Índice

sario comprar una plataforma hardware mayor, quizás incluso una máquina dedicada a ejecutar en exclusiva el SGBD. Estas compras de hardware adicional hacen que se incrementen los costes.

Costes de conversión

En algunas situaciones, el coste del SGBD y del hardware adicional puede ser insignificante si lo comparamos con el coste de convertir las aplicaciones existentes para que se ejecuten sobre el nuevo SGBD y sobre la nueva plataforma hardware. Este coste también incluye el coste de formar al personal en la utilización de los nuevos sistemas y, posiblemente, la contratación de personal especializado como ayuda durante la conversión y la operación del sistema. Este coste es una de las razones principales por las que algunas organizaciones se sienten atadas a sus sistemas actuales y no se deciden a cambiar a tecnologías de base de datos más modernas. Algunas veces se utiliza el término **sistema heredado** para referirse a un sistema más antiguo y usualmente inferior.

Prestaciones

Normalmente, los sistemas basados en archivos se escriben para una aplicación específica, como por ejemplo una aplicación de facturación. Como resultado, las prestaciones suelen ser muy buenas. Por el contrario, un SGBD se escribe para constituir una solución más general, con el fin de que puedan utilizarlo muchas aplicaciones y no sólo una aplicación concreta. El efecto es que algunas aplicaciones pueden ejecutarse más lentamente de lo que solían.

Mayor impacto de los fallos

La centralización de los recursos implementa la vulnerabilidad del sistema. Puesto que todos los usuarios y aplicaciones dependen de la disponibilidad del SGBD, el fallo de ciertos componentes puede hacer que se detengan todas las operaciones.

Resumen del capítulo

- El **sistema de gestión de bases de datos** (SGBD) constituye hoy en día la base fundamental de los sistemas de información y ha cambiado de modo radical la forma en que muchas organizaciones operan. Los sistemas de base de datos continúan siendo un área muy activa de investigación y son todavía muchos los problemas significativos que hay que resolver de manera satisfactoria.
- Los predecesores de los SGBD eran los **sistemas basados en archivos**, que son una colección de programas de aplicación que realizan una serie de servicios para los usuarios finales, usualmente la generación de informes. Cada programa define y gestiona sus propios datos. Aunque los sistemas basados en archivos constituyeron una gran mejora con respecto a los sistemas de archivo manual, siguen presentado problemas significativos, principalmente la redundancia de datos existente y la dependencia entre los programas y los datos.
- La técnica de bases de datos hizo su aparición para resolver los problemas asociados con los sistemas basados en archivos. Una **base de datos** es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, diseñada para satisfacer las necesidades de información de una organización. Un **SGBD** es un sistema software que permite a los usuarios definir, emplear, mantener y controlar el acceso a la base de datos. Un **programa de aplicación** es un programa informático que interactúa con la base de datos emitiendo solicitudes (normalmente instrucciones SQL) dirigidas al SGBD. Normalmente se utiliza el término **sistema de bases de datos** para definir una colección de programas de aplicación que interactúan con la base de datos, junto con el SGBD y la propia base de datos.
- Todos los accesos a la base de datos se realizan a través del SGBD. El SGBD proporciona un **lenguaje de definición de datos** (DDL, Data Definition Language), que permite a los usuarios definir la base de datos, y un **lenguaje de manipulación de datos** (DML, Data Manipulation Language), que permite a los usuarios insertar, actualizar, borrar y extraer datos de la base de datos.

- El SGBD proporciona un acceso controlado a la base de datos. Proporciona mecanismos de seguridad, integridad, control de concurrencia y control de recuperación, así como un catálogo accesible por el usuario. También proporciona un mecanismo de vistas para simplificar los datos con los que el usuario tiene que tratar.
- El entorno SGBD está compuesto de hardware (la computadora), de software (el SGBD, el sistema operativo y los sistemas de aplicación), de datos, de procedimientos y de personas. Las personas que se relacionan con el SGBD son los administradores de datos y de base de datos, los diseñadores de bases de datos, los desarrolladores de aplicaciones y los usuarios finales.
- Las raíces de los SGBD están en los sistemas basados en archivos. Los sistemas jerárquicos y CODASYL representan la primera generación de sistemas de gestión de bases de datos. El **modelo jerárquico** está representado por IMS (Information Management System), mientras que el **modelo en red** o **CODASYL** está ejemplificado por IDS (Integrate Data Store), ambos desarrollados a mediados de la década de 1960. El **modelo relacional**, propuesto por E. F. Codd en 1970, representa la segunda generación de sistemas de gestión de bases de datos. Este modelo tuvo una gran influencia sobre la comunidad de los SGBD y hoy en día existen más de 100 SGBD relacionales. La tercera generación de sistemas de gestión de bases de datos está representada por los SGBD **objeto-relacional** y los SGBD **orientados a objetos**.
- Entre las ventajas de la técnica de bases de datos podemos citar el control de la redundancia de los datos, la coherencia de los datos, la compartición de datos y unos mejores mecanismos de seguridad e integridad. Entre sus desventajas cabe resaltar la complejidad, el coste, la reducción de las prestaciones y el mayor impacto de los fallos sobre el sistema.

Cuestiones de repaso

1.1. Proporcione cuatro ejemplos de sistemas de bases de datos distintos de los enumerados en la Sección 1.1.

1.2. Explique cada uno de los siguientes términos:
   (a) datos
   (b) base de datos
   (c) sistema de gestión de bases de datos
   (d) programa de aplicación de bases de datos
   (e) independencia de los datos
   (f) seguridad
   (g) integridad
   (h) vistas.

1.3. Describa el enfoque de tratamiento de los datos adoptado en los antiguos sistemas basados en archivos. Indique las desventajas de este enfoque.

1.4. Describa las principales características del enfoque de base de datos y compárelas con la técnica basada en archivos.

1.5. Describa los cinco componentes del entorno SGBD y explique cómo se relacionan entre sí.

1.6. Explique el papel de cada una de las siguientes personas en un entorno de base de datos:
   (a) administrador de los datos
   (b) administrador de la base de datos
   (c) diseñador lógico de la base de datos
   (d) diseñador físico de la base de datos
   (e) desarrollador de aplicaciones
   (f) usuarios finales

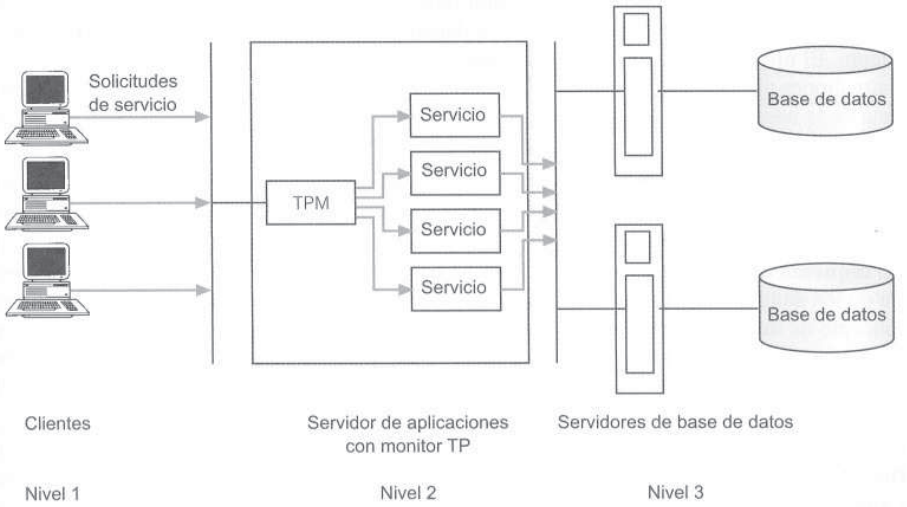

Figura 2.16. Monitor de procesamiento de transacciones como nivel intermedio en una arquitectura cliente-servidor de tres niveles.

Processing, procesamiento distribuido de transacciones). Un SGBD que cumpla este estándar puede funcionar como gestor de recursos bajo control de un monitor TP que actúe como gestor de transacciones. Hablaremos de las transacciones distribuidas y del estándar DTP en los Capítulos 22 y 23.

- *Equilibrado de carga.* El monitor TP puede equilibrar las solicitudes de los clientes distribuyéndolas entre múltiples SGBD situados en una o más computadoras, dirigiendo en todo momento las solicitudes de servicio de los clientes al servidor que menos carga de procesamiento tenga asignada. Además, puede poner en marcha dinámica otros SGBD adicionales según sea necesario para conseguir las prestaciones deseadas.
- *Multiplexación.* En entornos con un gran número de usuarios, puede resultar difícil en ocasiones que todos los usuarios mantengan activa una sesión simultáneamente con el SGBD. En muchos casos, los usuarios no necesitan disponer de un acceso continuo al SGBD. En lugar de hacer que cada usuario se conecte al SGBD, el monitor BD puede establecer una serie de conexiones con el SGBD de la forma necesaria y multiplexar las solicitudes de los usuarios a través de estas conexiones. Esto permite que pueda acceder un mayor número de usuarios a los SGBD disponibles utilizando un número de conexiones mucho más pequeño, lo que implica un menor gasto de recursos.
- *Mejora de la fiabilidad.* El monitor TP actúa como *gestor de transacciones*, llevando a cabo las acciones necesarias para mantener la coherencia de la base de datos, y actuando el SGBD como un *gestor de recursos*. Si el SGBD falla, el monitor TP puede ser capaz de reenviar la transacción a otro SGBD o puede retenerla hasta que el SGBD vuelva a estar disponible.

Los monitores TP suelen utilizarse en entornos que tengan un volumen muy alto de transacciones, en los que el monitor TP puede emplearse para descargar tareas de procesamiento del servidor SGBD. Como ejemplos destacados de monitores TP podemos citar CICS y Encina de IBM (que se utilizan principalmente en IBM AIX o Windows NT y que ahora se incluyen en los productos IBM TXSeries) y Tuxedo de BEA Systems.

Resumen del capítulo

- La arquitectura de base de datos ANSI-SPARC utiliza **tres niveles** de abstracción: **externo**, **conceptual** e **interno**. El **nivel externo** está compuesto por las vistas que los usuarios tienen de la base de

datos. El **nivel conceptual** es la vista comunitaria de la base de datos. Especifica el contenido de información de la base de datos completa, con independencia de las consideraciones relativas al almacenamiento. El nivel conceptual representa todas las entidades, sus atributos y sus relaciones, así como las restricciones aplicables a los datos y la información de seguridad e integridad. El **nivel interno** es la vista de la base de datos que tiene la computadora. Especifica cómo se representan los datos, cómo se secuencian los registros, qué índices y punteros existen, etc.

- La **correspondencia entre los niveles externo y conceptual** transforma las solicitudes y resultados entre esos dos niveles. La **correspondencia entre los niveles conceptual a interno** transforma las solicitudes y resultados entres los niveles interno y conceptual de la base de datos.
- Un **esquema de base de datos** es una descripción de la estructura de la base de datos. La independencia de los datos hace que cada nivel sea inmune a los cambios efectuados en los niveles inferiores. El concepto de **independencia lógica de los datos** hace referencia a la inmunidad de los esquemas externos con respecto a los cambios efectuados en el esquema conceptual. La **independencia física de los datos** hace referencia a la inmunidad del esquema conceptual frente a los cambios que se efectúen en el esquema interno.
- Un sublenguaje de datos está compuesto de dos partes: un **lenguaje de definición de datos (DDL, Data Definition Language)** y un **lenguaje de manipulación de datos (DML, Data Manipulation Language)**. El DDL se utiliza para especificar el esquema de la base de datos y el DML se emplea para consultar y actualizar la base de datos. La parte de un DML relacionada con la extracción de datos se denomina **lenguaje de consulta**.
- Un **modelo de datos** es una colección de conceptos que pueden usarse para describir un conjunto de datos, las operaciones de manipulación de los mismos y una serie de restricciones de integridad aplicables a los datos. Los modelos de datos pueden clasificarse en tres categorías amplias: modelos de datos **basados en objetos**, modelos de datos **basados en registros** y modelos de datos **físicos**. Los primeros dos se utilizan para describir los datos en los niveles conceptual y externo, mientras que el último se emplea para describir los datos en el nivel interno.
- Los modelos de datos basados en objetos incluyen el modelo Entidad-Relación, el modelo semántico, el modelo funcional y el modelo orientado a objetos. Los modelos de datos basados en registros incluyen el modelo relacional, el modelo en red y el modelo jerárquico.
- El **modelado conceptual** es el proceso de construir una arquitectura detallada para una base de datos que sea independiente de los detalles de implementación, como por ejemplo el SGBD de destino, los programas de aplicación, los lenguajes de programación o cualquier otra consideración física. El diseño del esquema conceptual es crítico para el éxito del sistema. Merece la pena invertir el tiempo y la energía necesarios para producir el mejor diseño conceptual posible.
- Las **funciones** y **servicios** de un SGBD multiusuario incluyen el almacenamiento, extracción y actualización de los datos; un catálogo accesible por los usuarios; el soporte de transacciones; los servicios de control de concurrencia y recuperación; los servicios de autorización; el soporte para la comunicación de datos; los servicios de integridad; servicios para promover la independencia de los datos y servicios de utilidad.
- El **catálogo del sistema** es uno de los componentes fundamentales de un SGBD. Contiene 'datos acerca de los datos' o **metadatos**. El catálogo debe ser accesible por los usuarios. El IRDS (Information Resource Dictionary System) es un estándar ISO que define una serie de métodos de acceso para diccionario de datos. Esto permite compartir los diccionarios y transferirlos de un sistema a otro.
- El concepto de arquitectura **cliente-servidor** hace referencia a la forma en que los componentes software interactúan. Existe un proceso **cliente** que requiere algún tipo de recurso y un **servidor** que proporciona el recurso. En el modelo en dos niveles, el cliente gestiona la interfaz de usuario y la lógica de procesamiento del negocio y el servidor gestiona la funcionalidad de la base de datos. En el entorno web, el modelo tradicional en dos niveles ha sido sustituido por un modelo en tres niveles, compuesto por un modelo de interfaz de usuario (el **cliente**), un nivel de lógica de negocios y de proce-

samiento de datos (el **servidor de aplicaciones**) y el SGBD (el **servidor de base de datos**), distribuidos en máquinas distintas.

- Un **monitor de procesamiento de transacciones (TP, Transaction Processing)** es un programa que controla la transferencia de datos entre clientes y servidores para proporcionar un entorno coherente, particularmente para el procesamiento de transacciones en línea (OLTP). Entre las ventajas podemos citar el encaminamiento de transacciones, las transacciones distribuidas, el equilibrado de la carga de procesamiento, la multiplexación y una mayor fiabilidad.

Cuestiones de repaso

2.1. Explique el concepto de independencia de los datos y explique su importancia en un entorno de base de datos.

2.2. Para resolver la cuestión de la independencia de los datos, se propuso la arquitectura en tres niveles de ANSI-SPARC. Compare y contraste los tres niveles de este modelo.

2.3. ¿Qué es un modelo de datos? Indique y explique los tipos principales de modelos de datos.

2.4. Explique la función y la importancia del modelado conceptual.

2.5. Describa los tipos de servicios que cabe esperar que un SGBD multiusuario proporcione.

2.6. De las distintas funciones descritas en respuesta a la Cuestión 2.5, ¿cuáles cree que *no* serían necesarias en un SGBD autónomo para PC? Justifique su respuesta.

2.7. Explique la función y la importancia del catálogo del sistema.

2.8. Describa los componentes principales de un SGBD e indique qué componentes podrían ser responsables de cada uno de los servicios identificados en la Cuestión 2.5.

2.9. ¿Qué quiere decir el término 'arquitectura cliente-servidor' y cuáles son las ventajas de este enfoque? Compare la arquitectura cliente-servidor con las otras dos arquitecturas.

2.10. Compare la arquitectura cliente-servidor en dos niveles para un SGBD tradicional con la arquitectura cliente-servidor en tres niveles. ¿Por qué resulta esta última arquitectura más apropiada para la Web?

2.11. ¿Qué es un monitor TP? ¿Qué ventajas aporta un monitor TP a un entorno OLTP?

Ejercicios

2.12. Analice los SGBD que esté utilizando actualmente. Determine hasta qué punto proporciona cada uno de los sistemas las funciones que cabe esperar de un SGBD. ¿Qué tipo de lenguaje proporciona cada sistema? ¿Qué tipo de arquitectura utiliza cada SGBD? Compruebe la accesibilidad y la facilidad de ampliación del catálogo del sistema. ¿Es posible exportar el catálogo del sistema a otro sistema?

2.13. Escriba un programa que almacene nombres y números telefónicos en una base de datos. Escriba otro programa que almacene nombres y direcciones en una base de datos. Modifique los programas para utilizar esquemas externos, un esquema conceptual y un esquema interno. ¿Cuáles son las ventajas y desventajas de esta modificación?

2.14. Escriba un programa que almacene nombres y fechas de nacimiento en una base de datos. Amplíe el programa para que almacene el formato de los datos en la base de datos; en otras palabras, cree un catálogo del sistema. Proporcione una interfaz que haga que este catálogo del sistema sea accesible por los usuarios externos.

2.15. Cómo modificaría el programa del Ejercicio 2.13 para adaptarlo a una arquitectura cliente-servidor. ¿Cuáles son las ventajas y desventajas de esta modificación?

Chapter Summary

- The Relational Database Management System (RDBMS) has become the dominant data-processing software in use today, with estimated new licence sales of between US$6 billion and US$10 billion per year (US$25 billion with tools sales included). This software represents the second generation of DBMSs and is based on the relational data model proposed by E. F. Codd.
- A mathematical **relation** is a subset of the Cartesian product of two or more sets. In database terms, a relation is any subset of the Cartesian product of the domains of the attributes. A relation is normally written as a set of *n*-tuples, in which each element is chosen from the appropriate domain.
- Relations are physically represented as **tables**, with the rows corresponding to individual tuples and the columns to attributes.
- The structure of the relation, with domain specifications and other constraints, is part of the **intension** of the database, while the relation with all its tuples written out represents an **instance** or **extension** of the database.
- Properties of database relations are: each cell contains exactly one atomic value, attribute names are distinct, attribute values come from the same domain, attribute order is immaterial, tuple order is immaterial, and there are no duplicate tuples.
- The **degree** of a relation is the number of attributes, while the **cardinality** is the number of tuples. A **unary** relation has one attribute, a **binary** relation has two, a **ternary** relation has three, and an ***n*-ary** relation has *n* attributes.
- A **superkey** is an attribute, or set of attributes, that identifies tuples of a relation uniquely, while a **candidate key** is a minimal superkey. A **primary key** is the candidate key chosen for use in identification of tuples. A relation must always have a primary key. A **foreign key** is an attribute, or set of attributes, within one relation that is the candidate key of another relation.
- A **null** represents a value for an attribute that is unknown at the present time or is not applicable for this tuple.
- **Entity integrity** is a constraint that states that in a base relation no attribute of a primary key can be null. **Referential integrity** states that foreign key values must match a candidate key value of some tuple in the home relation or be wholly null. Apart from relational integrity, integrity constraints include, required data, domain, and multiplicity constraints; other integrity constraints are called **general constraints**.
- A **view** in the relational model is a **virtual** or **derived relation** that is dynamically created from the underlying base relation(s) when required. Views provide security and allow the designer to customize a user's model. Not all views are updatable.

Chapter Summary

- The **relational algebra** is a (high-level) procedural language: it can be used to tell the DBMS how to build a new relation from one or more relations in the database. The **relational calculus** is a non-procedural language: it can be used to formulate the definition of a relation in terms of one or more database relations. However, formally the relational algebra and relational calculus are equivalent to one another: for every expression in the algebra, there is an equivalent expression in the calculus (and vice versa).
- The relational calculus is used to measure the selective power of relational languages. A language that can be used to produce any relation that can be derived using the relational calculus is said to be **relationally complete**. Most relational query languages are relationally complete but have more expressive power than the relational algebra or relational calculus because of additional operations such as calculated, summary, and ordering functions.
- The five fundamental operations in relational algebra, *Selection*, *Projection*, *Cartesian product*, *Union*, and *Set difference*, perform most of the data retrieval operations that we are interested in. In addition, there are also the *Join*, *Intersection*, and *Division* operations, which can be expressed in terms of the five basic operations.
- The **relational calculus** is a formal non-procedural language that uses predicates. There are two forms of the relational calculus: tuple relational calculus and domain relational calculus.
- In the **tuple relational calculus**, we are interested in finding tuples for which a predicate is true. A tuple variable is a variable that 'ranges over' a named relation: that is, a variable whose only permitted values are tuples of the relation.
- In the **domain relational calculus**, domain variables take their values from domains of attributes rather than tuples of relations.
- The relational algebra is logically equivalent to a safe subset of the relational calculus (and vice versa).
- Relational data manipulation languages are sometimes classified as **procedural** or **non-procedural**, **transform-oriented**, **graphical**, **fourth-generation**, or **fifth-generation**.

Review Questions

4.1 What is the difference between a procedural and a non-procedural language? How would you classify the relational algebra and relational calculus?

4.2 Explain the following terms:
(a) relationally complete
(b) closure of relational operations.

4.3 Define the five basic relational algebra operations. Define the Join, Intersection, and Division operations in terms of these five basic operations.

4.4 Discuss the differences between the five Join operations: Theta join, Equijoin, Natural join, Outer join, and Semijoin. Give examples to illustrate your answer.

4.5 Compare and contrast the tuple relational calculus with domain relational calculus. In particular, discuss the distinction between tuple and domain variables.

4.6 Define the structure of a (well-formed) formula in both the tuple relational calculus and domain relational calculus.

4.7 Explain how a relational calculus expression can be unsafe. Illustrate your answer with an example. Discuss how to ensure that a relational calculus expression is safe.

Example 5.42 DELETE all rows

Delete all rows from the Viewing *table.*

```
DELETE FROM Viewing;
```

No WHERE clause has been specified, so the delete operation applies to all rows in the table. This removes all rows from the table leaving only the table definition, so that we are still able to insert data into the table at a later stage.

Chapter Summary

- SQL is a non-procedural language, consisting of standard English words such as SELECT, INSERT, DELETE, that can be used by professionals and non-professionals alike. It is both the formal and *de facto* standard language for defining and manipulating relational databases.
- The **SELECT** statement is the most important statement in the language and is used to express a query. It combines the three fundamental relational algebra operations of *Selection*, *Projection*, and *Join*. Every SELECT statement produces a query result table consisting of one or more columns and zero or more rows.
- The SELECT clause identifies the columns and/or calculated data to appear in the result table. All column names that appear in the SELECT clause must have their corresponding tables or views listed in the FROM clause.
- The WHERE clause selects rows to be included in the result table by applying a search condition to the rows of the named table(s). The ORDER BY clause allows the result table to be sorted on the values in one or more columns. Each column can be sorted in ascending or descending order. If specified, the ORDER BY clause must be the last clause in the SELECT statement.
- SQL supports five aggregate functions (COUNT, SUM, AVG, MIN, and MAX) that take an entire column as an argument and compute a single value as the result. It is illegal to mix aggregate functions with column names in a SELECT clause, unless the GROUP BY clause is used.
- The GROUP BY clause allows summary information to be included in the result table. Rows that have the same value for one or more columns can be grouped together and treated as a unit for using the aggregate functions. In this case the aggregate functions take each group as an argument and compute a single value for each group as the result. The HAVING clause acts as a WHERE clause for groups, restricting the groups that appear in the final result table. However, unlike the WHERE clause, the HAVING clause can include aggregate functions.
- A **subselect** is a complete SELECT statement embedded in another query. A subselect may appear within the WHERE or HAVING clauses of an outer SELECT statement, where it is called a **subquery** or **nested query**. Conceptually, a subquery produces a temporary table whose contents can be accessed by the outer query. A subquery can be embedded in another subquery.
- There are three types of subquery: **scalar**, **row**, and **table**. A *scalar subquery* returns a single column and a single row; that is, a single value. In principle, a scalar subquery can be used whenever a single value is needed. A *row subquery* returns multiple columns, but again only a single row. A row subquery can be used whenever a row value constructor is needed, typically in predicates. A *table subquery* returns one or more columns and multiple rows. A table subquery can be used whenever a table is needed, for example, as an operand for the IN predicate.

Ejemplo 6.11 Revocación de privilegios específicos de un cierto usuario mediante REVOKE

Revocar todos los privilegios concedidos a Director *sobre la tabla* Staff.

```
REVOKE ALL PRIVILEGES
ON Staff
FROM Director;
```

Esto es equivalente a REVOKE SELECT . . ., ya que éste fue el único privilegio que se le había concedido a Director.

Resumen del capítulo

- El estándar ISO proporciona ocho tipos de datos base: booleano, carácter, bit, numérico exacto, numérico aproximado, fecha y hora, intervalo y objetos de carácter/binario de gran tamaño.
- Las instrucciones DDL de SQL permiten definir objetos de la base de datos. Las instrucciones CREATE y DROP SCHEMA permiten crear y destruir esquemas; las instrucciones CREATE, ALTER y DROP TABLE permiten crear, modificar y destruir tablas; las instrucciones CREATE y DROP INDEX permiten crear y destruir índices.
- El estándar SQL de ISO proporciona una serie de cláusulas en las instrucciones **CREATE** y **ALTER TABLE** para definir **restricciones de integridad** que se encargan de gestionar: datos requeridos, restricciones de dominio, integridad de entidades, integridad referencial y restricciones generales. Los **datos requeridos** pueden especificarse utilizando NOT NULL. Las **restricciones de dominio** pueden especificarse utilizando la cláusula CHECK o definiendo dominios mediante la instrucción CREATE DOMAIN. Las **claves primarias** deben definirse utilizando la cláusula PRIMARY KEY y las **claves alternativas** se definen utilizando la combinación de NOT NULL y UNIQUE. Las **claves externas** deben definirse utilizando la cláusula FOREIGN KEY y las reglas de actualización y borrado mediante las subcláusulas ON UPDATE y ON DELETE. Las **restricciones generales** pueden definirse utilizando las cláusulas CHECK y UNIQUE. Las restricciones generales también pueden crearse mediante la instrucción CREATE ASSERTION.
- Una **vista** es una tabla virtual que representa un subconjunto de columnas y/o filas y/o expresiones de columna de una o más tablas base o vistas. Las vistas se crean utilizando la instrucción CREATE VIEW, en las que se especifica la **consulta de definición**. Las vistas pueden no necesariamente ser una tabla físicamente almacenada, en cuyo caso se las recrea cada vez que se hace referencia a ellas.
- Las vistas pueden emplearse para simplificar la estructura de la base de datos y hacer que las consultas sean más fáciles de escribir. También se las puede usar para proteger ciertas columnas y/o filas frente a accesos no autorizados. No todas las vistas son actualizables.
- El proceso de **resolución de vistas** combina la consulta efectuada sobre una vista con la definición de la propia vista, generando una consulta sobre la tabla o tablas base subyacentes. Este proceso se realiza cada vez que el SGBD tiene que procesar una consulta referida a una vista. Otra técnica alternativa, denominada **materialización de vistas**, almacena las vistas como tablas temporales dentro de la base de datos la primera vez que se efectúa una consulta sobre dichas vistas. Posteriormente, las consultas basadas en la vista materializada pueden ejecutarse mucho más rápidamente de lo que se podría si se recalculara la vista cada vez que se accede a ella. Una de las desventajas de las vistas materializadas es que se hace preciso mantener actualizada la tabla temporal.
- La instrucción COMMIT indica que se ha completado con éxito una transacción y que todos los cambios realizados en la base de datos son ya permanentes. La instrucción ROLLBACK indica que la transacción debe abortarse y que los cambios en la base de datos deben deshacerse.

- El control de acceso en SQL está modelado en tornos a los conceptos de identificadores de autorización, propiedad y privilegios. Los **identificadores de autorización** son asignados a los usuarios de la base de datos por el DBA y permiten identificar a cada usuario. Cada objeto que se crea en SQL tiene un **propietario**. El propietario puede pasar **privilegios** a otros usuarios utilizando la instrucción GRANT y puede revocar los privilegios que les haya pasado mediante la instrucción REVOKE. Los privilegios que pueden pasarse son USAGE, SELECT, DELETE, INSERT, UPDATE y REFERENCES; estos últimos tres pueden restringirse a columnas específicas. El usuario puede autorizar al usuario que recibe el privilegio para que pase ese privilegio a otros, utilizándose para ello la cláusula WITH GRANT OPTION, y puede revocar este privilegio mediante la cláusula GRANT OPTION FOR.

Cuestiones de repaso

6.1. Describa los ocho tipos de datos base de SQL.

6.2. Explique cuál es la funcionalidad y la importancia de la característica de mejora de la integridad.

6.3. Explique cada una de las cláusulas de la instrucción CREATE TABLE.

6.4. Explique las ventajas y desventajas de las vistas.

6.5. Describa cómo funciona el proceso de resolución de vistas.

6.6. ¿Qué restricciones son necesarias para garantizar que una vista sea actualizable?

6.7. ¿Qué es una vista materializada y cuáles son las ventajas de mantener una vista materializada, en lugar de utilizar el proceso de resolución de vistas?

6.8. Describa la diferencia existente entre el control de acceso discrecional y el control de acceso obligatorio. ¿Qué tipo de mecanismo de control soporta SQL?

6.9. Describa cómo funcionan los mecanismos de control de acceso de SQL.

Ejercicios

Responda a las siguientes preguntas utilizando el esquema relacional especificado en la sección de Ejercicios del Capítulo 3:

6.10. Cree la tabla Hotel utilizando las características de mejora de la integridad de SQL.

6.11. Ahora cree las tablas Room, Booking y Guest utilizando las características de mejora de la integridad de SQL con las siguientes restricciones:

(a) type puede tener los siguientes valores: Single, Double o Family.

(b) price debe estar comprendido entre 10 euros y 100 euros.

(c) roomNo debe estar comprendido entre 1 y 100.

(d) dateFrom y dateTo deben ser mayores que la fecha actual..

(e) No se puede reservar dos veces una misma habitación.

(f) Un mismo huésped no puede tener reservas que se solapen.

6.12. Cree una tabla separada con la misma estructura que la tabla Booking para almacenar los registros antiguos. Utilizando la instrucción INSERT, copie los registros de la tabla Booking a la tabla de archivo, traspasando únicamente las reservas anteriores al 1 de enero de 2003. Borre todas las reservas anteriores al 1 de enero de 2003 de la tabla Booking.

6.13. Cree una vista que contenga el nombre del hotel y los nombres de los huéspedes albergados en el hotel.

6.14. Cree una vista que contenga la cuenta correspondiente a cada huésped del Grovenor Hotel.

6.15. Proporcione a los usuarios Manager y Director acceso completo a estas vistas, con el privilegio de pasar el acceso a otros usuarios.

Tabla 9.9. Administración de datos y administración de la base de datos: principales diferencias en cuanto a tareas. (*Cont.*)

Administración de datos	Administración de la base de datos
Lleva a cabo los diseños conceptual y lógico de la base de datos.	Lleva a cabo los diseños lógico y físico de la base de datos.
Desarrolla y mantiene el modelo de datos corporativo.	Implementa el diseño físico de la base de datos.
Coordina el desarrollo del sistema.	Monitoriza y controla la base de datos.
Orientación de gestión.	Orientación técnica.
Independiente del SGBD.	Dependiente del SGBD.

Resumen del capítulo

- Un **sistema de información** está constituido por los recursos que permiten la recopilación, gestión, control y diseminación de la información a lo largo de una organización
- Un sistema de información basado en computadora incluye los siguientes componentes: base de datos, software de base de datos, software de aplicación, hardware informático (incluyendo los medios de almacenamiento) y el personal que utiliza y desarrolla el sistema.
- La base de datos es un componente fundamental de los sistemas de información y su desarrollo y utilización deben contemplarse desde la perspectiva de los requisitos globales de la organización. Por tanto, el ciclo de vida de un sistema de información corporativo está inherentemente enlazado con el ciclo de vida de la base de datos que le da soporte.
- Las etapas principales del **ciclo de desarrollo de un sistema de base de datos** incluyen: planificación de la base de datos, definición del sistema, recopilación y análisis de requisitos, diseño de la base de datos, selección del SGBD (opcional), diseño de la aplicación, prototipado (opcional), implementación, conversión y carga de los datos, pruebas y mantenimiento operativo.
- La **planificación de la base de datos** está compuesta por las actividades de gestión que permiten llevar a cabo las distintas etapas del ciclo de desarrollo de un sistema de base de datos de la forma más eficiente y efectiva posible.
- La **definición del sistema** implica identificar el ámbito y los límites del sistema de base de datos, así como las vistas de usuario. Una **vista de usuario** define lo que se requiere del sistema de base de datos desde la perspectiva de un puesto de trabajo concreto (como por ejemplo Gerente o Supervisor) o de una aplicación corporativa (como por ejemplo marketing, personal o control de almacén).
- La **recopilación y análisis de requisitos** es el proceso de recopilar y analizar información acerca de aquella parte de la organización a la que el sistema de base de datos deba dar soporte, así como utilizar esta información para identificar los requisitos del nuevo sistema. Hay tres técnicas principales para gestionar los requisitos de un sistema de bases de datos que tenga múltiples vistas de usuario: el enfoque **centralizado**, el enfoque **de integración de vistas** y una combinación de ambos enfoques.
- El enfoque **centralizado** implica combinar los requisitos de cada vista de usuario en un único conjunto de requisitos para el nuevo sistema de base de datos. Durante la etapa de diseño de la base de datos se creará un modelo de datos que represente todas las vistas de usuario. En el enfoque de **integración de vistas**, los requisitos de cada vista de usuario se conservan como listas independientes; durante la etapa de diseño de la base de datos se crean después modelos de datos que representan cada una de las vistas de usuario.
- El **diseño de la base de datos** es el proceso de crear un diseño que dé soporte a la misión y a los objetivos de la empresa para el sistema de base de datos requerido. Hay tres fases en el diseño de bases de datos: diseño conceptual, lógico y físico.

- El **diseño conceptual de la base de datos** es el proceso de construir el modelo de los datos utilizados en una empresa, independientemente de *todas* las consideraciones físicas.
- El **diseño lógico de la base de datos** es el proceso de construir un modelo de los datos utilizados en una empresa basándose en un modelo de datos específico, pero con independencia del SGBD concreto y del resto de consideraciones físicas.
- El **diseño físico de la base de datos** es el proceso de generar una descripción de la implementación de la base de datos en almacenamiento secundario; describe las relaciones básicas, la organización de los archivos y los índices empleados para conseguir un acceso eficiente a los datos, así como cualesquiera restricciones de integridad y medidas de seguridad asociadas.
- La **selección del SGBD** implica seleccionar un SGBD apropiado para el sistema de base de datos.
- El **diseño de la aplicación** implica el diseño de la interfaz de usuario y el diseño de las transacciones, que describen los programas de aplicación que utilizan y procesan la base de datos. Una **transacción** de base de datos es una acción, o serie de acciones, llevada a cabo por un único usuario o programa de aplicación y que accede al contenido de la base de datos o lo modifica.
- El **prototipado** implica la construcción de un modelo operativo de sistema de la base de datos que permita a los diseñadores o usuarios visualizar y evaluar el sistema.
- La **implementación** es la realización física del diseño de la base de datos y del diseño de las aplicaciones.
- La **conversión y carga de los datos** implica la transferencia de los datos existentes a la nueva base de datos y la conversión de las aplicaciones existentes para que se ejecuten sobre la nueva base de datos.
- Las **pruebas** son el proceso de operar el sistema de base de datos con la intención de localizar posibles errores.
- El **mantenimiento operativo** es el proceso de monitorizar y mantener el sistema después de la instalación.
- El término **CASE (Computer-Aided Software Engineering)** se aplica a cualquier herramienta que de soporte a la ingeniería del software y que permita que las actividades de desarrollo del sistema de base de datos se lleven a cabo de la forma más eficiente y efectiva posible. Las herramientas CASE pueden clasificarse en tres categorías: CASE de alto nivel, CASE de bajo nivel y CASE integrado.
- La **administración de datos** es la gestión de los recursos de datos, incluyendo las tareas de planificación de base de datos, desarrollo y mantenimiento de estándares, políticas y procedimientos y diseño conceptual y lógico de la base de datos.
- La **administración de la base de datos** es la gestión de la realización física de un sistema de base de datos, incluyendo las tareas de diseño físico de la base de datos e implementación de la misma, configuración de la seguridad y los controles de integridad, monitorización de las prestaciones del sistema y reorganización de la base de datos según sea necesario.

Cuestiones de repaso

9.1. Describa los componentes principales de un sistema de información.

9.2. Explique la relación existente entre el ciclo de vida de los sistemas de información y el ciclo de desarrollo de los sistemas de bases de datos.

9.3. Describa el propósito y actividades principales asociados con cada etapa del ciclo de desarrollo de un sistema de base de datos.

9.4. Explique lo que representa una vista de usuario en el contexto de un sistema de base de datos.

9.5. Explique las técnicas principales para la gestión del diseño de un sistema de base de datos que tenga múltiples vistas de usuario.

9.6. Realice una comparación entre las tres fases del diseño de una base de datos.

Si tratamos de responder a la pregunta; '¿En qué sucursal está disponible el inmueble de número PA14?', no podremos responder a esta pregunta, ya que este inmueble no ha sido todavía asignado a ningún empleado que trabaje en una sucursal. La incapacidad de responder a esta pregunta se considera una pérdida de información (ya que sabemos que todo inmueble debe estar disponible en una sucursal) y es el resultado de una trampa de corte. La multiplicidad de las entidades Staff y PropertyForRent en la relación *Oversees* tiene un valor mínimo de cero, lo que significa que algunos inmuebles no pueden asociarse con una sucursal a través de un empleado. Por tanto, para resolver este problema, necesitamos identificar la relación que falta, que en este caso es la relación *Offers* (ofrece) entre las entidades Branch y PropertyForRent. El modelo ER mostrado en la Figura 11.22(a) representa la verdadera asociación entre estas entidades. Este modelo garantiza que se puedan conocer en todo momento los inmuebles asociados con cada sucursal, incluyendo aquellos inmuebles que todavía no hayan sido asignados a ningún empleado.

Si ahora examinamos las instancias de los tipos de relación *Has*, *Oversees* y *Offers*, como se muestra en la Figura 11.22(b), vemos que somos capaces de determinar que el inmueble número PA14 está disponible en la sucursal número B007.

Figura 11.22(a). El modelo ER mostrado en la Figura 11.21(a) reestructurado para eliminar la trampa de corte.

Resumen del capítulo

- Un **tipo de entidad** es un grupo de objetos con las mismas propiedades y que son identificados por la organización como poseedores de una existencia independiente. Una **instancia de entidad** es un objeto unívocamente identificable de un cierto tipo de entidad.
- Un **tipo de relación** es un conjunto de asociaciones significativas entre tipos de entidad. Una instancia de relación es una asociación unívocamente identificable que incluye una instancia de cada tipo de entidad participante.
- El **grado de un tipo de relación** es el número de tipos de entidad participantes en una relación.
- Una **relación recursiva** es un tipo de relación en la que el *mismo* tipo de entidad participa más de una vez en roles *diferentes*.
- Un **atributo** es una propiedad de un tipo de entidad o un tipo de relación.
- Un **dominio de atributo** es el conjunto de valores permitidos para uno o más atributos.
- Un **atributo simple** está formado por un único componente que tiene existencia independiente.
- Un **atributo compuesto** está formado por múltiples componentes cada uno de los cuales tiene una existencia independiente.
- Un **atributo univaluado** contiene un único valor para cada instancia de un tipo de entidad.
- Un **atributo multivaluado** contiene múltiples valores para cada instancia de un tipo de entidad.
- Un **atributo derivado** representa un valor que se puede derivar a partir del valor de un atributo o conjunto de atributos relacionados, no necesariamente de la misma entidad.

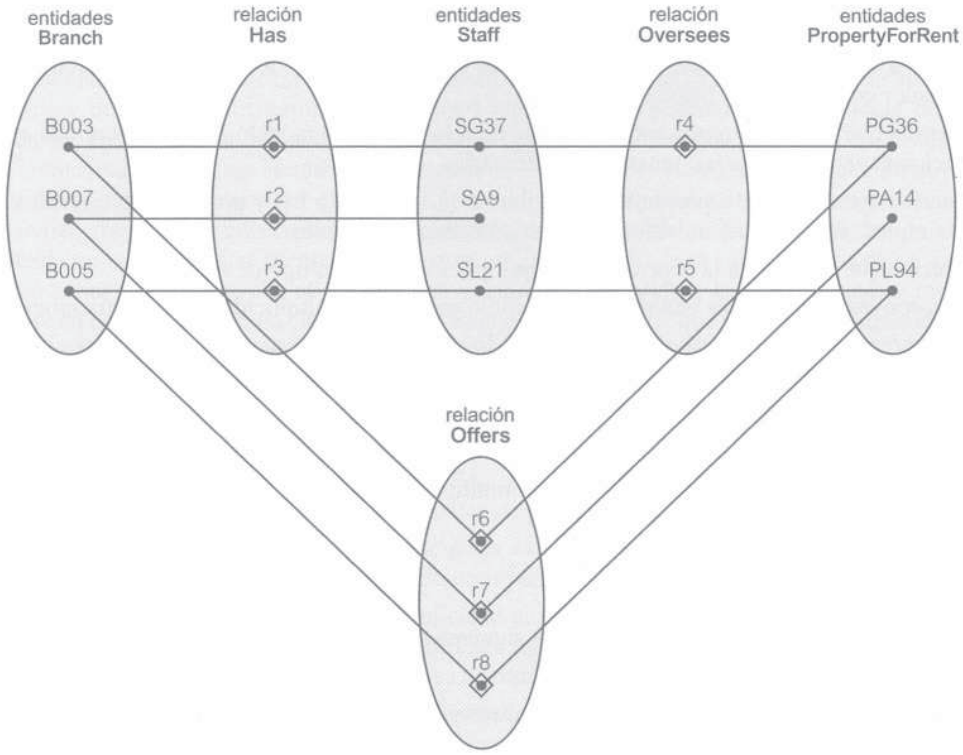

Figura 11.22(b). Red semántica del modelo ER mostrado en la Figura 11.22(a).

- Una **clave candidata** es un conjunto mínimo de atributos que identifica unívocamente cada instancia de un tipo de entidad.
- Una **clave principal** es la clave candidata que se selecciona para identificar de forma unívoca cada instancia de un tipo de entidad-
- Una **clave compuesta** es una clave candidata que está compuesta de dos o más atributos.
- Un **tipo de entidad fuerte** es un tipo de entidad cuya existencia *no depende* de ningún otro tipo de entidad. Un **tipo de entidad débil** es un tipo de entidad cuya existencia depende de algún otro tipo de entidad.
- La **multiplicidad** es el número (o rango) de posibles instancias de un tipo de entidad que pueden estar relacionadas con una única instancia de un tipo de entidad asociado a través de una relación concreta.
- La **multiplicidad para una relación compleja** es el número (o rango) de posibles instancias de un tipo de entidad en una relación n-aria cuando los otros $(n–1)$ valores están fijos.
- La **cardinalidad** describe el número máximo de posibles instancias de relación para una entidad que participa en un tipo de relación dado.
- La **participación** determina si todas las instancias de una entidad participan en una relación dada o si sólo lo hacen algunas de ellas.
- Las **trampas multiplicativas** aparecen cuando un modelo representa una relación entre tipos de entidad, pero la ruta que conecta ciertas instancias de entidad es ambigua.
- Las **trampas de corte** aparecen cuando un modelo sugiere la existencia de una relación entre tipos de entidad, pero no existe una ruta entre ciertas instancias de entidad.

Resumen del capítulo

- Una **superclase** es un tipo de entidad que incluye uno o más subgrupos diferenciados de instancias, los cuales es preciso representar en un modelo de datos. Una **subclase** es un subgrupo diferenciado de instancias de un tipo de entidad, el cual necesita ser representado en un modelo de datos.
- La **especialización** es el proceso de maximizar las diferencias entre miembros de una entidad identificando sus características distintivas.
- La **generalización** es el proceso de minimizar las diferencias entre entidades identificando sus características comunes.
- Hay dos restricciones que pueden aplicarse a las relaciones de especialización/generalización; dichas restricciones son las **restricciones de participación** y las **restricciones de disyunción**.
- Una **restricción de participación** determina si todo miembro de la superclase debe ser miembro de una subclase.
- Una **restricción de disyunción** describe la relación entre miembros de las subclases e indica si es posible que un miembro de una superclase sea miembro de una subclase o de más de una.
- La **agregación** representa una relación de tipo 'tiene' o 'es parte de' entre tipos de entidad, en la que uno de los tipos de entidad representa el 'todo' y el otro la 'parte'.
- La **composición** es una forma específica de agregación que representa una asociación entre entidades, en la que existe una pertenencia fuerte y una existencia coincidente entre el 'todo' y la 'parte'.

Cuestiones de repaso

12.1. Describa lo que representan una superclase y una subclase.

12.2. Describa la relación entre una superclase y su subclase.

12.3. Describa el proceso de herencia de atributos e ilústrelo con un ejemplo.

12.4. ¿Cuáles son las razones principales para introducir los conceptos de superclases y subclases en un modelo ER?

12.5. Describa qué es lo que representa una subclase compartida y cómo se relaciona este concepto con el de herencia múltiple.

12.6. Describa los procesos de especialización y generalización e indique sus diferencias..

12.7. Describa las dos restricciones principales que se aplican a las relaciones de especialización/generalización.

12.8. Describa los conceptos de agregación y composición, indique en qué se diferencian y proporcione un ejemplo de cada uno de esos conceptos.

Ejercicios

12.9. Considere si sería apropiado introducir los conceptos avanzados de especialización/generalización, agregación y/o composición para los casos de estudio descritos en el Apéndice B.

12.10. Considere si sería apropiado introducir los conceptos avanzados de especialización/generalización, agregación y/o composición en el modelo ER del caso de estudio descrito en el Ejercicio 11.12. Si resulta apropiado, vuelva a dibujar el diagrama ER como un diagrama EER, añadiendo los conceptos adicionales avanzados.

Tercera forma normal (3NF) Una relación que se encuentra en primera y segunda formas normales y en la que ningún atributo que no sea de clave candidata depende transitivamente de *ninguna clave candidata*.

Cuando se usan las definiciones generales de las formas 2NF y 3NF, es preciso ser consciente de las dependencias parciales y transitivas con respecto a todas las claves candidatas, y no sólo con respecto a la clave principal. Esto puede hacer más complejo el proceso de normalización, pero las definiciones generales imponen restricciones adicionales a las relaciones y pueden identificar redundancias ocultas en las relaciones que de otro modo podrían pasar desapercibidas.

El compromiso al que hay que hacer frente es si resulta mejor mantener la simplicidad del proceso de normalización examinando únicamente las dependencias con respecto a las claves principales, lo que permite la identificación de las redundancias más problemáticas y obvias en las relaciones, o si es mejor usar las definiciones generales e incrementar la probabilidad de identificar redundancias que de otro modo podrían no percibirse. De hecho, a menudo sucede que, independientemente de si usamos las definiciones basadas en las claves principales o las definiciones generales de las formas 2NF y 3NF, la descomposición de las relaciones es la misma. Por ejemplo, si aplicamos las definiciones generales de las formas 2NF y 3NF a los Ejemplos 13.10 y 13.11 descritos en las Secciones 13.7 y 13.8, resulta la misma descomposición de las relaciones mayores en relaciones de menor tamaño. El lector interesado puede verificar por su cuenta este resultado.

En el siguiente capítulo volveremos a examinar el proceso de identificación de las dependencias funcionales que resultan útiles para el proceso de normalización y llevaremos a este proceso un paso más allá exponiendo las formas normales más avanzadas que la forma 3NF, como por ejemplo la forma normal de Boyce–Codd (BCNF). Asimismo, en dicho capítulo presentaremos un segundo ejemplo resuelto tomado del caso de estudio *DreamHome* donde se revisa el proceso de normalización desde la forma UNF hasta la BCNF.

Resumen del capítulo

- La **normalización** es una técnica para generar un conjunto de relaciones con una serie de propiedades deseables, dados los requisitos de datos de una organización. La normalización es un método formal que puede utilizarse para identificar relaciones basándose en sus claves y en las dependencias funcionales existentes entre sus atributos.
- Las relaciones con redundancia en los datos sufren de **anomalías de actualización**, que pueden clasificarse como anomalías de inserción, de borrado y de modificación.
- Uno de los conceptos principales asociados con la normalización es el de **dependencia funcional**, que describe la relación entre atributos de una tabla. Por ejemplo, si A y B son atributos de la relación R, B dependerá funcionalmente de A (lo que se denota $A \rightarrow B$) si cada valor de A está asociado con exactamente un valor de B. (A y B pueden estar compuestos, cada uno de ellos, por uno o más atributos).
- El **determinante** de una dependencia funcional es el atributo o grupo de atributos situado del lado izquierdo de la flecha que describe la dependencia.
- Las características principales de las dependencias funcionales que utilizamos para la normalización son que dichas dependencias presentan una relación uno a uno entre los atributos del lado izquierdo y los del lado derecho de la dependencia; que dichas dependencias se cumplen para todo instante de tiempo; y que se trata de dependencias funcionales completas.
- Una tabla en **forma no normalizada (Unnormalized Form, UNF)** es una tabla que contiene uno o más grupos repetitivos.
- Una relación en **primera forma normal (First Normal Form, 1NF)** es una relación en la que la intersección de cada fila y columna contiene un valor y sólo uno.
- Una relación en **segunda forma normal (2NF)** es una relación que está en primera forma normal y en la que todo atributo que no sea de clave principal depende funcionalmente de modo completo de la *clave principal*. La **dependencia funcional completa** indica que si A y B son atributos de una relación, B depen-

de funcionalmente de modo completo de A si B depende funcionalmente de A pero no de ningún subconjunto propio de A.

- Una relación en **tercera forma normal (3NF)** es una relación que está en primera y segunda formas normales y en la que ningún atributo que no sea de clave principal depende transitivamente de la *clave principal*. La **dependencia transitiva** es una condición en la que A, B y C son atributos de una relación tales que si $A \rightarrow B$ y $B \rightarrow C$, entonces C depende transitivamente de A a través de B (supuesto que A no dependa funcionalmente de B o C).
- La **definición general de la segunda forma normal (2NF)** indica que se trata de una relación que se encuentra en primera forma normal y en la que todo atributo que no sea de clave candidata depende funcionalmente de modo completo de *cualquier clave candidata*. En esta definición, los atributos de clave candidata son los atributos que forman parte de alguna clave candidata.
- La **definición general de la tercera forma normal (3NF)** indica que se trata de una relación que se encuentra en primera y segunda formas normales y en la que no hay ningún atributo que no sea de clave candidata que dependa transitivamente de *alguna clave candidata*. En esta definición, los atributos de clave candidata son los que forman parte de alguna clave candidata.

Cuestiones de repaso

13.1. Describa el propósito del proceso de normalización de los datos.

13.2. Explique las formas alternativas en que puede usarse la normalización como ayuda al diseño de una base de datos.

13.3. Describa los tipos de anomalías de actualización que puede introducirse en una normalización que tenga datos redundantes.

13.4. Describa el concepto de dependencia funcional.

13.5. ¿Cuáles son las características principales de las dependencias funcionales que se utilizan para la normalización?

13.6. Describa cómo identifica normalmente un diseñador de base de datos el conjunto de dependencias funcionales asociadas con una relación.

13.7. Describa las características de una tabla en forma no normalizada (UNF, Unnormalized Form) y describa cómo puede convertirse dicha tabla en una relación en primera forma normal (1NF).

13.8. ¿Cuál es la mínima forma normal que una relación debe satisfacer? Proporcione una definición para dicha forma normal.

13.9. Describa las dos técnicas existentes para convertir una tabla en forma no normalizada (UNF) en una relación o relaciones en primera forma normal (1NF).

13.10. Describa el concepto de dependencia funcional completa e indique cómo se relaciona este concepto con la forma 2NF. Proporcione un ejemplo para ilustrar su respuesta.

13.11. Describa el concepto de dependencia transitiva e indique cómo se relaciona este concepto con la forma 3NF. Proporcione un ejemplo para ilustrar su respuesta.

13.12. Explique cómo difieren las definiciones de las formas 2NF y 3NF basadas en las claves principales de las definiciones generales de formas 2NF y 3NF. Proporcione un ejemplo para ilustrar su respuesta.

Ejercicios

13.13. Continúe el proceso de normalización de las relaciones 1NF Client y PropertyRentalOwner mostradas en la Figura 13.13, hasta obtener relaciones 3NF. Al final del proceso, compruebe que las relaciones 3NF resultantes son iguales que las generadas a partir de la relación 1NF alternativa ClientRental mostrada en la Figura 13.16

Es importante observar que si se realiza una combinación natural con cualesquiera dos de las relaciones se generarán tuplas espurias. Sin embargo, si se realiza la combinación de las tres se obtendrá la relación PropertyItemSupplier original.

El lector interesado puede encontrar un análisis detallado de la forma normal 5NF en Date (2003), Elmasri y Navathe (2003) y Hawryszkiewycz (1994).

Resumen del capítulo

- Pueden usarse **reglas de inferencia** para identificar el conjunto de *todas* las dependencias funcionales asociadas con una relación. Este conjunto de dependencias puede ser muy grande para una relación dada.
- Puede emplearse una serie de reglas de inferencia denominadas **axiomas de Armstrong** para identificar el conjunto *mínimo* de dependencias funcionales a partir del conjunto de todas las dependencias funcionales de una relación.
- La **forma normal de Boyce–Codd (Boyce–Codd Normal Form, BCNF)** es una relación en la que todo determinante es una clave candidata.
- La **cuarta forma normal (Fourth Normal Form, 4NF)** es una relación que está en forma BCNF y no contiene dependencias multivaluadas no triviales. Una **dependencia multivaluada (multi-valued dependency, MVD)** representa una dependencia entre atributos (A, B y C) de una relación tal que para cada valor de A hay un conjunto de valores de B y un conjunto de valores de C, siendo los conjuntos de valores de B y C independientes entre sí.
- Una **dependencia de combinación sin pérdidas** es una propiedad de la descomposición que significa que no se generan tuplas espurias cuando se combinan las relaciones mediante una operación de combinación natural.
- La **quinta forma normal (Fifth Normal Form, 5NF)** es una relación que no contiene dependencias de combinación. Para una relación R con subconjuntos de atributos denominados A, B, . . . , Z, la relación R satisfará una **dependencia de combinación** si y sólo si todo valor legal de R es igual a la combinación de sus proyecciones sobre A, B, . . . , Z.

Cuestiones de repaso

14.1. Describa el propósito de utilizar reglas de inferencia para identificar dependencias funcionales en una relación dada.

14.2. Explique el propósito de los Axiomas de Armstrong.

14.3. Explique el propósito de la forma normal de Boyce–Codd (BCNF) y explique cómo difiere la forma BCNF de la forma 3NF. Proporcione un ejemplo para ilustrar su respuesta.

14.4. Describa el concepto de dependencia multivaluada y explique cómo se relaciona este concepto con la forma 4NF. Proporcione un ejemplo para ilustrar su respuesta.

14.5. Describa el concepto de dependencia de combinación y explique cómo se relaciona este concepto con la forma 5NF. Proporcione un ejemplo para ilustrar su respuesta.

Ejercicios

14.6. Después de completar el Ejercicio 13.14, examine las relaciones 3NF creadas para representar los atributos mostrados en el formulario del *Wellmeadows Hospital* mostrado en la Figura 13.18. Determine si estas relaciones están también en forma BCNF. Si no es así, transforme a BCNF las relaciones que no lo estén.

14.7. Después de completar el Ejercicio 13.15, examine las relaciones 3NF creadas para representar los atributos mostrados en la relación que contiene los datos de citas de dentistas/pacientes de la Figura 13.19.

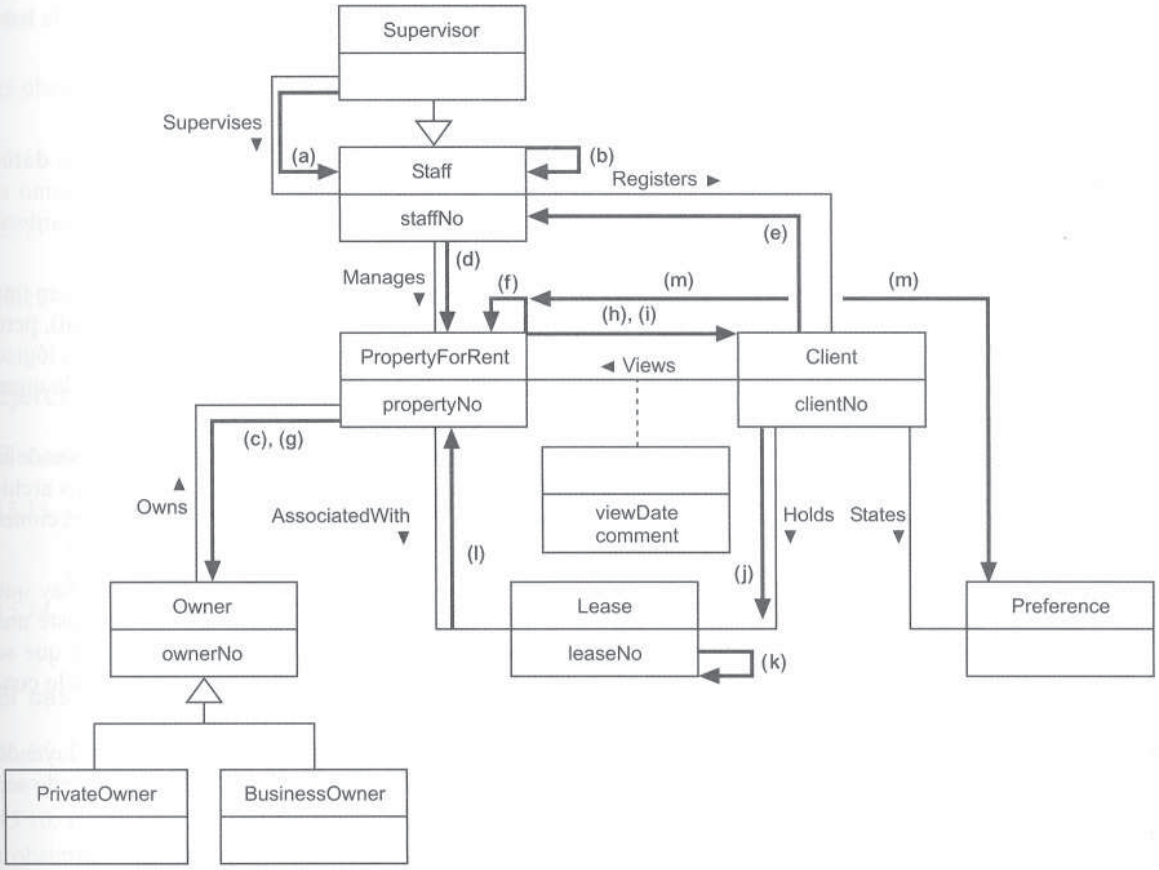

Figura 15.9. Utilización de las rutas de transacción para comprobar que el modelo conceptual soporta las transacciones de los usuarios.

Paso 1.9 Repasar el modelo de datos conceptual con los usuarios

Objetivo Revisar el modelo conceptual de los datos con los usuario para comprobar si éstos consideran el modelo como una representación 'verdadera' de los requisitos de datos de la organización.

Antes de completar el Paso 1, revisamos el modelo conceptual de los datos con el usuario. El modelo conceptual de los datos incluye el diagrama ER y la documentación de soporte que describe el modelo. Si hay alguna anomalía en el modelo de datos, será necesario realizar los cambios apropiados, lo cual puede requerir que se repitan algunos de los pasos anteriores. Deberemos repetir este proceso hasta que el usuario esté dispuesto a 'autorizar' el modelo, considerándolo como una representación 'verdadera' de la parte de la organización que estamos modelando.

Los pasos de esta metodología se resumen en el Apéndice F. El siguiente capítulo describe los pasos relativos a la metodología del diseño lógico de la base de datos.

Resumen del capítulo

- Una **metodología de diseño** es un enfoque estructurado que utiliza procedimientos, técnicas, herramientas y ayudas documentales para facilitar el proceso de diseño y darlo soporte.

- El diseño de una base de datos incluye tres fases principales: diseño **conceptual**, **lógico** y **físico** de la base de datos.
- El **diseño conceptual de la base de datos** es el proceso de construir un modelo de los datos utilizado en una organización, de forma independiente de *todas* las consideraciones físicas.
- El diseño conceptual de la base de datos comienza con la creación de un **modelo conceptual de los datos** de la organización, el cual es enteramente independiente de detalles de implementación tales como el SGBD seleccionado, los programas de aplicación, los lenguajes de programación, la plataforma hardware, cuestiones de rendimiento o cualquier otra consideración física.
- El **diseño lógico de la base de datos** es el proceso de construir un modelo de los datos utilizados en una empresa basado en un modelo de datos específico (como por ejemplo el modelo de datos relacional), pero de forma independiente de cualquier SGBD concreto y de otras consideraciones físicas. El diseño lógico de la base de datos traduce el modelo conceptual de los datos a un **modelo lógico de los datos** de la organización.
- El **diseño físico de la base de datos** es el proceso de producir una descripción de la implementación de la base de datos en el almacenamiento secundario; describe las relaciones base, la organización de los archivos y de los índices utilizados para conseguir un acceso eficiente a los datos y cualesquiera restricciones de integridad asociadas y medidas de seguridad implementadas.
- La fase de diseño físico de la base de datos permite al diseñador tomar decisiones sobre cómo hay que implementar la base de datos. Por tanto, el **diseño físico** está adaptado a un SGBD concreto. Existe una cierta realimentación entre el diseño físico y el diseño conceptual/lógico, porque las decisiones que se tomen durante el diseño físico para mejorar las prestaciones pueden afectar a la estructura del modelo conceptual/lógico de los datos.
- Hay varios factores críticos que determinan el éxito de la etapa de diseño de la base de datos, incluyendo por ejemplo el trabajar interactivamente con los usuarios y estar dispuesto a repetir los pasos cuando sea necesario.
- El objetivo principal del Paso 1 de la metodología consiste en construir un modelo conceptual adecuado a los requisitos de datos de la organización. Un modelo conceptual de los datos comprende: tipos de entidad, tipos de relación, atributos, dominios de atributos, claves principales y claves alternativas.
- Un modelo conceptual de los datos está soportado en una serie de documentos, como diagramas ER y un diccionario de datos, que se generan durante el desarrollo del modelo.
- El modelo conceptual de los datos se valida para garantizar que soporte las transacciones requeridas. Hay dos técnicas posibles para garantizar que el modelo conceptual de los datos soporte las transacciones necesarias: (1) comprobar que toda la información (entidades, relaciones y sus atributos) requerida por cada transacción está incluida en el modelo, documentando una descripción de los requisitos de cada transacción; (2) representar diagramáticamente la ruta que cada transacción toma, dibujándola directamente en el diagrama ER.

Cuestiones de repaso

15.1. Describa el propósito de una metodología de diseño.

15.2. Describa las fases principales del diseño de una base de datos.

15.3. Identifique diversos factores de importancia para que el diseño de una base de datos resulte adecuado.

15.4. Explique el importante papel que juegan los usuarios en el proceso de diseño de una base de datos.

15.5. Describa el objetivo principal del diseño conceptual de la base de datos.

15.6. Identifique los pasos principales asociados con el diseño conceptual de la base de datos.

15.7. ¿Cómo identificaría los tipos de entidad y los tipos de relación a partir de la especificación de requisitos del usuario?

Paso 2.6.3 Revisión del modelo lógico global de los datos con los usuarios

Objetivo Revisar el modelo lógico de datos global con los usuarios para verificar que éstos consideran el modelo como una representación real de los requisitos de datos de la organización.

El modelo lógico global de los datos de la organización debería estar completo y ser lo suficientemente preciso. Es necesario revisar el modelo y la documentación que lo describe con los usuarios para verificar que se trata de una representación real de la organización.

Para facilitar la descripción de las tareas asociadas con el Paso 2.6, es necesario utilizar los términos 'modelo lógico de datos *local*' y 'modelo lógico de datos *global*'. Sin embargo, al final de este paso, cuando los modelos de datos locales hayan sido combinados en un *único* modelo de datos global, dejará de ser necesaria la distinción entre los modelos de datos que hacen referencia a algunas de las vistas de usuario o a todas ellas. Por tanto, al completar este paso, utilizaremos el término 'modelo lógico de datos' para hacer referencia al único modelo de datos global, y emplearemos dicho término en lo que resta de la metodología.

Paso 2.7 Verificar las consideraciones derivadas del crecimiento futuro

Objetivo Determinar si cabe prever para el futuro cambios significativos y evaluar si el modelo lógico de datos puede aceptar dichos cambios.

El diseño lógico de la base de dato concluye tomando en consideración si el modelo lógico de datos (que puede o no haber sido desarrollado utilizando el Paso 2.6) es susceptible de ampliación para soportar posibles desarrollos futuros. Si el modelo sólo puede aceptar los actuales requisitos, la vida del modelo puede ser relativamente corta y requerir un trabajo de reconstrucción significativo para satisfacer los nuevos requisitos. Es importante desarrollar un modelo que sea *ampliable* y que pueda evolucionar para soportar los nuevos requisitos con un efecto mínimo sobre los usuarios existentes. Por supuesto, esto puede ser muy difícil de conseguir, ya que la organización puede no saber lo que desea hacer en el futuro. Incluso si sabe qué quiere hacer en el futuro, puede que sea prohibitivamente caro tanto en tiempo como en dinero prever desde el principio los posibles desarrollos futuros. Por tanto, es posible que sea necesario ser un tanto selectivo en cuanto a qué funcionalidades soportar. En consecuencia, merece la pena examinar el modelo para comprobar si puede ser ampliado con un impacto mínimo. Sin embargo, no es necesario incorporar ningún cambio en el modelo de datos a menos que el usuario lo solicite.

Al final del Paso 2, se utiliza el modelo lógico de dato como fuente de información para el diseño físico de la base de datos, diseño que se describirá en los siguientes dos capítulos y que forma los Pasos 3 a 8 de la metodología.

Para los lectores que ya estén familiarizados con el diseño de bases de datos, en el Apéndice G se proporciona un resumen de los pasos de esta metodología.

Resumen del capítulo

- La metodología de diseño de base de datos incluye tres fases principales: diseño conceptual, diseño lógico y diseño físico de la base de datos.
- El **diseño lógico de la base de datos** es el proceso de construir un modelo de los datos utilizados en una empresa basándose en un modelo de datos específico pero sin prestar atención al SGBD concreto que se vaya a utilizar ni a otras consideraciones físicas.
- Un **modelo lógico de los datos** incluye diagramas ER, un esquema relacional y la documentación de soporte, como por ejemplo el diccionario de datos, que se genera al desarrollar el modelo.
- El propósito del Paso 2.1 de la metodología de diseño lógico de bases de datos es construir un **esquema relacional** a partir del modelo de datos conceptual creado en el Paso 1.

- En el Paso 2.2 se valida un esquema relacional utilizando las reglas de normalización para garantizar que cada tabla sea estructuralmente correcta. La **normalización** se utiliza para mejorar el modelo de modo que satisfaga diversas restricciones que eviten la innecesaria duplicación de los datos. En el Paso 2.3 se valida también el esquema relacional para garantizar que soporte las transacciones incluidas en la especificación de requisitos del usuario.
- En el Paso 2.4 se comprueban las restricciones de integridad del modelo lógico de los datos. Las **restricciones de integridad** son las restricciones que hay que imponer a la base de datos para proteger ésta frente a la posibilidad de llegar a ser incompleta, imprecisa o incoherente. Los tipos principales de restricciones de integridad son: datos requeridos, restricciones relativas a los dominios de los atributos, multiplicidad, integridad de las entidades, integridad referencial y restricciones generales.
- En el Paso 2.5 se valida el modelo lógico de los datos junto con los usuarios.
- El Paso 2.6 del diseño lógico de bases de datos es un paso opcional que sólo se necesita si la base de datos tiene múltiples vistas de usuario que están siendo gestionadas mediante la técnica de integración de vistas (*véase* la Sección 9.5), que hace que se creen dos o más modelos lógicos locales de los datos. Un **modelo lógico local de los datos** representa los requisitos de datos de una o más de las vistas de usuario de la base de datos, pero no de todas ellas. En el Paso 2.6, estos modelos de datos se combinan en un **modelo lógico global de los datos** que representa los requisitos de todas las vistas de usuario. Este modelo lógico de los datos se vuelve a validar mediante las técnicas de normalización, comprobándose también si soporta las transacciones requeridas y validándoselo junto con los usuarios.
- El diseño lógico de base de datos concluye con el Paso 2.7, donde se considera si el modelo es susceptible de ampliación para soportar posibles desarrollos futuros. Al final del Paso 2, el modelo lógico de los datos, que puede o no haber sido desarrollado mediante el Paso 2.6, es la fuente de información para el diseño físico de la base de datos que se describe en los Capítulos 17 y 18 y que constituye los Pasos 3 a 8 de la metodología.

Cuestiones de repaso

16.1. Explique el propósito del diseño lógico de bases de datos.

16.2. Describa las reglas para derivar tablas que representen:
   (a) tipos de entidad fuertes;
   (b) tipos de entidad débiles;
   (c) tipos de relaciones binarias uno a muchos (1:*);
   (d) tipos de relaciones binarias uno a uno (1:1);
   (e) tipos de relaciones recursivas uno a uno (1:1);
   (f) tipos de relaciones superclase/subclase;
   (g) tipos de relaciones recursivas muchos a muchos (*:*);
   (h) tipos de relaciones complejas;
   (i) atributos multivaluados.

   Proporcione ejemplos para ilustrar sus respuestas.

16.3. Explique cómo puede utilizarse la técnica de normalización para derivar las tablas derivadas a partir del modelo conceptual de los datos.

16.4. Explique dos técnicas que pueden usarse para verificar que el esquema relacional es capaz de soportar las transacciones necesarias.

16.5. Describa el propósito de las restricciones de integridad e identifique los tipos principales de restricciones de integridad existentes en un modelo lógico de los datos.

16.6. Describa las estrategias alternativas que pueden aplicarse si existe una tupla hija que hace referencia a una tupla padre que queremos borrar.