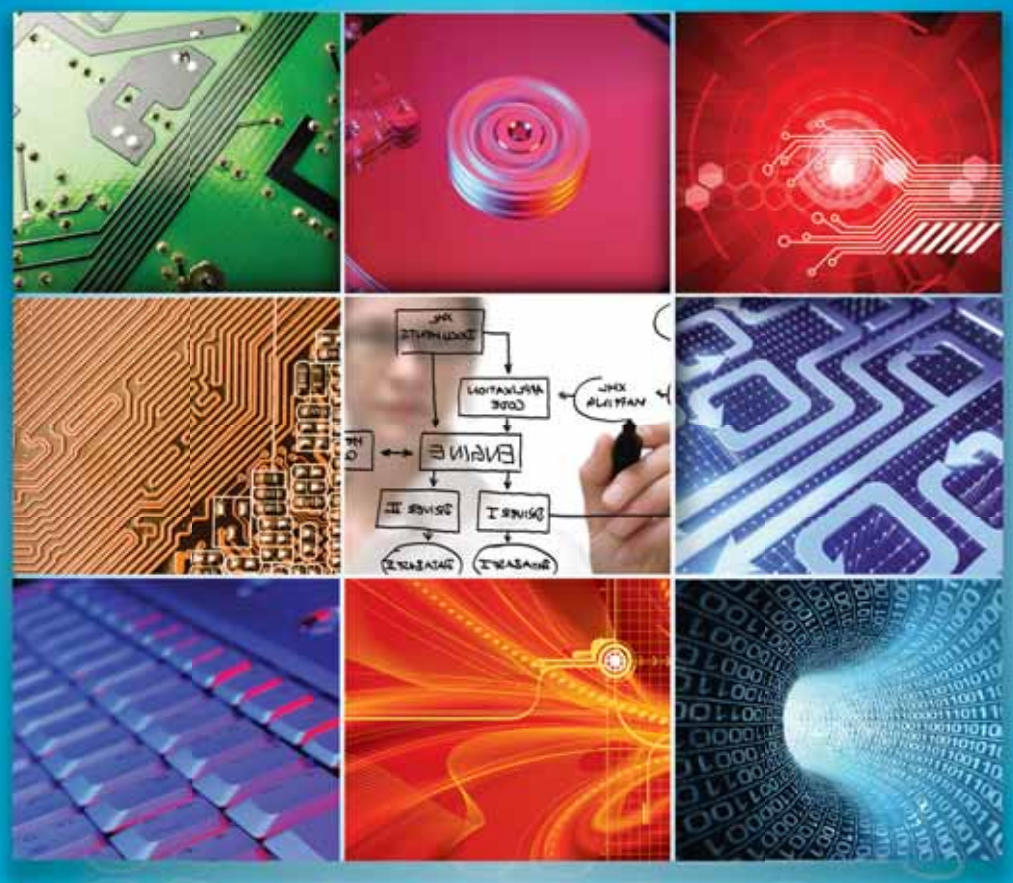
CORONEL / MORRIS / ROB

BASES DE DATOS

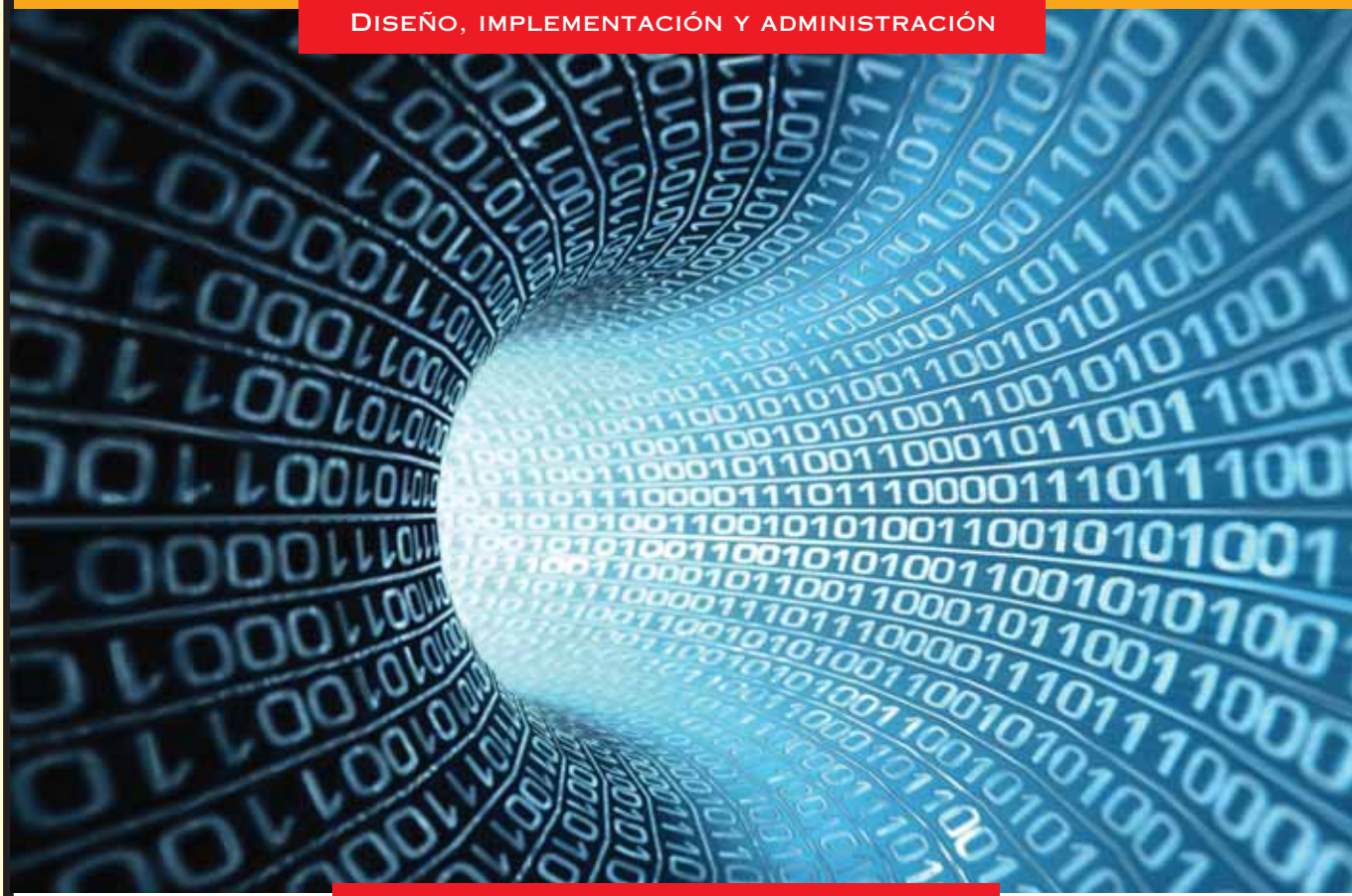
Diseño, implementación y administración



NOVENA EDICIÓN

BASES DE DATOS

Diseño, implementación y administración



Carlos Coronel • Steven Morris • Peter Rob

Traducción
Jorge Humberto Romo Muñoz
Traductor profesional

Revisión técnica
José Sánchez Juárez
Roberto de Luna Caballero
Fabiola Ocampo Botello
Profesores de la Escuela Superior de cómputo
Instituto Politécnico Nacional

CENGAGE Learning™

Australia • Brasil • Corea • España • Estados Unidos • Japón • México • Reino Unido • Singapur

Bases de datos
Diseño, implementación y administración.
Novena edición
Carlos Coronel, Steven Morris, Peter Rob

Director de producto y desarrollo Latinoamérica
Daniel Oti Yvonnet

Director editorial y de producción Latinoamérica
Raúl D. Zendejas Espejel

Editor de desarrollo
Sergio R. Cervantes González

Coordinadora de producción editorial
Abril Vega Orozco

Editora de producción
Gloria Luz Olguín Sarmiento

Coordinador de producción
Rafael Pérez González

Diseño de portada
Itzhack Shelomi

Imagen de portada
Stock Images

Composición tipográfica
Mónica Cervantes González
Heriberto Gachúz Chávez

Traducido del libro:
Database systems
Design, Implementation
and Management Ninth Edition.
Carlos Coronel, Steven Morris, Peter Rob
Publicado en inglés por
Course Technology / Cengage Learning, ©2011
ISBN 13: 978-0-538-46968-5
ISBN 10: 0-538-46968-4

Datos para catalogación bibliográfica:
Coronel, Carlos, Steven Morris, Peter Rob
Bases de datos
Diseño, implementación y administración
Novena edición
ISBN-13: 978-607-481-618-1
ISBN-10: 607-481-618-2

Visite nuestro sitio en:
http://latinoamerica.cengage.com

Impreso en México
1 2 3 4 5 6 7 8 9 12 11

TABLA DE CONTENIDO

PARTE I CONCEPTOS DE BASES DE DATOS

TABLA DE CONTENIDO

PARTE III: DISEÑO E IMPLEMENTACIÓN AVANZADOS

TABLA DE CONTENIDO

TABLA DE CONTENIDO

PARTE V BASES DE DATOS E INTERNET

PARTE VI ADMINISTRACIÓN DE BASES DE DATOS

EN EL SITIO WEB PREMIUM

El sitio web Premium se puede hallar en *cengage.com/mis/coronel*. Localice su tarjeta de acceso premium en el frente de cada compra de un libro nuevo y haga clic en "Create My Account" para iniciar el proceso de registro. Si ha comprado un libro usado, busque en *Database Systems, Ninth Edition* en *www.ichapters.com* en donde puede comprar un acceso al instante.

CAPÍTULO 6

NORMALIZACIÓN DE TABLAS DE BASES DE DATOS

En este capítulo, el lector aprenderá:

- Qué es la normalización y qué función desempeña en el proceso de diseño de bases de datos
- Acerca de las formas normales 1NF, 2NF, 3NF, BCNF 4NF
- Cómo las formas normales se pueden transformar de inferiores a superiores
- Que la normalización y el modelado de ER se usan de manera concurrente para producir un buen diseño de bases de datos
- Que algunas situaciones requieren desnormalización para generar información de manera eficiente

Un buen diseño de base de datos debe corresponderse con buenas estructuras de tablas. En este capítulo, el lector aprenderá a evaluar y diseñar buenas estructuras de tablas para controlar redundancias de datos, con lo cual evitará anomalías de datos. El proceso que da estos resultados deseables se conoce como normalización.

Para reconocer y apreciar las características de una buena estructura de tablas es útil examinar una mala. Por tanto, el capítulo empieza por analizar las características de una mala estructura de tablas y los problemas que acarrea. A continuación aprenderemos a corregir una mala estructura de tablas. Esta metodología dará importantes dividendos: sabremos cómo diseñar una buena estructura de tablas y cómo reparar una mala.

Descubriremos que no sólo mediante la normalización se pueden eliminar anomalías de datos, sino que también un conjunto de estructuras de tablas correctamente normalizado es en realidad menos complicado de usar que uno no normalizado. Además, aprenderemos que el conjunto normalizado de estructuras de tablas refleja más fielmente las operaciones reales de una organización.

6.1 TABLAS DE BASES DE DATOS Y NORMALIZACIÓN

Tener un buen software de base de datos relacional no es suficiente para evitar la redundancia de datos explicada en el capítulo 1. Si las tablas de una base de datos se tratan como si fueran archivos en un sistema de archivos, el sistema de administración de base de datos relacional (RDBMS) nunca tendrá oportunidad para demostrar sus excelentes características para manejar datos.

La tabla es el elemento angular del diseño de bases de datos y, en consecuencia, su estructura es de gran interés. En el ideal, el proceso de diseño de bases de datos que exploramos en el capítulo 4, da buenas estructuras de tablas, pero es posible crear malas estructuras incluso en un buen diseño de bases de datos. Entonces, ¿cómo se reconoce una mala estructura de tabla y cómo se produce una tabla buena? La respuesta a ambas preguntas es la normalización. **Normalización** es un proceso para evaluar y corregir estructuras de tablas a fin de minimizar redundancias de datos, con lo cual se reduce la probabilidad de anomalías de datos. El proceso de normalización comprende asignar atributos a tablas a partir del concepto de determinación que aprendimos en el capítulo 3.

La normalización funciona por medio de una serie de etapas llamadas formas normales. Las primeras tres etapas se describen como primera forma normal (1NF), segunda forma normal (2NF) y tercera forma normal (3NF). Desde un punto de vista estructural, 2NF es mejor que 1NF y 3NF es mejor que 2NF. Para casi todos los propósitos en el diseño de bases de datos de negocios, 3NF es tan alto como sea necesario llegar en el proceso de normalización, pero descubriremos que las estructuras 3NF correctamente diseñadas también satisfacen los requisitos de la cuarta forma normales (4NF).

Aun cuando la normalización es un ingrediente muy importante de diseño de bases de datos, no se debe suponer que el nivel más alto de normalización es siempre el más deseable. En general, cuanto más alta es la forma normal, más operaciones relacionales de combinación se requieren para producir una salida especificada y más recursos son requeridos por el sistema de bases de datos para responder a consultas del usuario final. Un diseño exitoso también debe considerar la demanda del usuario final para una rápida operación. Por tanto, ocasionalmente se espera *desnormalizar* algunas partes de un diseño de base de datos para satisfacer requisitos de operación. La **desnormalización** produce una forma normal más baja; esto es, una 3NF se convertirá a 2NF por medio de desnormalización. No obstante, *el precio que se paga por una mejor operación por medio de desnormalización es una mayor redundancia de datos*.

NOTA

Aun cuando la palabra *tabla* se usa en todo este capítulo, formalmente, la normalización se refiere a *relaciones*. En el capítulo 3 aprendimos que es frecuente que los términos *tabla* y *relación* se usen indistintamente. De hecho, se puede decir que una tabla es la vista de la implementación de una relación lógica que satisface algunas condiciones específicas (tabla 3.1). No obstante, siendo más rigurosos, la relación matemática no permite tuplas duplicadas, mientras que las tuplas duplicadas podrían existir en tablas (vea sección 6.5). También, en terminología de normalización, cualquier atributo que sea al menos parte de una llave se conoce como **atributo primo** en lugar del término más común de **atributo llave**, que se introdujo ya antes. Por el contrario, un **atributo no primo**, o un **atributo no llave**, no es parte de ninguna llave candidata.

6.2 NECESIDAD DE NORMALIZACIÓN

La normalización se utiliza por lo general en coordinación con el modelado entidad-relación que aprendimos en los capítulos previos. Hay dos situaciones comunes en las que los diseñadores de bases de datos la usan. Cuando diseñan una nueva estructura de bases de datos fundamentada en las necesidades de negocios de usuarios finales, el diseñador construirá un modelo de datos usando una técnica como los ERD con notación de "*pata de gallo*". Después que el diseño inicial está completo, el diseñador puede usar normalización para analizar las relaciones que existen entre los atributos dentro de cada entidad, para determinar si la estructura se puede mejorar por medio de normalización. De manera opcional, a los diseñadores de bases de datos se les pide con frecuencia que modifiquen estructuras de datos existentes que puedan estar en la forma de archivos planos, hojas de cálculo o estructuras de bases de datos anteriores. De nueva cuenta, por medio de un análisis de relaciones entre los atributos o campos de la estructura de datos, el diseñador puede usar el proceso de normalización para mejorarla a fin de crear un diseño apropiado de bases de datos. Ya sea para diseñar una nueva estructura o modificar una ya existente, el proceso de normalización es el mismo.

Para tener una mejor idea del proceso de normalización, considere las actividades simplificadas de una base de datos de una compañía constructora que maneja varios proyectos. Cada proyecto tiene su propio número de identificación, nombre, empleados asignados al mismo, etc. Cada empleado tiene un número, nombre y clasificación de trabajo; por ejemplo, ingeniero o técnico en computadoras.

La compañía cobra a sus clientes al facturar las horas empleadas en cada contrato. El costo por hora depende del puesto del empleado. Por ejemplo, una hora de tiempo de un técnico en computadoras se factura a un costo diferente que una hora de tiempo de un ingeniero. Periódicamente se genera un informe que contiene la información que se ve en la tabla 6.1.

El cargo total de la tabla 6.1 es un atributo derivado y, en este punto, no se guarda en la tabla.

La forma más fácil a corto plazo para generar el informe requerido parece ser una tabla cuyo contenido corresponde a necesidades de información (figura 6.1).

FIGURA 6.1 **Representación tabular del formato de informe**

Nombre de la tabla: RPT_FORMAT **Nombre de la base de datos: Ch06_ConstructCo**

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

CONTENIDO EN LÍNEA

Las bases de datos para ilustrar el material de este capítulo se encuentran en el sitio web Premium para este libro.

Observe que los datos de la figura 6.1 reflejan la asignación de empleados a proyectos. En apariencia, un empleado puede ser asignado a más de un proyecto. Por ejemplo, Darlene Smithson (EMP_NUM = 112) ha sido asignada a dos proyectos: Amber Wave y Starflight. Dada la estructura del conjunto de datos, cada proyecto incluye sólo una instancia de cualquier empleado. Por tanto, saber el valor de PROJ_NUM y EMP_NUM permitirá encontrar la clasificación del trabajo y su cargo por hora. Además, el lector conocerá el número total de horas que cada empleado trabajó en cada proyecto. (El cargo total, que es atributo derivado cuyo valor se puede calcular multiplicando las horas facturadas y el cargo por hora, no se ha incluido en la figura 6.1. No hay daño estructural si se incluye este atributo derivado.)

TABLA 6.1

NÚMERO DE PROYECTO	NOMBRE DEL PROYECTO	NÚMERO DE EMPLEADO	NOMBRE DE EMPLEADO	CLASE DE TRABAJO	CARGO/ HORA	HORAS FACTURADAS	CARGO TOTAL
15	Evergreen	103	June E. Arbough	Ingeniero electricista	$ 85.50	23.8	$ 2,034.90
		101	John G. News	Diseñador de base de datos	$105.00	19.4	$ 2,037.00
		105	Alice K. Johnson *	Diseñador de base de datos	$105.00	35.7	$ 3,748.50
		106	William Smithfield	Programador	$ 35.75	12.6	$ 450.45
		102	David H. Senior	Analista de sistemas	$ 96.75	23.8	$ 2,302.65
				Subtotal			**$10,573.50**
18	Amber Wave	114	Annelise Jones	Diseñador de aplicaciones	$ 48.10	25.6	$ 1,183.26
		118	James J. Frommer	Apoyo general	$ 18.36	45.3	$ 831.71
		104	Anne K. Ramoras *	Analista de sistemas	$ 96.75	32.4	$ 3,134.70
		112	Darlene M. Smithson	Analista de DSS	$ 45.95	45.0	$ 2,067.75
				Subtotal			**$ 7,265.52**
22	Rolling Tide	105	Alice K. Johnson	Diseñador de base de datos	$105.00	65.7	$ 6,998.50
		104	Anne K. Ramoras	Analista de sistemas	$ 96.75	48.4	$ 4,682.70
		113	Delbert K. Joenbrood	Diseñador de aplicaciones	$ 48.10	23.6	$ 1,135.16
		111	Geoff B. Wabash	Apoyo de oficina	$ 26.87	22.0	$ 591.14
		106	William Smithfield	Programador	$ 35.75	12.8	$ 457.60
				Subtotal			**$13,765.10**
25	Starflight	107	Maria D. Alonzo	Programador	$ 35.75	25.6	$ 915.20
		115	Travis B. Bawangi	Analista de sistemas	$ 96.75	45.8	$ 4,431.15
		101	John G. News *	Diseñador de base de datos	$105.00	56.3	$ 5,911.50
		114	Annelise Jones	Diseñador de aplicaciones	$ 48.10	33.1	$ 1,592.11
		108	Ralph B. Washington	Analista de sistemas	$ 96.75	23.6	$ 2,283.30
		118	James J. Frommer	Apoyo general	$ 18.36	30.5	$ 559.98
		112	Darlene M. Smithson	Analista de DSS	$ 45.95	41.4	$ 1,902.33
				Subtotal			**$17,595.57**
				Total			**$49,199.69**

Nota: * indica líder de proyecto

Desafortunadamente, la estructura del conjunto de datos de la figura 6.1 no se apega a los requisitos expresados en el capítulo 3 ni maneja datos muy bien. Considere las siguientes deficiencias:

1. El número de proyecto (PROJ_NUM) está aparentemente destinado a ser una llave primaria o al menos parte de una PK, pero contiene valores nulos. (Dada la exposición precedente, se sabe que PROJ_NUM + EMP_NUM definirá cada renglón.)
2. Las entradas de la tabla invitan a inconsistencias de datos. Por ejemplo, el valor JOB_CLASS "Elect. Engineer" podría introducirse como "Elect.Eng". en algunos casos, "El.Eng". en otros y "EE" en otros.
3. La tabla presenta redundancias de datos, las cuales dan las siguientes anomalías:
   a) *Anomalías de actualización.* Modificar la JOB_CLASS para el empleado 105 requiere (potencialmente) muchas alteraciones, una para cada EMP_NUM = 105.
   b) *Anomalías de inserción.* Sólo para completar la definición de un renglón, un nuevo empleado debe asignarse a un proyecto. Si el empleado todavía no es asignado, un proyecto fantasma debe crearse para completar la entrada de datos del empleado.
   c) *Anomalías de eliminación.* Suponga que sólo un empleado está asociado con un proyecto dado. Si ese empleado abandona la compañía y sus datos se borran, la información del proyecto también se borrará. Para impedir la pérdida de la información del proyecto, debe ser creado un empleado ficticio sólo para salvar la información del proyecto.

A pesar de esas deficiencias estructurales, la estructura de tabla *parece* funcionar; el informe se genera con facilidad. Desafortunadamente, podría dar resultados variables dependiendo de qué anomalía de datos haya ocurrido. Por ejemplo, si se desea imprimir un informe para mostrar el valor total de "horas trabajadas" por la clasificación de trabajo "Diseñador de base de datos", ese informe no incluirá datos para los registros "DB Design" y "Database Design". Estas anomalías en los informes causan una multitud de problemas a gerentes y no pueden ser corregidas mediante programación de aplicación.

Incluso si una muy cuidadosa auditoría de entrada de datos puede eliminar casi todos los problemas en los informes (a un alto costo), es fácil demostrar que hasta una entrada de datos sencilla se hace ineficiente. Dada la existencia de anomalías de actualización, suponga que Darlene M. Smithson es asignada a trabajar en el proyecto Evergreen. La capturista debe actualizar el archivo PROJECT con la entrada:

15 Evergreen 112 Darlene M Smithson Analista DSS $45.95 0.0

para que haya correspondencia con los atributos PROJ_NUM, PROJ_NAME, EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR y HOURS. (Cuando Ms. Smithson acaba de ser asignada al proyecto, todavía no ha trabajado, de modo que el número total de horas trabajadas es 0.0.)

NOTA

Recuerde que la convención de dar nombres facilita ver qué significa cada atributo y cuál es su probable origen. Por ejemplo, PROJ_NAME usa el prefijo PROJ para indicar que el atributo está asociado con la tabla PROJECT, en tanto que el componente NAME se explica por sí mismo, también. No obstante, recuerde que la longitud del nombre también debe tenerse en cuenta, en especial en la designación del prefijo. Por esa razón, el prefijo CHG se utilizó en lugar de CHARGE. (Dado el contexto de la base de datos, no es probable que ese prefijo se malentienda.)

Cada vez que otro empleado sea asignado a un proyecto, algunas entradas de datos (por ejemplo PROJ_NAME, EMP_NAME y CHG_HOUR) serán innecesariamente repetidos. Imagine el trabajo de introducir datos cuando deban hacerse 200 o 300 entradas de tabla. Nótese que la entrada del número de empleado debe ser suficiente para identificar a Darlene M. Smithson, la descripción de su trabajo y su cargo por hora. Como hay sólo una persona identificada por el número 112, las características de esa persona (nombre, clasificación de trabajo, etc.), no debe introducirse cada vez que se actualice el archivo. Desafortunadamente, la estructura que se ve en la figura 6.1 no tiene tolerancia para esa posibilidad.

La redundancia de datos evidente en la figura 6.1 lleva a espacio desperdiciado en el disco. Aún más, produce anomalías de datos. Suponga que la capturista ha introducido los datos como

15 Evergreen 112 Darla Smithson Analista DCS $4.95 0.0

A primera vista parece que la entrada de datos es correcta pero, ¿Evergreen es el mismo proyecto que Evergreen? Y ¿se supone que Analista DCS es Analista DSS? ¿Darla Smithson es la misma persona que Darlene M. Smithson? Esta confusión es un problema de integridad de datos que fue causado porque la entrada de datos no se apegaba a la regla de que todas las copias de datos redundantes deben ser idénticas.

La posibilidad de introducir problemas de integridad de datos causados por redundancia debe ser considerada cuando se diseñe una base de datos. El ambiente de bases de datos relacional se presta especialmente bien para ayudar al diseñador a superar esos problemas.

6.3 EL PROCESO DE NORMALIZACIÓN

En esta sección aprenderemos a usar normalización para producir un conjunto de tablas normalizadas para guardar los datos que se usarán para generar la información requerida. El objetivo de normalización es asegurar que cada tabla se apegue al concepto de relaciones bien formadas, es decir, tablas que tienen las siguientes características:

- Cada tabla representa un solo tema. Por ejemplo, una tabla de curso contendrá sólo datos que directamente pertenecen a cursos. Del mismo modo, una tabla de estudiante contendrá sólo datos de estudiante.
- Ningún ítem de datos se guardará *innecesariamente* en más de una tabla (en pocas palabras, las tablas tienen mínima redundancia controlada). La razón para este requisito es asegurar que los datos se actualicen en sólo un lugar.
- Todos los atributos no primos de una tabla son dependientes de la llave primaria; toda la llave primaria es nada más que la llave primaria. La razón para este requisito es asegurar que los datos sean identificables de manera única por un valor de llave primaria.
- Cada tabla está libre de anomalías de inserción, actualización o eliminación. Esto es para asegurar la integridad y consistencia de los datos.

Para lograr el objetivo, el proceso de normalización nos lleva por los pasos que conducen a formas normales sucesivamente más altas. Las formas normales más comunes y sus características básicas aparecen en la tabla 6.2. Aprenderemos los detalles correspondientes en las secciones indicadas.

TABLA 6.2 Formas normales

FORMA NORMAL	CARACTERÍSTICA	SECCIÓN
Primera forma normal (1NF)	Formato de tabla, sin grupos repetidos y PK identificada	6.3.1
Segunda forma normal (2NF)	1NF y sin dependencias parciales	6.3.2
Tercera forma normal (3NF)	2NF y sin dependencias transitivas	6.3.3
Forma normal Boyce-Codd (BCNF)	Todo determinante es llave candidata (caso especial de 3NF)	6.6.1
Cuarta forma normal (4NF)	3NF y sin dependencias de valor múltiple independientes	6.6.2

El concepto de llaves es central para entender la normalización. Recuerde del capítulo 3 que una llave candidata es una superllave mínima (irreductible). La llave primaria es la llave candidata seleccionada para ser el medio primario empleado para identificar los renglones de la tabla. Aun cuando la normalización se presenta por lo general desde la perspectiva de llaves candidatas, por sencillez mientras se explica inicialmente el proceso de normalización, haremos la suposición de que por cada tabla hay sólo una llave candidata y, por tanto, esta es la llave primaria.

Desde el punto de vista del modelador de datos, el objetivo de la normalización es asegurar que todas las tablas queden al menos en tercera forma normal (3NF). Incluso existen formas normales de orden superior, pero las formas normales, como la quinta forma normal (5NF), y forma normal llave dominio (DKNF) no son probables de encontrarse

en un ambiente de negocios y son, principalmente, de interés teórico. Con más frecuencia, estas formas normales de orden superior aumentan combinaciones (haciendo lenta la operación) sin agregar ningún valor en la eliminación de redundancia de datos. Algunas aplicaciones muy especializadas, por ejemplo investigación estadística, podrían requerir normalización superior a 4NF, pero caen fuera del propósito de casi todas las operaciones de negocios. Como esta obra se enfoca en aplicaciones prácticas de técnicas de bases de datos, las formas normales de orden superior no se tratan.

Dependencia funcional

Antes de resumir el proceso de normalización, es buena idea repasar los conceptos de determinación y dependencia funcional que se trataron en detalle en el capítulo 3. La tabla 6.3 resume los conceptos principales.

TABLA 6.3 Conceptos de dependencia funcional

CONCEPTO	DEFINICIÓN
Dependencia funcional	El atributo *B* es funcionalmente dependiente en forma completa en el atributo *A* si cada valor de *A* determina un valor de *B* y sólo uno. Ejemplo: PROJ_NUM → PROJ_NAME (léase como "PROJ_NUM funcionalmente determina PROJ_NAME") En este caso, el atributo PROJ_NUM se conoce como atributo "determinante" y el atributo PROJ_NAME como atributo "dependiente".
Dependencia funcional (definición generalizada)	El atributo *A* determina al atributo *B* (esto es, *B* es funcionalmente dependiente de *A*) si todos los renglones de la tabla que concuerdan en valor para el atributo *A* también concuerdan en valor para el atributo *B*.
Dependencia totalmente funcional (llave compuesta)	Si el atributo *B* es funcionalmente dependiente de una llave compuesta *A* pero no de ningún subconjunto de ella, el atributo *B* es funcionalmente dependiente en forma completa de *A*.

Es esencial entender estos conceptos porque se usan para derivar el conjunto de dependencias funcionales para una relación determinada. El proceso de normalización trabaja una relación a la vez, identificando sus dependencias y normalizando la relación. Como se verá en las siguientes secciones, la normalización empieza por identificar las dependencias de una relación determinada y progresivamente descomponiendo la relación (tabla) en un conjunto de nuevas relaciones (tablas) basadas en las dependencias identificadas.

Dos tipos de dependencias funcionales que son de especial interés en la normalización son las parciales y las transitivas. Una **dependencia parcial** existe cuando hay una dependencia funcional en la que el determinante es sólo parte de la llave primaria (recuerde que estamos suponiendo que hay sólo una llave candidata). Por ejemplo, si (A, B) → (C, D), B → C y (A, B) es la llave primaria, entonces la dependencia funcional B → C es una dependencia parcial porque sólo parte de la llave primaria (B) se necesita para determinar el valor de C. Las dependencias parciales tienden a ser más bien sencillas y fáciles de identificar.

Existe **dependencia transitiva** cuando hay dependencias funcionales tales que X → Y, Y → Z y X es la llave primaria. En ese caso, la dependencia X → Z es una dependencia transitiva porque X determina el valor de Z por medio de Y. A diferencia de las dependencias parciales, las dependencias transitivas son más difíciles de identificar entre un conjunto de datos. Por fortuna, hay una forma más fácil para identificar dependencias transitivas. Se presentará una dependencia transitiva sólo cuando existe dependencia funcional entre atributos no primos. En el ejemplo previo, la dependencia transitiva real es X → Z. No obstante, la dependencia Y → Z es señal de que existe una dependencia transitiva. En consecuencia, en toda la exposición del proceso de normalización, la existencia de una dependencia funcional entre atributos no primos será considerada como signo de una dependencia transitiva. Para resolver los problemas relacionados con dependencias transitivas, los cambios a la estructura de la tabla se hacen con base en la dependencia funcional que señala la existencia de dependencia transitiva. Por tanto, para simplificar la descripción de la normalización, de ahora en adelante nos referiremos a la dependencia indicadora como la dependencia transitiva.

6.3.1 CONVERSIÓN A LA PRIMERA FORMA NORMAL

Debido a que el modelo relacional los ve como parte de una tabla o un conjunto de tablas en las que todos los valores llave deben estar identificados, los datos descritos en la figura 6.1 no podrían guardarse como se muestra. Nótese que esa figura contiene lo que se conoce como grupos repetidores. Un **grupo repetidor** deriva su nombre del hecho de que un grupo de múltiples entradas del mismo tipo pueden existir para cualquier instancia de atributo llave *individual*. En la figura 6.1, observe que cada instancia de número individual de proyecto (PROJ_NUM) puede hacer referencia a un grupo de entradas de datos relacionados. Por ejemplo, el proyecto Evergreen (PROJ_NUM = 15) muestra cinco entradas en este punto y esas entradas están relacionadas porque cada una de ellas comparte la característica PROJ_NUM = 15. Cada vez que un nuevo registro se introduzca para el proyecto Evergreen, el número de entradas del grupo crece en uno.

Una tabla relacional no debe contener grupos repetidores. La existencia de grupos repetidores da evidencia de que la tabla RPT_FORMAT de la figura 6.1 no satisface incluso los requisitos mínimos de la forma normal, reflejando así redundancias de datos.

Normalizar la estructura de tabla reducirá las redundancias de datos. Si existen grupos repetidores, deben ser eliminados al asegurarse que cada renglón define una sola entidad. Además, las dependencias deben ser identificadas para diagnosticar la forma normal, cuya identificación permitirá saber en dónde estamos en el proceso de normalización. Éste empieza con un sencillo procedimiento de tres pasos.

Paso 1: eliminar los *grupos repetidores*

Empecemos por presentar los datos en un formato tabular, donde cada celda tiene un solo valor y donde no hay *grupos repetidores*. Para eliminar éstos, elimine los valores nulos asegurándose que el atributo de cada uno de los *grupos repetidores* contiene un valor apropiado de datos. Ese cambio convierte la tabla de la figura 6.1 a 1NF en la figura 6.2.

FIGURA 6.2 **Tabla en primera forma normal**

Nombre de la tabla: DATA_ORG_1NF **Nombre de la base de datos: Ch06_ConstructCo**

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
15	Evergreen	101	John G. News	Database Designer	105.00	19.4
15	Evergreen	105	Alice K. Johnson *	Database Designer	105.00	35.7
15	Evergreen	106	William Smithfield	Programmer	35.75	12.6
15	Evergreen	102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
18	Amber Wave	118	James J. Frommer	General Support	18.36	45.3
18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	96.75	48.4
22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	26.87	22.0
22	Rolling Tide	106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
25	Starflight	115	Travis B. Bawangi	Systems Analyst	96.75	45.8
25	Starflight	101	John G. News *	Database Designer	105.00	56.3
25	Starflight	114	Annelise Jones	Applications Designer	48.10	33.1
25	Starflight	108	Ralph B. Washington	Systems Analyst	96.75	23.6
25	Starflight	118	James J. Frommer	General Support	18.36	30.5
25	Starflight	112	Darlene M. Smithson	DSS Analyst	45.95	41.4

Paso 2: identificar la llave primaria

El diseño de la figura 6.2 representa más de un simple cambio cosmético. Incluso un observador indiferente notará que PROJ_NUM no es una llave primaria adecuada porque el número de proyecto no identifica de manera única a todos los atributos restantes de entidad (renglón). Por ejemplo, el valor 15 de PROJ_NUM puede identificar a cualquiera de

cinco empleados. Para mantener una llave primaria apropiada que identificará *de manera única* el valor de cualquier atributo, la nueva llave debe estar compuesta de una *combinación* de PROJ_NUM y EMP_NUM. Por ejemplo, usando los datos que se muestran en la figura 6.2, si sabemos que PROJ_NUM = 15 y EMP_NUM = 103, las entradas para los atributos PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR y HOURS debe ser Evergreen, June E. Arbough, ingeniero electricista, $84.50 y 23.8, respectivamente.

Paso 3: identificar todas las dependencias

La identificación de la llave primaria (PK) en el paso 2 significa que ya ha identificado la siguiente dependencia:

PROJ_NUM, EMP_NUM → PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR, HOURS

Esto es, los valores de PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR y HOURS son todos ellos dependientes de (es decir, están determinados por) la combinación de PROJ_NUM y EMP_NUM. Hay dependencias adicionales. Por ejemplo, el número de proyecto identifica (determina) el nombre del proyecto. En otras palabras, el nombre del proyecto es dependiente del número de proyecto. Se puede escribir esa dependencia como

PROJ_NUM → PROJ_NAME

También, si conocemos el número de empleado, también conocemos su nombre, clasificación del trabajo y el cargo por hora. Por tanto, se puede identificar la dependencia mostrada a continuación:

EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR

No obstante, dados los componentes previos de dependencia, se puede ver que conocer la clasificación del trabajo significa conocer el cargo por hora correspondiente. En otras palabras, se puede identificar una última dependencia:

JOB_CLASS → CHG_HOUR

Existe esta dependencia entre dos atributos no primos; en consecuencia, es una señal de que existe dependencia transitiva y nos referiremos a ella de esta manera. Las dependencias que acabamos de examinar también se pueden describir con ayuda del diagrama que se muestra en la figura 6.3. Debido a que ese diagrama describe todas las dependencias encontradas dentro de una estructura determinada de tabla, se conoce como **diagrama de dependencia.** Los diagramas de dependencia son muy útiles para tener una vista global de todas las relaciones entre los atributos de una tabla y su uso los hace menos probables de que descuidemos una dependencia importante.

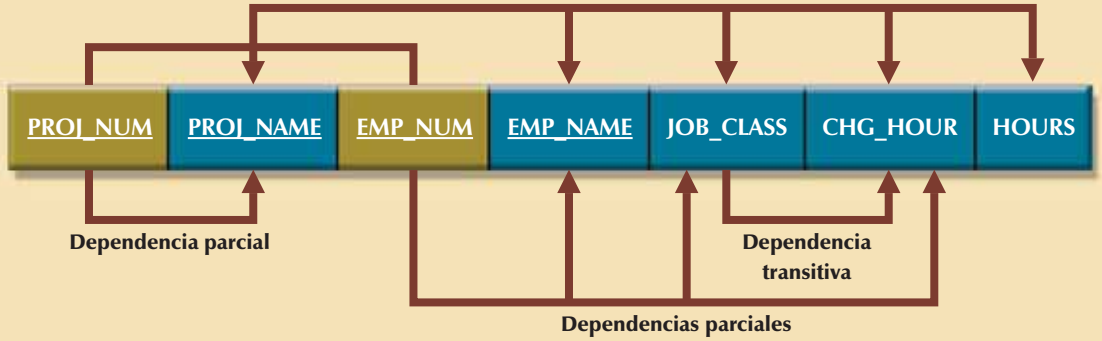
Al examinar la figura 6.3, observe las siguientes características de un diagrama de dependencia:

1. Los atributos de llave primaria están en negritas, subrayados y sombreados en color diferente.
2. Las flechas arriba de los atributos indican todas las dependencias deseables, es decir, las que están basadas en la llave primaria. En este caso, nótese que los atributos de entidad son dependientes de la *combinación* de PROJ_NUM y EMP_NUM.
3. Las flechas abajo del diagrama de dependencia indican dependencias menos deseables. Existen dos tipos de éstas:

   a) *Dependencias parciales*. Es necesario conocer sólo PROJ_NUM para determinar el PROJ_NAME; esto es, el PROJ_NAME depende sólo de parte de la llave primaria. Y sólo es necesario conocer el EMP_NUM para encontrar el EMP_NAME, la JOB_CLASS y el CHG_HOUR. Una dependencia basada en sólo una parte de una llave primaria compuesta es una dependencia parcial.

   b) *Dependencias transitivas*. Observe que CHG_HOUR depende de JOB_CLASS. Como ni CHG_HOUR ni JOB_CLASS es un atributo primo, esto es, ningun atributo es al menos parte de una llave, la condición es una dependencia transitiva. En otras palabras, una dependencia transitiva es una dependencia de un atributo no primo de otro atributo no primo. El problema con dependencias transitivas es que todavía dan anomalías de datos.

FIGURA 6.3 **Diagrama de dependencia de primera forma normales (1NF)**



1NF (PROJ_NUM, EMP_NUM, PROJ_NAME, EMP_NAME, JOB_CLASS, HOURS, HOUR)

DEPENDENCIAS PARCIALES:

(PROJ_NUM → PROJ_NAME)
(EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR)

DEPENDENCIA TRANSITIVA:

JOB_CLASS → CHG_HOUR)

Observe que la figura 6.3 incluye el esquema relacional para la tabla en 1NF y una notación textual por cada dependencia identificada.

NOTA

El término **primera forma normal (1NF)** describe el formato tabular en el que:

- Todos los atributos llave están definidos.
- No hay grupos repetidores en la tabla. En otras palabras, cada intersección de renglón/columna contiene un y sólo un valor, no un conjunto de ellos.
- Todos los atributos son dependientes de la llave primaria.

Todas las tablas relacionales satisfacen los requisitos de la 1NF. El problema con la estructura de la tabla 1NF que se ve en la figura 6.3 es que contiene dependencias parciales, es decir, dependencias basadas en sólo una parte de la llave primaria.

Si bien es cierto que a veces se usan dependencias parciales por razones de operación, deben usarse con precaución. (Si los requisitos de información parecen dictar el uso de dependencias parciales, es tiempo de evaluar la necesidad de un diseño de almacén de datos, lo cual se estudia en el capítulo 13.) Esa precaución se justifica porque una tabla que contiene dependencias parciales está todavía sujeta a redundancias de datos y, por tanto, a varias anomalías. Ocurren redundancias de datos porque toda entrada de renglón requiere duplicación de datos. Por ejemplo, si Alice K. Johnson envía su bitácora de trabajo, entonces el usuario tendría que hacer múltiples entradas durante el curso de un día. Por cada entrada, las EMP_NAME, JOB_CLASS y CHG_HOUR deben introducirse cada vez, aun cuando los valores de atributo sean idénticos por cada renglón introducido. Esa duplicación de trabajo es muy ineficiente. Lo que es más, el trabajo de duplicación ayuda a crear anomalías; nada impide que el usuario teclee versiones ligeramente diferentes del nombre del empleado, su puesto y la paga por hora. Por ejemplo, el nombre del empleado para EMP_NUM = 102 podría escribirse como Dave Senior o D. Senior. El nombre del proyecto también podría introducirse correctamente como Evergreen o escribirse mal como Evergreen. Tales anomalías de datos violan las reglas de integridad y consistencia de la base de datos relacional.

6.3.2 Conversión a la segunda forma normal

La conversión a 2NF se hace sólo cuando la 1NF tiene una llave primaria compuesta. Si la 1NF tiene una llave primaria de un solo atributo, entonces la tabla está automáticamente en la 2NF. La conversión de 1NF a 2NF es sencilla. Empezando con el formato 1NF que se ve en la figura 6.3, se hace lo siguiente:

Paso 1: hacer nuevas tablas para eliminar dependencias parciales

Por cada componente de la llave primaria que actúe como determinante en una dependencia parcial, genere una nueva tabla con una copia de ese componente como llave primaria. Mientras estos componentes se coloquen en las nuevas tablas, es importante que también permanezcan en la original. Es primordial que los determinantes continúen en la tabla original porque serán llaves foráneas para las relaciones que se requieren para vincular estas nuevas tablas a la original. Para la construcción de nuestro diagrama de dependencia revisado, escriba cada componente de llave en un renglón separado; a continuación escriba la llave original (compuesta) en el último renglón. Por ejemplo,

PROJ_NUM

EMP_NUM

PROJ_NUM EMP_NUM

Cada componente se convertirá en la llave en una nueva tabla. En otras palabras, la tabla original ahora está dividida en tres tablas (PROJECT, EMPLOYEE y ASSIGNMENT).

Paso 2: reasignar atributos dependientes correspondientes

Use la figura 6.3 para determinar los atributos que sean dependientes en las dependencias parciales. Las dependencias para los componentes de la llave original se encuentran si se examinan las flechas que apuntan hacia abajo en el diagrama de dependencia que se ve en la figura 6.3. Los atributos que son dependientes en una dependencia parcial se remueven de la tabla original y se colocan en la nueva tabla con su determinante. Cualesquier atributos que no sean dependientes en una dependencia parcial permanecerán en la tabla original. En otras palabras, a las tres tablas que resulten de la conversión a 2NF se les dan nombres apropiados (PROJECT, EMPLOYEE y ASSIGNMENT) y son descritas por los siguientes esquemas relacionales:

PROJECT (**PROJ_NUM,** PROJ_NAME)

EMPLOYEE (**EMP_NUM,** EMP_NAME, JOB_CLASS, CHG_HOUR)

ASSIGNMENT (**PROJ_NUM**, **EMP_NUM**, ASSIGN_HOURS)

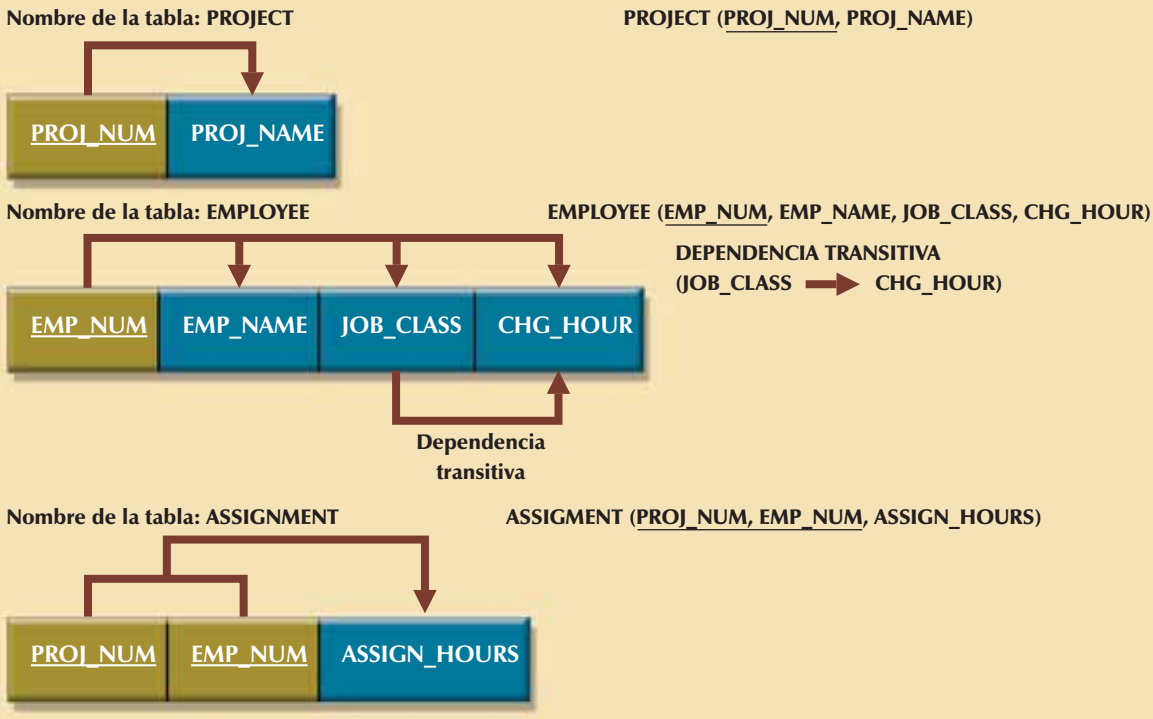
Como el número de horas empleadas en cada proyecto por cada trabajador es dependiente de PROJ_NUM y de EMP_NUM en la tabla ASSIGNMENT, se dejan esas horas en la tabla ASSIGNMENT como ASSIGN_HOURS. Nótese que la tabla ASSIGNMENT contiene una llave primaria compuesta formada por los atributos PROJ_NUM y EMP_NUM. Observe que dejando los determinantes de la tabla original, así como convirtiéndolos en las llaves primarias de las nuevas tablas, se han creado las relaciones de llave primaria/(llave foránea). Por ejemplo, en la tabla EMPLOYEE, EMP_NUM es la llave primaria. En la tabla ASSIGNMENT, EMP_NUM es parte de la llave primaria compuesta (PROJ_NUM, EMP_NUM) y es una llave foránea que relaciona la tabla EMPLOYEE con la tabla ASSIGNMENT.

Los resultados de los pasos 1 y 2 se muestran en la figura 6.4. En este punto, casi todas las anomalías estudiadas antes se han eliminado. Por ejemplo, si se desea agregar, cambiar o eliminar un registro PROJECT, sólo es necesario ir a la tabla PROJECT y hacer el cambio a sólo un renglón.

Debido a que puede existir una dependencia parcial sólo cuando la llave primaria de una tabla está compuesta de varios atributos, una tabla cuya llave primaria está formada por un solo atributo está automáticamente en 2NF una vez que esté en 1NF.

La figura 6.4 todavía muestra una dependencia transitiva, que puede generar anomalías. Por ejemplo, si el cargo por hora cambia para una clasificación de trabajo obtenida por muchos empleados, ese cambio debe ser hecho *por cada*

FIGURA 6.4 **Resultados de la conversión a segunda forma normales (2NF)**



uno de ellos. Si usted olvida actualizar algunos de los registros de empleado que son afectados por el cambio del cargo por hora, diferentes empleados con la misma descripción de trabajo van a generar distintos cargos por hora.

NOTA

Una tabla está en **segunda forma normal (2NF)** cuando:

- Está en **1F.**

y también

- No incluye dependencias parciales; esto es, ningún atributo es dependiente de sólo una parte de la llave primaria.

Nótese que todavía es posible que una tabla en 2NF exhiba dependencia transitiva; esto es, la llave primaria puede apoyarse en uno o más atributos no primos para determinar funcionalmente otros atributos no primos, como está indicado por una dependencia funcional entre los atributos no primos.

6.3.3 CONVERSIÓN A LA TERCERA FORMA NORMAL

Las anomalías de datos creadas por la organización de base de datos que se ven en la figura 6.4 se eliminan fácilmente al completar los dos pasos siguientes:

Paso 1: hacer nuevas tablas para eliminar dependencias transitivas

Por cada dependencia transitiva, escriba una copia de su determinante como llave primaria para una nueva tabla. Un **determinante** es cualquier atributo cuyo valor determina otros valores dentro de un renglón. Si tenemos tres dependencias transitivas diferentes, tendremos tres determinantes. Al igual que con la conversión a 2NF, es importante que el determinante permanezca en la tabla original para servir como llave foránea. La figura 6.4 muestra sólo una tabla que contiene una dependencia transitiva.

CAPÍTULO 12

SISTEMAS PARA ADMINISTRACIÓN DE BASES DE DATOS DISTRIBUIDAS

En este capítulo, el lector aprenderá:

- Qué es un sistema de administración de bases de datos distribuidas (DDBMS) y cuáles son sus componentes
- Cómo resulta afectada la implementación de una base de datos por diferentes niveles de datos y distribución de proceso
- Cómo se administran las transacciones en un ambiente de bases de datos distribuidas
- La forma en que el diseño de una base de datos es afectado por el ambiente de bases de datos distribuidas

Vista Previa

En este capítulo aprenderá que una base de datos se puede dividir en varios fragmentos. Éstos se pueden guardar en computadoras diferentes dentro de una red. El procesamiento también se puede dispersar entre varios sitios o nodos de la red. La base de datos de múltiples sitios forma el núcleo del sistema de bases de datos distribuidas.

El crecimiento de sistemas de bases de datos distribuidas ha sido alentado por la dispersión de operaciones de negocios por todo el mundo, junto con el rápido cambio tecnológico que ha hecho que las redes locales y amplias sean prácticas y más confiables. El sistema de base de datos distribuida que se basa en la red es muy flexible: puede servir a las necesidades de un pequeño negocio que opere sólo dos tiendas en la misma población, mientras que al mismo tiempo satisface las necesidades de empresas que operan en todo el mundo.

Aunque un sistema de base de datos distribuida requiere de un DBMS más refinado, el usuario final no debe tener más carga por la mayor complejidad operacional. Esto es, la mayor complejidad de un sistema de base de datos distribuida debe ser transparente para el usuario final.

El sistema de administración de base de datos distribuida (DDBMS) trata a una base de datos distribuida como una sola base de datos lógica; por tanto, se le aplican los conceptos básicos de diseño que usted aprendió en capítulos previos. Aunque el usuario final no necesite estar consciente de las características especiales de la base de datos distribuida, la distribución de datos entre diferentes sitios en una red claramente se agrega a la complejidad del sistema. Por ejemplo, el diseño de una base de datos distribuida debe considerar la ubicación de los datos y su división en fragmentos de la base de datos. Examinaremos estos problemas en este capítulo.

12.1 LA EVOLUCIÓN DE LOS SISTEMAS DE ADMINISTRACIÓN DE UNA BASE DE DATOS DISTRIBUIDA

Un **sistema de administración de una base de datos distribuida** (**DDBMS**) gobierna el almacenamiento y procesamiento de datos lógicamente relacionados en sistemas de cómputo interconectados en los que los datos y el procesamiento están distribuidos en varios sitios. Para entender cómo y por qué el DDBMS es diferente del DBMS, es útil examinar brevemente los cambios en el ambiente de negocios que prepararon el escenario para el desarrollo del DDBMS.

Durante la década de 1970, algunas corporaciones implementaron sistemas de administración de bases de datos centralizadas para satisfacer sus necesidades estructuradas de información. La información estructurada por lo general se presenta como reportes formales emitidos regularmente en formato estándar. Esa información, generada por lenguajes de programación procedimentales, es creada por especialistas en responder a peticiones canalizadas de manera precisa. Así, las necesidades de información estructurada están bien proporcionadas por sistemas centralizados.

El uso de una base de datos centralizada requería que los datos corporativos se guardaran en un solo sitio central, por lo general un mainframe. El acceso a datos se obtenía mediante terminales "tontas". El método centralizado, que se ilustra en la figura 12.1, funcionaba bien para llenar las necesidades de información estructurada de las corporaciones, pero ya no tenía capacidad cuando eventos que ocurrían con gran celeridad requerían tiempos de respuesta más rápidos y un acceso igualmente rápido a la información. El lento avance de las solicitudes de información, para la aprobación al especialista y al usuario, simplemente no servía a quienes tomaban decisiones y estaban en un ambiente muy dinámico. Lo que se necesitaba era un acceso rápido y no estructurado a bases de datos, usando consultas *ad hoc* para generar información en el acto.

FIGURA 12.1 Sistema de administración de bases de datos centralizadas

Los sistemas de administración de bases de datos fundamentados en el modelo relacional podrían proporcionar el ambiente en el que se satisficieran las necesidades de información no estructurada empleando consultas *ad hoc*. A los usuarios finales se les daría la capacidad de tener acceso a los datos cuando fuera necesario. Desafortunadamente, las primeras implementaciones del modelo relacional no dieron un rendimiento efectivo total cuando se comparaban con los bien establecidos modelos de bases de datos jerárquicos o de red.

Las últimas dos décadas vieron nacer una serie de decisivos cambios sociales y tecnológicos que afectaron el desarrollo y diseño de las bases de datos. Entre esos cambios se hallan:

- Las operaciones de negocios se descentralizaron.
- La competencia aumentó a nivel mundial.
- Las demandas de clientes y necesidades del mercado favorecieron un estilo de administración descentralizada.
- El rápido cambio tecnológico creó computadoras de bajo costo con potencia semejante a la de un mainframe, al igual que equipos inalámbricos portátiles que manejan impresionantes funciones múltiples con telefonía celular y servicios de datos, así como redes cada vez más complejas y rápidas para conectarlos. En consecuencia, las corporaciones han adoptado cada día más tecnologías de red como la plataforma para sus soluciones computarizadas.

- El gran número de aplicaciones basadas en los DBMS y la necesidad de proteger las inversiones en software de DBMS centralizados hicieron atractiva la noción de compartir datos. Los diversos reinos de los datos convergen hoy en el mundo digital cada vez más. En consecuencia, aplicaciones individuales manejan múltiples tipos de datos (voz, video, música, imágenes, etc.) y a esos datos se tiene acceso desde numerosos lugares geográficamente dispersos.

Esos factores crearon un dinámico ambiente de negocios en el que las compañías tenían que responder rápidamente a presiones tecnológicas y de competitividad. Cuando las grandes unidades de negocios se reestructuraron para formar operaciones dispersas, menos complejas y de reacción rápida, se hicieron evidentes dos necesidades en bases de datos:

- *El rápido acceso a datos ad hoc* se hizo decisivo en el ambiente de rápida respuesta para tomar decisiones.
- *La descentralización de estructuras de administración*, basada en la descentralización de unidades de negocios, convirtió en una necesidad las bases de datos de acceso múltiple descentralizado y ubicadas en diversos lugares.

Durante años recientes, los factores que acabamos de describir se afianzaron todavía más. No obstante, la forma en que se abordaron estuvo fuertemente influida por:

- *La creciente aceptación de internet como plataforma para acceso y distribución de datos.* La World Wide Web (la web) es, en efecto, el *depósito* de los datos distribuidos.
- *La revolución inalámbrica.* El uso generalizado de equipos digitales inalámbricos, por ejemplo, teléfonos inteligentes como el iPhone y BlackBerry y las agendas electrónicas personales (PDA), ha creado una alta demanda de acceso a los datos. Estos equipos tienen acceso a los datos desde lugares geográficamente dispersos y requieren variados intercambios de datos en gran cantidad de formatos (datos, voz, video, música, imágenes, etc.). Aunque el acceso a los datos distribuidos no necesariamente implica bases de datos distribuidas, es frecuente que el rendimiento y los requisitos de tolerancia a fallas hagan uso de técnicas de copia de datos similares a las que encontramos en bases de datos distribuidas.
- *El acelerado crecimiento de compañías que suministran tipos de servicios como "aplicaciones como un servicio".* Este nuevo tipo de servicio proporciona servicios de aplicación remota a compañías que desean subcontratar su desarrollo de aplicación, mantenimiento y operaciones. Los datos de la compañía por lo general se guardan en servidores centrales y no están necesariamente distribuidos. Al igual que con el acceso inalámbrico a los datos, este tipo de servicio puede no requerir una funcionalidad de datos totalmente distribuida; sin embargo, otros factores como el rendimiento y la tolerancia a fallas a veces requieren el uso de técnicas de copia de datos semejante a las que encontramos en bases de datos distribuidas.
- *El mayor interés en el análisis de datos que llevan a minería y almacenamiento de éstos.* Aunque un almacén de datos no suele ser una base de datos distribuida, se apoya en técnicas como la copia de datos y consultas distribuidas que facilitan la extracción e integración de datos. (El diseño, implementación y uso de un almacén de datos se explican en el capítulo 13.)

CONTENIDO EN LÍNEA

Para saber más acerca del impacto de internet en el acceso y distribución de datos, vea el **apéndice I**, publicado en el sitio web Premium para este libro.

En este punto, todavía no está claro el impacto a largo plazo de internet y de la revolución inalámbrica en el diseño y administración de bases de datos *distribuidas*. Quizás el éxito de internet y las tecnologías inalámbricas aliente el uso de bases de datos distribuidas, a medida que el ancho de banda se hace un cuello de botella más problemático. Quizá la resolución de problemas de ancho de banda simplemente confirmará el estándar de bases de datos centralizadas. En cualquier caso, hoy existen bases de datos distribuidas y es probable que muchos conceptos y componentes de operación encuentren un lugar en futuros desarrollos de bases de datos.

La base de datos descentralizada es especialmente deseable porque la administración de una centralizada está sujeta a problemas como:

- *Degradación del rendimiento* debido a un creciente número de lugares remotos en grandes distancias.
- *Altos costos* asociados con mantener y operar sistemas de bases de datos de mainframes.

- *Problemas de confiabilidad* creados por dependencia de un sitio central (un solo punto de síndrome de falla) y la necesidad de copia de datos.
- *Problemas de escalabilidad* asociados con los límites físicos impuestos por un solo factor (potencia, acondicionamiento de temperatura y consumo de potencia.)
- *Rigidez organizacional* impuesta por la base de datos, que podría no soportar la flexibilidad y agilidad requeridas por organizaciones mundiales modernas.

El dinámico ambiente de negocios y los defectos de una base de datos centralizada generaron una demanda para aplicaciones basada en tener acceso a datos desde diferentes fuentes en múltiples lugares. Ese ambiente de base de datos de múltiples fuentes/múltiples lugares es manejado mejor por un sistema de administración de base de datos distribuida (DDBMS).

12.2 VENTAJAS Y DESVENTAJAS DE UN DDBMS

Los sistemas de administración de bases de datos distribuidas presentan varias ventajas sobre los sistemas tradicionales; al mismo tiempo, están sujetos a algunos problemas. La tabla 12.1 resume las ventajas y desventajas asociadas con un DDBMS.

TABLA 12.1 Ventajas y desventajas de un DBMS distribuido

VENTAJAS	DESVENTAJAS
• *Los datos están ubicados cerca del sitio de mayor demanda.* Los datos en un sistema de base de datos distribuida están dispersos para satisfacer necesidades de negocios. • *Más rápido acceso a datos.* Es frecuente que usuarios finales trabajen con sólo un subconjunto localmente almacenado de datos de la compañía. • *Más rápido procesamiento de datos.* Un sistema de base de datos distribuida extiende la carga de trabajo de sistemas al procesar datos en varios sitios. • *Facilita crecimiento.* Se pueden agregar nuevos sitios a la red sin afectar las operaciones de otros sitios. • *Mejores comunicaciones.* Como los sitios locales son más pequeños y cercanos, los clientes alientan una mejor comunicación entre departamentos y entre clientes y personal de la compañía. • *Reducidos costos de operación.* Es más eficiente en costo agregar estaciones de trabajo a una red que actualizar un sistema de mainframes. El trabajo de desarrollo se hace a menos costo y más rapidez en PC de bajo costo que en mainframes. • *Interfaz de uso fácil.* Las PC y estaciones de trabajo suelen estar equipadas con una interfaz gráfica de uso fácil (GUI). La GUI simplifica la capacitación y uso para usuarios finales. • *Menor riesgo de falla en un solo punto.* Cuando falla una de las computadoras, la carga de trabajo es recogida por otras estaciones de trabajo. Los datos también son distribuidos en múltiples sitios. • *Independencia del procesador.* El usuario final puede tener acceso a cualquier copia disponible de los datos y la solicitud de un usuario final es procesada por cualquier procesador en la ubicación de los datos.	• *Complejidad de administración y control.* Las aplicaciones deben reconocer la ubicación de los datos y unir éstos desde varios lugares. Los administradores de bases de datos deben tener capacidad para coordinar actividades de las bases de datos a fin de evitar degradación de éstas debido a anomalías en datos. • *Dificultad tecnológica.* Integridad de datos, administración de transacciones, control de concurrencia, seguridad, respaldo, recuperación, optimización de consulta, selección de vía de acceso, etc., deben abordarse y resolverse. • *Seguridad.* La probabilidad de lapsos de seguridad aumenta cuando los datos están ubicados en varios sitios. La responsabilidad de la administración de datos será compartida por diferentes personas en varios lugares. • *Falta de normas.* No hay protocolos estándar de comunicación al nivel de la base de datos. (Aunque el TCP/IP es de hecho la norma al nivel de la red, no hay norma al nivel de aplicación.) Por ejemplo, diferentes vendedores de bases de datos emplean técnicas distintas (a veces incompatibles) para administrar la distribución y procesamiento de datos en un ambiente de DDBMS. • *Más necesidades de almacenamiento e infraestructura.* Se requieren múltiples copias de datos en diferentes lugares, lo cual precisa más espacio para almacenamiento en disco. • *Más costo en capacitación.* Los costos de capacitación son por lo general más altos en un modelo distribuido de lo que serían en un modelo centralizado, a veces hasta el grado de neutralizar ahorros operacionales y de hardware. • *Costos.* Las bases de datos distribuidas requieren infraestructura duplicada para operar (lugar físico, ambiente, personal, software, licencias, etcétera).

Las bases de datos distribuidas se usan con éxito pero están lejos de tener toda la flexibilidad y potencia de las que teóricamente son capaces. El ambiente inherentemente complejo de los datos distribuidos aumenta la urgencia de contar con protocolos estándar que gobiernen la administración de transacciones, control de concurrencia, seguridad, respaldo, recuperación, optimización de consultas, selección de vía de acceso, etc. Entonces, estos problemas deben abordarse y resolverse antes que la tecnología de los DDBMS se generalice.

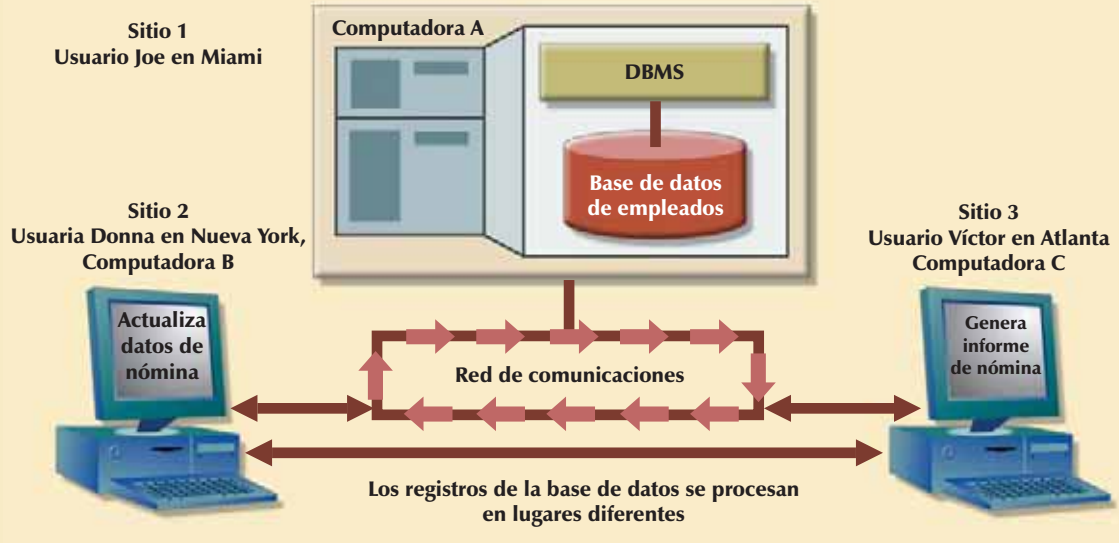
El resto de este capítulo explora los componentes y conceptos básicos de la base de datos distribuida. Debido a que ésta suele estar basada en el modelo relacional, se usa terminología relacional para explicar los conceptos y componentes básicos de una base de datos distribuida.

12.3 Procesamiento distribuido y bases de datos distribuidas

En **procesamiento distribuido**, el procesamiento lógico de una base de datos es compartido entre dos o más sitios físicamente independientes que estén conectados a través de una red. Por ejemplo, la entrada/salida (I/O), la selección y la validación de datos podría ser realizada en una computadora y el informe basado en esos datos podría ser creado en otra computadora.

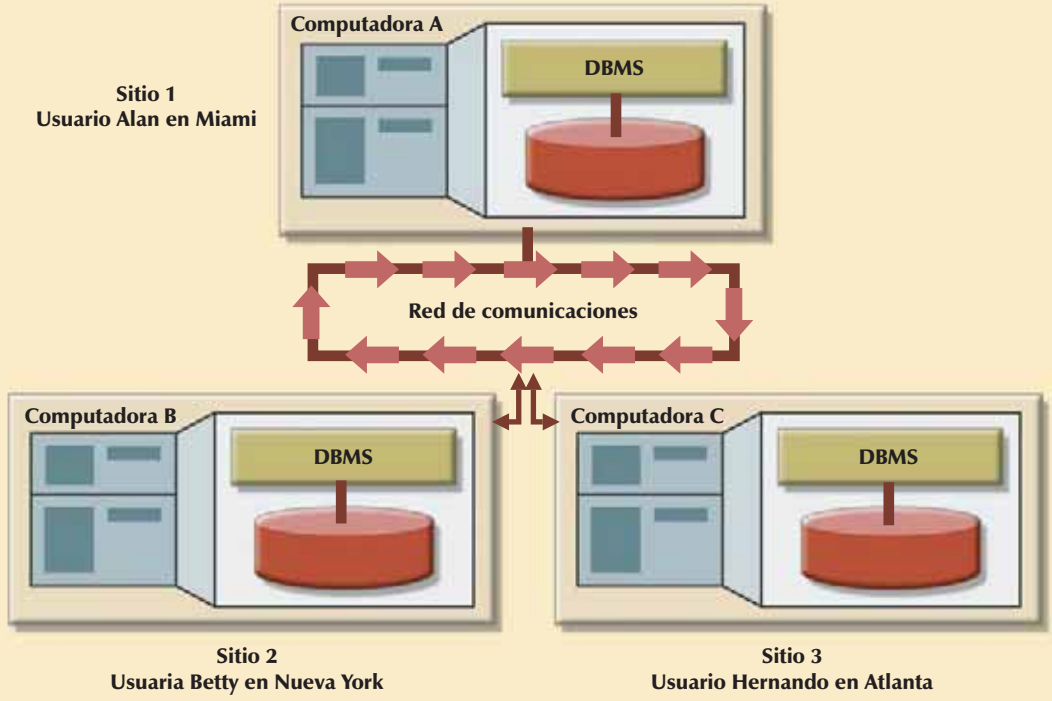
Un ambiente de procesamiento distribuido básico se ilustra en la figura 12.2, que muestra que un sistema de procesamiento distribuido comparte las tareas de procesamiento de una base de datos entre tres sitios conectados a través de una red de comunicaciones. Aunque la base de datos resida sólo en un sitio (Miami), cada uno de los sitios puede tener acceso a sus datos y actualizarla. La base de datos está ubicada en la computadora A, que es una computadora de red conocida como *servidor de base de datos*.

FIGURA 12.2 Ambiente de procesamiento distribuido



Una **base de datos distribuida**, por otra parte, guarda una base de datos relacionada lógicamente en dos o más sitios físicamente independientes. Los sitios están conectados mediante una red computarizada. En contraste, el sistema de procesamiento distribuido usa sólo una base de datos en un solo lugar, pero comparte las tareas de procesamiento entre varios sitios. En un sistema de base de datos distribuida, una base de datos está compuesta de varias partes conocidas como **fragmentos de base de datos**, los cuales están ubicados en sitios diferentes y pueden replicarse entre varios sitios. Cada fragmento de base de datos es, a su vez, administrado por su proceso local de base de datos. En la figura 12.3 se ilustra un ejemplo de un ambiente de base de datos distribuida.

FIGURA 12.3 **Ambiente de base de datos distribuida**



La base de datos de la figura 12.3 está dividida en tres fragmentos de base de datos (E1, E2 y E3) situados en lugares diferentes. Las computadoras están conectadas mediante un sistema de red. En una base de datos completamente distribuida, los usuarios Alan, Betty y Hernando no necesitan saber el nombre o ubicación de cada fragmento de la base de datos para tener acceso a esta última. También, los usuarios podrían estar ubicados en sitios que no sean Miami, Nueva York o Atlanta y aun así tener acceso a la base de datos como una sola unidad lógica.

Al examinar las figuras 12.2 y 12.3, deben recordarse los siguientes puntos:

- El procesamiento distribuido no requiere una base de datos distribuida, sino que una base de datos distribuida requiere procesamiento distribuido (cada fragmento de una base de datos es administrado por su propio proceso local de base de datos).
- El procesamiento distribuido puede estar basado en una sola base de datos ubicada en una sola computadora. Para que ocurra la administración de datos distribuidos, copias o partes de las funciones de procesamiento de la base de datos deben estar distribuidas a todos los sitios de almacenamiento de datos.
- Tanto el procesamiento distribuido como las bases de datos distribuidas requieren de una red para conectar todos los componentes.

12.4 CARACTERÍSTICAS DE LOS SISTEMAS DE ADMINISTRACIÓN DE BASES DE DATOS DISTRIBUIDAS

Un DDBMS gobierna el almacenamiento y procesamiento de datos lógicamente relacionados, mediante sistemas interconectados computarizados en los que los datos y funciones de procesamiento están distribuidos entre varios sitios. Un DBMS debe tener al menos las siguientes funciones para ser clasificado como distribuido:

- *Interfaz de aplicación* para interactuar con el usuario final, programas de aplicación y otros DBMS dentro de la base de datos distribuida.
- *Validación* para analizar solicitudes de datos para corrección de la sintaxis.
- *Transformación* para descomponer las solicitudes complejas en componentes atómicos de solicitud de datos.

- *Optimización de consulta* para encontrar la mejor estrategia de acceso. (¿A qué fragmentos de la base de datos debe tener acceso la consulta y cómo deben sincronizarse las actualizaciones de datos, si las hay?)
- *Mapeo* para determinar la ubicación de los datos de fragmentos locales y remotos.
- *Interfaz de I/O* para leer o escribir datos desde o hacia un almacenamiento local permanente.
- *Formateo* para preparar los datos para presentación al usuario final o a un programa de aplicación.
- *Seguridad* para dar privacidad a bases de datos locales y remotas.
- *Respaldo y recuperación* para asegurar la disponibilidad y capacidad de recuperación de la base de datos en caso de falla.
- *Funciones de administración de una base de datos* para el administrador de la base de datos.
- *Control de concurrencia* para administrar acceso simultáneo a datos y asegurar consistencia de datos en fragmentos de la base de datos del DDBMS.
- *Administración de transacción* para asegurar que los datos se muevan de un estado consistente a otro. Esta actividad incluye la sincronización de transacciones locales y remotas, así como transacciones por segmentos distribuidos múltiples.

Un sistema de administración de bases de datos totalmente distribuido debe ejecutar todas las funciones de un DBMS centralizado, como sigue:

1. Recibir la solicitud de una aplicación (o de un usuario final).
2. Validar, analizar y descomponer la solicitud, la cual debe incluir operaciones matemáticas o lógicas como las siguientes: seleccionar todos los clientes con un saldo mayor a $1000. La solicitud podría requerir datos de una o varias tablas.
3. Mapear (transferir) los componentes de datos lógicos a físicos de una solicitud.
4. Descomponer la solicitud en varias operaciones de I/O de disco.
5. Buscar, localizar, leer y validar los datos.
6. Asegurar consistencia, seguridad e integridad de la base de datos.
7. Validar los datos para las condiciones, si las hay, especificadas por la solicitud.
8. Presentar los datos seleccionados en el formato requerido.

Además, un DBMS distribuido debe manejar todas las funciones necesarias impuestas por la distribución y el procesamiento de los datos y debe ejecutar esas funciones adicionales *en forma transparente* al usuario final. Las características de acceso transparente a datos del DDBMS se ilustran en la figura 12.4.

La base de datos lógica individual de la figura 12.4 está formada por dos fragmentos, A1 y A2, ubicados en los sitios 1 y 2, respectivamente. Mary puede consultar la base de datos como si fuera una base de datos local; también Tom puede hacerlo. Ambos usuarios "ven" sólo una base de datos lógica y *no necesitan saber los nombres de los fragmentos.* De hecho, los usuarios finales ni siquiera necesitan saber que la base de datos está dividida en fragmentos, *ni necesitan saber dónde están ubicados éstos.*

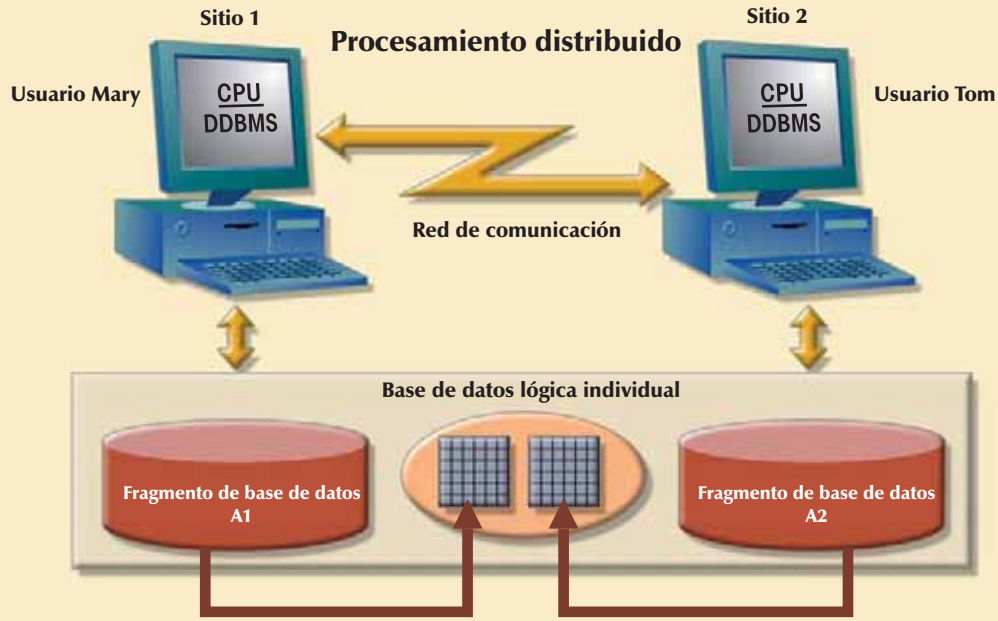
Para entender mejor los diferentes tipos de situaciones de una base de datos distribuida, definamos primero los componentes del sistema de administración de base de datos distribuida.

12.5 COMPONENTES DE UN DDBMS

El DDBMS debe incluir al menos los siguientes componentes:

- *Estaciones de trabajo o equipos remotos* (sitios o nodos) que forman el sistema de red. El sistema de base de datos distribuida debe ser independiente del hardware del sistema computarizado.
- *Componentes del hardware y software de la red* que residen en cada estación de trabajo o equipo. Los componentes de la red posibilitan que todos los sitios interactúen e intercambien datos. Debido a que es probable que los componentes (computadoras, sistemas operativos, hardware de red, etc.) sean suministrados por diferentes vendedores, es mejor asegurar que las funciones de una base de datos distribuida puedan ser ejecutadas en plataformas distintas.

FIGURA 12.4 **Sistema de administración de base de datos totalmente distribuido**



- *Medios de comunicación* que lleven los datos de un nodo a otro. El DDBMS debe ser independiente de medios de comunicación; esto es, debe ser capaz de soportar varios tipos de medios.
- El **procesador de transacciones (TP)**, que es el componente de software que se encuentra en cada computadora o equipo que pide datos. El procesador de transacciones recibe y procesa las solicitudes de datos de la aplicación (remota y local). El TP también se conoce como **procesador de aplicación** (**AP**) o **administrador de transacciones** (**TM)**
- El **procesador de datos** (**DP**), que es el componente de software que reside en cada computadora o equipo que guarda y recupera datos localizados en el sitio. El DP también se conoce como **administrador de datos** (**DM**). Un procesador de datos puede incluso ser un DBMS centralizado.

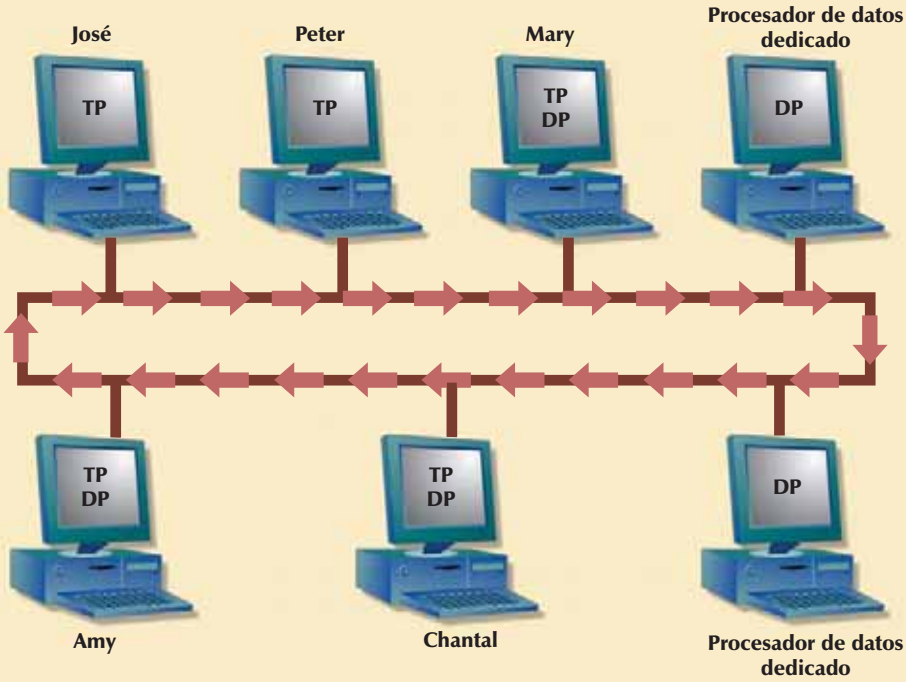
La figura 12.5 ilustra la colocación de los componentes y la interacción entre ellos. La comunicación entre los TP y los DP que se ven en la figura 12.5 es posible mediante un conjunto de reglas específico, o *protocolos*, usados por el DDBMS.

Los protocolos determinan la forma en que el sistema de base de datos distribuido:

- Se entrelaza con la red para transportar datos y comandos entre procesadores de datos (DP) y procesadores de transacciones (TP).
- Sincroniza todos los datos recibidos de los DP (lado de TP) y enruta los recuperados a los TP apropiados (lado de DP).
- Asegura funciones de base de datos comunes en un sistema distribuido. Tales funciones incluyen seguridad, control de concurrencia, respaldo y recuperación.

Los DP y los TP se pueden agregar al sistema sin afectar la operación de los otros componentes. Un TP y un DP pueden residir en la misma computadora, permitiendo al usuario final tener acceso transparente a datos locales y remotos. En teoría, un DP puede ser un DBMS centralizado independiente con interfaces propias para soportar acceso remoto desde otros DBMS independientes en la red.

FIGURA 12.5 **Componentes para administración de un sistema de base de datos distribuida**



***Nota*: cada uno de los TP puede tener acceso a datos en cualquier DP y cada DP maneja todas las solicitudes de datos locales desde cualquier TP.**

12.6 NIVELES DE DATOS Y DISTRIBUCIÓN DE PROCESOS

Los sistemas actuales de bases de datos pueden ser clasificados a partir de cómo soportan la distribución de procesos y la de datos. Por ejemplo, un DBMS puede almacenar datos en un solo sitio (DB centralizada) o en múltiples sitios (DB distribuida) y puede soportar procesamiento de datos en un solo sitio o en múltiples sitios. La tabla 12.2 usa una matriz sencilla para clasificar sistemas de bases de datos de acuerdo con la distribución de datos y de procesos. Estos tipos de procesos se estudian en las secciones que siguen.

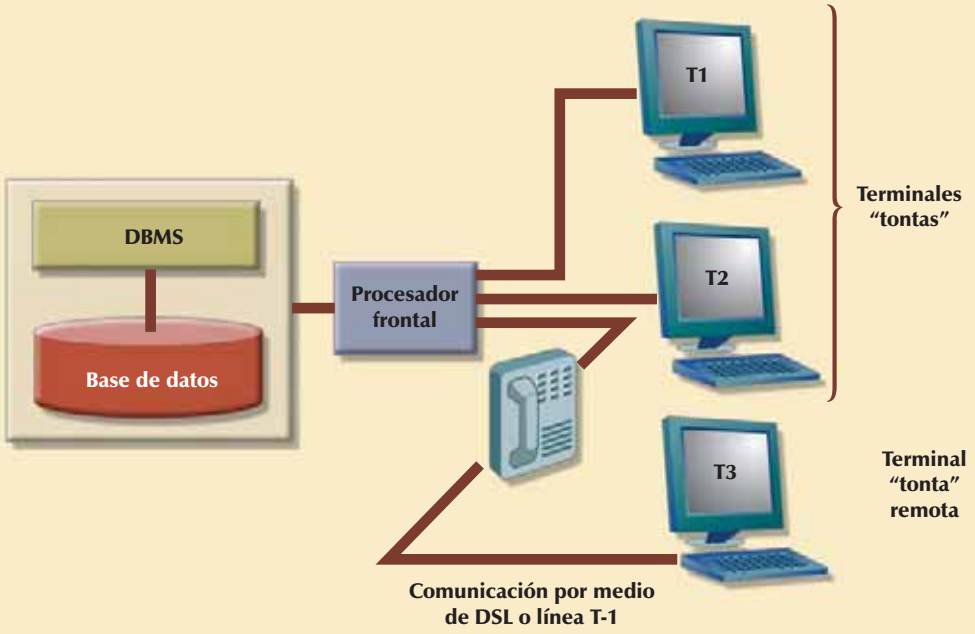
TABLA 12.2 **Sistemas de bases de datos: niveles de distribución de datos y de procesos**

	PROCESO EN UN SITIO	PROCESO EN SITIOS MÚLTIPLES
Datos de un solo sitio	DBMS anfitrión	No aplicable (Requiere procesos múltiples)
Datos de sitios múltiples	Servidor de archivo DBMS de cliente/servidor (LAN DBMS)	Totalmente distribuido DBMS cliente/servidor

12.6.1 PROCESAMIENTO DE UN SOLO SITIO, DATOS DE UN SOLO SITIO (SPSD)

En una situación de **procesamiento de un solo sitio, datos de un solo sitio** (**SPSD**), todo el procesamiento se hace en una sola computadora anfitrión (servidor de un solo procesador, servidor de multiprocesador, sistema de mainframe) y todos los datos se guardan en el sistema local de discos del computador anfitrión. El procesamiento no puede ser realizado en el lado del usuario del sistema. Esta situación es típica de casi todos los DBMS de mainframe y servidores de mediana capacidad. El DBMS se encuentra en el computador anfitrión, al que se tiene acceso por terminales "tontas" conectadas al mismo (figura 12.6). Esta situación también es típica de la primera generación de bases de datos de microcomputadoras de un solo usuario.

FIGURA 12.6 **Procesamiento de un solo sitio, datos de un solo sitio (centralizados)**



Usando la figura 12.6 como ejemplo, se puede ver que las funciones del TP y el DP están insertadas dentro del DBMS ubicado en una sola computadora. El DBMS por lo general corre bajo un sistema operativo de tiempo compartido y multitareas, que permite que varios procesos se ejecuten de manera concurrente en una computadora anfitrión multitarea que tiene acceso a un solo DP. Todo el almacenamiento y procesamiento de datos es manejado por una sola computadora anfitrión.

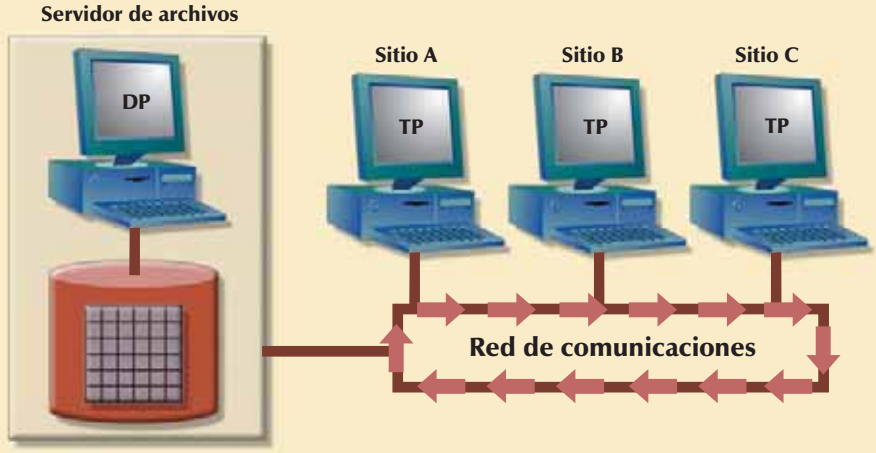
12.6.2 PROCESAMIENTO EN MÚLTIPLES SITIOS, DATOS EN UN SITIO (MPSD)

En una situación de un **procesamiento en múltiples sitios, datos en un solo sitio** (**MPSD**), varios procesos se ejecutan en computadoras diferentes que comparten un solo depósito de datos. Por lo general, la situación de MPSD requiere de un servidor de archivos de red que ejecute aplicaciones convencionales a las que se tiene acceso mediante una red. Numerosas aplicaciones de contabilidad multiusuarios que se ejecutan bajo una red de computadoras personales se ajustan a esta descripción (figura 12.7).

Al examinar la figura 12.7, observe que:

- El TP de cada estación de trabajo actúa sólo como redirector para enrutar todas las solicitudes de datos de la red al servidor de archivos.
- El usuario final ve al servidor de archivos sólo como otro disco duro. Debido a que sólo el almacenamiento de datos de entrada/salida (I/O) es manejado por la computadora del servidor de archivos, el MPSD ofrece capacidad limitada para procesamiento distribuido.
- El usuario final debe hacer una referencia directa al servidor de archivos para tener acceso a datos remotos. Todas las actividades de registro y bloqueo de archivos son realizadas en el lugar del usuario final.
- Todas las funciones de selección, búsqueda y actualización de datos tienen lugar en la estación de trabajo, requiriendo así que todos los archivos viajen por la red para ser procesada por ella. Este requisito aumenta el tráfico de red, reduce el tiempo de respuesta y aumenta costos de comunicaciones.

FIGURA 12.7 Procesamiento en sitios múltiples, datos en un sitio



La ineficiencia de la última condición se puede ilustrar fácilmente. Por ejemplo, suponga que la computadora del servidor de archivos guarda una tabla CUSTOMER que contiene 10 000 renglones de datos, 50 de los cuales tienen saldos mayores a $1 000. Suponga que el sitio A emite la siguiente consulta de SQL:

```
SELECT      *
FROM        CUSTOMER
WHERE       CUS_BALANCE > 1000;
```

Los 10 000 renglones de CUSTOMER deben viajar por la red para ser evaluados en el sitio A. Una variación del método de procesamiento de multiples sitios y datos de un solo sitio se conoce como arquitectura cliente/servidor. La **arquitectura cliente/servidor** es semejante a la del servidor de archivos de red *excepto que todo el procesamiento de la base de datos se realiza en el sitio del servidor, reduciendo así el tráfico de red.* Aunque el servidor de archivos de red y los sistemas cliente/servidor ejecutan un procesamiento de sitios múltiples, el procesamiento del último es distribuido. Nótese que el método de servidor de archivos de red requiere que la base de datos se encuentre ubicada en un solo sitio. En contraste, la arquitectura cliente/servidor es capaz de soportar datos en sitios múltiples.

CONTENIDO EN LÍNEA

El **apéndice F**, se encuentra en el sitio web Premium para este libro.

12.6.3 PROCESAMIENTO EN SITIOS MÚLTIPLES, DATOS EN SITIOS MÚLTIPLES (MPMD)

La situación de **procesamiento en sitios múltiples, datos en sitios múltiples** (**MPMD**) describe un DBMS totalmente distribuido con soporte para procesadores de datos múltiples y procesadores de transacciones en sitios múltiples. Dependiendo del nivel de soporte para diversos tipos de DBMS centralizados, los DDBMS se clasifican como homogéneos o heterogéneos.

Los **DDBMS homogéneos** integran sólo un tipo de DBMS centralizado en una red. Así, el mismo DBMS estará ejecutándose en diferentes plataformas de servidor (servidor de un procesador, servidor multiprocesador, granjas de servidores u hojas de servidor). En contraste, los **DDBMS heterogéneos** integran diferentes tipos de DBMS centralizados en una red. La tabla 12.3 es una lista de varios sistemas que podrían estar integrados dentro de un solo DDBMS heterogéneo en una red. Un **DDBMS totalmente heterogéneo** soportará diferentes DBMS que puedan incluso dar soporte a diferentes modelos de datos (relacionales, jerárquicos o de red) que se ejecutan bajo diferentes sistemas computarizados; por ejemplo, mainframes y computadoras personales (PC).

TABLA 12.3 **Situación de una base de datos distribuida heterogénea**

PLATAFORMA	DBMS	SISTEMA OPERATIVO	PROTOCOLO DE COMUNICACIONES DE RED
IBM 3090	DB2	MVS	APPC LU 6.2
DEC/VAX	VAX rdb	OpenVMS	DECnet
IBM AS/400	SQL/400	OS/400	3270
Computadora RISC	Informix	UNIX	TCP/IP
CPU Pentium	Oracle	Windows Server 2008	TCP/IP

Algunas implementaciones de DDBMS soportan a varias plataformas, sistemas operativos y redes, permitiendo acceso remoto a datos a otro DBMS. No obstante, esos DDBMS son todavía objeto de ciertas restricciones. Por ejemplo:

- El acceso remoto es proporcionado para sólo lectura y no da soporte a privilegios de escritura.
- Se ponen restricciones en el número de tablas remotas a las que se tenga acceso en una sola transacción.
- Se ponen restricciones al número de bases de datos a las que podría accederse.
- Se ponen restricciones en el modelo de base de datos al que se tenga acceso. Así, el acceso puede ser proporcionado a bases de datos relacionales pero no a bases de datos de red o jerárquicas.

La lista de restricciones precedente de ninguna manera es exhaustiva. La tecnología de los DDBMS continúa cambiando con gran rapidez y con frecuencia se agregan nuevas funciones. El manejo de datos en múltiples sitios lleva a varios problemas que deben abordarse y entenderse. La siguiente sección examina varias características clave de sistemas de administración de bases de datos distribuidas.

12.7 CARACTERÍSTICAS DE TRANSPARENCIA DE LAS BASES DE DATOS DISTRIBUIDAS

Un sistema de base de datos distribuida requiere características funcionales que puedan agruparse y describirse como características de transparencia, que tienen la propiedad común de permitir que el usuario final se sienta como el único usuario de la base de datos. En otras palabras, el usuario piensa que está trabajando con un DBMS centralizado; todas las complejidades de una base de datos distribuida están ocultas, o transparentes, al usuario.

Las características de transparencia del DDBMS son:

- **Transparencia de distribución**, que permite que una base de datos distribuida sea tratada como una sola base de datos lógica. Si un DDBMS exhibe transparencia de distribución, el usuario no necesita saber:
  - Que los datos están divididos, lo que significa que renglones y columnas de la tabla están divididos vertical u horizontalmente y guardados en varios sitios.
  - Que los datos pueden replicarse en varios sitios.
  - La ubicación de los datos.
- **Transparencia de transacción**, que permite que una transacción actualice datos en más de un sitio de red. La transparencia de transacción asegura que la transacción será totalmente terminada o abortada, manteniendo así la integridad de la base de datos.
- **Transparencia de fallas**, que asegura que el sistema continuará operando en el caso de una falla de nodo. Las funciones que se perdieron por la falla serán captadas por otro nodo de red.
- **Transparencia de desempeño**, que permite al sistema funcionar como si fuera un DBMS centralizado. El sistema no sufrirá ninguna degradación de desempeño debido a su uso en una red o a diferencias en la plataforma de ésta. La transparencia de desempeño también asegura que el sistema encontrará la vía más eficiente en costo para tener acceso a datos remotos.

CORONEL / MORRIS / ROB

Bases de datos

Diseño, implementación y administración

Novena edición

Bases de datos: Diseño, implementación y administración, Novena edición, un líder del mercado para los textos de este tema, proporciona a los lectores fundamentos sólidos en la práctica de diseño e implementación de bases de datos. El libro ofrece cobertura detallada de diseño de bases de datos, demostrando que la clave para la implementación exitosa está en un diseño apropiado que tenga cabida dentro de una visión estratégica más grande del entorno de datos. Actualizado en el tratamiento de modelos de datos, mejora la cobertura de la normalización con una lista de comprobación de modelado de datos. Además cubre mejor el diseño de bases de datos y ciclo de vida; contiene nuevas preguntas de repaso, conjuntos de problemas y casos en todo el libro.

Con un fuerte componente práctico que incluye ejercicios y ejemplos del mundo real, este libro ayudará a los estudiantes a desarrollar habilidades de diseño de base de datos que tienen aplicación valiosa y significativa en el mundo real.

Características

- Impreso a todo color, con ilustraciones detalladas, tablas y diagramas para mejorar y facilitar la comprensión de los conceptos más complejos.
- Se utiliza una variedad de bases de datos en diversos formatos para los lectores con experiencia en la implementación utilizando las bases de datos de MS Access, MS SQL Server y Oracle.
- Las viñetas de negocios nuevos y actualizados ofrecen una visión sobre las cuestiones y desafíos que enfrenta la aplicación de bases de datos hoy en día.
- El estilo de escritura clara y directa proporciona un equilibrio de la teoría y la práctica.
- El manual del instructor incluye todas las funciones finales del capítulo, material de análisis sugerido y otros recursos, dando al profesor una herramienta de enseñanza completa.

ISBN-13: 978-607481618-1
ISBN-10: 607481618-2

CENGAGE Learning™

http://latinoamerica.cengage.com