadenas. Para fines de comparación, las cadenas se consideran ordenadas en orden alfabético (o lexicográfico); si una cadena str1 aparece antes de otra cadena str2 en orden alfabético, entonces str1 se considera menor que str2.
También hay un operador de concatenación indicado por || (barra vertical doble) que puede concatenar dos cadenas
en SQL. Por ejemplo, "abc" || "XYZ" da como resultado una sola cadena "abcXYZ". Otro tipo de datos de cadena de longitud variable llamado CHARACTER LARGE OBJECT o CLOB también está disponible para especificar columnas que tienen valores de texto grandes, como documentos. La longitud máxima de CLOB se puede especificar en kilobytes (K), megabytes (M) o gigabytes (G). Por ejemplo, CLOB (20M) especifica una longitud máxima de 20 megabytes.

■ Los tipos de datos de cadena de bits son de longitud fija n — BIT (n) —o de longitud variable— BIT VARYING (n), donde n es el número máximo de bits. El valor predeterminado para n, la longitud de una cadena de caracteres o una cadena de bits, es 1. Las cadenas de bits literales se colocan entre comillas simples pero precedidas por una B para distinguirlas de las cadenas de caracteres; por ejemplo, B‘10101 ’. Otro tipo de datos de cadena de bits de longitud variable llamado BINARY LARGE OBJECT o BLOB también está disponible para especificar columnas que tienen valores binarios grandes, como imágenes. En cuanto a CLOB, la longitud máxima de un BLOB se puede especificar en kilobits (K), megabits (M) o gigabits (G).
Por ejemplo, BLOB (30G) especifica una longitud máxima de 30 gigabits.

■ Un tipo de datos booleano tiene los valores tradicionales de VERDADERO o FALSO. En SQL, debido a la presencia de valores NULL, se utiliza una lógica de tres valores, por lo

que un tercer valor posible para un tipo de datos booleano es UNKNOWN. Discutimos la necesidad de DESCONOCIDO y la lógica de tres valores en el Capítulo 7.
■ El tipo de datos FECHA tiene diez posiciones y sus componentes son AÑO, MES y DÍA en el formato AAAA-MM-DD. El tipo de datos TIEMPO tiene al menos ocho posiciones, con los componentes HORA, MINUTO y SEGUNDO en el formato HH: MM: SS. La implementación de SQL solo debe permitir fechas y horas válidas. Esto implica que los meses deben estar entre 1 y 12 y los días deben estar entre 01 y 31; además, un día debe ser válido para el mes correspondiente. La comparación <(menor que) se puede usar con fechas u horas; una fecha anterior se considera más pequeña que una fecha posterior y, de manera similar, con la hora. Los valores literales están representados por cadenas entre comillas simples precedidas por la palabra clave DATE o TIME; por ejemplo, DATE "2014-09-27" o TIME "09: 12: 47 ". Además, un tipo de datos TIEMPO (i), donde i se denomina precisión de segundos de fracción de tiempo, especifica i + 1 posiciones adicionales para TIEMPO: una posición para un carácter separador de punto adicional (.), Y posiciones i para especificar fracciones decimales de un segundo. Un tipo de datos HORA CON ZONA HORARIA incluye seis posiciones adicionales para especificar el desplazamiento de la zona horaria universal estándar, que está en el rango de +13: 00 a –12: 59 en unidades de HORAS: MINUTOS. Si CON HUSO HORARIO no está incluido, el valor predeterminado es el huso horario local para la sesión SQL. Algunos tipos de datos adicionales se analizan a continuación. La lista de tipos discutidos aquí no es exhaustiva; diferentes implementaciones han agregado más tipos de datos a SQL.
■ Un tipo de datos de marca de tiempo (TIMESTAMP) incluye los campos DATE y TIME, más un mínimo de seis posiciones para fracciones decimales de segundos y un calificador WITH TIME ZONE opcional. Los valores literales están representados por comillas simples precedidas por la palabra clave TIMESTAMP, con un espacio en blanco entre los datos y el tiempo; por ejemplo, TIMESTAMP '2014-09-27 09: 12: 47.648302'.
■ Otro tipo de datos relacionado con DATE, TIME y TIMESTAMP es el tipo de datos INTERVAL. Esto especifica un intervalo, un valor relativo que se puede usar para incrementar o disminuir un valor absoluto de una fecha, hora o marca de tiempo. Los intervalos están calificados para ser intervalos AÑO / MES o intervalos DÍA / HORA. El formato de DATE, TIME y TIMESTAMP se puede considerar como un tipo especial de cadena. Por lo tanto, generalmente se pueden usar en comparaciones de cadenas al ser lanzados (o coaccionados o convertidos) en cadenas equivalentes. Es posible especificar el tipo de datos de cada atributo directamente, como en la Figura 6.1; alternativamente, se puede declarar un dominio y el nombre de dominio se puede utilizar con la especificación de atributo. Esto facilita el cambio del tipo de datos de un dominio que utilizan numerosos atributos en un esquema y mejora la legibilidad del esquema. Por ejemplo, podemos crear un dominio SSN_TYPE mediante la siguiente declaración:
CREAR DOMINIO SSN_TYPE COMO CHAR (9); Podemos usar SSN_TYPE en lugar de CHAR (9) en la Figura 6.1 para los atributos Ssn y Super_ssn de EMPLOYEE, Mgr_ssn de DEPARTMENT, Essn de WORKS_ON y Essn de DEPENDENT. Un dominio también puede tener una especificación predeterminada opcional a través de una cláusula DEFAULT, como veremos más adelante para los atributos. Tenga en cuenta que es posible que los dominios no estén disponibles en algunas implementaciones de SQL. En SQL, también hay un comando CREATE TYPE, que se puede utilizar para crear tipos definidos por el usuario o UDT. Estos se pueden utilizar como tipos de datos para atributos o como base para crear tablas. Discutiremos CREATE TYPE en detalle en el Capítulo 12, porque a menudo se usa junto con la especificación de características de la base de datos de objetos.
que se han incorporado en versiones más recientes de SQL.

6.2 Especificación de restricciones en SQL

Esta sección describe las restricciones básicas que se pueden especificar en SQL como parte de la creación de tablas. Estos incluyen restricciones de integridad clave y referencial, restricciones en dominios de atributos y NULL, y restricciones en tuplas individuales dentro de una relación que usa la cláusula CHECK. Analizamos la especificación de restricciones más generales, llamadas aserciones, en el Capítulo 7.

6.2.1 Especificación de restricciones de atributo y valores predeterminados de atributo

Debido a que SQL permite NULL como valores de atributo, se puede especificar una restricción NOT NULL si no se permite NULL para un atributo en particular. Esto siempre se especifica implícitamente para los atributos que forman parte de la clave primaria de cada relación, pero se puede especificar para cualquier otro atributo cuyos valores no deben ser NULL, como se muestra en la Figura 6.1. También es posible definir un valor por defecto para un atributo añadiendo la cláusula DEFAULT <valor> a una definición de atributo. El valor predeterminado se incluye en cualquier tupla nueva si no se proporciona un valor explícito para ese atributo. La figura 6.2 ilustra un ejemplo de cómo especificar un gerente predeterminado para un nuevo departamento y un departamento predeterminado para un nuevo empleado. Si no se especifica una cláusula predeterminada, el valor predeterminado predeterminado es NULL para los atributos que no tienen la restricción NOT NULL. Otro tipo de restricción puede restringir los valores de atributo o dominio utilizando la cláusula CHECK que sigue a una definición de atributo o dominio.6 Por ejemplo, suponga que los números de departamento están restringidos a números enteros entre 1 y 20; luego, podemos cambiar la declaración de atributo de Dnumber en la tabla DEPARTMENT (ver Figura 6.1) a lo siguiente: Dnumber INT NOT NULL CHECK (Dnumber> 0 AND Dnumber <21); La cláusula CHECK también se puede utilizar junto con la sentencia CREATE DOMAIN. Por ejemplo, podemos escribir la siguiente declaración: CREATE DOMAIN D_NUM AS INTEGER CHECK (D_NUM> 0 AND D_NUM <21); A continuación, podemos utilizar el dominio creado D_NUM como el tipo de atributo para todos los atributos que se refieren a los números de departamento en la Figura 6.1, como Dnumber de DEPARTAMENTO, Dnum de PROYECTO, Dno de EMPLEADO, etc.

6.2.2 Especificación de restricciones de integridad clave y referencial

Debido a que las claves y las restricciones de integridad referencial son muy importantes, existen cláusulas especiales dentro de la instrucción CREATE TABLE para especificarlas. En la Figura 6.1.7 se muestran algunos ejemplos para ilustrar la especificación de claves y la integridad referencial. La cláusula PRIMARY KEY especifica uno o más atributos que componen la clave primaria de una relación. Si una clave primaria tiene un solo atributo, la cláusula puede seguir al atributo directamente. Por ejemplo, la clave primaria de DEPARTAMENTO se puede especificar de la siguiente manera (en lugar de la forma en que se especifica en la Figura 6.1): Dnumber INT PRIMARY KEY, la cláusula UNIQUE especifica claves alternativas (únicas), también conocidas como claves candidatas como se ilustra en la Declaraciones de la tabla DEPARTAMENTO y PROYECTO en la Figura 6.1. La cláusula UNIQUE también se puede especificar directamente para una clave única si es un atributo único, como en el siguiente ejemplo: Dname VARCHAR (15) UNIQUE, la integridad referencial se especifica mediante la cláusula FOREIGN KEY, como se muestra en la Figura 6.1. Como discutimos en la Sección 5.2.4, una restricción de integridad referencial se puede violar cuando se insertan o eliminan tuplas, o cuando se actualiza un valor de atributo de clave externa o clave primaria. La acción predeterminada que toma SQL para una violación de integridad es rechazar la operación de actualización que causará una violación, lo que se conoce como la opción RESTRICT. Sin embargo, el diseñador de esquemas puede especificar una acción alternativa a realizar adjuntando una cláusula de acción activada referencial a cualquier restricción de clave externa. Las

opciones incluyen SET NULL, CASCADE y SET DEFAULT. Una opción debe estar calificada con ON DELETE o ON UPDATE. Ilustramos esto con los ejemplos que se muestran en la Figura 6.2. Aquí, el diseñador de la base de datos elige ON DELETE SET NULL y ON UPDATE CASCADE para la clave externa Super_ssn de EMPLOYEE. Esto significa que si se elimina la tupla de un empleado supervisor, el valor de Super_ssn se establece automáticamente en NULL para todas las tuplas de empleados que hacían referencia a la tupla de empleados eliminada. Por otro lado, si el valor de Ssn para un empleado supervisor se actualiza (digamos, porque se ingresó incorrectamente), el nuevo valor se envía en cascada a Super_ssn para todas las tuplas de empleados que hacen referencia a la tupla de empleados actualizada.8 En general, la acción realizada por el DBMS para SET NULL o SET DEFAULT es el mismo para ON DELETE y ON UPDATE: el valor de los atributos de referencia afectados se cambia a NULL para SET NULL y al valor predeterminado especificado del atributo de referencia para SET DEFAULT. La acción de CASCADE ON DELETE es eliminar todas las tuplas de referencia, mientras que la acción de CASCADE ON UPDATE es cambiar el valor de los atributos de clave externa de referencia al valor de clave primaria actualizado (nuevo) para todas las tuplas de referencia. Es responsabilidad del diseñador de la base de datos elegir la acción apropiada y especificarla en el esquema de la base de datos. Como regla general, la opción CASCADE es adecuada para relaciones de “relación” (consulte la Sección 9.1), como WORKS_ON; para relaciones que representan atributos de varios valores, como DEPT_LOCATIONS; y para relaciones que representan tipos de entidades débiles, como DEPENDENT.

6.2.3 Dar nombre a las restricciones

La figura 6.2 también ilustra cómo una restricción puede recibir un nombre de restricción, siguiendo la palabra clave CONSTRAINT. Los nombres de todas las restricciones dentro de un esquema particular deben ser únicos. Un nombre de restricción se usa para identificar una restricción en particular en caso de que la restricción deba eliminarse más tarde y reemplazarse por otra restricción, como discutimos en el Capítulo 7. Dar nombres a las restricciones es opcional. También es posible diferir temporalmente una restricción hasta el final de una transacción, como veremos en el capítulo 20 cuando presentemos conceptos de transacción.

6.2.4 Especificación de restricciones en tuplas mediante CHECK

Además de las restricciones de integridad clave y referencial, que se especifican mediante palabras clave especiales, se pueden especificar otras restricciones de tabla mediante cláusulas CHECK adicionales al final de una instrucción CREATE TABLE. Estas pueden denominarse restricciones basadas en filas porque se aplican a cada fila individualmente y se comprueban cada vez que se inserta o modifica una fila. Por ejemplo, suponga que la tabla DEPARTMENT en la Figura 6.1 tiene un atributo adicional Dept_create_date, que almacena la fecha en que se creó el departamento. Entonces podríamos agregar la siguiente cláusula CHECK en el
final de la instrucción CREATE TABLE para la tabla DEPARTMENT para asegurarse de que la fecha de inicio de un gerente sea posterior a la fecha de creación del departamento. VERIFICAR (Dpto_create_date <= Mgr_start_date); La cláusula CHECK también se puede utilizar para especificar restricciones más generales utilizando la sentencia CREATE ASSERTION de SQL. Discutimos esto en el Capítulo 7 porque requiere todo el poder de las consultas, que se analizan en las Secciones 6.3 y 7.1.

6.3 Consultas de recuperación básicas en SQL

SQL tiene una declaración básica para recuperar información de una base de datos: la declaración SELECT. La instrucción SELECT no es lo mismo que la operación SELECT del álgebra relacional, que discutiremos en el Capítulo 8. Hay muchas opciones y

variantes de la instrucción SELECT en SQL, por lo que presentaremos sus características gradualmente. Usaremos consultas de ejemplo especificadas en el esquema de la Figura 5.5 y nos referiremos al estado de la base de datos de muestra que se muestra en la Figura 5.6 para mostrar los resultados de algunas de estas consultas. En esta sección, presentamos las características de SQL para consultas de recuperación simples. Las características de SQL para especificar consultas de recuperación más complejas se presentan en la Sección 7.1. Antes de continuar, debemos señalar una distinción importante entre el modelo SQL práctico y el modelo relacional formal discutido en el Capítulo 5: SQL permite que una tabla (relación) tenga dos o más tuplas que son idénticas en todos sus valores de atributo. Por tanto, en general, una tabla SQL no es un conjunto de tuplas, porque un conjunto no permite dos miembros idénticos; más bien, es un conjunto múltiple (a veces llamado bolsa) de tuplas. Algunas relaciones SQL están limitadas a ser conjuntos porque se ha declarado una restricción de clave o porque la opción DISTINCT se ha utilizado con la instrucción SELECT (descrita más adelante en esta sección). Debemos ser conscientes de esta distinción al analizar los ejemplos.

6.3.1 La estructura SELECT-FROM-WHERE de las consultas SQL básicas

Las consultas en SQL pueden ser muy complejas. Comenzaremos con consultas simples y luego avanzaremos a otras más complejas paso a paso. La forma básica de la instrucción SELECT, a veces denominada mapeo o bloque select-from-where, está formada por las tres cláusulas SELECT, FROM y WHERE y tiene la siguiente forma:

```
SELECT   <attribute list>
FROM     <table list>
WHERE    <condition>;
```

where

- <attribute list> is a list of attribute names whose values are to be retrieved by the query.
- <table list> is a list of the relation names required to process the query.
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query.

En SQL, los operadores de comparación lógica básica para comparar valores de atributo entre sí y con constantes literales son =, <, <=,>,> = y <>. Éstos corresponden a los operadores de álgebra relacional =, <, ≤,>, ≥ y ≠, respectivamente, y a los operadores del lenguaje de programación C / C ++ =, <, <=,>,> = y! =. La principal diferencia sintáctica es el operador no igual. SQL tiene operadores de comparación adicionales que presentaremos gradualmente. Ilustramos la instrucción SELECT básica en SQL con algunas consultas de muestra. Las consultas se etiquetan aquí con los mismos números de consulta utilizados en el Capítulo 8 para facilitar la referencia cruzada.
Consulta 0. Recupera la fecha de nacimiento y la dirección de los empleados) cuyo nombre es "John B. Smith".

```
Q0:   SELECT   Bdate, Address
      FROM     EMPLOYEE
      WHERE    Fname = 'John' AND Minit = 'B' AND Lname = 'Smith';
```

Esta consulta solo involucra la relación EMPLOYEE listada en la cláusula FROM. La consulta selecciona las tuplas EMPLOYEE individuales que satisfacen la condición de la cláusula WHERE, luego proyecta el resultado en los atributos Bdate y Address enumerados en la cláusula SELECT.
La cláusula SELECT de SQL especifica los atributos cuyos valores se recuperarán, que se denominan atributos de proyección en álgebra relacional (consulte el Capítulo 8) y la cláusula WHERE especifica la condición booleana que debe ser verdadera para cualquier tupla recuperada, que se conoce como la condición de selección en álgebra relacional. La

Figura 6.3 (a) muestra el resultado de la consulta Q0 en la base de datos de la Figura 5.6. Podemos pensar en una variable tupla implícita o un iterador en la consulta SQL que recorre o recorre cada tupla individual en la tabla EMPLOYEE y evalúa la condición en la cláusula WHERE. Solo se seleccionan aquellas tuplas que satisfacen la condición, es decir, aquellas tuplas para las que la condición se evalúa como VERDADERA después de sustituir sus valores de atributo correspondientes.
Consulta 1. Obtenga el nombre y la dirección de todos los empleados que trabajan para el departamento de "Investigación"

```
Q1:   SELECT   Fname, Lname, Address
      FROM     EMPLOYEE, DEPARTMENT
      WHERE    Dname = 'Research' AND Dnumber = Dno;
```

En la cláusula WHERE de Q1, la condición Dname = "Investigación" es una condición de selección que elige la tupla particular de interés en la tabla DEPARTMENT, porque Dname es un atributo de DEPARTMENT. La condición Dnumber = Dno se denomina condición de unión, porque combina dos tuplas: una de DEPARTMENT y otra de EMPLOYEE, siempre que el valor de Dnumber en DEPARTMENT sea igual al valor de Dno en EMPLOYEE. El resultado de la consulta Q1 se muestra en la Figura 6.3 (b). En general, se puede especificar cualquier número de condiciones de selección y combinación en una sola consulta SQL. Una consulta que involucra solo condiciones de selección y unión más atributos de proyección se conoce como consulta de selección-proyecto-unión. El siguiente ejemplo es una consulta select-project-join con dos condiciones de combinación.
Consulta 2. Para cada proyecto ubicado en "Stafford", indique el número de proyecto, el número del departamento de control y el apellido, la dirección y la fecha de nacimiento del gerente del departamento.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum = Dnumber AND Mgr_ssn = Ssn AND
               Plocation = 'Stafford'
```

6.3.2 Nombres de atributos ambiguos, alias, cambio de nombre y variables de tupla

En SQL, se puede usar el mismo nombre para dos (o más) atributos siempre que los atributos estén en tablas diferentes. Si este es el caso, y una consulta de múltiples tablas hace referencia a dos o más atributos con el mismo nombre, debemos calificar el nombre del atributo con el nombre de la relación para evitar ambigüedad. Esto se hace anteponiendo el nombre de la relación al nombre del atributo y separando los dos por un punto. Los nombres de atributo completamente calificados se pueden usar para mayor claridad incluso si no hay ambigüedad en los nombres de atributo. La ambigüedad de los nombres de los atributos también surge en el caso de consultas que hacen referencia a la misma relación dos veces. También es posible cambiar el nombre de los atributos de relación dentro de la consulta en SQL dándoles alias. Siempre que se asignen uno o más alias a una relación, podemos usar estos nombres para representar diferentes referencias a esa misma relación. Esto permite múltiples referencias a la misma relación dentro de una consulta. Podemos usar este mecanismo de cambio de nombre o nombre de alias en cualquier consulta SQL para especificar variables de tupla para cada tabla en la cláusula WHERE, ya sea que la misma relación necesite o no ser referenciada más de una vez. De hecho, se recomienda esta práctica ya que da como resultado consultas más fáciles de comprender.

6.3.3 Cláusula WHERE no especificada y uso del asterisco

Una cláusula WHERE faltante indica que no hay condición en la selección de tuplas; por lo tanto, todas las tuplas de la relación especificada en la cláusula FROM califican y se seleccionan para el resultado de la consulta. Si se especifica más de una relación en la

cláusula FROM y no hay cláusula WHERE, entonces se selecciona el PRODUCTO CRUZADO (todas las combinaciones posibles de tuplas) de estas relaciones.

Es extremadamente importante especificar cada selección y condición de combinación en la cláusula WHERE; si se pasa por alto alguna de estas condiciones, pueden resultar relaciones incorrectas y muy grandes.
Para recuperar todos los valores de atributo de las tuplas seleccionadas, no tenemos que listar los nombres de atributo explícitamente en SQL; simplemente especificamos un asterisco (*), que representa todos los atributos. El * también puede tener como prefijo el nombre de la relación o alias,

6.3.4 Tablas como conjuntos en SQL
Como mencionamos anteriormente, SQL generalmente trata una tabla no como un conjunto, sino más bien como un conjunto múltiple; las tuplas duplicadas pueden aparecer más de una vez en una tabla y en el resultado de una consulta. SQL no elimina automáticamente las tuplas duplicadas en los resultados de las consultas, por las siguientes razones:
■ La eliminación de duplicados es una operación costosa. Una forma de implementarlo es ordenar las tuplas primero y luego eliminar los duplicados.
■ Es posible que el usuario desee ver tuplas duplicadas en el resultado de una consulta.
■ Cuando se aplica una función agregada (consulte la Sección 7.1.7) a tuplas, en la mayoría de los casos no queremos eliminar los duplicados.

Una tabla SQL con una clave está restringida a ser un conjunto, ya que el valor de la clave debe ser distinto en cada tupla, Si queremos eliminar las tuplas duplicadas del resultado de una consulta SQL, usamos la palabra clave DISTINCT en la cláusula SELECT , lo que significa que solo deben permanecer tuplas distintas en el resultado. En general, una consulta con SELECT DISTINCT elimina los duplicados, mientras que una consulta con SELECT ALL no lo hace.

Las relaciones resultantes de estas operaciones de conjuntos son conjuntos de tuplas; es decir, las tuplas duplicadas se eliminan del resultado. Estas operaciones de conjunto se aplican solo a relaciones compatibles con tipos, por lo que debemos asegurarnos de que las dos relaciones en las que aplicamos la operación tengan los mismos atributos y que los atributos aparezcan en el mismo orden en ambas relaciones. SQL también tiene las correspondientes operaciones de varios conjuntos, seguidas de la palabra clave ALL (UNION ALL, EXCEPT ALL, INTERSECT ALL). Sus resultados son conjuntos múltiples (no se eliminan los duplicados).

6.3.5 Operadores aritméticos y de coincidencia de patrones de subcadenas
La primera característica permite condiciones de comparación solo en partes de una cadena de caracteres, utilizando el operador de comparación LIKE. Esto se puede utilizar para hacer coincidir patrones de cuerdas. Las cadenas parciales se especifican mediante dos caracteres reservados:% reemplaza un número arbitrario de cero o más caracteres, y el guión bajo (_) reemplaza un solo carácter.
Si se necesita un guión bajo o% como carácter literal en la cadena, el carácter debe ir precedido de un carácter de escape, que se especifica después de la cadena utilizando la palabra clave ESCAPE. Por ejemplo, "AB \ _CD \% EF" ESCAPE "\" representa la cadena literal "AB_CD% EF" porque \ se especifica como el carácter de escape. Cualquier carácter que no se utilice en la cadena se puede elegir como carácter de escape.
Además, necesitamos una regla para especificar apóstrofos o comillas simples ("") si se van a incluir en una cadena porque se utilizan para comenzar y terminar cadenas. Si se

necesita un apóstrofo (’), se representa como dos apóstrofos consecutivos (") para que no se interprete
como final de la cadena. Observe que la comparación de subcadenas implica que los valores de los atributos no son valores atómicos (indivisibles), como asumimos en el modelo relacional formal,

Otra característica permite el uso de aritmética en consultas. Los operadores aritméticos estándar para la suma (+), la resta (-), la multiplicación (*) y la división (/) se pueden aplicar a valores numéricos o atributos con dominios numéricos.

ara los tipos de datos de cadena, el operador de concatenación || se puede utilizar en una consulta para agregar dos valores de cadena. Para los tipos de datos de fecha, hora, marca de tiempo e intervalo, los operadores incluyen incrementar (+) o disminuir (-) una fecha, hora o marca de tiempo en un intervalo. Además, un valor de intervalo es el resultado de la diferencia entre dos valores de fecha, hora o marca de tiempo. Otro operador de comparación, que puede utilizarse por conveniencia, es BETWEEN.

6.3.6 Orden de los resultados de la consulta
SQL permite al usuario ordenar las tuplas en el resultado de una consulta por los valores de uno o más de los atributos que aparecen en el resultado de la consulta, utilizando la cláusula ORDER BY. Esto se ilustra en la Consulta 15.
Consulta 15. Recupere una lista de empleados y los proyectos en los que están trabajando, ordenados por departamento y, dentro de cada departamento, ordenados alfabéticamente por apellido y luego por nombre.
P15: SELECCIONE D.Dname, E.Lname, E.Fname, P.Pname
DEL DEPARTAMENTO COMO D, EMPLEADO COMO E, TRABAJA EN COMO W, PROYECTO COMO P DONDE D.Dnumber = E.Dno Y E.Ssn = W.Essn Y W.Pno = P.Pnumber ORDEN POR D.Dname, E.Lname, E.Fname;
El orden predeterminado es el orden ascendente de valores. Podemos especificar la palabra clave DESC si queremos ver el resultado en un orden descendente de valores. La palabra clave ASC se puede utilizar para especificar el orden ascendente de forma explícita. Por ejemplo, si queremos orden alfabético descendente en Dname y orden ascendente en Lname, Fname, la cláusula ORDER BY de Q15 puede escribirse como ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC

6.3.7 Discusión y resumen de consultas básicas de recuperación de SQL
Una consulta de recuperación simple en SQL puede constar de hasta cuatro cláusulas, pero solo las dos primeras, SELECT y FROM, son obligatorias. Las cláusulas se especifican en el siguiente orden, siendo las cláusulas entre corchetes […] opcionales:
SELECCIONAR <lista de atributos>
DESDE <lista de tablas>
[DONDE <condición>]
[ORDER BY <lista de atributos>];

La cláusula SELECT enumera los atributos que se recuperarán y la cláusula FROM especifica todas las relaciones (tablas) necesarias en la consulta simple. La cláusula WHERE identifica las condiciones para seleccionar las tuplas de estas relaciones, incluidas las condiciones de unión si es necesario. ORDER BY especifica un orden para mostrar los resultados de una consulta. En la Sección 7.1.8 se describirán dos cláusulas adicionales GROUP BY y HAVING.
En el Capítulo 7, presentaremos características más complejas de las consultas de recuperación de SQL. Estos incluyen lo siguiente: consultas anidadas que permiten incluir

una consulta como parte de otra consulta; funciones agregadas que se utilizan para proporcionar resúmenes de la información en las tablas; dos cláusulas adicionales (GROUP BY y HAVING) que se pueden utilizar para proporcionar poder adicional para agregar funciones; y varios tipos de combinaciones que pueden combinar registros de varias tablas de diferentes formas.

6.4 INSERTAR, ELIMINAR y ACTUALIZAR

Declaraciones en SQL En SQL, se pueden usar tres comandos para modificar la base de datos: INSERT, DELETE y UPDATE.

6.4.1 El comando INSERT

En su forma más simple, INSERT se usa para agregar una sola tupla (fila) a una relación (tabla). Debemos especificar el nombre de la relación y una lista de valores para la tupla. Los valores deben aparecer en el mismo orden en que se especificaron los atributos correspondientes en el comando CREATE TABLE.

Una segunda forma de la instrucción INSERT permite al usuario especificar nombres de atributos explícitos que corresponden a los valores proporcionados en el comando INSERT. Esto es útil si una relación tiene muchos atributos, pero solo a algunos de esos atributos se les asignan valores en la nueva tupla. Sin embargo, los valores deben incluir todos los atributos con especificación NOT NULL y sin valor predeterminado. Los atributos con valores NULL permitidos o DEFAULT son los que se pueden omitir.

Los atributos no especificados en U1A se establecen en DEFAULT o NULL, y los valores se enumeran en el mismo orden en que se enumeran los atributos en el comando INSERT. También es posible insertar en una relación múltiples tuplas separadas por comas en un solo comando INSERT. Los valores de los atributos que forman cada tupla están entre paréntesis.

Un DBMS que implemente SQL por completo debe admitir y hacer cumplir todas las restricciones de integridad que se pueden especificar en el DDL. Una variación del comando INSERT inserta múltiples tuplas en una relación junto con la creación de la relación y la carga con el resultado de una consulta. La mayoría de los DBMS tienen herramientas de carga masiva que permiten al usuario cargar datos formateados desde un archivo en una tabla sin tener que escribir una gran cantidad de comandos INSERT. El usuario también puede escribir un programa para leer cada registro en el archivo, formatearlo como una fila
en la tabla e insértelo usando las construcciones de bucle de un lenguaje de programación . Otra variación para cargar datos es crear una nueva tabla TNEW que tenga la mismos atributos que una tabla T existente, y cargue algunos de los datos actualmente en T en TNEW. La sintaxis para hacer esto usa la cláusula LIKE.

6.4.2 El comando DELETE

El comando DELETE elimina las tuplas de una relación. Incluye una cláusula WHERE, similar a la utilizada en una consulta SQL, para seleccionar las tuplas que se eliminarán. Las tuplas se eliminan explícitamente de una sola tabla a la vez. Sin embargo, la eliminación puede propagarse a tuplas en otras relaciones si las acciones desencadenadas referenciales se especifican en las restricciones de integridad referencial del DDL. Dependiendo del número de tuplas seleccionadas por la condición en la cláusula WHERE, cero , una o varias tuplas se pueden eliminar con un solo comando DELETE. Una cláusula WHERE faltante especifica que todas las tuplas de la relación deben

eliminarse; sin embargo, la tabla permanece en la base de datos como una tabla vacía. Debemos usar el comando DROP TABLE para eliminar la definición de la tabla.

6.4.3 El comando UPDATE

El comando UPDATE se usa para modificar los valores de los atributos de una o más tuplas seleccionadas. Como en el comando DELETE, una cláusula WHERE en el comando UPDATE selecciona las tuplas que se modificarán a partir de una sola relación. Sin embargo, la actualización de un valor de clave primaria puede propagarse a los valores de clave externa de tuplas en otras relaciones si dicha acción desencadenada referencial se especifica en las restricciones de integridad referencial del DDL. Una cláusula SET adicional en el comando UPDATE especifica los atributos que se modificarán y sus nuevos valores. Se pueden modificar varias tuplas con un solo comando UPDATE, ambién es posible especificar NULL o DEFAULT como el nuevo valor de atributo. Para modificar múltiples relaciones, debemos emitir varios comandos UPDATE.

6.5 Funciones adicionales de SQL

SQL tiene una serie de características adicionales que no hemos descrito en este capítulo, pero que discutimos en otra parte del libro. Estos son los siguientes:

■ En el Capítulo 7, que es una continuación de este capítulo, presentaremos las siguientes características de SQL: varias técnicas para especificar consultas de recuperación complejas, incluidas consultas anidadas, funciones agregadas, agrupación, tablas unidas, combinaciones externas, declaraciones de casos y consultas recursivas. ; Vistas, disparadores y afirmaciones de SQL; y comandos para la modificación del esquema.

■ SQL tiene varias técnicas para escribir programas en varios lenguajes de programación que incluyen sentencias SQL para acceder a una o más bases de datos. Estos incluyen SQL integrado (y dinámico), SQL / CLI (Interfaz de nivel de llamada) y su predecesor ODBC (Conectividad de base de datos abierta) y SQL / PSM (Módulos almacenados persistentes). Discutimos estas técnicas en el Capítulo 10.
también describe cómo acceder a bases de datos SQL a través del lenguaje de programación Java usando JDBC y SQLJ.

■ Cada RDBMS comercial tendrá, además de los comandos SQL, un conjunto de comandos para especificar parámetros de diseño de bases de datos físicas, estructuras de archivos para relaciones y rutas de acceso como índices. En el Capítulo 2, llamamos a estos comandos lenguaje de definición de almacenamiento (SDL). Las versiones anteriores de SQL tenían comandos para crear índices, pero se eliminaron del lenguaje porque no estaban en el nivel de esquema conceptual. Muchos sistemas todavía tienen los comandos CREATE INDEX; pero requieren un privilegio especial. Describimos esto en el Capítulo 17.

■ SQL tiene comandos de control de transacciones. Se utilizan para especificar unidades de procesamiento de bases de datos con fines de control y recuperación de simultaneidad. Analizamos estos comandos en el Capítulo 20 después de analizar el concepto de transacciones con más detalle.

■ SQL tiene construcciones de lenguaje para especificar la concesión y revocación de privilegios a los usuarios. Los privilegios suelen corresponder al derecho a utilizar determinados comandos SQL para acceder a determinadas relaciones. A cada relación se le asigna un propietario, y el propietario o el personal del DBA pueden otorgar a los usuarios seleccionados el privilegio de usar una declaración SQL, como SELECT, INSERT, DELETE o UPDATE, para acceder a la relación. Además, el personal de DBA puede otorgar privilegios para crear esquemas, tablas o vistas a ciertos usuarios. Estos

comandos SQL, llamados GRANT y REVOKE, se analizan en el Capítulo 20, donde discutimos la seguridad y autorización de la base de datos.
■ SQL tiene construcciones de lenguaje para crear disparadores. Por lo general, se denominan técnicas de base de datos activa, ya que especifican acciones que se desencadenan automáticamente por eventos como actualizaciones de la base de datos. Discutimos estas características en la Sección 26.1, donde discutimos conceptos de bases de datos activas.
■ SQL ha incorporado muchas características de modelos orientados a objetos para tener capacidades más poderosas, lo que lleva a sistemas relacionales mejorados conocidos como relacionales a objetos. En el Capítulo 12 se analizan capacidades como la creación de atributos de estructura compleja, la especificación de tipos de datos abstractos (denominados UDT o tipos definidos por el usuario) para atributos y tablas, la creación de identificadores de objeto para hacer referencia a tuplas y la especificación de operaciones sobre tipos.
■ SQL y las bases de datos relacionales pueden interactuar con nuevas tecnologías como XML (consulte el Capítulo 13) y OLAP / almacenes de datos (Capítulo 29).

6.6 Resumen

En este capítulo, presentamos el lenguaje de base de datos SQL. Este lenguaje y sus variaciones se han implementado como interfaces para muchos DBMS relacionales comerciales, incluido Oracle de Oracle; DB2 de ibm; SQL Server de Microsoft; y muchos otros sistemas, incluidos Sybase e INGRES. Algunos sistemas de código abierto también proporcionan SQL, como MySQL y PostgreSQL. La versión original de SQL se implementó en el DBMS experimental llamado SYSTEM R, que fue desarrollado en IBM Research. SQL está diseñado para ser un lenguaje integral que incluye declaraciones para la definición de datos, consultas, actualizaciones, especificación de restricciones y definición de vista. En este capítulo analizamos las siguientes características de SQL: los comandos de definición de datos para crear tablas, los tipos de datos básicos de SQL, los comandos para la especificación de restricciones, las consultas de recuperación simples y los comandos de actualización de la base de datos. En el próximo capítulo, presentaremos las siguientes características de SQL: consultas de recuperación complejas; puntos de vista; disparadores y afirmaciones; y comandos de modificación de esquema.