M. Tamer Özsu
Patrick Valduriez

Principles of Distributed Database Systems

Third Edition

Springer

Principios de los sistemas de bases de datos distribuidas

M. Tamer Özsu • Patrick Valduriez

Principios de la distribución Sistemas de bases de datos

Tercera edición

Springer

Sr. Tamer Özsu
Escuela de David R. Cheriton
Ciencias de la Computación
Universidad de Waterloo
Waterloo, Ontario
Canadá N2L 3G1
Tamer.Ozsu@uwaterloo.ca

Patrick Valduriez
INRIA
LIRMM
161 rue Ada
34392 Montpellier Cedex
Francia
Patrick.Valduriez@inria.fr

Este libro fue publicado anteriormente por: Pearson Education, Inc.

ISBN 978-1-4419-8833-1 DOI 10.1007/978-1-4419-8834-8 Springer
ISBN electrónico 978-1-4419-8834-8
Nueva York Dordrecht Heidelberg Londres

Número de control de la Biblioteca del Congreso: 2011922491

Impreso en papel libre de ácido.

Springer es parte de Springer Science+Business Media (www.springer.com)

A mi familia y
mis padres MTO

A Esther, mis hijas Anna, Juliette y
Sarah y mis padres PV

Prefacio

Han pasado casi veinte años desde la publicación de la primera edición de este libro y diez desde la publicación de la segunda. Como es de suponer, en un ámbito tan cambiante como este, se han producido cambios significativos en este periodo. La gestión distribuida de datos pasó de ser una tecnología potencialmente significativa a una práctica común. La llegada de Internet y la World Wide Web sin duda ha cambiado la forma en que solemos considerar la distribución. La aparición en los últimos años de diferentes formas de computación distribuida, ejemplificadas por los flujos de datos y la computación en la nube, ha revitalizado el interés en la gestión distribuida de datos. Por lo tanto, era hora de una revisión a fondo del material.

Empezamos a trabajar en esta edición hace cinco años, y nos ha llevado bastante tiempo completarla. Sin embargo, el resultado final es un libro con una revisión exhaustiva: si bien mantuvimos y actualizamos los capítulos principales, también añadimos nuevos. Los principales cambios son los siguientes:

1. La integración y las consultas de bases de datos se abordan ahora con mucho más detalle, lo que refleja la atención que estos temas han recibido en la comunidad durante la última década. El capítulo 4 se centra en el proceso de integración, mientras que el capítulo 9 aborda las consultas en sistemas multibase de datos.
2. Las ediciones anteriores sólo tenían una breve discusión de los protocolos de replicación de datos. Este tema ahora se trata en un capítulo separado (Capítulo 13) donde ofrecemos un análisis en profundidad de los protocolos y cómo pueden integrarse con la gestión de transacciones.
3. La gestión de datos punto a punto se analiza en profundidad en el capítulo 16. Estos sistemas se han convertido en una alternativa arquitectónica importante e interesante a los sistemas de bases de datos distribuidas clásicos. Si bien las primeras arquitecturas de estos sistemas seguían el paradigma punto a punto, su versión moderna presenta características fundamentalmente diferentes, por lo que merecen un análisis a fondo en un capítulo aparte.
4. La gestión de datos web se aborda en el capítulo 17. Este es un tema complejo de abordar, ya que no existe un marco unificado. Se abordan diversos aspectos.

del tema que abarca desde modelos web hasta motores de búsqueda y procesamiento XML distribuido.

5. Las ediciones anteriores incluían un capítulo donde abordábamos los "problemas recientes" de aquel momento. En esta edición, contamos con un capítulo similar (Capítulo 18) donde abordamos la gestión de datos de flujo y la computación en la nube. Estos temas aún están en constante evolución y son objeto de considerable investigación en curso. Destacamos los problemas y las posibles líneas de investigación.

El manuscrito resultante logra un equilibrio entre nuestros dos objetivos, es decir, abordar cuestiones nuevas y emergentes, y mantener las características principales del libro al abordar los principios de la gestión de datos distribuidos.

La organización del libro se puede dividir en dos partes principales. La primera parte abarca los principios fundamentales de la gestión distribuida de datos y consta de los capítulos 1 a 14. El capítulo 2 , en esta parte, abarca los antecedentes y puede omitirse si los estudiantes ya poseen conocimientos suficientes sobre los conceptos de bases de datos relacionales y la tecnología de redes informáticas. La única parte esencial de este capítulo es el Ejemplo 2.3, que presenta el ejemplo de ejecución que utilizamos a lo largo de gran parte del libro. La segunda parte abarca temas más avanzados e incluye los capítulos 15 a 18. El contenido de un curso depende en gran medida de su duración y sus objetivos. Si el curso se centra en las técnicas fundamentales, podría abarcar los capítulos 1, 3, 5, 6-8 y 10-12. Una cobertura más extensa incluiría, además de lo anterior, los capítulos 4, 9 y 13. Los cursos que dispongan de tiempo para cubrir más material pueden seleccionar uno o más de los capítulos 15 a 18 de la segunda parte.

Muchos colegas han colaborado en esta edición del libro. S. Keshav (Universidad de Waterloo) leyó y proporcionó muchas sugerencias para actualizar las secciones sobre redes de computadoras. Renee Miller (Universidad de Toronto) y Erhard Rahm (Universidad de Leipzig) leyeron un borrador inicial del Capítulo 4 y proporcionaron muchos comentarios, Alon Halevy (Google) respondió a varias preguntas sobre este capítulo y proporcionó un borrador de su próximo libro sobre este tema, además de leer y proporcionar comentarios sobre el Capítulo 9, Avigdor Gal (Technion) también revisó y criticó este capítulo muy exhaustivamente. Matthias Jarke y Xiang Li (Universidad de Aachen), Gottfried Vossen (Universidad de Muenster), Erhard Rahm y Andreas Thor (Universidad de Leipzig) contribuyeron con ejercicios para este capítulo. Hubert Naacke (Universidad de París 6) contribuyó a la sección sobre modelos de costos heterogéneos y Fabio Porto (LNCC, Petrópolis) a la sección sobre procesamiento de consultas adaptativas del Capítulo 9. La replicación de datos (Capítulo 13) no podría haberse escrito sin la ayuda de Gustavo Alonso (ETH Zurich) y Bettina Kemm
Tamer pasó cuatro meses en la primavera de 2006 visitando a Gustavo, donde comenzó el trabajo en este capítulo , lo que implicó largas discusiones. Bettina leyó varias versiones de este capítulo durante el año siguiente, criticándolo todo y señalando mejores maneras de explicar el material. Esther Pacitti (Universidad de Montpellier) también contribuyó a este capítulo, revisándolo y proporcionando material de referencia; también contribuyó a la sección sobre replicación en clústeres de bases de datos del capítulo 14. Ricardo Jiménez-Peris también contribuyó a ese capítulo en la sección sobre tolerancia a fallos en clústeres de bases de datos. Khuzaima Daudjee (Universidad de Waterloo) leyó y proporcionó

Comentarios sobre este capítulo también. El capítulo 15, sobre Gestión de Bases de Datos de Objetos Distribuidos, fue revisado por Serge Abiteboul (INRIA), quien ofreció importantes críticas al material y sugerencias para su mejora. La gestión de datos entre pares (Capítulo 16) se debe en gran medida a las conversaciones con Beng Chin Ooi (Universidad Nacional de Singapur) durante los cuatro meses que Tamer visitó la NUS en otoño de 2006.

La sección del Capítulo 16 sobre procesamiento de consultas en sistemas P2P utiliza material de los trabajos de doctorado de Reza Akbarinia (INRIA) y Wenceslao Palma (PUC-Valparaíso, Chile), mientras que la sección sobre replicación utiliza material del trabajo de doctorado de Vidal Martins (PUCPR, Curitiba). La sección de procesamiento distribuido de XML del Capítulo 17 utiliza material de los trabajos de doctorado de Ning Zhang (Facebook) y Patrick Kling de la Universidad de Waterloo, y de Ying Zhang de CWI. Los tres también leyeron el material y proporcionaron comentarios significativos. Victor Muntes-i Mulero (Universitat Politécnica de Catalunya) contribuyó a los ejercicios de dicho capítulo. ur Ulusoy Ozg (Universidad de Bilkent) proporcionó comentarios y correcciones a los Capítulos 16 y 17. La sección de gestión del flujo de datos del Capítulo 18 se basa en el trabajo de doctorado de Lukasz Golab (AT&T Labs-Research) y Yingying Tao en la Universidad de Waterloo. Walid Aref (Universidad de Purdue) y Avigdor Gal (Technion) utilizaron el borrador del libro en sus cursos, lo cual les resultó muy útil para depurar ciertas partes. Les agradecemos, así como a muchos colegas que nos ayudaron con las dos primeras ediciones, toda su ayuda. No siempre hemos seguido sus consejos y, huelga decirlo, los problemas y errores resultantes son nuestros. Los estudiantes de dos cursos de la Universidad de Waterloo (Gestión de Datos Web en invierno de 2005 y Distribución de Datos a Escala de Internet en otoño de 2005) elaboraron encuestas como parte de sus trabajos académicos que resultaron muy útiles para estructurar algunos capítulos. Tamer impartió cursos en la ETH de Zúrich (PDDBS: Bases de Datos Paralelas y Distribuidas en primavera de 2006) y en la NUS (CS5225: Sistemas de Bases de Datos Paralelas y Distribuidas en otoño de 2010) utilizando partes de esta Agradecemos a los estudiantes de todos estos cursos por sus contribuciones y su paciencia mientras tenían que lidiar con capítulos que eran trabajos en progreso; el material se limpió considerablemente como resultado de estas experiencias de enseñanza.

Notarán que la editorial de la tercera edición del libro es diferente a la de las dos primeras. Pearson, nuestra editorial anterior, decidió no participar en la tercera edición. Posteriormente, Springer mostró un gran interés en el libro. Agradecemos a Susan Lagerstrom-Fife y Jennifer Evans, de Springer, por su rápida decisión de publicar el libro, y a Jennifer Mauer por su gran apoyo durante el proceso de conversión. También agradecemos a Tracy Dunkelberger, de Pearson, quien gestionó la reversión de los derechos de autor sin demora.

Al igual que en ediciones anteriores, contaremos con diapositivas de presentación que podrán utilizarse para la enseñanza a partir del libro, así como con soluciones para la mayoría de los ejercicios. Estas estarán disponibles en Springer para los instructores que adopten el libro y habrá un enlace a ellas desde el sitio web del libro: springer.com.

Finalmente, nos interesaría mucho conocer sus comentarios y sugerencias sobre el material. Agradecemos cualquier comentario, pero nos gustaría especialmente recibir comentarios sobre los siguientes aspectos:

1. cualquier error que pueda haber quedado a pesar de nuestros mejores esfuerzos (aunque esperamos no hay muchos);
2. cualquier tema que ya no debería incluirse y cualquier tema que debería incluirse añadido o ampliado; y
3. cualquier ejercicio que hayas diseñado y que te gustaría incluir en el libro.

Sr. Tamer Ozsu (Tamer.Ozsu@uwaterloo.ca)
Patrick Valduriez (Patrick.Valduriez@inria.fr)
Noviembre de 2010

Contenido

Capítulo 1
Introducción

La tecnología de sistemas de bases de datos distribuidas (DDBS) es la unión de dos enfoques aparentemente opuestos para el procesamiento de datos: los sistemas de bases de datos y las tecnologías de redes informáticas. Los sistemas de bases de datos nos han llevado de un paradigma de procesamiento de datos en el que cada aplicación definía y mantenía sus propios datos (Figura 1.1) a uno en el que estos se definen y administran de forma centralizada (Figura 1.2). Esta nueva orientación da como resultado la independencia de los datos, lo que permite que los programas de aplicación sean inmunes a los cambios en la organización lógica o física de los datos, y viceversa.

Una de las principales motivaciones tras el uso de sistemas de bases de datos es el deseo de integrar los datos operativos de una empresa y proporcionar un acceso centralizado y, por lo tanto, controlado a dichos datos. La tecnología de las redes informáticas, por otro lado, promueve un modo de trabajo que contradice cualquier intento de centralización. A primera vista, puede resultar difícil comprender cómo estos dos enfoques contrastantes pueden sintetizarse para producir una tecnología más potente y prometedora que cualquiera de ellos por separado. La clave para comprenderlo es comprender...

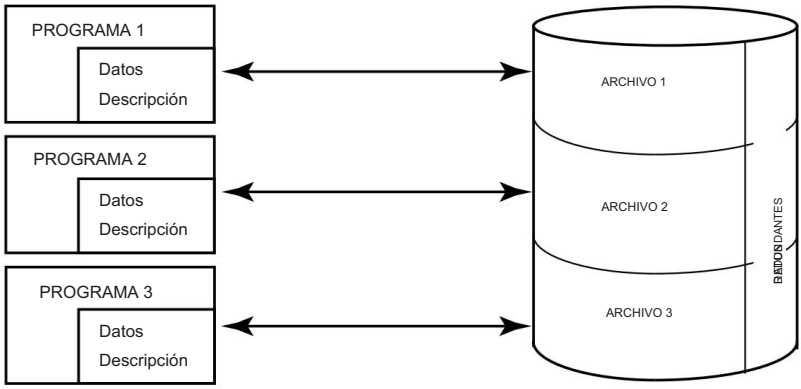

Fig. 1.1 Procesamiento tradicional de archivos

MT Özsu y P. Valduriez, Principios de sistemas de bases de datos distribuidas: Tercera edición,
DOI 10.1007/978-1-4419-8834-8_1, © Springer Science+Business Media, LLC 2011

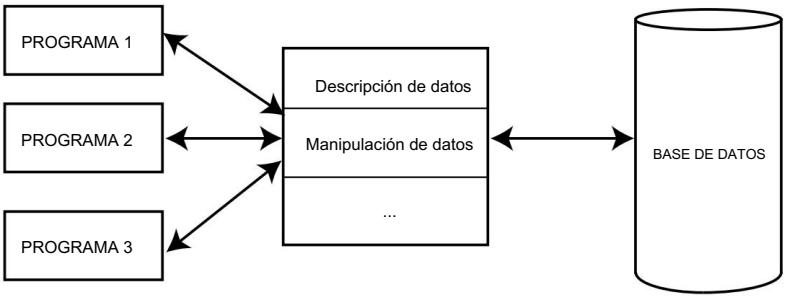

Fig. 1.2 Procesamiento de la base de datos

Que el objetivo más importante de la tecnología de bases de datos es la integración, no la centralización. Es importante comprender que uno de estos términos no implica necesariamente el otro. Es posible lograr la integración sin centralización, y eso es precisamente lo que la tecnología de bases de datos distribuidas intenta lograr.

En este capítulo, definimos los conceptos fundamentales y establecemos el marco para analizar las bases de datos distribuidas. Comenzamos examinando los sistemas distribuidos en general para aclarar el papel de la tecnología de bases de datos en el procesamiento distribuido de datos, y luego abordamos temas más directamente relacionados con los DDBS.

1.1 Procesamiento de datos distribuidos

El término procesamiento distribuido (o computación distribuida) es difícil de definir con precisión. Obviamente, existe cierto grado de procesamiento distribuido en cualquier sistema informático, incluso en ordenadores de un solo procesador, donde la unidad central de procesamiento (CPU) y las funciones de entrada/salida (E/S) están separadas y se superponen. Esta separación y superposición puede considerarse una forma de procesamiento distribuido. La aparición generalizada de ordenadores paralelos ha complicado aún más el panorama, ya que la distinción entre sistemas de computación distribuida y algunas formas de ordenadores paralelos es bastante vaga.

En este libro, definimos el procesamiento distribuido de tal manera que nos lleva a la definición de un sistema de bases de datos distribuidas. La definición práctica que utilizamos para un sistema de computación distribuida establece que se trata de un conjunto de elementos de procesamiento autónomos (no necesariamente homogéneos) interconectados por una red informática que cooperan en la realización de las tareas asignadas. El «elemento de procesamiento» al que se refiere esta definición es un dispositivo informático que puede ejecutar un programa por sí solo. Esta definición es similar a las que se ofrecen en los libros de texto sobre sistemas distribuidos (p. ej., [Tanenbaum y van Steen, 2002] y [Colouris et al., 2001]).

Una pregunta fundamental que debemos plantearnos es ¿Qué se está distribuyendo? Uno de los elementos que podrían distribuirse es la lógica de procesamiento. De hecho, la definición de un sistema de computación distribuida dada anteriormente asume implícitamente que...

La lógica o los elementos de procesamiento se distribuyen. Otra distribución posible es la funcional. Diversas funciones de un sistema informático podrían delegarse en diversos componentes de hardware o software. Un tercer modo de distribución posible es la de los datos. Los datos utilizados por diversas aplicaciones pueden distribuirse a varios sitios de procesamiento. Finalmente, el control puede ser distribuido. El control de la ejecución de diversas tareas podría distribuirse en lugar de que lo realice un solo sistema informático. Desde la perspectiva de los sistemas de bases de datos distribuidas, todos estos modos de distribución son necesarios e importantes. En las siguientes secciones, los abordaremos con más detalle.

Otra pregunta razonable que podemos plantearnos en este momento es: ¿Por qué distribuimos? Las respuestas clásicas a esta pregunta indican que el procesamiento distribuido se adapta mejor a la estructura organizativa de las empresas actuales, ampliamente distribuidas, y que dicho sistema es más fiable y ágil. Más importante aún, muchas de las aplicaciones actuales de la tecnología informática son inherentemente distribuidas.
Las aplicaciones basadas en la Web, el comercio electrónico a través de Internet, las aplicaciones multimedia como las noticias a pedido o las imágenes médicas y los sistemas de control de fabricación son ejemplos de dichas aplicaciones.

Sin embargo, desde una perspectiva más global, se puede afirmar que la razón fundamental del procesamiento distribuido es la capacidad de afrontar mejor los problemas de gestión de datos a gran escala que enfrentamos hoy en día, mediante una variante de la conocida regla de "divide y vencerás". Si se desarrolla el soporte de software necesario para el procesamiento distribuido, podría ser posible resolver estos complejos problemas simplemente dividiéndolos en partes más pequeñas y asignándolas a diferentes grupos de software, que funcionan en diferentes ordenadores y producen un sistema que se ejecuta en múltiples elementos de procesamiento, pero que puede trabajar eficientemente para la ejecución de una tarea común.

Los sistemas de bases de datos distribuidas también deben considerarse dentro de este marco y considerarse como herramientas que podrían simplificar y hacer más eficiente el procesamiento distribuido. Es razonable establecer una analogía entre lo que las bases de datos distribuidas podrían ofrecer al mundo del procesamiento de datos y lo que la tecnología de bases de datos ya ha proporcionado. Sin duda, el desarrollo de sistemas de bases de datos distribuidas de propósito general, adaptables y eficientes ha contribuido enormemente al desarrollo de software distribuido.

1.2 ¿Qué es un sistema de base de datos distribuida?

Definimos una base de datos distribuida como un conjunto de múltiples bases de datos lógicamente interrelacionadas y distribuidas en una red informática. Un sistema de gestión de bases de datos distribuidas (SGBD distribuido) se define entonces como el sistema de software que permite la gestión de la base de datos distribuida y hace que la distribución sea transparente para los usuarios. En ocasiones, el término «sistema de bases de datos distribuidas» (SBDD) se utiliza para referirse conjuntamente a la base de datos distribuida y al SGBD distribuido. Los dos términos importantes en estas definiciones son «lógicamente interrelacionado» y «distribuido en una red informática». Ayudan a eliminar ciertos casos que a veces han sido aceptados para representar un DDBS.

Un DDBS no es una colección de archivos que se puedan almacenar individualmente en cada nodo de una red informática. Para formar un DDBS, los archivos no solo deben estar lógicamente relacionados, sino que también deben estar estructurados entre sí y el acceso debe realizarse mediante una interfaz común. Cabe destacar que recientemente se ha desarrollado una amplia gama de funcionalidades de DBMS para datos semiestructurados almacenados en archivos de Internet (como páginas web). En vista de esta actividad, el requisito anterior puede parecer innecesariamente estricto. Sin embargo, es importante distinguir entre un DDBS que cumple este requisito y sistemas de gestión de datos distribuidos más generales que proporcionan un acceso a los datos similar a un DBMS. En varios capítulos de este libro, ampliaremos nuestra exposición para abarcar estos sistemas más generales.

A veces se ha asumido que la distribución física de los datos no es el problema más significativo. Por lo tanto, los defensores de este punto de vista se sentirían cómodos al etiquetar como una base de datos distribuida un número de bases de datos (relacionadas) que residen en el mismo sistema informático. Sin embargo, la distribución física de los datos es importante. Crea problemas que no se encuentran cuando las bases de datos residen en el mismo ordenador . Estas dificultades se tratan en la Sección 1.5. Tenga en cuenta que la distribución física no implica necesariamente que los sistemas informáticos estén geográficamente alejados; en realidad podrían estar en la misma habitación. Simplemente implica que la comunicación entre ellos se realiza a través de una red en lugar de a través de memoria compartida o disco compartido (como sería el caso con los sistemas multiprocesador), con la red como el único recurso compartido.

Esto sugiere que los sistemas multiprocesador no deberían considerarse como DDBS. Aunque los multiprocesadores sin recursos compartidos, donde cada nodo de procesador tiene su propia memoria primaria y secundaria, y puede tener también sus propios periféricos, son bastante similares al entorno distribuido que analizamos, existen diferencias. La diferencia fundamental reside en el modo de operación. Un diseño de sistema multiprocesador es bastante simétrico, compuesto por varios componentes idénticos de procesador y memoria, y controlado por una o más copias del mismo sistema operativo, responsable de un control estricto de la asignación de tareas a cada procesador. Esto no ocurre en los sistemas de computación distribuida, donde la heterogeneidad del sistema operativo y del hardware es bastante común. Los sistemas de bases de datos que se ejecutan sobre sistemas multiprocesador se denominan sistemas de bases de datos paralelos y se describen en el capítulo 14.

Un DDBS tampoco es un sistema donde, a pesar de existir una red, la base de datos reside en un solo nodo de la misma (Figura 1.3). En este caso, los problemas de gestión de bases de datos no difieren de los que se presentan en un entorno de bases de datos centralizado (próximamente, analizaremos los SGBD cliente-servidor, que flexibilizan este requisito en cierta medida). La base de datos es gestionada centralmente por un sistema informático (sitio 2 en la Figura 1.3) y todas las solicitudes se enrutan a dicho sitio. La única consideración adicional se refiere a los retrasos en la transmisión. Es obvio que la existencia de una red informática o un conjunto de archivos no es suficiente para formar un sistema de bases de datos distribuido. Nos interesa un entorno donde los datos se distribuyan entre varios sitios (Figura 1.4).

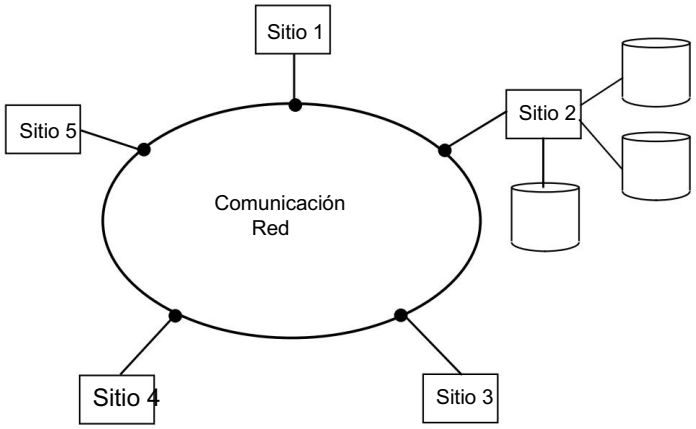

Fig. 1.3 Base de datos central en una red

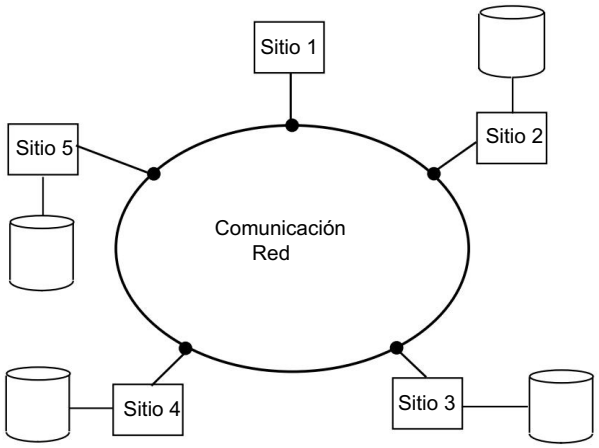

Figura 1.4 Entorno DDBS

1.3 Alternativas de entrega de datos

En bases de datos distribuidas, los datos se entregan desde los sitios donde se almacenan hasta donde se formula la consulta. Caracterizamos las alternativas de entrega de datos según tres dimensiones ortogonales: modos de entrega, frecuencia y métodos de comunicación. Las combinaciones de alternativas en cada una de estas dimensiones (que analizamos a continuación) proporcionan un amplio espacio de diseño.

Los modos de entrega alternativos son solo extracción, solo inserción e híbrido. En el modo de entrega de datos solo extracción, la transferencia de datos de los servidores a los clientes se inicia mediante una extracción del cliente. Cuando un servidor recibe una solicitud de un cliente, este responde localizando la información solicitada. La característica principal de la entrega basada en extracción es que la llegada de nuevos elementos de datos o las actualizaciones de los existentes se realiza en un

El servidor se envía sin notificar a los clientes, a menos que estos lo consulten explícitamente. Además, en el modo de extracción, los servidores deben interrumpirse continuamente para atender las solicitudes de los clientes. Además, la información que los clientes pueden obtener de un servidor se limita a cuándo y qué datos deben solicitar. Los SGBD convencionales ofrecen principalmente entrega de datos mediante extracción.

En el modo de entrega de datos de solo inserción, la transferencia de datos desde los servidores a los clientes se inicia mediante una inserción del servidor en ausencia de cualquier solicitud específica de los clientes. La principal dificultad del enfoque push reside en decidir qué datos serían de interés común y cuándo enviarlos a los clientes: las alternativas son periódicas, irregulares o condicionales. Por lo tanto, la utilidad del envío push del servidor depende en gran medida de la precisión del servidor para predecir las necesidades de los clientes. En el modo push, los servidores difunden información a un conjunto ilimitado de clientes (difusión aleatoria) que pueden escuchar un medio, o a un conjunto selectivo de clientes (multidifusión), que pertenecen a ciertas categorías de destinatarios que podrían recibir los datos.

El modo híbrido de entrega de datos combina los mecanismos de extracción de cliente y envío de servidor . El enfoque de consulta continua (p. ej., [Liu et al., 1996], [Terry et al., 1992], [Chen et al., 2000], [Pandey et al., 2003]) presenta una posible manera de combinar los modos de extracción e inserción: la transferencia de información de los servidores a los clientes se inicia primero mediante una extracción del cliente (al plantear la consulta) y la posterior transferencia de información actualizada a los clientes se inicia mediante una inserción del servidor.

Existen tres medidas de frecuencia típicas que permiten clasificar la regularidad de la entrega de datos: periódica, condicional y ad hoc o irregular.

En la entrega periódica, los datos se envían desde el servidor a los clientes a intervalos regulares. Los intervalos pueden definirse por defecto del sistema o por los clientes mediante sus perfiles. Tanto la extracción como la inserción pueden realizarse periódicamente. La entrega periódica se realiza según un programa regular y predefinido. Una solicitud semanal de un cliente para consultar el precio de las acciones de IBM es un ejemplo de extracción periódica. Un ejemplo de inserción periódica es cuando una aplicación puede enviar la lista de precios de las acciones regularmente, por ejemplo, todas las mañanas. La inserción periódica es especialmente útil en situaciones en las que los clientes podrían no estar disponibles en todo momento o no poder responder a lo que se les ha enviado, como en entornos móviles, donde los clientes pueden desconectarse.

En la entrega condicional, los datos se envían desde los servidores siempre que se cumplan ciertas condiciones instaladas por los clientes en sus perfiles. Dichas condiciones pueden ser tan simples como un lapso de tiempo determinado o tan complejas como reglas de evento-condición-acción. La entrega condicional se utiliza principalmente en los sistemas de entrega híbridos o de solo inserción. Mediante la inserción condicional, los datos se envían según una condición preestablecida, en lugar de una programación repetitiva particular. Una aplicación que envía precios de acciones solo cuando cambian es un ejemplo de inserción condicional. Una aplicación que envía un estado de cuenta solo cuando el saldo total es un 5% inferior al umbral de saldo predefinido es un ejemplo de inserción condicional híbrida. La inserción condicional asume que los cambios son críticos para los clientes y que los clientes siempre están atentos y necesitan responder a lo que se les envía. La inserción condicional híbrida asume además que la falta de alguna información de actualización no es crucial para los clientes.

La entrega ad hoc es irregular y se realiza mayoritariamente en un sistema basado puramente en extracción. Los datos se extraen de los servidores a los clientes de manera ad hoc cuando los clientes lo solicitan.

Por el contrario, la extracción periódica surge cuando un cliente utiliza el sondeo para obtener datos de los servidores según un período regular (programación).

El tercer componente del espacio de diseño de alternativas de entrega de información es el método de comunicación. Estos métodos determinan las diversas maneras en que servidores y clientes se comunican para entregar información a los clientes. Las alternativas son unidifusión y uno a muchos. En unidifusión, la comunicación entre un servidor y un cliente es uno a uno: el servidor envía datos a un cliente utilizando un modo de entrega específico con cierta frecuencia. En uno a muchos, como su nombre indica, el servidor envía datos a varios clientes. Cabe destacar que no nos referimos a un protocolo específico; la comunicación uno a muchos puede utilizar un protocolo de multidifusión o de difusión.

Cabe señalar que esta caracterización es objeto de considerable debate. No está claro que todos los puntos del espacio de diseño sean significativos. Además, la especificación de alternativas como las condicionales y periódicas (que podrían tener sentido) es difícil. Sin embargo, sirve como una caracterización de primer orden de la complejidad de los sistemas emergentes de gestión de datos distribuidos. En este libro, nos centramos principalmente en sistemas de entrega de datos ad hoc, basados únicamente en la extracción, aunque en algunos capítulos se abordan ejemplos de otros enfoques.

1.4 Promesas de los DDBS

Se han citado numerosas ventajas de los DDBS en la literatura, desde razones sociológicas para la descentralización [D'Oliviera, 1977] hasta mejoras económicas. Todas estas se resumen en cuatro fundamentos que también pueden considerarse promesas de la tecnología DDBS: gestión transparente de datos distribuidos y replicados, acceso fiable a los datos mediante transacciones distribuidas, mejor rendimiento y mayor facilidad de expansión del sistema. En esta sección, analizamos estas promesas y, en el proceso, introducimos muchos de los conceptos que estudiaremos en capítulos posteriores.

1.4.1 Gestión transparente de datos distribuidos y replicados

La transparencia se refiere a la separación de la semántica de alto nivel de un sistema de los aspectos de implementación de bajo nivel. En otras palabras, un sistema transparente oculta los detalles de la implementación a los usuarios. La ventaja de un SGBD totalmente transparente es el alto nivel de soporte que ofrece para el desarrollo de aplicaciones complejas. Es evidente que queremos que todos los SGBD (centralizados o distribuidos) sean totalmente transparentes.

Comencemos nuestra discusión con un ejemplo. Consideremos una empresa de ingeniería con oficinas en Boston, Waterloo, París y San Francisco. Ejecutan proyectos en cada una de estas sedes y desean mantener una base de datos de sus empleados, los proyectos y otros datos relacionados. Suponiendo que la base de datos sea relacional, podemos almacenar...

Esta información se almacena en dos relaciones: EMP(ENO, ENAME, TITLE)[1] y PROJ(PNO, PNAME, BUDGET). También introducimos una tercera relación para almacenar información salarial: SAL(TITLE, AMT) y una cuarta relación, ASG, que indica qué empleados han sido asignados a qué proyectos, con qué duración y con qué responsabilidad: ASG(ENO, PNO, RESP, DUR). Si todos estos datos se almacenaran en un SGBD centralizado y quisiéramos conocer los nombres y empleados que trabajaron en un proyecto durante más de 12 meses, lo especificaríamos mediante la siguiente consulta SQL:

```
SELECCIONE ENAME, AMT
DE EMP, ASG, SAL DONDE ASG.DUR
> 12
Y EMP.ENO = ASG.ENO
Y SAL.TÍTULO = EMP.TÍTULO
```

Sin embargo, dada la naturaleza distribuida del negocio de esta empresa, es preferible, en estas circunstancias, localizar los datos de forma que los datos de los empleados de la oficina de Waterloo se almacenen en Waterloo, los de la oficina de Boston en Boston, y así sucesivamente. Lo mismo aplica a la información sobre proyectos y salarios. Por lo tanto, estamos realizando un proceso en el que particionamos cada una de las relaciones y almacenamos cada partición en un sitio diferente. Esto se conoce como fragmentación y se analiza con más detalle más adelante, en el capítulo 3.

Además, puede ser preferible duplicar algunos de estos datos en otros sitios por razones de rendimiento y fiabilidad. El resultado es una base de datos distribuida, fragmentada y replicada (Figura 1.5). El acceso totalmente transparente permite a los usuarios realizar la consulta como se especificó anteriormente, sin tener que preocuparse por la fragmentación, la ubicación ni la replicación de los datos, y dejar que el sistema se encargue de resolver estos problemas.

Para que un sistema gestione adecuadamente este tipo de consulta en una base de datos distribuida, fragmentada y replicada, debe ser capaz de gestionar diversos tipos de transparencias. Los analizamos en esta sección.

1.4.1.1 Independencia de los datos

La independencia de los datos es un aspecto fundamental de la transparencia que buscamos en un SGBD. Además, es el único tipo importante en el contexto de un SGBD centralizado. Se refiere a la inmunidad de las aplicaciones de usuario a los cambios en la definición y organización de los datos, y viceversa.

Como es bien sabido, la definición de datos se produce en dos niveles. En un nivel se especifica la estructura lógica de los datos, y en el otro, su estructura física. La primera se conoce comúnmente como definición del esquema, mientras que la segunda se denomina descripción física de los datos. Por lo tanto, podemos hablar de dos tipos de datos.

---

[1] Analizamos los sistemas relacionales en el Capítulo 2 (Sección 2.1), donde desarrollamos este ejemplo más a fondo. Por el momento, basta con señalar que esta nomenclatura indica que acabamos de definir una relación con tres atributos: ENO (que es la clave, identificada mediante subrayado), ENAME y TITLE.

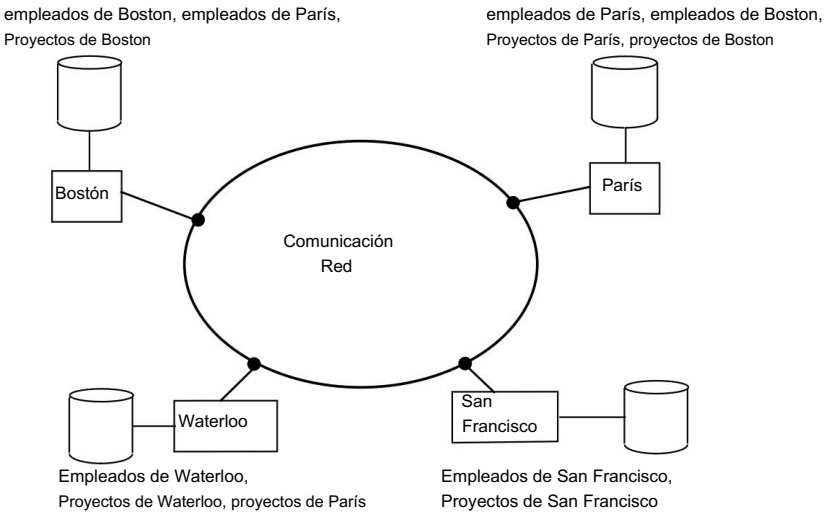

Fig. 1.5 Una aplicación distribuida

Independencia: independencia lógica de datos e independencia física de datos. La independencia lógica de datos se refiere a la inmunidad de las aplicaciones de usuario a cambios en la estructura lógica (es decir, el esquema) de la base de datos. La independencia física de datos, por otro lado, se encarga de ocultar los detalles de la estructura de almacenamiento a las aplicaciones de usuario. Al escribir una aplicación de usuario, no debería preocuparse por los detalles de la organización física de los datos. Por lo tanto, no debería ser necesario modificarla cuando se produzcan cambios en la organización de los datos por cuestiones de rendimiento.

1.4.1.2 Transparencia de la red

En sistemas de bases de datos centralizados, el único recurso disponible que debe protegerse del usuario son los datos (es decir, el sistema de almacenamiento). Sin embargo, en un entorno de bases de datos distribuidas, existe un segundo recurso que debe gestionarse de forma similar: la red. Preferiblemente, el usuario debe estar protegido de los detalles operativos de la red; posiblemente incluso ocultando su existencia. En tal caso, no habría diferencia entre las aplicaciones de bases de datos que se ejecutan en una base de datos centralizada y las que se ejecutan en una base de datos distribuida. Este tipo de transparencia se conoce como transparencia de red o transparencia de distribución.

La transparencia de la red se puede considerar desde la perspectiva de los servicios prestados o de los datos. Desde la primera perspectiva, es deseable contar con un método uniforme para acceder a los servicios. Desde la perspectiva de un SGBD, la transparencia de la distribución exige que los usuarios no tengan que especificar la ubicación de los datos.

A veces se identifican dos tipos de transparencia de distribución: transparencia de ubicación y transparencia de nombres. La transparencia de ubicación se refiere al hecho de que

El comando utilizado para realizar una tarea es independiente tanto de la ubicación de los datos como del sistema en el que se realiza la operación. La transparencia de nombres implica que se proporciona un nombre único para cada objeto en la base de datos. En ausencia de transparencia de nombres, los usuarios deben integrar el nombre de la ubicación (o un identificador) como parte del nombre del objeto.

1.4.1.3 Transparencia de replicación

El tema de la replicación de datos dentro de una base de datos distribuida se presenta en el Capítulo 3 y se analiza en detalle en el Capítulo 13. En este punto, cabe mencionar que, por razones de rendimiento, fiabilidad y disponibilidad, suele ser deseable poder distribuir los datos de forma replicada entre las máquinas de una red. Dicha replicación mejora el rendimiento, ya que se pueden satisfacer con mayor facilidad los diversos y conflictivos requisitos de los usuarios. Por ejemplo, los datos a los que un usuario accede habitualmente pueden ubicarse tanto en su máquina local como en la de otro usuario con los mismos requisitos de acceso. Esto aumenta la localización de la referencia. Además, si una de las máquinas falla, una copia de los datos sigue disponible en otra máquina de la red. Por supuesto, esta es una descripción muy simple de la situación. De hecho, la decisión de replicar o no, y cuántas copias de cualquier objeto de base de datos tener, depende en gran medida de las aplicaciones del usuario. Analizaremos esto en capítulos posteriores.

Suponiendo que los datos se replican, la cuestión de la transparencia radica en si los usuarios deben ser conscientes de la existencia de copias o si el sistema debe gestionarlas y el usuario actuar como si existiera una única copia de los datos (cabe destacar que no nos referimos a la ubicación de las copias, sino solo a su existencia). Desde la perspectiva del usuario, la respuesta es obvia. Es preferible no involucrarse en la gestión de copias ni tener que especificar que una determinada acción puede o debe realizarse en múltiples copias. Sin embargo, desde la perspectiva del sistema, la respuesta no es tan sencilla. Como veremos en el Capítulo 11, al delegar al usuario la responsabilidad de especificar que una acción debe ejecutarse en múltiples copias, se simplifica la gestión de transacciones en los SGBD distribuidos. Por otro lado, esto inevitablemente conlleva la pérdida de flexibilidad. No es el sistema quien decide si se deben tener copias ni cuántas, sino la aplicación Cualquier cambio en estas decisiones, debido a diversas consideraciones, afecta sin duda a la aplicación del usuario y, por lo tanto, reduce considerablemente la independencia de los datos. Por ello, es deseable que la transparencia de replicación se incluya como característica estándar de los SGBD. Recuerde que la transparencia de replicación se refiere únicamente a la existencia de réplicas, no a su ubicación real. Tenga en cuenta también que la distribución transparente de estas réplicas en la red es competencia de la transparencia de red.

1.4.1.4 Transparencia de fragmentación

La última forma de transparencia que debe abordarse en el contexto de un sistema de bases de datos distribuidas es la transparencia de fragmentación. En el capítulo 3, analizamos y justificamos que suele ser conveniente dividir cada relación de base de datos en fragmentos más pequeños y tratar cada fragmento como un objeto de base de datos independiente (es decir, otra relación). Esto se suele hacer por razones de rendimiento, disponibilidad y fiabilidad. Además, la fragmentación puede reducir los efectos negativos de la replicación. Cada réplica no representa la relación completa, sino solo un subconjunto de ella; por lo tanto, se requiere menos espacio y se deben gestionar menos elementos de datos.

Existen dos tipos generales de fragmentación. En un caso, denominado fragmentación horizontal, una relación se divide en un conjunto de subrelaciones, cada una con un subconjunto de las tuplas (filas) de la relación original. La segunda alternativa es la fragmentación vertical, donde cada subrelación se define en un subconjunto de los atributos (columnas) de la relación original.

Cuando los objetos de la base de datos están fragmentados, debemos abordar el problema de gestionar las consultas de usuario que se especifican en relaciones completas, pero que deben ejecutarse en subrelaciones. En otras palabras, el problema radica en encontrar una estrategia de procesamiento de consultas basada en los fragmentos en lugar de en las relaciones, aunque las consultas se especifiquen en estas últimas. Normalmente, esto requiere una traducción de lo que se denomina una consulta global a varias consultas de fragmentos. Dado que el problema fundamental para gestionar la transparencia de la fragmentación reside en el procesamiento de consultas, posponemos la discusión de las técnicas que permiten realizar esta traducción hasta el capítulo 7.

1.4.1.5 ¿Quién debe proporcionar transparencia?

En secciones anteriores, analizamos diversas formas posibles de transparencia en un entorno informático distribuido. Obviamente, para que los usuarios principiantes puedan acceder de forma fácil y eficiente a los servicios del SGBD, es deseable una transparencia total , que incluya todos los tipos que hemos analizado. Sin embargo, el nivel de transparencia es inevitablemente un equilibrio entre la facilidad de uso y la dificultad y el coste general que supone proporcionar altos niveles de transparencia. Por ejemplo, Gray argumenta que la transparencia total dificulta enormemente la gestión de datos distribuidos y afirma que «las aplicaciones codificadas con acceso transparente a bases de datos distribuidas geográficamente presentan: baja manejabilidad, baja modularidad y bajo rendimiento de los mensajes». [Gray, 1989]. Propone un mecanismo de llamada a procedimiento remoto entre los usuarios solicitantes y los SGBD del servidor, mediante el cual los usuarios dirigirían sus consultas a un SGBD específico. Este es, de hecho, el enfoque comúnmente adoptado por los sistemas cliente-servidor que analizaremos a continuación.

Lo que aún no se ha discutido es quién es responsable de proporcionar estos servicios. Es posible identificar tres niveles distintos en los que se pueden prestar los servicios de transparencia. Es bastante común considerarlos como medios mutuamente excluyentes , aunque es más apropiado considerarlos complementarios.

Podríamos dejar la responsabilidad de proporcionar acceso transparente a los recursos de datos a la capa de acceso. Las características de transparencia pueden integrarse en el lenguaje del usuario, que traduce los servicios solicitados en las operaciones requeridas. En otras palabras, el compilador o el intérprete asumen la tarea y no se proporciona ningún servicio transparente al implementador del compilador o del intérprete.

La segunda capa donde se puede proporcionar transparencia es el sistema operativo . Los sistemas operativos de vanguardia ofrecen cierto nivel de transparencia a los usuarios. Por ejemplo, los controladores de dispositivo del sistema operativo gestionan los detalles para que cada periférico realice la función solicitada. El usuario típico de una computadora, o incluso un programador de aplicaciones, no suele escribir controladores de dispositivo para interactuar con cada periférico; esa operación es transparente para el usuario.

Proporcionar acceso transparente a los recursos a nivel de sistema operativo puede extenderse obviamente al entorno distribuido, donde la gestión de los recursos de red la asume el sistema operativo distribuido o el middleware si el SGBD distribuido se implementa sobre uno. Este enfoque presenta dos posibles problemas. El primero es que no todos los sistemas operativos distribuidos disponibles comercialmente ofrecen un nivel razonable de transparencia en la gestión de la red. El segundo problema es que algunas aplicaciones no desean estar protegidas de los detalles de la distribución y necesitan acceder a ellos para realizar ajustes específicos del rendimiento.

La tercera capa que permite la transparencia se encuentra en el SGBD. La transparencia y el soporte para las funciones de base de datos que proporciona el sistema operativo subyacente a los diseñadores del SGBD son generalmente mínimos y se limitan a operaciones fundamentales para realizar ciertas tareas. El SGBD es responsable de realizar todas las traducciones necesarias del sistema operativo a la interfaz de usuario de nivel superior. Este modo de operación es el más común hoy en día.

Sin embargo, existen varios problemas asociados a dejar la tarea de proporcionar transparencia total al SGBD. Estos se relacionan con la interacción del sistema operativo con el SGBD distribuido y se abordan a lo largo de este libro.

La Figura 1.6 muestra una jerarquía de estas transparencias . No siempre es fácil delinear con claridad los niveles de transparencia, pero dicha figura cumple una importante función instructiva, aunque no sea completamente correcta. Para completar la imagen, hemos añadido una capa de "transparencia de lenguaje", aunque no se trata en este capítulo. Con esta capa genérica, los usuarios tienen acceso de alto nivel a los datos (p. ej., lenguajes de cuarta generación, interfaces gráficas de usuario, acceso a lenguaje natural).

1.4.2 Confiabilidad a través de transacciones distribuidas

Los SGBD distribuidos están diseñados para mejorar la fiabilidad, ya que cuentan con componentes replicados y, por lo tanto, eliminan los puntos únicos de fallo. El fallo de un solo sitio, o el fallo de un enlace de comunicación que hace que uno o más sitios sean inaccesibles, no es suficiente para provocar la caída de todo el sistema. En el caso de una base de datos distribuida, esto significa que algunos datos pueden ser inaccesibles, pero con el cuidado adecuado, los usuarios...

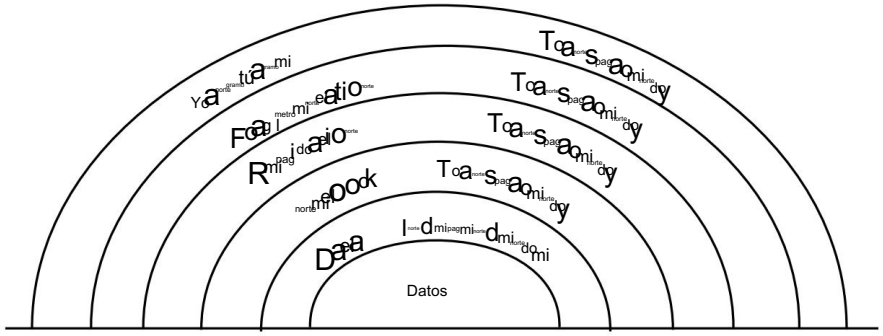

Fig. 1.6 Capas de transparencia

Se les puede permitir acceder a otras partes de la base de datos distribuida. El "cuidado adecuado"
Viene en forma de soporte para transacciones distribuidas y protocolos de aplicación.

Analizamos las transacciones y el procesamiento de transacciones en detalle en los capítulos 10 a 12.
Una transacción es una unidad básica de computación consistente y confiable, que consiste en una
secuencia de operaciones de base de datos ejecutadas como una acción atómica. Transforma un estado
de base de datos consistente en otro estado de base de datos consistente incluso cuando varias de estas
Las transacciones se ejecutan simultáneamente (a veces denominada transparencia de concurrencia),
e incluso cuando ocurren fallos (también llamado atomicidad de fallos). Por lo tanto, un SGBD
que proporciona soporte completo para transacciones garantiza la ejecución simultánea de acciones del usuario
Las transacciones no violarán la consistencia de la base de datos ante fallas del sistema.
siempre que cada transacción sea correcta, es decir, obedezca las reglas de integridad especificadas en la
base de datos.

Pongamos un ejemplo de una transacción basada en el ejemplo de la empresa de ingeniería.
que presentamos anteriormente. Supongamos que hay una aplicación que actualiza el
salarios de todos los empleados en un 10%. Es conveniente encapsular la consulta (o
el código del programa) que realiza esta tarea dentro de los límites de la transacción. Para
Por ejemplo, si se produce un fallo del sistema a mitad de la ejecución de este programa,
Nos gustaría que el DBMS pudiera determinar, al recuperarse, dónde se quedó.
y continuar con su funcionamiento (o empezar de nuevo). Este es el tema del fracaso.
atomicidad. Alternativamente, si otro usuario ejecuta una consulta que calcula el promedio
salarios de los empleados de esta empresa mientras se lleva a cabo la acción de actualización original,
El resultado calculado será erróneo. Por lo tanto, nos gustaría que el sistema pudiera...
Sincronizar la ejecución simultánea de estos dos programas. Para encapsular una consulta
(o un código de programa) dentro de los límites transaccionales, es suficiente declarar el
inicio de la transacción y su fin:

```
Iniciar transacción ACTUALIZACIÓN ÐE SALARIO
comenzar
   EJECUTAR SQL ACTUALIZAR PAGO
                        COLOCAR        SAL = SAL*1.1
fin.
```

Las transacciones distribuidas se ejecutan en varios sitios desde los que acceden a la base de datos local. La transacción anterior, por ejemplo, se ejecutará en Boston, Waterloo, París y San Francisco, ya que los datos se distribuyen en estos sitios. Con soporte completo para transacciones distribuidas, las aplicaciones de usuario pueden acceder a una única imagen lógica de la base de datos y confiar en el SGBD distribuido para garantizar que sus solicitudes se ejecuten correctamente, independientemente de lo que ocurra en el sistema. "Correctamente" significa que las aplicaciones de usuario no necesitan preocuparse por coordinar sus accesos a bases de datos locales individuales ni por la posibilidad de fallos en el sitio o en el enlace de comunicación durante la ejecución de sus transacciones. Esto ilustra la relación entre las transacciones distribuidas y la transparencia, ya que ambas implican cuestiones relacionadas con la asignación de nombres y la gestión de directorios, entre otras.

Proporcionar soporte transaccional requiere la implementación de protocolos de control de concurrencia distribuida (Capítulo 11) y confiabilidad distribuida (Capítulo 12) , en particular, los protocolos de confirmación en dos fases (2PC) y de recuperación distribuida, que son significativamente más complejos que sus contrapartes centralizadas. El soporte de réplicas requiere la implementación de protocolos de control de réplicas que implementen una semántica específica para acceder a ellas (Capítulo 13).

1.4.3 Rendimiento mejorado

La mejora del rendimiento de los SGBD distribuidos se basa generalmente en dos puntos. En primer lugar, un SGBD distribuido fragmenta la base de datos conceptual, lo que permite almacenar los datos cerca de sus puntos de uso (lo que se conoce como localización de datos). Esto presenta dos posibles ventajas:

1. Dado que cada sitio maneja solo una parte de la base de datos, la contención de los servicios de CPU y E/S no es tan grave como en las bases de datos centralizadas.
2. La localización reduce los retrasos de acceso remoto que suelen estar asociados a las redes de área amplia (por ejemplo, el retraso mínimo de propagación de mensajes de ida y vuelta en sistemas basados en satélites es de aproximadamente 1 segundo).

La mayoría de los SGBD distribuidos están estructurados para maximizar el beneficio de la localización de datos . La reducción de la contención y la sobrecarga de comunicación solo se puede obtener mediante una fragmentación y distribución adecuadas de la base de datos.

Este punto se relaciona con la sobrecarga de la computación distribuida si los datos deben residir en sitios remotos y se debe acceder a ellos mediante comunicación remota. El argumento es que, en estas circunstancias, es mejor distribuir la funcionalidad de gestión de datos donde se encuentran los datos en lugar de mover grandes cantidades de datos.
Esto se ha convertido últimamente en un tema de controversia. Algunos argumentan que, con el uso generalizado de redes de alta velocidad y alta capacidad, la distribución y la gestión de datos ya no tienen sentido, y que podría ser mucho más sencillo almacenar los datos en un sitio central y acceder a ellos (mediante descargas) a través de redes de alta velocidad. Este argumento, si bien atractivo, pasa por alto el propósito de las bases de datos distribuidas. En primer lugar, en

En la mayoría de las aplicaciones actuales, los datos se distribuyen; lo que podría ser objeto de debate es cómo y dónde los procesamos. En segundo lugar, y más importante, este argumento no distingue entre ancho de banda (la capacidad de los enlaces informáticos) y latencia (el tiempo que tardan en transmitirse los datos). La latencia es inherente a los entornos distribuidos y existen límites físicos a la velocidad de transmisión de datos a través de redes informáticas. Como se indicó anteriormente, por ejemplo, los enlaces satelitales tardan aproximadamente medio segundo en transmitir datos entre dos estaciones terrestres. Esto depende de la distancia de los satélites a la Tierra, y no hay nada que podamos hacer para mejorar ese rendimiento. Para algunas aplicaciones, esto podría representar un retraso inaceptable.

El segundo caso es que el paralelismo inherente de los sistemas distribuidos puede aprovecharse para el paralelismo entre consultas y dentro de ellas. El paralelismo entre consultas resulta de la capacidad de ejecutar múltiples consultas simultáneamente, mientras que el paralelismo dentro de consultas se logra fragmentando una consulta en varias subconsultas, cada una de las cuales se ejecuta en un sitio diferente y accede a una parte distinta de la base de datos distribuida .

Si el acceso del usuario a la base de datos distribuida consistiera únicamente en consultas (es decir, acceso de solo lectura), la provisión de paralelismo entre consultas e intraconsultas implicaría replicar la mayor cantidad posible de la base de datos. Sin embargo, dado que la mayoría de los accesos a la base de datos no son de solo lectura, la combinación de operaciones de lectura y actualización requiere la implementación de protocolos complejos de control de concurrencia y confirmación.

1.4.4 Expansión más sencilla del sistema

En un entorno distribuido, es mucho más fácil adaptarse al aumento del tamaño de las bases de datos. Rara vez se requieren revisiones importantes del sistema; la expansión suele gestionarse añadiendo potencia de procesamiento y almacenamiento a la red. Obviamente, puede que no sea posible obtener un aumento lineal de la potencia, ya que esta también depende de la sobrecarga de la distribución. Sin embargo, aún es posible realizar mejoras significativas.

Un aspecto que facilita la expansión de sistemas es la economía. Normalmente cuesta mucho menos ensamblar un sistema de computadoras "más pequeñas" con la potencia equivalente a una sola máquina grande. Anteriormente, se creía que era posible comprar una computadora cuatro veces más potente gastando el doble. Esto se conocía como la ley de Grosh. Con la llegada de las microcomputadoras y las estaciones de trabajo, y sus características de precio y rendimiento, esta ley se considera inválida.

Esto no debe interpretarse como la desaparición de los mainframes; no es ese el punto que planteamos aquí. De hecho, en los últimos años, hemos observado un resurgimiento en la venta mundial de mainframes. La cuestión es que, para muchas aplicaciones, resulta más económico crear un sistema informático distribuido (ya sea compuesto por mainframes o estaciones de trabajo) con suficiente potencia que establecer un sistema único y centralizado para ejecutar estas tareas. De hecho, esto último podría no ser viable hoy en día.

1.5 Complicaciones introducidas por la distribución

Los problemas encontrados en los sistemas de bases de datos adquieren una complejidad adicional en un entorno distribuido, aunque los principios básicos subyacentes son los mismos.
Además, esta complejidad adicional da lugar a nuevos problemas influenciados principalmente por tres factores.

En primer lugar, los datos pueden replicarse en un entorno distribuido. Una base de datos distribuida puede diseñarse de modo que toda la base de datos, o parte de ella, resida en diferentes sitios de una red informática. No es imprescindible que todos los sitios de la red contengan la base de datos; basta con que haya más de un sitio donde resida. La posible duplicación de datos se debe principalmente a consideraciones de fiabilidad y eficiencia. En consecuencia, el sistema de base de datos distribuida es responsable de (1) seleccionar una de las copias almacenadas de los datos solicitados para acceder a ella en caso de recuperación, y (2) garantizar que el efecto de una actualización se refleje en todas y cada una de las copias de dicho dato.

En segundo lugar, si algunos sitios fallan (por ejemplo, por un mal funcionamiento del hardware o del software), o si algunos enlaces de comunicación fallan (lo que hace que algunos de los sitios sean inaccesibles) mientras se ejecuta una actualización, el sistema debe asegurarse de que los efectos se reflejarán en los datos que residen en los sitios que fallan o son inaccesibles tan pronto como el sistema pueda recuperarse de la falla.

El tercer punto es que como cada sitio no puede tener información instantánea de las acciones que se llevan a cabo en los otros sitios, la sincronización de transacciones en múltiples sitios es considerablemente más difícil que en un sistema centralizado.

Estas dificultades apuntan a una serie de problemas potenciales con los DBMS distribuidos.
Estas son la complejidad inherente a la creación de aplicaciones distribuidas, el mayor coste de replicar recursos y, lo que es más importante, la gestión de la distribución, la devolución del control a muchos centros y la dificultad de llegar a acuerdos, y las preocupaciones de seguridad exacerbadas (el problema del canal de comunicación seguro).
Estos son problemas bien conocidos en los sistemas distribuidos en general y, en este libro, analizamos sus manifestaciones dentro del contexto de los DBMS distribuidos y cómo pueden abordarse.

1.6 Problemas de diseño

En la Sección 1.4, analizamos las promesas de la tecnología de SGBD distribuidos, destacando los desafíos que deben superarse para hacerlas realidad. En esta sección, profundizamos en este análisis presentando los problemas de diseño que surgen al crear un SGBD distribuido. Estos problemas ocuparán gran parte del resto del libro.

1.6.1 Diseño de bases de datos distribuidas

La pregunta que se aborda es cómo distribuir la base de datos y las aplicaciones que se ejecutan en ella en los diferentes sitios. Existen dos alternativas básicas para la distribución de datos: particionada (o no replicada) y replicada. En el esquema particionado, la base de datos se divide en varias particiones separadas, cada una ubicada en un sitio diferente. Los diseños replicados pueden ser completamente replicados (también llamados completamente duplicados), donde la base de datos completa se almacena en cada sitio, o parcialmente replicados (o parcialmente duplicados), donde cada partición de la base de datos se almacena en más de un sitio, pero no en todos. Los dos aspectos fundamentales del diseño son la fragmentación (la separación de la base de datos en particiones llamadas fragmentos) y la distribución (la distribución óptima de los fragmentos).

La investigación en esta área se centra principalmente en la programación matemática para minimizar el coste combinado de almacenar la base de datos, procesar transacciones en ella y la comunicación de mensajes entre sitios. El problema general es NP-hard. Por lo tanto, las soluciones propuestas se basan en heurísticas. El diseño de bases de datos distribuidas es el tema del capítulo 3.

1.6.2 Gestión de directorios distribuidos

Un directorio contiene información (como descripciones y ubicaciones) sobre los elementos de datos de la base de datos. Los problemas relacionados con la gestión de directorios son similares al problema de ubicación de la base de datos, analizado en la sección anterior. Un directorio puede ser global para todo el DDBS o local para cada sitio; puede estar centralizado en un sitio o distribuido en varios; puede haber una o varias copias. Discutiremos brevemente estas cuestiones en el capítulo 3.

1.6.3 Procesamiento de consultas distribuidas

El procesamiento de consultas se ocupa del diseño de algoritmos que analizan las consultas y las convierten en una serie de operaciones de manipulación de datos. El problema radica en cómo decidir una estrategia para ejecutar cada consulta en la red de la manera más rentable, independientemente de cómo se defina el coste. Los factores a considerar son la distribución de datos, los costes de comunicación y la falta de suficiente información disponible localmente. El objetivo es optimizar dónde se utiliza el paralelismo inherente para mejorar el rendimiento de la ejecución de la transacción, sujeto a las restricciones mencionadas anteriormente. El problema es de naturaleza NP-hard, y los enfoques suelen ser heurísticos. El procesamiento distribuido de consultas se analiza en detalle en los capítulos 6 a 8.

1.6.4 Control de concurrencia distribuida

El control de concurrencia implica la sincronización de los accesos a la base de datos distribuida, de modo que se mantenga su integridad. Es, sin duda, uno de los problemas más estudiados en el campo de los DDBS. El problema del control de concurrencia en un contexto distribuido es algo diferente al de un entorno centralizado. No solo es necesario preocuparse por la integridad de una única base de datos, sino también por la consistencia de múltiples copias de la misma. La condición que exige que todos los valores de múltiples copias de cada dato converjan al mismo valor se denomina consistencia mutua.

Las soluciones alternativas son demasiado numerosas para discutirlas aquí, por lo que las examinamos en detalle en el Capítulo 11. Mencionemos solo que las dos clases generales son pesimista, que sincroniza la ejecución de las solicitudes del usuario antes de que comience la ejecución, y optimista, que ejecuta las solicitudes y luego verifica si la ejecución ha comprometido la consistencia de la base de datos. Dos primitivas fundamentales que se pueden usar con ambos enfoques son el bloqueo, que se basa en la exclusión mutua de los accesos a los elementos de datos, y el sellado de tiempo, donde las ejecuciones de las transacciones se ordenan según las marcas de tiempo. Existen variaciones de estos esquemas, así como algoritmos híbridos que intentan combinar los dos mecanismos básicos.

1.6.5 Gestión distribuida de bloqueos

El problema de interbloqueo en los DDBS es similar al que se presenta en los sistemas operativos. La competencia entre usuarios por el acceso a un conjunto de recursos (en este caso, datos) puede resultar en un interbloqueo si el mecanismo de sincronización se basa en el bloqueo. Las conocidas alternativas de prevención, evitación y detección/recuperación también se aplican a los DDBS. La gestión de interbloqueos se aborda en el capítulo 11.

1.6.6 Confiabilidad de los DBMS distribuidos

Mencionamos anteriormente que una de las ventajas potenciales de los sistemas distribuidos es la mejora de la confiabilidad y la disponibilidad. Sin embargo, esto no es una característica que se obtenga automáticamente. Es importante proporcionar mecanismos para garantizar la consistencia de la base de datos, así como para detectar fallos y recuperarse de ellos. Esto implica que, para los DDBS, cuando se produce un fallo y varios sitios dejan de funcionar o de ser accesibles, las bases de datos de los sitios operativos se mantienen consistentes y actualizadas. Además, cuando el sistema informático o la red se recupera de la falla, los DDBS deberían poder recuperarse y actualizar las bases de datos de los sitios fallidos.
Esto puede resultar especialmente difícil en el caso de la partición de red, donde los sitios se dividen en dos o más grupos sin comunicación entre ellos. Los protocolos de confiabilidad distribuida son el tema del Capítulo 12.

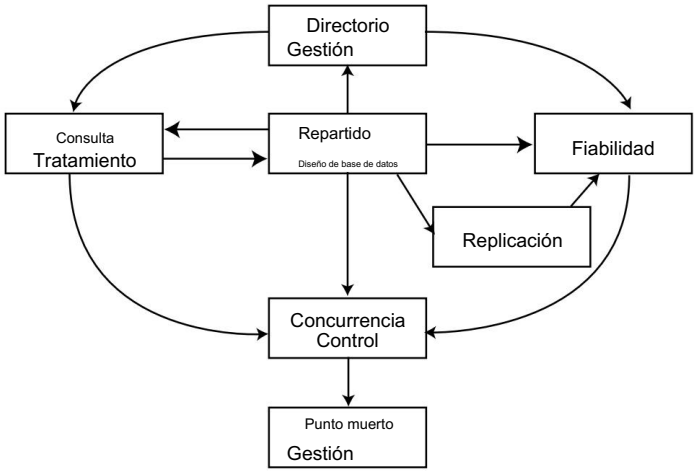

Figura 1.7 Relación entre los temas de investigación

1.6.7 Replicación

Si la base de datos distribuida se replica (parcial o totalmente), es necesario implementar protocolos que garanticen la consistencia de las réplicas; es decir, que las copias de un mismo dato tengan el mismo valor. Estos protocolos pueden ser diligentes, ya que fuerzan la aplicación de las actualizaciones a todas las réplicas antes de que se complete la transacción, o pueden ser perezosos, de modo que la transacción actualice una copia (denominada maestra), desde la cual se propagan las actualizaciones a las demás una vez completada la transacción. Los protocolos de replicación se abordan en el capítulo 13.

1.6.8 Relación entre problemas

Naturalmente, estos problemas no están aislados. Cada uno se ve afectado por las soluciones encontradas para los demás, lo que a su vez afecta el conjunto de soluciones factibles . En esta sección, analizamos su relación.

La relación entre los componentes se muestra en la Figura 1.7. El diseño de bases de datos distribuidas afecta a muchas áreas. Afecta la gestión de directorios, ya que la definición de fragmentos y su ubicación determinan el contenido del directorio (o directorios), así como las estrategias que se pueden emplear para gestionarlos.

El procesador de consultas utiliza la misma información (es decir, la estructura y la ubicación de los fragmentos) para determinar la estrategia de evaluación de consultas. Por otro lado, los patrones de acceso y uso determinados por el procesador de consultas se utilizan como entrada para los algoritmos de distribución y fragmentación de datos. De igual forma, la ubicación y el contenido de los directorios influyen en el procesamiento de las consultas.

La replicación de fragmentos al distribuirse afecta las estrategias de control de concurrencia que podrían emplearse. Como veremos en el capítulo 11, algunos algoritmos de control de concurrencia no son fáciles de usar con bases de datos replicadas.
De igual forma, los patrones de uso y acceso a la base de datos influirán en los algoritmos de control de concurrencia. Si el entorno requiere muchas actualizaciones, las precauciones necesarias son muy diferentes a las de un entorno de solo consultas.

Existe una fuerte relación entre el problema de control de concurrencia, el problema de gestión de interbloqueos y los problemas de fiabilidad. Esto es previsible, ya que en conjunto se les suele denominar problema de gestión de transacciones. El algoritmo de control de concurrencia empleado determinará si se requiere o no una función independiente de gestión de interbloqueos. Si se utiliza un algoritmo basado en bloqueos, se producirán interbloqueos, mientras que no ocurrirán si se opta por el sellado de tiempo.

Los mecanismos de confiabilidad involucran tanto técnicas de recuperación local como protocolos de confiabilidad distribuida. En ese sentido, ambos influyen en la elección de las técnicas de control de concurrencia y se basan en ellas. Las técnicas para proporcionar confiabilidad también utilizan la información sobre la ubicación de los datos, ya que la existencia de copias duplicadas de estos sirve como medida de seguridad para mantener un funcionamiento confiable.

Finalmente, la necesidad de protocolos de replicación surge si la distribución de datos implica réplicas. Como se indicó anteriormente, existe una estrecha relación entre los protocolos de replicación y las técnicas de control de concurrencia, ya que ambos gestionan la consistencia de los datos, aunque desde perspectivas diferentes. Además, los protocolos de replicación influyen en las técnicas de fiabilidad distribuida, como los protocolos de confirmación. De hecho, a veces se sugiere (erróneamente, en nuestra opinión) que se pueden utilizar protocolos de replicación en lugar de implementar protocolos de confirmación distribuida.

1.6.9 Cuestiones adicionales

Los problemas de diseño mencionados abarcan lo que podríamos llamar sistemas de bases de datos distribuidas tradicionales . El entorno ha cambiado significativamente desde que se empezaron a investigar estos temas , lo que plantea nuevos desafíos y oportunidades.

Uno de los avances más importantes ha sido la transición hacia una federación más flexible entre las fuentes de datos, que también pueden ser heterogéneas. Como se explica en la siguiente sección, esto ha dado lugar al desarrollo de sistemas multibase de datos (también denominados bases de datos federadas y sistemas de integración de datos) que requieren una nueva investigación de algunas de las técnicas fundamentales de las bases de datos. Estos sistemas constituyen una parte importante del entorno distribuido actual. En el capítulo 4, abordamos los problemas de diseño de bases de datos en sistemas multibase de datos (es decir, la integración de bases de datos) y, en el capítulo 9, los desafíos del procesamiento de consultas.

El crecimiento de Internet como plataforma de red fundamental ha planteado preguntas importantes sobre los supuestos que subyacen a los sistemas de bases de datos distribuidas.
Dos cuestiones nos preocupan especialmente. Una es el resurgimiento de la computación entre pares (P2P), y la otra es el desarrollo y crecimiento de la World Wide Web (web, para abreviar).
Ambas buscan mejorar el intercambio de datos, pero adoptan diferentes enfoques.

Estos enfoques plantean diferentes desafíos para la gestión de datos. En el capítulo 16, analizamos la gestión de datos entre pares y la gestión de datos web en el capítulo 17.

Cabe destacar que el concepto peer-to-peer no es nuevo en las bases de datos distribuidas, como se explica en la siguiente sección. Sin embargo, su nueva versión presenta diferencias significativas con respecto a las versiones anteriores. En el capítulo 16, nos centraremos en estas nuevas versiones .

Finalmente, como se mencionó anteriormente, existe una estrecha relación entre las bases de datos distribuidas y las bases de datos paralelas. Si bien las primeras asumen que cada sitio es un único ordenador lógico, la mayoría de estas instalaciones son, de hecho, clústeres paralelos. Por lo tanto, si bien la mayor parte del libro se centra en los problemas que surgen al gestionar datos distribuidos en estos sitios, existen interesantes problemas de gestión de datos dentro de un único sitio lógico que puede ser un sistema paralelo. Abordamos estos problemas en el capítulo 14.

1.7 Arquitectura de DBMS distribuida

La arquitectura de un sistema define su estructura. Esto significa que se identifican los componentes del sistema, se especifica la función de cada componente y se definen las interrelaciones e interacciones entre estos componentes. La especificación de la arquitectura de un sistema requiere la identificación de los distintos módulos, con sus interfaces e interrelaciones, en términos del flujo de datos y control a través del sistema.

En esta sección desarrollamos tres arquitecturas de “referencia”[2] para un DBMS distribuido: sistemas cliente/servidor, DBMS distribuidos punto a punto y sistemas de múltiples bases de datos.
Estas son visiones “idealizadas” de un DBMS en el sentido de que muchos de los sistemas comercialmente disponibles pueden desviarse de ellas; sin embargo, las arquitecturas servirán como un marco razonable dentro del cual se pueden discutir las cuestiones relacionadas con los DBMS distribuidos.

Comenzamos con una breve presentación de la "arquitectura ANSI/SPARC", un enfoque datalógico para definir la arquitectura de un SGBD. Se centra en las diferentes clases y roles de usuario y sus diferentes perspectivas sobre los datos. Esta arquitectura ayuda a comprender adecuadamente ciertos conceptos que hemos tratado hasta ahora. A continuación, presentamos una breve descripción de la arquitectura genérica de un SGBD centralizado, que posteriormente ampliamos para identificar las arquitecturas alternativas para un SGBD distribuido. Dentro de esta caracterización, nos centramos en las tres alternativas identificadas anteriormente.

1.7.1 Arquitectura ANSI/SPARC

A finales de 1972, el Comité de Procesamiento de Información y Computadoras (X3) del Instituto Nacional de Estándares de Estados Unidos (ANSI) estableció un Grupo de Estudio sobre Bases de Datos.

[2] Los desarrolladores de estándares suelen crear una arquitectura de referencia para definir claramente interfaces que necesitan ser estandarizadas.

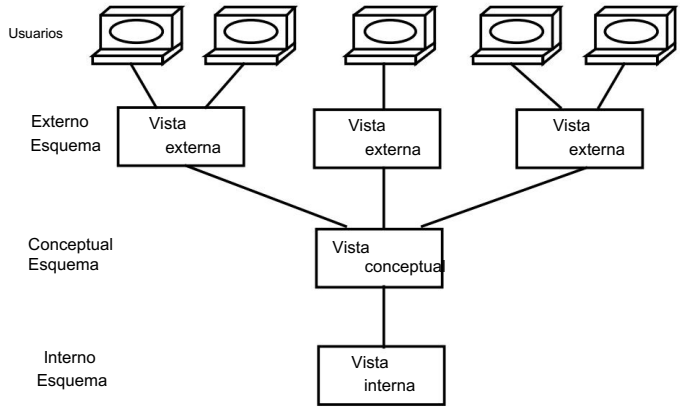

Figura 1.8 La arquitectura ANSI/SPARC

Sistemas de Gestión, bajo el auspicio de su Comité de Planificación y Requisitos de Normas (SPARC). La misión del grupo de estudio era estudiar la viabilidad de establecer normas en esta área, así como determinar qué aspectos deberían normalizarse, de ser viable. El grupo de estudio emitió su informe provisional en 1975 [ANSI/SPARC, 1975] y su informe final en 1977 [Tsichritzis y Klug, 1978].

El marco arquitectónico propuesto en estos informes se conoció como la «arquitectura ANSI/SPARC», cuyo nombre completo es «ANSI/X3/SPARC DBMS Framework». El grupo de estudio propuso la estandarización de las interfaces y definió un marco arquitectónico que contenía 43 interfaces, 14 de las cuales se encargarían del subsistema de almacenamiento físico del ordenador y, por lo tanto, no se considerarían partes esenciales de la arquitectura del DBMS.

En la Figura 1.8 se muestra una versión simplificada de la arquitectura ANSI/SPARC . Existen tres perspectivas de datos: la externa, que corresponde al usuario final, que puede ser un programador; la interna, que corresponde al sistema o la máquina; y la conceptual, que corresponde a la empresa. Para cada una de estas perspectivas, se requiere una definición de esquema adecuada .

En el nivel más bajo de la arquitectura se encuentra la vista interna, que se ocupa de la definición física y la organización de los datos. La ubicación de los datos en los diferentes dispositivos de almacenamiento y los mecanismos de acceso utilizados para acceder y manipular los datos son los aspectos que se abordan en este nivel. En el otro extremo se encuentra la vista externa, que se ocupa de cómo los usuarios ven la base de datos. La vista de un usuario individual representa la parte de la base de datos a la que accederá, así como las relaciones que desea ver entre los datos. Una vista puede ser compartida entre varios usuarios, y el conjunto de vistas de usuario conforma el esquema externo. Entre estos dos extremos se encuentra el esquema conceptual, que es una definición abstracta de la base de datos. Es la vista del "mundo real" de la empresa que se modela en la base de datos [Yormark, 1977]. Como tal, se supone que representa los datos y las relaciones entre ellos sin considerar los requisitos de las aplicaciones individuales ni las restricciones de los medios de almacenamiento físicos. Sin embargo, en la práctica, no es posible ignorar

Requisitos por completo, por razones de rendimiento. La transformación entre estos tres niveles se logra mediante asignaciones que especifican cómo se puede obtener una definición de un nivel a partir de una definición de otro.

Esta perspectiva es importante, ya que sienta las bases para la independencia de los datos que analizamos anteriormente. La separación de los esquemas externos del esquema conceptual permite la independencia lógica de los datos, mientras que la separación del esquema conceptual del esquema interno permite la independencia física de los datos.

1.7.2 Una arquitectura genérica de DBMS centralizada

Un SGBD es un programa reentrante compartido por múltiples procesos (transacciones) que ejecutan programas de bases de datos. Al ejecutarse en un ordenador de propósito general, un SGBD interactúa con otros dos componentes: el subsistema de comunicación y el sistema operativo. El subsistema de comunicación permite la interconexión del SGBD con otros subsistemas para la comunicación con las aplicaciones. Por ejemplo, el monitor del terminal necesita comunicarse con el SGBD para ejecutar transacciones interactivas. El sistema operativo proporciona la interfaz entre el SGBD y los recursos del ordenador (procesador, memoria, unidades de disco, etc.).

Las funciones que realiza un SGBD se pueden estructurar en capas, como se muestra en la Figura 1.9, donde las flechas indican la dirección de los datos y el flujo de control. Con un enfoque descendente , las capas son la interfaz, el control, la compilación, la ejecución, el acceso a los datos y la gestión de la consistencia.

La capa de interfaz gestiona la interfaz con las aplicaciones. Puede haber varias interfaces, como, en el caso de los SGBD relacionales que se describen en el Capítulo 2, SQL integrado en un lenguaje host, como C y QBE (Consulta por Ejemplo).
Los programas de aplicación de bases de datos se ejecutan en vistas externas de la base de datos. Para una aplicación, una vista es útil para representar su percepción particular de la base de datos (compartida por varias aplicaciones). En los SGBD relacionales, una vista es una relación virtual derivada de las relaciones base mediante la aplicación de operaciones de álgebra relacional.[3] Estos conceptos se definen con mayor precisión en el Capítulo 2, pero suelen abordarse en cursos de bases de datos de pregrado, por lo que esperamos que muchos lectores estén familiarizados con ellos. La gestión de vistas consiste en traducir la consulta del usuario de datos externos a datos conceptuales.

La capa de control controla la consulta añadiendo predicados de integridad semántica y predicados de autorización. Las restricciones de integridad semántica y las autorizaciones suelen especificarse en un lenguaje declarativo, como se explica en el capítulo 5. El resultado de esta capa es una consulta enriquecida en el lenguaje de alto nivel aceptado por la interfaz.

La capa de procesamiento (o compilación) de consultas asigna la consulta a una secuencia optimizada de operaciones de nivel inferior. Esta capa se encarga del rendimiento.

---

[3] Tenga en cuenta que esto no significa que las vistas del mundo real se especifiquen, o deban especificarse, en álgebra relacional. Al contrario, se especifican mediante un lenguaje de datos de alto nivel como SQL. La traducción de uno de estos lenguajes al álgebra relacional ya se comprende bien, y los efectos de la definición de la vista pueden especificarse en términos de operaciones de álgebra relacional.

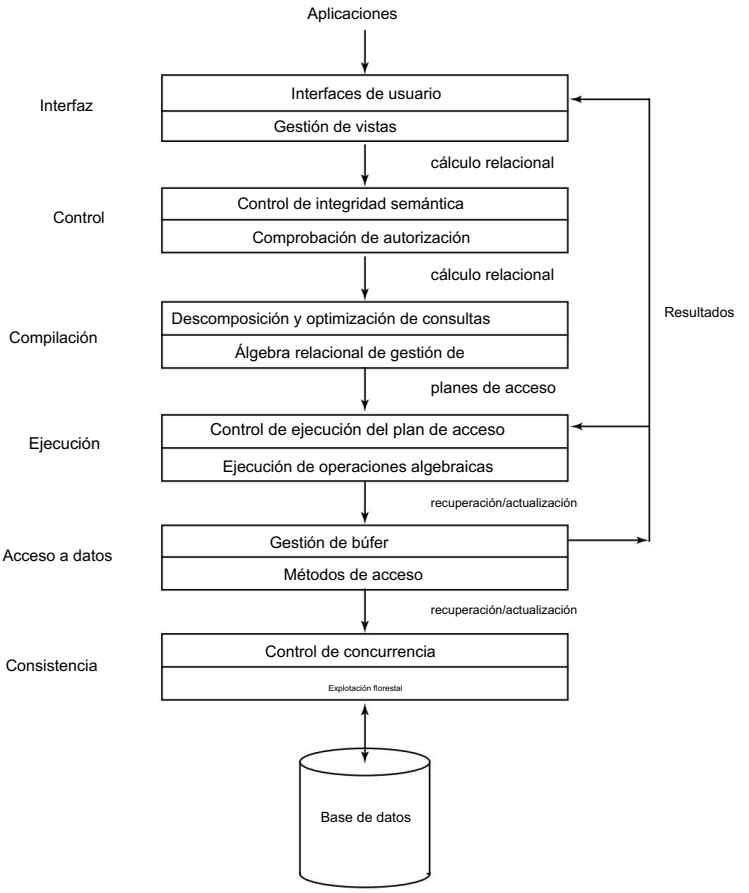

Figura 1.9 Capas funcionales de un DBMS centralizado

Descompone la consulta en un árbol de operaciones algebraicas e intenta encontrar el orden óptimo de las operaciones. El resultado se almacena en un plan de acceso. La salida de esta capa es una consulta expresada en código de nivel inferior (operaciones algebraicas).

La capa de ejecución dirige la ejecución de los planes de acceso, incluyendo la gestión de transacciones (confirmación, reinicio) y la sincronización de operaciones algebraicas. Interpreta las operaciones relacionales llamando a la capa de acceso a datos mediante las solicitudes de recuperación y actualización.

La capa de acceso a datos gestiona las estructuras de datos que implementan los archivos, índices, etc. También gestiona los búferes almacenando en caché los datos a los que se accede con mayor frecuencia. El uso cuidadoso de esta capa minimiza el acceso a los discos para obtener o escribir datos.

Finalmente, la capa de consistencia gestiona el control de concurrencia y el registro de actualizaciones. Solicitudes. Esta capa permite la recuperación de transacciones, sistemas y medios después de una falla.

1.7.3 Modelos arquitectónicos para sistemas DBMS distribuidos

Consideramos ahora las posibles formas en que se puede diseñar un DBMS distribuido. Utilizamos una clasificación (Figura 1.10) que organiza los sistemas según su caracterización con respecto a (1) la autonomía de los sistemas locales, (2) su distribución y (3) su heterogeneidad.

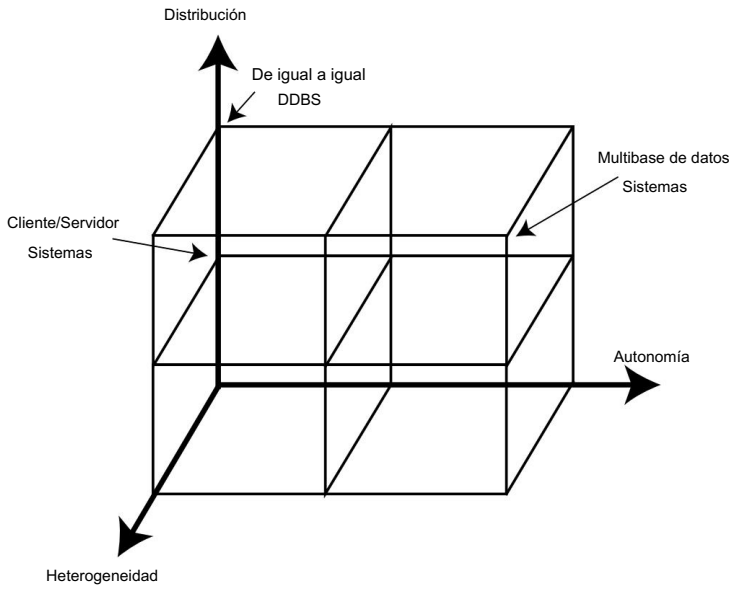

Figura 1.10 Alternativas de implementación del DBMS

1.7.4 Autonomía

La autonomía, en este contexto, se refiere a la distribución del control, no de los datos. Indica el grado en que los SGBD individuales pueden operar de forma independiente. La autonomía depende de diversos factores, como si los sistemas que los componen (es decir, los SGBD individuales) intercambian información, si pueden ejecutar transacciones de forma independiente y si se permite su modificación. Los requisitos de un sistema autónomo se especifican a continuación [Gligor y Popescu-Zeletin, 1986]:

1. Las operaciones locales de los DBMS individuales no se ven afectadas por su participación en el sistema distribuido.

2. La forma en que los DBMS individuales procesan las consultas y las optimizan no debería verse afectada por la ejecución de consultas globales que acceden a múltiples bases de datos.

3. La consistencia o el funcionamiento del sistema no deben verse comprometidos cuando se utilizan componentes individuales. Los DBMS se incorporan o abandonan el sistema distribuido.

Por otra parte, las dimensiones de la autonomía pueden especificarse de la siguiente manera [Du y Elmagarmid, 1989]:

1. Autonomía de diseño: Los DBMS individuales son libres de utilizar los modelos de datos y técnicas de gestión de transacciones que prefieran.
2. Autonomía de comunicación: Cada uno de los DBMS individuales es libre de tomar su propia decisión sobre qué tipo de información quiere proporcionar a los otros DBMS o al software que controla su ejecución global.
3. Autonomía de ejecución: Cada DBMS puede ejecutar las transacciones que le son subordinadas. comprometido a ello de cualquier manera que quiera.

Utilizaremos una clasificación que cubra los aspectos importantes de estas características. Una alternativa es la integración estrecha, donde una imagen única de toda la base de datos está disponible para cualquier usuario que desee compartir la información, que puede residir en varias bases de datos. Desde la perspectiva de los usuarios, los datos se integran lógicamente en una sola base de datos. En estos sistemas estrechamente integrados, los gestores de datos se implementan de forma que uno de ellos controle el procesamiento de cada solicitud de usuario, incluso si esta es atendida por más de un gestor de datos. Los gestores de datos no suelen funcionar como sistemas de gestión de bases de datos independientes, aunque suelen tener la funcionalidad necesaria.

A continuación, identificamos sistemas semiautónomos que consisten en SGBD que pueden (y suelen hacerlo) operar de forma independiente, pero que han decidido participar en una federación para que sus datos locales sean compartibles. Cada uno de estos SGBD determina qué partes de su propia base de datos hará accesibles a los usuarios de otros SGBD. No son sistemas totalmente autónomos, ya que deben modificarse para que puedan intercambiar información entre sí.

La última alternativa que consideramos es el aislamiento total, donde los sistemas individuales son SGBD independientes que desconocen la existencia de otros SGBD y cómo comunicarse con ellos. En estos sistemas, el procesamiento de las transacciones de usuario que acceden a múltiples bases de datos es especialmente difícil, ya que no existe control global sobre la ejecución de cada SGBD.

Es importante señalar que las tres alternativas que consideramos para sistemas autónomos no son las únicas. Simplemente destacamos las tres más populares.

1.7.5 Distribución

Mientras que la autonomía se refiere a la distribución (o descentralización) del control, la dimensión de distribución de la taxonomía se ocupa de los datos. Por supuesto, consideramos la distribución física de los datos en múltiples sitios; como se mencionó anteriormente, el usuario los ve como un conjunto lógico. Existen diversas maneras de distribuir los SGBD . Estas alternativas se resumen en dos clases: distribución cliente-servidor y distribución punto a punto (o distribución completa). Junto con la opción no distribuida , la taxonomía identifica tres arquitecturas alternativas.

La distribución cliente/servidor concentra las tareas de gestión de datos en los servidores, mientras que los clientes se centran en proporcionar el entorno de la aplicación, incluida la interfaz de usuario. Las tareas de comunicación se comparten entre los equipos cliente.
y servidores. Los SGBD cliente/servidor representan una solución práctica para distribuir la funcionalidad. Existen diversas maneras de estructurarlos, cada una con un nivel de distribución diferente. En cuanto al marco de trabajo, abstraemos estas diferencias y dejamos esa discusión para la Sección 1.7.8, dedicada a las arquitecturas de SGBD cliente/servidor. Lo importante en este punto es que los sitios de una red se distinguen como "clientes" y "servidores", y su funcionalidad es diferente.

En los sistemas peer to peer, no hay distinción entre máquinas cliente y servidores. Cada máquina cuenta con la funcionalidad completa de un SGBD y puede comunicarse con otras máquinas para ejecutar consultas y transacciones. La mayoría de los primeros trabajos sobre sistemas de bases de datos distribuidos partieron de la base de datos de arquitectura punto a punto. Por lo tanto, este libro se centra principalmente en los sistemas punto a punto (también conocidos como totalmente distribuidos), aunque muchas de las técnicas también se aplican a los sistemas cliente-servidor.

1.7.6 Heterogeneidad

La heterogeneidad puede presentarse de diversas formas en sistemas distribuidos, desde la heterogeneidad del hardware y las diferencias en los protocolos de red hasta las variaciones en los gestores de datos. Desde la perspectiva de este libro, las más importantes se relacionan con los modelos de datos, los lenguajes de consulta y los protocolos de gestión de transacciones. La representación de datos con diferentes herramientas de modelado genera heterogeneidad debido a las capacidades expresivas y limitaciones inherentes de cada modelo de datos. La heterogeneidad en los lenguajes de consulta no solo implica el uso de paradigmas de acceso a datos completamente distintos en distintos modelos de datos (acceso conjunto a conjunto en sistemas relacionales frente a acceso registro a registro en algunos sistemas orientados a objetos), sino que también abarca las diferencias entre lenguajes, incluso cuando los sistemas individuales utilizan el mismo modelo de datos. Si bien SQL es actualmente el lenguaje de consulta relacional estándar, existen numerosas implementaciones diferentes y el lenguaje de cada proveedor tiene una estructura ligeramente distinta (a veces incluso diferente semántica, lo que produce resultados distintos).

1.7.7 Alternativas arquitectónicas

La distribución de las bases de datos, su posible heterogeneidad y su autonomía son cuestiones ortogonales. Por consiguiente, según la caracterización anterior, existen 18 arquitecturas posibles. No todas estas alternativas arquitectónicas que conforman el espacio de diseño son significativas. Además, no todas son relevantes desde la perspectiva de este libro.

En la Figura 1.10, identificamos tres arquitecturas alternativas que constituyen el enfoque de este libro y que analizamos con más detalle en las siguientes tres subsecciones: (A0, D1, H0) que corresponde a sistemas de gestión de bases de datos (SGBD) distribuidos cliente-servidor, (A0, D2, H0) que es un SGBD distribuido punto a punto, y (A2, D2, H1) que representa un sistema multibase de datos heterogéneo, distribuido punto a punto. Cabe destacar que analizamos los problemas de heterogeneidad en el contexto de una arquitectura de sistema, aunque el problema también surge en otros modelos.

1.7.8 Sistemas cliente/servidor

Los SGBD cliente/servidor entraron en el panorama informático a principios de la década de 1990 y han tenido un impacto significativo tanto en la tecnología de los SGBD como en la forma en que trabajamos en informática. La idea general es muy simple y elegante: distinguir la funcionalidad necesaria y dividir estas funciones en dos clases: funciones de servidor y funciones de cliente. Esto proporciona una arquitectura de dos niveles que facilita la gestión de la complejidad de los SGBD modernos y la complejidad de la distribución.

Como ocurre con cualquier término popular, el concepto cliente/servidor ha sido objeto de un uso abusivo y ha adquirido diferentes significados. Desde una perspectiva centrada en los procesos, cualquier proceso que solicita los servicios de otro es su cliente, y viceversa. Sin embargo, es importante destacar que «computación cliente/servidor» y «SGBD cliente/servidor», tal como se utilizan en nuestro contexto, no se refieren a procesos, sino a máquinas reales. Por lo tanto, nos centraremos en qué software debe ejecutarse en las máquinas cliente y qué software debe ejecutarse en la máquina servidor.

Dicho de esta manera, el tema queda más claro y podemos empezar a estudiar las diferencias en la funcionalidad del cliente y del servidor. La asignación de funcionalidad entre clientes y servidores difiere según el tipo de SGBD distribuido (por ejemplo, relacional u orientado a objetos). En los sistemas relacionales, el servidor realiza la mayor parte de la gestión de datos. Esto significa que todo el procesamiento y la optimización de consultas, la gestión de transacciones y la gestión del almacenamiento se realizan en el servidor. El cliente, además de la aplicación y la interfaz de usuario, cuenta con un módulo cliente DBMS responsable de gestionar los datos almacenados en caché y (en ocasiones) los bloqueos de transacciones que también se hayan almacenado en caché. También es posible realizar la comprobación de consistencia de las consultas de usuario en el lado del cliente, pero esto no es habitual, ya que requiere la replicación del catálogo del sistema en las máquinas cliente. Por supuesto, existe un sistema operativo y software de comunicación que se ejecuta tanto en el cliente como en el servidor, pero solo nos centraremos en la funcionalidad relacionada con el DBMS. Esta arquitectura, representada en la F

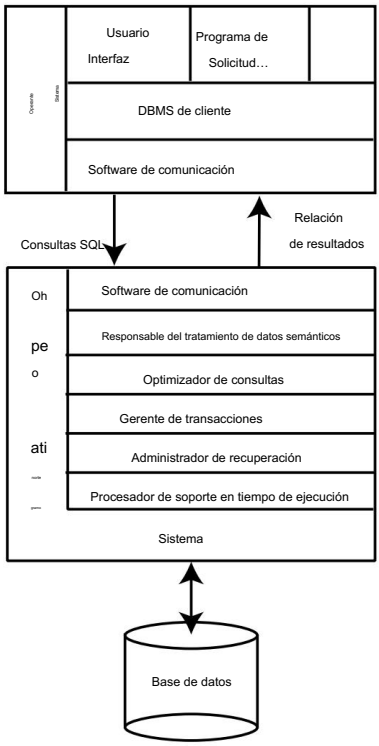

Fig. 1.11 Arquitectura de referencia cliente/servidor

Es bastante común en sistemas relacionales donde la comunicación entre los clientes y los servidores se realiza mediante sentencias SQL. En otras palabras, el cliente envía consultas SQL al servidor sin intentar comprenderlas ni optimizarlas. El servidor realiza la mayor parte del trabajo y devuelve la relación resultante al cliente.

Existen diversos tipos de arquitectura cliente/servidor. La más sencilla es aquella en la que solo existe un servidor al que acceden múltiples clientes. A esto lo denominamos "cliente múltiple/servidor único". Desde la perspectiva de la gestión de datos, esto no difiere mucho de las bases de datos centralizadas, ya que la base de datos se almacena en una sola máquina (el servidor) que también aloja el software para gestionarla. Sin embargo, existen algunas diferencias (importantes) con los sistemas centralizados en la forma en que se ejecutan las transacciones y se gestionan las cachés. No consideramos estos aspectos en este momento. Una arquitectura cliente/servidor más sofisticada es aquella en la que existen múltiples servidores en el sistema (el denominado enfoque cliente múltiple/servidor múltiple). En este caso, existen dos estrategias de gestión alternativas: cada cliente gestiona su propia conexión al servidor correspondiente o cada cliente solo conoce su "servidor local", que se comunica con otros servidores según sea necesario. El primer enfoque simplifica el código del servidor, pero impone responsabilidades adicionales Esto da lugar a lo que se ha denominado sistemas de "clientes pesados". Este último enfoque, en

Por otro lado, concentra la funcionalidad de gestión de datos en los servidores. Así, la transparencia del acceso a los datos se proporciona en la interfaz del servidor, lo que da lugar a " clientes ligeros".

Desde una perspectiva datalógica, los SGBD cliente/servidor ofrecen la misma visión de los datos que los sistemas peer-to-peer que analizaremos a continuación. Es decir, ofrecen al usuario la apariencia de una base de datos lógicamente única, mientras que a nivel físico los datos pueden estar distribuidos. Por lo tanto, la principal distinción entre los sistemas cliente/servidor y los peer-to-peer no reside en el nivel de transparencia que se proporciona a los usuarios y las aplicaciones, sino en el paradigma arquitectónico utilizado para lograr dicho nivel de transparencia.

El modelo cliente/servidor puede extenderse de forma natural para proporcionar una distribución de funciones más eficiente en diferentes tipos de servidores: los servidores cliente ejecutan la interfaz de usuario (p. ej., servidores web), los servidores de aplicaciones ejecutan los programas de aplicación y los servidores de bases de datos gestionan las bases de datos. Esto da lugar a la tendencia actual en la arquitectura de sistemas distribuidos de tres niveles, donde los sitios se organizan como servidores especializados en lugar de como ordenadores de propósito general.

La idea original, que consiste en delegar las funciones de gestión de bases de datos a un servidor especial, se remonta a principios de la década de 1970 [Canaday et al., 1974]. En aquel entonces, el ordenador donde se ejecutaba el sistema de base de datos se denominaba máquina de base de datos, ordenador de base de datos o ordenador backend, mientras que el ordenador que ejecutaba las aplicaciones se denominaba ordenador host. Los términos más recientes para estos son servidor de base de datos y servidor de aplicaciones, respectivamente. La Figura 1.12 ilustra una vista simple del enfoque del servidor de base de datos, con servidores de aplicaciones conectados a un servidor de base de datos mediante una red de comunicación.

El enfoque de servidor de bases de datos, como extensión de la arquitectura clásica cliente/ servidor , ofrece varias ventajas potenciales. En primer lugar, el enfoque único en la gestión de datos permite el desarrollo de técnicas específicas para aumentar la fiabilidad y la disponibilidad de los datos, por ejemplo, mediante el paralelismo. En segundo lugar, el rendimiento general de la gestión de bases de datos puede mejorarse significativamente mediante la estrecha integración del sistema de base de datos y un sistema operativo dedicado. Por último, un servidor de bases de datos también puede aprovechar las arquitecturas de hardware más recientes, como los multiprocesadores o los clústeres de servidores de PC, para mejorar tanto el rendimiento como la disponibilidad de los datos.

Si bien estas ventajas son significativas, pueden verse contrarrestadas por la sobrecarga que supone la comunicación adicional entre la aplicación y los servidores de datos. Esto también supone un problema en los sistemas cliente-servidor clásicos, pero en este caso existe una capa adicional de comunicación de la que preocuparse. El coste de la comunicación solo se amortiza si la interfaz del servidor es lo suficientemente avanzada como para permitir la expresión de consultas complejas que impliquen un procesamiento intensivo de datos.

El enfoque de servidor de aplicaciones (de hecho, un enfoque distribuido de n niveles) puede ampliarse mediante la introducción de múltiples servidores de bases de datos y de aplicaciones (Figura 1.13), como ocurre en las arquitecturas cliente-servidor clásicas. En este caso, cada servidor de aplicaciones suele estar dedicado a una o varias aplicaciones, mientras que los servidores de bases de datos operan con el modelo multiservidor descrito anteriormente.

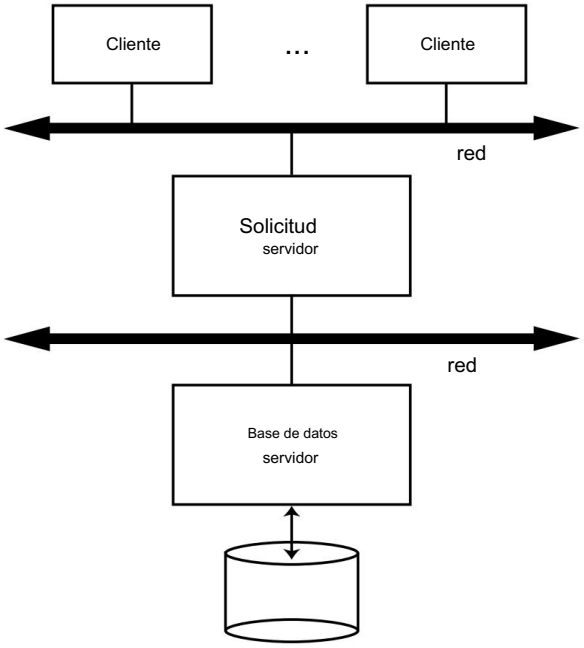

Fig. 1.12 Enfoque del servidor de base de datos

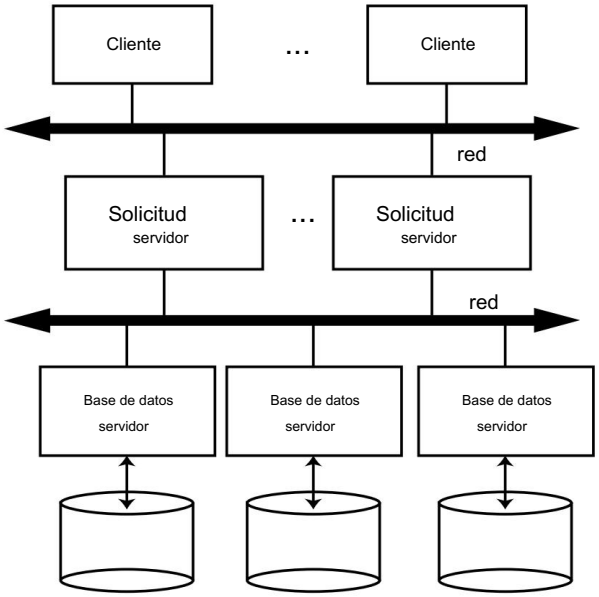

Fig. 1.13 Servidores de bases de datos distribuidas

1.7.9 Sistemas punto a punto

Si el término "cliente/servidor" está plagado de diferentes interpretaciones, "peer-to-peer" es aún peor, ya que su significado ha cambiado y evolucionado con el tiempo. Como se mencionó anteriormente, los primeros trabajos sobre sistemas de gestión de bases de datos distribuidos se centraron en arquitecturas peer-to-peer, donde no existía diferenciación entre la funcionalidad de cada sitio del sistema Tras una década de popularidad de la informática cliente-servidor, los sistemas peer-to-peer han regresado en los últimos años (impulsados principalmente por las aplicaciones de intercambio de archivos) y algunos incluso han posicionado la gestión de datos peer-to-peer como una alternativa a los sistemas de gestión de bases de datos (SGBD) distribuidos. Si bien esto puede parecer exagerado, los sistemas peer-to-peer modernos presentan dos diferencias importantes con respecto a sus predecesores. La primera es la distribución masiva de los sistemas actuales. Mientras que al principio nos centrábamos en unos pocos sitios (quizás decenas como máximo), los sistemas actuales consideran miles de sitios. La segunda es la heterogeneidad inherente de cada aspecto de los sitios y su autonomía. Si bien esto siempre ha sido una preocupación de las bases de datos distribuidas, como se mencionó anteriormente, junto con la distribución masiva, la heterogeneidad y la autonomía de los sitios adquieren una mayor importancia, lo que impide que se consideren algunos enfoques.

Analizar los sistemas de bases de datos peer-to-peer en este contexto plantea verdaderos desafíos ; las particularidades de la gestión de bases de datos en las arquitecturas peer-to-peer modernas aún se están investigando. En este libro, nos centraremos inicialmente en el significado clásico de peer-to-peer (la misma funcionalidad de cada sitio), dado que los principios y las técnicas fundamentales de estos sistemas son muy similares a los de los sistemas cliente-servidor, y abordaremos los problemas de las bases de datos peer-to-peer modernas en un capítulo aparte (Capítulo 16).

Comencemos la descripción de la arquitectura analizando la vista organizativa de los datos. En primer lugar, observamos que la organización física de los datos en cada máquina puede ser, y probablemente sea, diferente. Esto significa que se necesita una definición de esquema interno individual en cada sitio, denominada esquema interno local (LIS). La vista empresarial de los datos se describe mediante el esquema conceptual global (GCS), que es global porque describe la estructura lógica de los datos en todos los sitios.

Para gestionar la fragmentación y replicación de datos, es necesario describir la organización lógica de los datos en cada sitio. Por lo tanto, se requiere una tercera capa en la arquitectura: el esquema conceptual local (ECL). En el modelo arquitectónico elegido , el esquema conceptual global es la unión de los esquemas conceptuales locales. Finalmente, las aplicaciones de usuario y el acceso de los usuarios a la base de datos se sustentan en esquemas externos (ES), definidos como superiores al esquema conceptual global.

Este modelo de arquitectura, representado en la Figura 1.14, proporciona los niveles de transparencia mencionados anteriormente. La independencia de los datos se sustenta, ya que el modelo es una extensión de ANSI/SPARC, que proporciona dicha independencia de forma natural. La transparencia de ubicación y replicación se sustenta en la definición de los esquemas conceptuales local y global y la asignación entre ambos. La transparencia de red, por otro lado, se sustenta en la definición del esquema conceptual global. El usuario

---

[4] De hecho, en la primera edición de este libro que apareció a principios de 1990 y cuya redacción fue Completado en 1989, no hubo una sola mención del término "cliente/servidor".

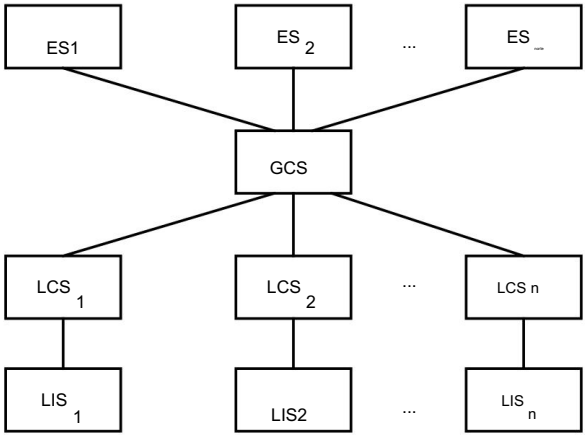

Fig. 1.14 Arquitectura de referencia de base de datos distribuida

Consulta datos independientemente de su ubicación o de qué componente local de la red distribuida
El sistema de base de datos lo atenderá. Como se mencionó anteriormente, el SGBD distribuido traduce
consultas globales en un grupo de consultas locales, que son ejecutadas por DBMS distribuido
componentes en diferentes sitios que se comunican entre sí.

Los componentes detallados de un DBMS distribuido se muestran en la Figura 1.15. Uno
Un componente gestiona la interacción con los usuarios y otro se encarga del almacenamiento.
El primer componente principal, al que llamamos procesador de usuario, consta de cuatro elementos:

1. El controlador de la interfaz de usuario es responsable de interpretar los comandos del usuario como
   Entran y dan formato a los datos del resultado a medida que se envían al usuario.
2. El responsable del tratamiento de datos semánticos utiliza las restricciones de integridad y las autorizaciones
   que se definen como parte del esquema conceptual global para comprobar si el usuario
   Se puede procesar la consulta. Este componente, que se estudia en detalle en el Capítulo
   5, también es responsable de la autorización y otras funciones.
3. El optimizador y descomponedor de consultas globales determina una estrategia de ejecución
   para minimizar una función de costo y traduce las consultas globales en locales
   utilizando los esquemas conceptuales globales y locales, así como el directorio global.
   El optimizador de consultas global es responsable, entre otras cosas, de generar
   La mejor estrategia para ejecutar operaciones de unión distribuidas. Estos problemas son...
   discutido en los capítulos 6 al 8.
4. El monitor de ejecución distribuida coordina la ejecución distribuida de la
   Solicitud del usuario. El monitor de ejecución también se denomina monitor de transacciones distribuidas.
   gerente. Al ejecutar consultas de forma distribuida, los monitores de ejecución
   en varios sitios pueden, y generalmente lo hacen, comunicarse entre sí.

El segundo componente principal de un DBMS distribuido es el procesador de datos y
consta de tres elementos:

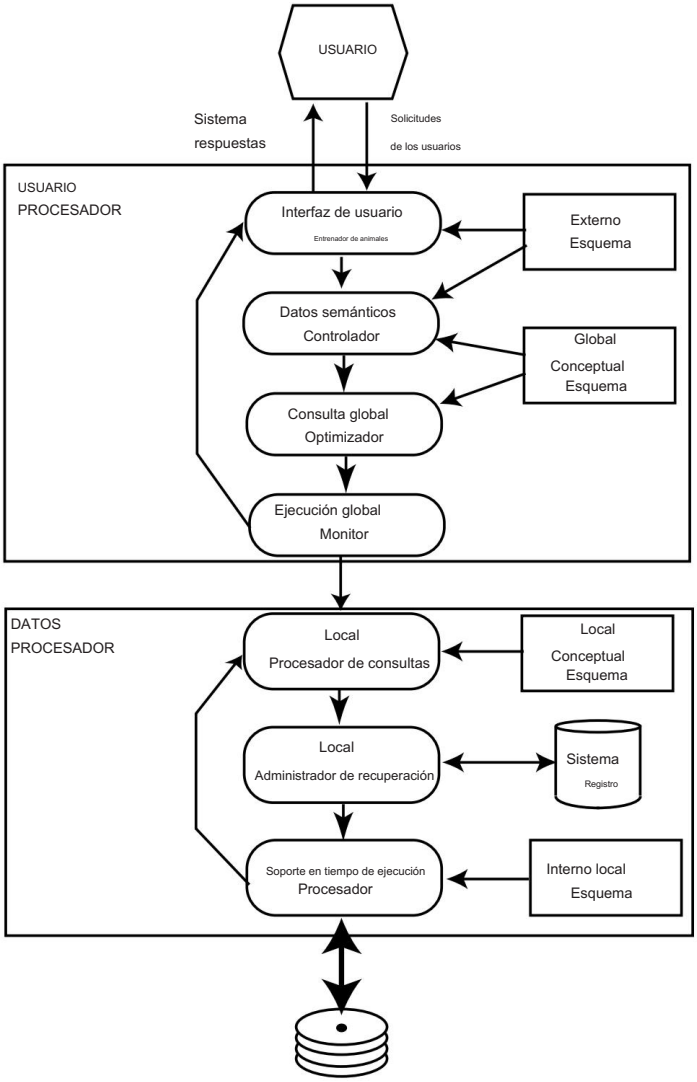

Fig. 1.15 Componentes de un DBMS distribuido

1. El optimizador de consultas local, que en realidad actúa como selector de ruta de acceso, es responsable de elegir la mejor ruta de acceso5 para acceder a cualquier elemento de datos (que se aborda brevemente en el Capítulo 8).
2. El administrador de recuperación local es responsable de garantizar que la base de datos local permanezca consistente incluso cuando ocurren fallas (Capítulo 12).
3. El procesador de soporte en tiempo de ejecución accede físicamente a la base de datos de acuerdo con los comandos físicos en la programación generada por el optimizador de consultas. El procesador de soporte de tiempo de ejecución es la interfaz con el sistema operativo y contiene el administrador del búfer de base de datos (o caché), que es responsable de mantener los búferes de memoria principal y administrar los accesos a los datos.

Es importante señalar, en este punto, que el uso de los términos "procesador de usuario" y "procesador de datos" no implica una división funcional similar a la de los sistemas cliente-servidor. Estas divisiones son meramente organizativas y no se sugiere que deban ubicarse en máquinas diferentes. En los sistemas peer-to-peer, se espera encontrar tanto los módulos de procesador de usuario como los de procesador de datos en cada máquina. Sin embargo, se ha sugerido separar los "sitios de solo consulta" de los de funcionalidad completa. En este caso, los primeros solo necesitarían tener el procesador de usuario.

En sistemas cliente/servidor con un solo servidor, el cliente gestiona el gestor de interfaz de usuario, mientras que el servidor gestiona toda la funcionalidad del procesador de datos, así como el controlador semántico de datos; no se requieren el optimizador de consultas global ni el monitor de ejecución global. Si existen varios servidores y se emplea el enfoque de servidor local descrito en la sección anterior, cada servidor aloja todos los módulos excepto el gestor de interfaz de usuario, que reside en el cliente. Sin embargo, si se espera que cada cliente contacte con servidores individuales por su cuenta, lo más probable es que los clientes alojen toda la funcionalidad del procesador de usuario, mientras que la funcionalidad del procesador de datos reside en los servidores.

1.7.10 Arquitectura del sistema multibase de datos

Los sistemas multibase de datos (MDBS) representan el caso en el que los SGBD individuales (distribuidos o no) son completamente autónomos y no tienen el concepto de cooperación; es posible que ni siquiera sepan de la existencia de los demás ni cómo comunicarse entre sí. Nos centraremos, naturalmente, en los MDBS distribuidos, a los que se referirá el término en adelante. En la mayor parte de la literatura actual se utiliza el término sistema de integración de datos. Evitamos usar ese término, ya que los sistemas de integración de datos también consideran fuentes de datos no basadas en bases de datos. Nos centramos estrictamente en las bases de datos. Analizamos estos sistemas y su relación con la integración de bases de datos en el capítulo 4. Sin embargo, observamos que existe una considerable variabilidad en el uso del término "multibase de datos" en la literatura. En este...

---

[5] El término "ruta de acceso" se refiere a las estructuras de datos y los algoritmos utilizados para acceder a ellos. Una ruta de acceso típica, por ejemplo, es un índice en uno o más atributos de una relación.

libro, lo utilizamos consistentemente como se define arriba, lo que puede desviarse de su uso en parte de la literatura existente.

Las diferencias en el nivel de autonomía entre los SGBD multidistribuidos y los SGBD distribuidos también se reflejan en sus modelos arquitectónicos. La diferencia fundamental se relaciona con la definición del esquema conceptual global. En el caso de los SGBD distribuidos lógicamente integrados, el esquema conceptual global define la vista conceptual de toda la base de datos, mientras que en el caso de los SGBD multidistribuidos, representa solo la colección de algunas de las bases de datos locales que cada SGBD local desea compartir. Los SGBD individuales pueden optar por hacer que algunos de sus datos estén disponibles para el acceso de otros (es decir, arquitecturas de bases de datos federadas) definiendo un esquema de exportación [Heimbigner y McLeod, 1985]. Por lo tanto, la definición de una base de datos global es diferente en los SGBD que en los SGBD distribuidos. En estos últimos, la base de datos global es igual a la unión de las bases de datos locales, mientras que en los primeros es solo un subconjunto (posiblemente propio) de la misma unión. En un MDBS, el GCS (que también se denomina esquema mediado) se define integrando los esquemas externos de bases de datos autónomas locales o (posiblemente partes de sus) esquemas conceptuales locales.

Además, los usuarios de un SGBD local definen sus propias vistas de la base de datos local y no necesitan modificar sus aplicaciones si no desean acceder a los datos de otra base de datos. Esto, nuevamente, es una cuestión de autonomía.

El diseño del esquema conceptual global en sistemas multibase de datos implica la integración de los esquemas conceptuales locales o los esquemas externos locales (Figura 1.16). Una diferencia importante entre el diseño del GCS en sistemas multi-DBMS y en sistemas distribuidos con integración lógica radica en que, en el primero, la correspondencia se realiza desde los esquemas conceptuales locales a un esquema global. En el segundo, sin embargo, la correspondencia se realiza en sentido inverso. Como se explica en los capítulos 3 y 4, esto se debe a que el diseño en el primero suele ser un proceso ascendente, mientras que en el segundo suele ser descendente. Además, si existe heterogeneidad en el sistema multibase de datos, se debe encontrar un modelo de datos canónico para definir el GCS.

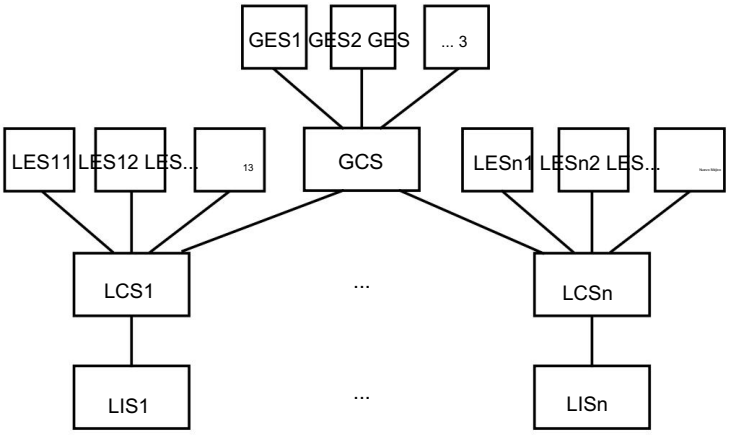

Fig. 1.16 Arquitectura MDBS con un GCS

Una vez diseñado el GCS, se pueden definir vistas del esquema global para los usuarios que requieren acceso global. No es necesario que el GES y el GCS se definan utilizando el mismo modelo de datos y lenguaje; su uso determina si el sistema es homogéneo o heterogéneo.

Si existe heterogeneidad en el sistema, existen dos alternativas de implementación: unilingüe y multilingüe. Un SGBD multilingüe requiere que los usuarios utilicen modelos de datos e idiomas posiblemente diferentes al acceder tanto a la base de datos local como a la global. La característica distintiva de los sistemas unilingües es que cualquier aplicación que acceda a datos de múltiples bases de datos debe hacerlo mediante una vista externa definida en el esquema conceptual global. Esto significa que el usuario de la base de datos global es, en efecto, un usuario diferente de quienes acceden solo a la base de datos local, utilizando un modelo de datos y un idioma de datos distintos.

Una alternativa es la arquitectura multilingüe, cuya filosofía básica es permitir que cada usuario acceda a la base de datos global (es decir, a los datos de otras bases de datos) mediante un esquema externo, definido en el lenguaje del SGBD local del usuario. La definición de GCS es bastante similar en la arquitectura multilingüe y en el enfoque unilingüe; la principal diferencia reside en la definición de los esquemas externos, que se describen en el lenguaje de los esquemas externos de la base de datos local. Suponiendo que la definición es puramente local, una consulta emitida según un esquema particular se gestiona exactamente como cualquier consulta en los SGBD centralizados. Las consultas a la base de datos global se realizan en el lenguaje del SGBD local, pero generalmente requieren cierto procesamiento para su mapeo al esquema conceptual global.

El modelo arquitectónico basado en componentes de un sistema multi-DBMS difiere significativamente de un sistema distribuido. La diferencia fundamental radica en la existencia de sistemas completos de gestión de bases de datos (DBMS), cada uno de los cuales gestiona una base de datos diferente. El sistema MDBS proporciona una capa de software que se ejecuta sobre estos sistemas individuales y proporciona a los usuarios la posibilidad de acceder a diversas bases de datos (Figura 1.17). Cabe destacar que, en un sistema MDBS distribuido, la capa multi-DBMS puede ejecutarse en varios sitios o puede haber un sitio central donde se ofrecen dichos servicios. Cabe destacar también que, en lo que respecta a los DBMS individuales , la capa MDBS es simplemente otra aplicación que envía solicitudes y recibe respuestas.

Una arquitectura de implementación popular para MDBSs es el enfoque de mediador/envoltorio (Figura 1.18) [Wiederhold, 1992]. Un mediador “es un módulo de software que explota el conocimiento codificado sobre ciertos conjuntos o subconjuntos de datos para crear información para una capa superior de aplicaciones”. Por lo tanto, cada mediador realiza una función particular con interfaces claramente definidas. Usando esta arquitectura para implementar un MDBS, cada módulo en la capa multi-DBMS de la Figura 1.17 se realiza como un mediador. Dado que los mediadores se pueden construir sobre otros mediadores, es posible construir una implementación en capas. Al mapear esta arquitectura a la vista datalogical de la Figura 1.16, el nivel de mediador implementa el GCS. Es este nivel el que maneja las consultas del usuario sobre el GCS y realiza la funcionalidad MDBS.

Los mediadores suelen operar utilizando un modelo de datos y un lenguaje de interfaz comunes. Para abordar las posibles heterogeneidades de los SGBD de origen, se implementan envoltorios cuya función es proporcionar una correspondencia entre la vista del SGBD de origen y la vista de los mediadores. Por ejemplo, si el SGBD de origen es relacional, pero...

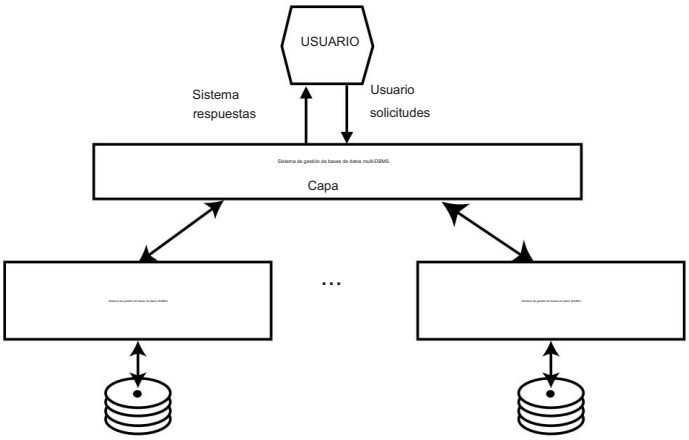

Figura 1.17 Componentes de un MDBS

Las implementaciones del mediador están orientadas a objetos, se establecen las asignaciones requeridas por los envoltorios. El rol y la función exactos de los mediadores difieren de una implementación a otra. En algunos casos, se han implementado mediadores delgados que no nada más que traducción. En otros casos, los wrappers se encargan de la ejecución de Algunas de las funciones de consulta.

Se puede ver la colección de mediadores como una capa de middleware que proporciona servicios por encima de los sistemas de origen. El middleware es un tema que ha sido objeto de debate. Se han realizado estudios importantes en la última década y se han desarrollado sistemas de middleware muy sofisticados. Se han desarrollado herramientas que proporcionan servicios avanzados para el desarrollo de aplicaciones distribuidas . Los mediadores que analizamos solo representan un subconjunto de la funcionalidad. proporcionadas por estos sistemas.

1.8 Notas bibliográficas

No existen muchos libros sobre sistemas de gestión de bases de datos distribuidos. El libro de Ceri y Pelagatti [Ceri y Pelagatti, 1983] fue el primero en este tema, aunque ahora está anticuado. El libro de Bell y Grimson [Bell y Grimson, 1992] también proporciona una descripción general de la temas abordados aquí. Además, casi todos los libros sobre bases de datos ahora tienen un capítulo sobre Sistemas de gestión de bases de datos distribuidos. Se ofrece una breve descripción general de la tecnología en [Ozsu y Valduriez, 1997]. Nuestros artículos [Ozsu y Valduriez, 1994, 1991] ofrecen análisis. del estado del arte en el momento en que fueron escritos.

El diseño de bases de datos se analiza de manera introductoria en [Levin y Morgan, 1975] y de forma más completa en [Ceri et al., 1987]. Se ofrece un estudio de los algoritmos de distribución de archivos en [Dowdy y Foster, 1982]. La gestión de directorios no ha Se han considerado en detalle en la comunidad de investigación, pero se pueden utilizar técnicas generales. encontrado en Chu y Nahouraii [1975] y [Chu, 1976]. Un estudio del procesamiento de consultas

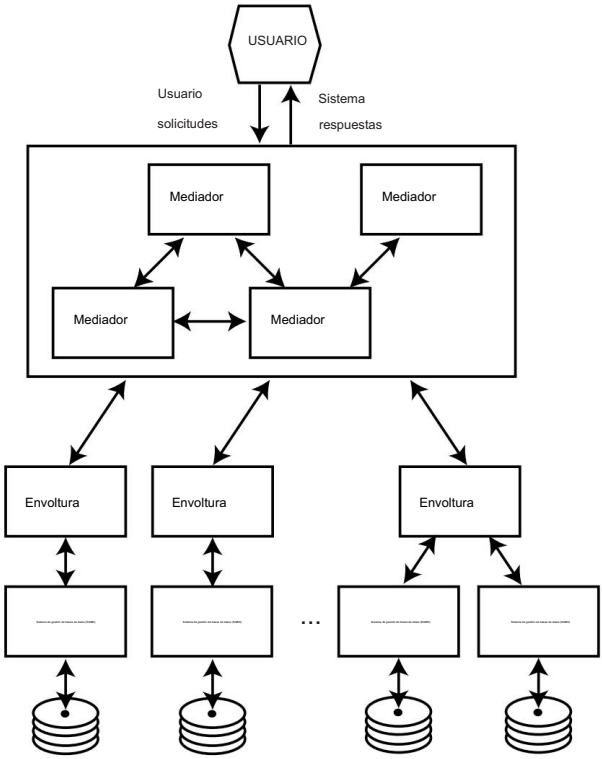

Fig. 1.18 Arquitectura del mediador/envoltorio

Las técnicas se pueden encontrar en [Sacco y Yao, 1982]. Algoritmos de control de concurrencia se revisan en [Bernstein y Goodman, 1981] y [Bernstein et al., 1987]. La gestión de bloqueos también ha sido objeto de una amplia investigación; una introducción El artículo es [Isloor y Marsland, 1980] y un artículo ampliamente citado es [Obermarck, 1982]. Para la detección de bloqueos, buenas encuestas son [Knapp, 1987] y [Elmagarmid, 1986]. La confiabilidad es uno de los temas discutidos en [Gray, 1979], que es uno de los Artículos fundamentales en el campo. Otros artículos importantes sobre este tema son [Verhofstadt, 1978] y [Harder¨y Reuter, 1983] . [Gray, 1979] es también el primer artículo que analiza las cuestiones de compatibilidad del sistema operativo con bases de datos distribuidas; el mismo tema es abordado en [Stonebraker, 1981]. Desafortunadamente, ambos artículos enfatizan la centralización sistemas de bases de datos.

Se han presentado varias propuestas de marcos arquitectónicos. Algunas de las interesantes incluyen la extensión bastante detallada de Schreiber del marco ANSI/ SPARC, que intenta acomodar la heterogeneidad de los modelos de datos [Schreiber, 1977], y la propuesta de Mohan y Yeh [Mohan y Yeh, 1978]. Como era de esperar, Estos se remontan a los primeros días de la introducción de la tecnología DBMS distribuida. La arquitectura detallada del sistema por componentes que se muestra en la Figura 1.15 se deriva de

[Rahimi, 1987]. Una alternativa a la clasificación que ofrecemos en la Figura 1.10 se puede encontrar en [Sheth y Larson, 1990].

La mayor parte del análisis sobre modelos arquitectónicos para sistemas de gestión de bases de datos (SGBD) multi-multi-DBMS proviene de [Ozsu y Barker, 1990]. Otros análisis arquitectónicos sobre sistemas SGBD multi-multi-DBMS se encuentran en [Gligor y Luckenbaugh, 1984], [Litwin, 1988] y [Sheth y Larson, 1990]. Todos estos artículos ofrecen análisis generales de diversos prototipos y sistemas comerciales. Un excelente resumen de los sistemas de bases de datos heterogéneos y federados se encuentra en [Sheth y Larson, 1990].

Capítulo 2
Fondo

Como se indicó en el capítulo anterior, existen dos bases tecnológicas para la tecnología de bases de datos distribuidas: la gestión de bases de datos y las redes de computadoras. En este capítulo, presentamos una visión general de los conceptos de estos dos campos que son más importantes desde la perspectiva de la tecnología de bases de datos distribuidas.

2.1 Descripción general de los DBMS relacionales

El objetivo de esta sección es definir la terminología y el marco de trabajo que se utilizarán en los capítulos posteriores, ya que la mayor parte de la tecnología de bases de datos distribuidas se ha desarrollado utilizando el modelo relacional. En capítulos posteriores, cuando corresponda, se presentarán otros modelos. Aquí nos centraremos en el lenguaje y los operadores.

2.1.1 Conceptos de bases de datos relacionales

Una base de datos es una colección estructurada de datos relacionados con fenómenos reales que intentamos modelar. Una base de datos relacional es aquella cuya estructura se presenta en forma de tablas. Formalmente, una relación R definida sobre n conjuntos D1, D2,..., Dn (no necesariamente distintos) es un conjunto de n-tuplas (o simplemente tuplas) d1, d2,..., dn tales que d1 D1, d2 D2,..., dn Dn.

Ejemplo 2.1. Como ejemplo, utilizamos una base de datos que modela una empresa de ingeniería. Las entidades que se modelarán son los empleados (EMP) y los proyectos (PROJ). Para cada empleado, se registrará su número (ENO), nombre (ENAME), cargo en la empresa (TITLE), salario (SAL), número de identificación del proyecto (PNO), responsabilidad dentro del proyecto (RESP) y duración de la asignación al proyecto (DUR) en meses. Asimismo, para cada proyecto, se registrará su número (PNO), nombre (PNAME) y presupuesto (BUDGET).

MT Özsu y P. Valduriez, Principios de sistemas de bases de datos distribuidas: Tercera edición,
DOI 10.1007/978-1-4419-8834-8_2, © Springer Science+Business Media, LLC 2011

Fig. 2.1 Esquema de base de datos de muestra

Los esquemas de relación para esta base de datos se pueden definir de la siguiente manera:

EMP(ENO, ENAME, TÍTULO, SAL, PNO, RESP, DUR)

PROY(PNO,PNAME, PRESUPUESTO)

En el esquema de relación EMP, existen siete atributos: ENO, ENAME, TITLE, SAL, PNO, RESP, DUR. Los valores de ENO provienen del dominio de todos los números de empleado válidos (por ejemplo, D1); los valores de ENAME, del dominio de todos los nombres válidos (por ejemplo , D2); y así sucesivamente. Cabe destacar que cada atributo de cada relación no tiene que provenir de un dominio distinto. Se pueden definir varios atributos dentro de una relación o de varias relaciones en el mismo dominio.

La clave de un esquema de relación es el subconjunto mínimo no vacío de sus atributos, de modo que los valores de los atributos que la componen identifiquen de forma única cada tupla de la relación. Los atributos que conforman la clave se denominan atributos primos. El superconjunto de una clave se suele denominar superclave. Así, en nuestro ejemplo, la clave de PROJ es PNO y la de EMP es el conjunto (ENO, PNO). Cada relación tiene al menos una clave. A veces, puede haber más de una posibilidad para la clave. En tales casos, cada alternativa se considera una clave candidata, y una de ellas se elige como clave principal, lo cual se indica mediante un subrayado. El número de atributos de una relación define su grado, mientras que el número de tuplas de la relación define su cardinalidad.

En formato tabular, la base de datos de ejemplo consta de dos tablas, como se muestra en la Figura 2.1. Las columnas de las tablas corresponden a los atributos de las relaciones; si se hubiera ingresado información como filas, estas corresponderían a las tuplas. La tabla vacía, que muestra la estructura de la tabla, corresponde al esquema de la relación; cuando la tabla se llena con filas, corresponde a una instancia de la relación. Dado que la información dentro de una tabla varía con el tiempo, se pueden generar múltiples instancias a partir de un esquema de relación. Tenga en cuenta que, de ahora en adelante, el término "relación" se refiere a una instancia de la relación. En la Figura 2.2 se muestran instancias de las dos relaciones definidas en la Figura 2.1.

Un valor de atributo puede ser indefinido. Esta falta de definición puede tener diversas interpretaciones, siendo las más comunes «desconocido» o «no aplicable». Este valor especial del atributo se conoce generalmente como valor nulo. La representación de un valor nulo debe ser diferente de la de cualquier otro valor del dominio, y se debe prestar especial atención para diferenciarlo de cero. Por ejemplo, el valor «0» para el atributo DUR es

Pulso electromagnético

ENO	TÍTULO DEL	ENNAME	SAL	PNO	RESP	DUR
E1	J. Doe,	Ingeniero Electrónico	40000		Gerente P1	12
E2	M. Smith	Analista	34000		Analista P1	24
E2	Señor Smith	Analista	34000		Analista P2	6
E3	A. Lee	Mech. Ing.	27000		Consultor P3	10
E3	A. Lee	Ingeniería Mecánica	27000		Ingeniero P4	48
E4	J. Miller	Programador	24000		Programador P2	18
E5	B. Casey	Anal. Sist.	34000		Gerente de P2	24
E6	L. Chu	Ing. Elect.	40000		Gerente P4	48
E7	R. Davis,	Ingeniero Mecánico	27000		Ingeniero P3	36
E8	J. Jones	Syst. Anal.	34000		Gerente de P3	40

PROYECTO

PNO	Nombre de usuario	PRESUPUESTO
P1	Instrumentación	150000
P2	Desarrollo de base de datos	135000
P3	CAD/CAM	250000
P4	Mantenimiento	310000

Fig. 2.2 Instancia de base de datos de muestra

información conocida (por ejemplo, en el caso de un empleado recién contratado), mientras que el valor es “nulo” Para DUR, significa desconocido. Admitir valores nulos es una característica importante y necesaria. para tratar posibles consultas [Codd, 1979].

2.1.2 Normalización

El objetivo de la normalización es eliminar diversas anomalías (o aspectos indeseables) de una relación para obtener relaciones “mejores”. Los siguientes cuatro problemas podrían existir en un esquema de relación:

1. Anomalía de repetición. Cierta información puede repetirse innecesariamente. Considere , por ejemplo, la relación EMP en la Figura 2.2. El nombre, el cargo y el salario de un empleado se repiten para cada proyecto en el que trabaja esta persona. Esto Obviamente es un desperdicio de almacenamiento y es contrario al espíritu de las bases de datos.

2. Anomalía en la actualización. Debido a la repetición de datos, realizar actualizaciones puede ser problemático. Por ejemplo, si cambia el salario de un empleado, es necesario actualizar varias tuplas para reflejar este cambio.
3. Anomalía de inserción. Es posible que no se pueda añadir nueva información a la base de datos. Por ejemplo, cuando un nuevo empleado se incorpora a la empresa, no podemos añadir información personal (nombre, cargo, salario) a la relación EMP a menos que se le asigne un puesto en un proyecto. Esto se debe a que la clave de EMP incluye el atributo PNO y los valores nulos no pueden formar parte de la clave.
4. Anomalía de eliminación. Es lo opuesto a la anomalía de inserción. Si un empleado trabaja en un solo proyecto y este se cancela, no es posible eliminar la información del proyecto de la relación EMP. Hacerlo resultaría en la eliminación de la única tupla del empleado, lo que resultaría en la pérdida de información personal que podríamos querer conservar.

La normalización transforma esquemas de relación arbitrarios en otros sin estos problemas. Una relación con una o más de las anomalías mencionadas se divide en dos o más relaciones de una forma normal superior. Se dice que una relación está en forma normal si cumple las condiciones asociadas a dicha forma normal. Codd definió inicialmente la primera, segunda y tercera forma normal (1NF, 2NF y 3NF, respectivamente). Posteriormente, Boyce y Codd [Codd, 1974] definieron una versión modificada de la tercera forma normal, comúnmente conocida como forma normal de Boyce-Codd (FNBC). A esta le siguieron las definiciones de la cuarta (4FN) [Fagin, 1977] y la quinta (5FN). [Fagin, 1979].

Las formas normales se basan en ciertas estructuras de dependencia. La FNBC y las formas normales inferiores se basan en dependencias funcionales (FD), la FN4 se basa en dependencias multivaluadas y la FN5 se basa en dependencias de proyección-unión. Solo introducimos la dependencia funcional, ya que es la única relevante para el ejemplo que estamos considerando.

Sea R una relación definida sobre el conjunto de atributos A = {A1,A2,...,An} y sea X A, Y A. Si para cada valor de X en R, solo hay un valor de Y asociado , decimos que “X determina funcionalmente a Y” o que “Y es funcionalmente dependiente de X”.
Notacionalmente, esto se muestra como X →Y. La clave de una relación determina funcionalmente los atributos no clave de la misma relación.

Ejemplo 2.2. Por ejemplo, en la relación PROJ del Ejemplo 2.1 (también se puede observar en la Figura 2.2), la FD válida es

PNO → (PNAME, PRESUPUESTO)

En la relación EMP tenemos

(ENO, PNO) → (ENOMBRE,TÍTULO,SAL,RESP,DUR)

Sin embargo, este último FD no es el único FD en EMP. Si a cada empleado se le asigna un único números de empleados, podemos escribir

ENO → (NOMBRE, TÍTULO, SAL)

(ENO, PNO) → (RESP, DUR)

También puede ocurrir que el salario para un puesto determinado sea fijo, lo que da lugar a la FD.

TÍTULO → SAL

No analizamos en detalle las formas normales ni los algoritmos de normalización; estos se pueden encontrar en libros de texto sobre bases de datos. El siguiente ejemplo muestra el resultado de la normalización en la base de datos de ejemplo que presentamos en el Ejemplo 2.1.

Ejemplo 2.3. El siguiente conjunto de esquemas de relación se normaliza en BCNF con respecto a las dependencias funcionales definidas sobre las relaciones.

EMP(ENO, ENAME, TÍTULO)

PAGAR(TÍTULO, SALARIO)

PROY(PNO, PNAME, PRESUPUESTO)

ASG(ENO, PNO, RESP, DUR)

Las instancias normalizadas de estas relaciones se muestran en la Figura 2.3.

2.1.3 Lenguajes de datos relacionales

Los lenguajes de manipulación de datos desarrollados para el modelo relacional (comúnmente llamados lenguajes de consulta) se dividen en dos grupos fundamentales: lenguajes de álgebra relacional y lenguajes de cálculo relacional. La diferencia entre ellos radica en cómo se formula la consulta del usuario. El álgebra relacional es procedimental, ya que se espera que el usuario especifique, mediante ciertos operadores de alto nivel, cómo obtener el resultado. El cálculo relacional, por otro lado, no es procedimental; el usuario solo especifica las relaciones que deben existir en el resultado. Ambos lenguajes fueron propuestos originalmente por Codd [1970], quien también demostró su equivalencia en términos de capacidad expresiva [Codd, 1972].

2.1.3.1 Álgebra relacional

El álgebra relacional consiste en un conjunto de operadores que operan sobre relaciones. Cada operador toma una o dos relaciones como operandos y produce una relación resultante, que, a su vez, puede ser operando de otro operador. Estas operaciones permiten consultar y actualizar una base de datos relacional.

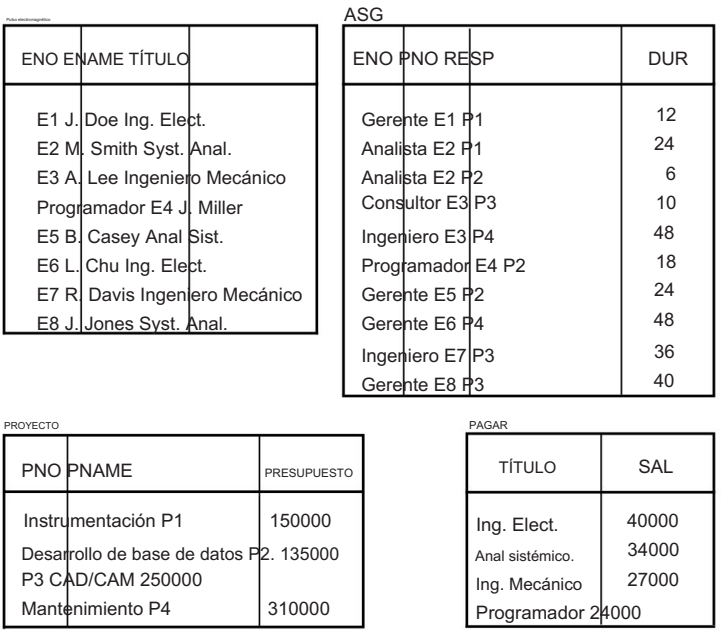

Fig. 2.3 Relaciones normalizadas

Hay cinco operadores fundamentales del álgebra relacional y otros cinco que pueden utilizarse. definido en términos de estos. Los operadores fundamentales son selección, proyección, unión, Diferencia de conjuntos y producto cartesiano. Los dos primeros operadores son unarios. operadores, y los tres últimos son operadores binarios. Los operadores adicionales que pueden ser definidos en términos de estos operadores fundamentales son intersección, θ − unión, natural Unión, semiunión y división. En la práctica, el álgebra relacional se extiende con operadores. para agrupar u ordenar los resultados, y para realizar operaciones aritméticas y agregadas funciones. Otros operadores, como la unión externa y el cierre transitivo, a veces son También se utiliza para proporcionar funcionalidad adicional. Solo analizamos las más comunes. operadores.

Los operandos de algunas de las relaciones binarias deben ser compatibles con la unión. Dos Las relaciones R y S son compatibles con la unión si y sólo si son del mismo grado y el i-ésimo atributo de cada uno se define sobre el mismo dominio. La segunda parte de La definición es válida, obviamente, sólo cuando se identifican los atributos de una relación. por sus posiciones relativas dentro de la relación y no por sus nombres. Si son relativos El orden de los atributos no es importante, es necesario reemplazar la segunda parte del definición por la frase “los atributos correspondientes de las dos relaciones deben ser definida sobre el mismo dominio”. La correspondencia se define aquí de forma bastante vaga.

Muchas definiciones de operadores se refieren a “fórmula”, que también aparece en lenguaje relacional. Expresiones de cálculo que analizaremos más adelante. Por lo tanto, definamos con precisión, en este punto, qué Nos referimos a una fórmula. Definimos una fórmula en el contexto del predicado de primer orden.

cálculo (ya que utilizamos ese formalismo más adelante) y seguimos la notación de Gallaire et al. [1984]. El cálculo de predicados de primer orden se basa en un alfabeto de símbolos que consta de (1) variables, constantes, funciones y símbolos de predicado; (2) paréntesis; (3) los conectores lógicos (y), (o), ¬ (no), → (implicación) y ↔ (equivalencia); y (4) los cuantificadores (para todo) y (existe). Un término es una constante o una variable. Recursivamente, si f es una función n-aria y t1,...,tn son términos, f(t1,...,tn) también es un término. Una fórmula atómica tiene la forma P(t1,...,tn), donde P es un símbolo de predicado n-ario y los t1 son términos. Una fórmula bien formada (fbf) se puede definir recursivamente de la siguiente manera: Si wi y wj son fbf, entonces (wi), ¬(wi),(wi) (wj),(wi) (wj),(wi) → (wj) y (wi) ↔ (wj) son todas fbf. Las variables en una fbf pueden ser libres o estar limitadas por uno de los dos cuantificadores.

Selección.

La selección produce un subconjunto horizontal de una relación dada. El subconjunto consta de todas las tuplas que satisfacen una fórmula (condición). La selección de una relación R es

σF(R)

donde R es la relación y F es una fórmula.

La fórmula de la operación de selección se denomina predicado de selección y es una fórmula atómica cuyos términos tienen la forma AθC, donde A es un atributo de R y θ es uno de los operadores de comparación aritmética <, >, =, =, ≤ y ≥. Los términos se pueden conectar mediante los conectores lógicos , y ¬. Además, el predicado de selección no contiene cuantificadores.

Ejemplo 2.4. Considere la relación EMP que se muestra en la Figura 2.3. El resultado de seleccionar estas tuplas para ingenieros eléctricos se muestra en la Figura 2.4.

σTÍTULO="Ingeniería Eléctrica" (EMP)

ENO	ENAME	TÍTULO
E1	J. Doe	Ing. Elect.
E6	L. Chu	Ing. Elect.

Fig. 2.4 Resultado de la selección

Proyección.

La proyección produce un subconjunto vertical de una relación. La relación resultante contiene únicamente los atributos de la relación original sobre la que se realiza la proyección. Por lo tanto, el grado del resultado es menor o igual que el grado de la relación original.

La proyección de la relación R sobre los atributos A y B se denota como

ΠA,B(R)

Tenga en cuenta que el resultado de una proyección puede contener tuplas idénticas. En ese caso, las tuplas duplicadas pueden eliminarse de la relación resultante. Es posible especificar la proyección con o sin eliminación de duplicados.

Ejemplo 2.5. La proyección de la relación PROJ, mostrada en la Figura 2.3, sobre los atributos PNO y BUDGET se muestra en la Figura 2.5.

ΠPNO,PRESUPUESTO (PROY)

PNO	PRESUPUESTO
P1	150000
P2	135000
P3	250000
P4	310000

Fig. 2.5 Resultado de la proyección

Unión.

La unión de dos relaciones R y S (denotada como R S) es el conjunto de todas las tuplas que se encuentran en R, S o ambas. Cabe destacar que R y S deben ser compatibles con la unión. Al igual que en el caso de la proyección, las tuplas duplicadas se eliminan normalmente. La unión puede utilizarse para insertar nuevas tuplas en una relación existente, donde estas tuplas forman una de las relaciones de operandos.

Diferencia de conjuntos.

La diferencia de conjuntos de dos relaciones R y S (R− S) es el conjunto de todas las tuplas que están en R pero no en S. En este caso, no solo R y S deberían ser compatibles en la unión, sino que la operación también es asimétrica (es decir, R − S = S − R). Esta operación permite

EMP x PAY

ENO	ENAME	EMP.TÍTULO	PAGAR.TÍTULO	SAL
E1	J. Doe	Ing. Elect.	Ing . Elect .	40000
E1	J. Doe	Ing. Elect.	Anal sistémico.	34000
E1	J. Doe	Ing. Elect.	Ing. Mecánico	27000
E1	J. Doe	Ing. Elect.	Programador	24000
E2	Señor Smith	Anal sistémico.	Ingeniería Eléctrica	40000
E2	Señor Smith	Anal sistémico.	Anal sistémico.	34000
E2	Señor Smith	Anal sistémico.	Ing. Mecánico	27000
E2	Señor Smith	Anal sistémico.	Programador	24000
E3	A. Lee	Ing. Mecánico	Ing. Elect.	40000
E3	A. Lee	Ing. Mecánico	Anal sistémico.	34000
E3	A. Lee	Ing. Mecánico	Ing. Mecánico	27000
E3	A. Lee	Ing. Mecánico	Programador	24000
≈	≈	≈≈	≈	≈
E8	J. Jones	Anal sistémico.	Ing. Elect.	40000
E8	J. Jones	Anal sistémico.	Anal sistémico.	34000
E8	J. Jones	Anal sistémico.	Ing. Mecánico	27000
E8	J. Jones	Anal sistémico.	Programador	24000

Fig. 2.6 Resultado parcial del producto cartesiano

Eliminación de tuplas de una relación. Junto con la operación de unión, podemos realizar modificación de tuplas mediante eliminación seguida de inserción.

Producto cartesiano.

El producto cartesiano de dos relaciones R de grado k1 y S de grado k2 es el conjunto de (k1 + k2)-tuplas, donde cada tupla resultante es una concatenación de una tupla de R con una tupla de S, para todas las tuplas de R y S. El producto cartesiano de R y S se denota como R×S.

Es posible que las dos relaciones tengan atributos con el mismo nombre. En En este caso, los nombres de los atributos se anteponen con el nombre de la relación para mantener la unicidad de los nombres de atributos dentro de una relación.

Ejemplo 2.6. Considere las relaciones EMP y PAY en la Figura 2.3. Se muestra EMP × PAY. en la Figura 2.6. Nótese que el atributo TÍTULO, que es común a ambas relaciones, aparece dos veces, precedido por el nombre de la relación.

Intersección.

La intersección de dos relaciones R y S (R ∩ S) consiste en el conjunto de todas las tuplas que están tanto en R como en S. En términos de los operadores básicos, se puede especificar de la siguiente manera:

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

Unión θ.

La unión es un derivado del producto cartesiano. Existen varias formas de unión; la principal clasificación es entre unión interna y unión externa. Primero, analizaremos la unión interna y sus variantes, y luego describiremos la unión externa.

El tipo más general de unión interna es la unión θ. La unión θ de dos relaciones R y S se denota como

$$R \quad F \; S$$

Donde F es una fórmula que especifica el predicado de unión. Un predicado de unión se especifica de forma similar a un predicado de selección, excepto que los términos tienen la forma R.AθS.B, donde A y B son atributos de R y S, respectivamente.

La unión de dos relaciones equivale a realizar una selección, utilizando el predicado de unión como fórmula de selección, sobre el producto cartesiano de las dos relaciones de operandos. Por lo tanto

$$R \quad F \; S = \sigma F(R \times S)$$

En la equivalencia anterior, debemos tener en cuenta que si F involucra atributos de las dos relaciones que son comunes a ambas, es necesaria una proyección para asegurarse de que esos atributos no aparezcan dos veces en el resultado.

Ejemplo 2.7. Consideremos la relación EMP de la Figura 2.3 y añadamos dos tuplas más, como se muestra en la Figura 2.7(a). La Figura 2.7(b) muestra la unión θ de las relaciones EMP y ASG sobre el predicado de unión EMP.ENO=ASG.ENO.

Se podría haber obtenido el mismo resultado si

$$EMP \quad EMP.ENO=ASG.ENO \; ASG = \Pi ENO, ENOMBRE, TÍTULO, SAL(\sigma EMP.ENO = PAGO.ENO(EMP \times ASG))$$

Observe que el resultado no incluye las tuplas E9 y E10, ya que estos empleados aún no han sido asignados a ningún proyecto. Además, la información sobre algunos empleados (p. ej., E2 y E3) que han sido asignados a varios proyectos aparece más de una vez en el resultado.

Este ejemplo demuestra un caso especial de unión θ, llamado unión equi. Este es un caso en el que la fórmula F solo contiene la igualdad (=) como operador aritmético. Sin embargo, cabe destacar que no es necesario especificar una unión equitativa sobre un atributo común, como podría sugerir el ejemplo anterior.

EMP.ENO=ASG.ENO ASG

ENO	ENAME	TÍTULO
E1	J. Doe	Ing. Elect.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Ingeniero Mecánico
E4	J. Miller	Programador
E5	B. Casey	Anal Syst.
E6	L. Chu	Ing. Elect.
E7	R. Davis	Ingeniero Mecánico
E8	J. Jones	Syst. Anal.
E9	A. Hsu	Programador
E10	T. Wong	Syst. Anal.

(a)

ENO	ENAME	TÍTULO	PNO	RESP	DUR
E1	J. Doe	Ing. Elect.	P1	Gerente	12
E2	M. Smith	Anal sistémico.	P1	Analista	12
E2	M. Smith	Anal sistémico.	P2	Analista	12
E3	A. Lee	Ing. Mecánico	P3	Consultor	12
E3	A. Lee	Ingeniero Mecánico	P4	Ingeniero	12
E4	J. Miller	Programador	P2	Programador	12
E5	J. Miller	Syst. Anal.	P2	Gerente	12
E6	L. Chu	Ing. Electo	P4	Gerente	12
E7	R. Davis	Ingeniero Mecánico	P3	Ingeniero	12
E8	J. Jones	Syst. Anal.	P3	Gerente	12

(b)

Fig. 2.7 El resultado de la unión

Una unión natural es una unión equitativa de dos relaciones sobre un atributo específico, más específicamente, sobre atributos con el mismo dominio. Sin embargo, existe una diferencia en que normalmente los atributos sobre los que se realiza la unión natural aparecen solo una vez En el resultado. Una unión natural se denota como la unión sin la fórmula.

R A S

donde A es el atributo común a R y S. Debemos notar aquí que la función natural El atributo de unión puede tener nombres diferentes en las dos relaciones; lo que se requiere es que Provienen del mismo dominio. En este caso, la unión se denota como

RA B S

donde B es el atributo de unión correspondiente de S.

Ejemplo 2.8. La unión de EMP y ASG en el Ejemplo 2.7 es, en realidad, una unión natural. He aquí otro ejemplo: la Figura 2.8 muestra la unión natural de las relaciones EMP y PAGAR en la Figura 2.3 sobre el atributo TÍTULO.

La unión interna requiere que las tuplas unidas de las dos relaciones de operandos satisfagan la predicado de unión. Por el contrario, la unión externa no tiene este requisito: las tuplas existen en La relación resultante, independientemente. La unión externa puede ser de tres tipos: unión externa izquierda (), Unión externa derecha () y unión externa completa (). En la unión externa izquierda, las tuplas de La relación del operando izquierdo siempre está en el resultado, en el caso de la unión externa derecha, Las tuplas del operando derecho siempre están en el resultado y, en el caso de operaciones externas completas, En la relación, las tuplas de ambas relaciones siempre aparecen en el resultado. La unión externa es útil en aquellos casos en los que deseamos incluir información de una o ambas relaciones incluso si no satisfacen el predicado de unión.

Pulso electromagnético ⋈ TÍTULO PAGAR

ENO	ENAME	TÍTULO	SAL
E1	J. Doe	Ing. Elect.	40000
E2	M. Smith	Analista	34000
E3	A. Lee	Ingeniería mecánica	27000
E4	J. Miller	Programador	24000
E5	B. Casey	Anal. Sist.	34000
E6	L. Chu	Ing. Elect.	40000
E7	R. Davis	Ingeniero Mecánico	27000
E8	J. Jones	Anal sistémico.	34000

Fig. 2.8 El resultado de la unión natural

Ejemplo 2.9. Considere la unión externa izquierda de EMP (revisada en el Ejemplo 2.7) y ASG sobre el atributo ENO (es decir, EMP ENO ASG). El resultado se muestra en la Figura 2.9. Tenga en cuenta que la información sobre dos empleados, E9 y E10, está incluida en el Resultado incluso aunque aún no hayan sido asignados a un proyecto con valores “Nulos” para los atributos de la relación ASG.

Pulso electromagnético ⋈ ENO ASG

ENO	ENAME	TÍTULO	PNO	RESP	DUR
E1	J. Doe	Ing. Elect.	P1	Gerente	12
E2	Señor Smith	Anal sistémico.	P1	Analista	12
E2	Señor Smith	Anal sistémico.	P2	Analista	12
E3	A. Lee	Ing. Mecánico	P3	Consultor	12
E3	A. Lee	Ing. Mecánico	P4	Ingeniero	12
E4	J. Miller	Programador	P2	Programador	12
E5	J. Miller	Anal sistémico.	P2	Gerente	12
E6	L. Chu	Ing. Elect.	P4	Gerente	12
E7	R. Davis	Ing. Mecánico	P3	Ingeniero	12
E8	J. Jones	Anal sistémico.	P3	Gerente	12
E9	A. Hsu	Programador	Nulo	Nulo	Nulo
E10	T. Wong	Syst. Anal.	Nulo	Nulo	Nulo

Fig. 2.9 El resultado de la unión externa izquierda

Semiunión.

La semiunión de la relación R, definida sobre el conjunto de atributos A, por la relación S, definida sobre el conjunto de atributos B, es el subconjunto de las tuplas de R que participan en la unión de R con S. Se denota como RF S (donde F es un predicado como se definió anteriormente) y Se puede obtener de la siguiente manera:

$$RF\ S = \Pi A(R \quad F\ S) = \Pi A(R) \quad F\ \Pi A \cap B(S)$$
$$= R \quad F\ \Pi A \cap B(S)$$

La ventaja de la semiunión es que disminuye la cantidad de tuplas que se deben unir. manejado para formar la unión. En sistemas de bases de datos centralizadas, esto es importante porque Por lo general, esto da como resultado una disminución en el número de accesos al almacenamiento secundario al hacer Mejor uso de la memoria. Esto es aún más importante en bases de datos distribuidas, ya que Por lo general, reduce la cantidad de datos que deben transmitirse entre sitios. para evaluar una consulta. Hablamos de esto con más detalle en los capítulos 3 y 8. En En este punto tenga en cuenta que la operación es asimétrica (es decir, RF S = SF R).

Ejemplo 2.10. Para demostrar la diferencia entre unión y semiunión, consideremos la semiunión de EMP con PAY sobre el predicado EMP.TITLE = PAY.TITLE, eso es,

$$\text{Pulso electromagnético} \quad EMP.TÍTULO = PAGO.TÍTULO \quad \text{PAGAR}$$

El resultado de la operación se muestra en la Figura 2.10. Animamos a los lectores a Compare las figuras 2.7 y 2.10 para ver la diferencia entre la unión y la semiunión operaciones. Tenga en cuenta que la relación resultante no tiene el atributo PAY y es Por lo tanto, más pequeño.

$$\text{Pulso electromagnético} \quad \ltimes_{EMP.TÍTULO=PAGAR.TÍTULO} \quad \text{PAGAR}$$

ENO	ENAME	TÍTULO
E1	J. Doe	Ing. Elect.
E2	M. Smith	Analista
E3	A. Lee	Ing. Mecánico
E4	J. Miller	Programador
E5	B. Casey	Syst. Anal.
E6	L. Chu	Ing. Elect.
E7	R. Davis	Ing. Mecánico
E8	J. Jones	Anal sistémico.

Fig. 2.10 El resultado de la semiunión

División.

La división de la relación R de grado r con la relación S de grado s (donde r > s y s = 0) es el conjunto de (r − s)-tuplas t tales que para todas las s-tuplas u en S, la tupla tu está en R. La operación de división se denota como R÷S y se puede especificar en términos de los operadores fundamentales de la siguiente manera:

$$R \div S = \Pi_{\bar{A}}(R) - \Pi_{\bar{A}}((\Pi_{\bar{A}}(R) \times S) - R)$$

donde $\bar{A}$ es el conjunto de atributos de R que no están en S [es decir, las (r−s)-tuplas].

Ejemplo 2.11. Supongamos que tenemos una versión modificada de la relación ASG (llamada ASG) representada en la Figura 2.11a y definida de la siguiente manera:

ASG = ΠENO,PNO (ASG) PNO PROJ

Si se desea encontrar el número de empleados asignados a todos los proyectos con un presupuesto superior a $200,000, es necesario dividir ASG con una versión restringida de PROJ, denominada PROJ (véase la Figura 2.11b). El resultado de la división (ASG÷ PROJ) se muestra en la Figura 2.11c.

La palabra clave en la consulta anterior es "all". Esto descarta la posibilidad de hacer una selección en ASG para encontrar las tuplas necesarias, ya que solo arrojaría aquellas que corresponden a empleados que trabajan en algún proyecto con un presupuesto mayor a $200,000, no a aquellos que trabajan en todos los proyectos. Tenga en cuenta que el resultado contiene solo la tupla E3 ya que las tuplas E3, P3, CAD/CAM, 250000 y E3, P4, Maintenance, 310000 existen en ASG. Por otro lado, por ejemplo, E7 no está en el resultado, ya que aunque la tupla E7, P3, CAD/CAM, 250000 está en ASG, la tupla E7, P4, Maintenance, 310000 no lo está. ,

Dado que todas las operaciones toman relaciones como entrada y producen relaciones como salida, podemos anidar operaciones mediante una notación entre paréntesis y representar programas de álgebra relacional. Los paréntesis indican el orden de ejecución. A continuación, se presentan algunos ejemplos que ilustran este problema.

Ejemplo 2.12. Considere las relaciones de la Figura 2.3. La consulta de recuperación

“Encuentre los nombres de los empleados que trabajan en el proyecto CAD/CAM”

puede ser respondido por el programa de álgebra relacional

ΠENAME(((σPNAME = PROYECTO “CAD/CAM”) PNO ASG) ENO EMP)

El orden de ejecución es: la selección en PROJ, seguida de la unión con ASG, seguido por la unión con EMP, y finalmente el proyecto en ENAME.

Un programa equivalente donde el tamaño de las relaciones intermedias es menor es

ΠENAME (EMP ENO(ΠENO(ASG PNO(σPNAME= “CAD/CAM” PROJ))))

ASG'

ENO	PNO	PNAME	PRESUPUESTO
E1	P1	Instrumentación	150000
E2	P1	Instrumentación	150000
E2	P2	Desarrollo de base de datos.	135000
E3	P3	CAD/CAM	250000
E3	P4	Mantenimiento	310000
E4	P2	Desarrollo de bases de datos.	135000
E5	P2	Desarrollo de bases de datos.	135000
E6	P4	Mantenimiento	310000
E7	P3	CAD/CAM	250000
E8	P3	CAD/CAM	250000

(a)

PROYECTO

PNO	Nombre de usuario	PRESUPUESTO
P3	CAD/CAM	250000
P4	Mantenimiento	310000

(b)

(ASG' ÷ PROY')

ENO
E3

(do)

Fig. 2.11 El resultado de la división

Ejemplo 2.13. La consulta de actualización

“Reemplazar el salario de los programadores por 25.000 dólares”

se puede calcular mediante

(PAY −(σTITLE = “Programador” PAY)) (Programador, 25000)

2.1.3.2 Cálculo relacional

En los lenguajes basados en cálculo relacional, en lugar de especificar cómo obtener el resultado,
Se especifica cuál es el resultado al indicar la relación que se supone que debe mantenerse.
Para el resultado. Los lenguajes de cálculo relacional se dividen en dos grupos: relacionales de tuplas
Cálculo y cálculo relacional de dominio. La diferencia entre ambos radica en términos

de la variable primitiva utilizada para especificar las consultas. Revisamos brevemente estos dos tipos de lenguajes.

Los lenguajes de cálculo relacional tienen una sólida base teórica, ya que se basan en la lógica de predicados de primer orden, como ya se explicó. La semántica de las fórmulas se proporciona interpretándolas como aserciones en la base de datos. Una base de datos relacional puede considerarse como una colección de tuplas o de dominios. El cálculo relacional de tuplas interpreta una variable en una fórmula como una tupla de una relación, mientras que el cálculo relacional de dominios interpreta una variable como el valor de un dominio.

Cálculo relacional de tuplas.

La variable primitiva utilizada en el cálculo relacional de tuplas es una variable de tupla que especifica una tupla de una relación. En otras palabras, abarca las tuplas de una relación.
El cálculo de tuplas es el cálculo relacional original desarrollado por Codd [1970].

En el cálculo relacional de tuplas, las consultas se especifican como {t|F(t)}, donde t es una tupla variable y F es una fórmula bien formada. Las fórmulas atómicas tienen dos formas:

1. Expresiones de pertenencia de variables tuplas. Si t es una variable tupla que abarca las tuplas de la relación R (símbolo de predicado), la expresión «la tupla t pertenece a la relación R» es una fórmula atómica, que suele especificarse como Rt o R(t).
2. Condiciones. Estas pueden definirse como sigue:

   (a) s[A]θt[B], donde s y t son variables de tupla, y A y B son componentes de s y t, respectivamente. θ es uno de los operadores de comparación aritmética <, >, =, =, ≤ y ≥. Esta condición especifica que el componente A de s se relaciona con θ respecto al componente B de t: por ejemplo, s[SAL] > t[SAL]. (b) s[A]θc, donde s, A y θ se definen como se indicó

   anteriormente y c es una constante. Por ejemplo, s[ENAME] = “Smith”.

Nótese que A se define como un componente de la variable de tupla s. Dado que el rango de s es una instancia de relación, digamos S, es obvio que el componente A de s corresponde al atributo A de la relación S. Lo mismo es cierto para B.

Existen numerosos lenguajes basados en el cálculo de tuplas relacionales, siendo los más populares SQL1 [Fecha: 1987] y QUEL [Stonebraker et al., 1976]. SQL es actualmente un estándar internacional (de hecho, el único) con diversas versiones: SQL1 se publicó en 1986, se incluyeron modificaciones en la versión de 1989, SQL2 se publicó en 1992 y SQL3, con extensiones de lenguaje orientadas a objetos, se publicó en 1999.

---

[1] A veces se cita a SQL como un término intermedio entre el álgebra relacional y el cálculo relacional. Sus creadores lo denominaron "lenguaje de mapeo". Sin embargo, se ajusta bastante a la definición de cálculo de tuplas ; por ello, lo clasificamos como tal.

SQL proporciona un enfoque uniforme para la manipulación de datos (recuperación, actualización), la definición de datos (manipulación de esquemas) y el control (autorización, integridad, etc.). Nos limitamos a la expresión, en SQL, de las consultas de los ejemplos 2.14 y 2.15.

Ejemplo 2.14. La consulta del Ejemplo 2.12,

"Encuentre los nombres de los empleados que trabajan en el proyecto CAD/CAM"

se puede expresar de la siguiente manera:

```
SELECCIONAR EMP.ENAME
DE EMP,ASG,PROJ DONDE EMP.ENO
= ASG.ENO
Y ASG.PNO = PROY.PNO
Y PROY.PNAME = "CAD/CAM"
```

Tenga en cuenta que una consulta de recuperación genera una nueva relación similar a las operaciones del álgebra relacional.

Ejemplo 2.15. La consulta de actualización del Ejemplo 2.13,

"Reemplazar el salario de los programadores por 25.000 dólares"

se expresa como

```
ACTUALIZAR PAGO
COLOCAR SAL = 25000
DONDE PAY.TITLE = "Programador"
```

Cálculo relacional de dominio.

El cálculo relacional de dominio fue propuesto por primera vez por Lacroix y Pirotte [1977]. La diferencia fundamental entre un lenguaje relacional de tuplas y un lenguaje relacional de dominios radica en el uso de una variable de dominio en este último. Una variable de dominio abarca los valores de un dominio y especifica un componente de una tupla. En otras palabras, el rango de una variable de dominio consiste en los dominios sobre los que se define la relación. Las funciones de función de trabajo (WFF) se formulan en consecuencia. Las consultas se especifican de la siguiente manera:

$$x1, x2,..., xn|F(x1,x2,..., xn)$$

donde F es una función funcional compuesta en la que x1,..., xn son las variables libres.

El éxito de los lenguajes de cálculo relacional de dominio se debe principalmente a QBE [Zloof, 1977], una aplicación visual del cálculo de dominio. QBE, diseñado exclusivamente para su uso interactivo desde una terminal visual, es intuitivo. El concepto básico es un ejemplo: el usuario formula consultas proporcionando un posible ejemplo de la respuesta.
Al escribir los nombres de las relaciones, se activa la impresión en pantalla de sus esquemas. Luego, al introducir palabras clave en las columnas (dominios), el usuario especifica la consulta. Por ejemplo, los atributos de la relación del proyecto se especifican con P, que significa "Imprimir".

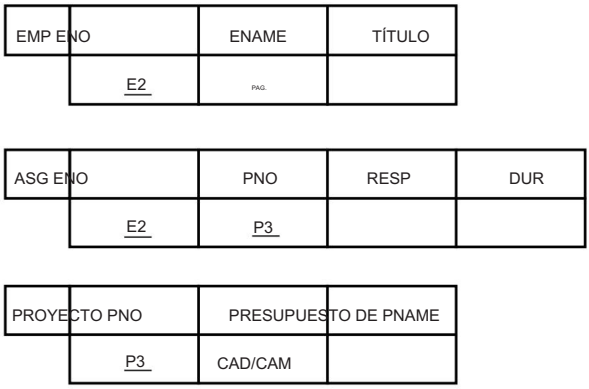

Fig. 2.12 Consulta de recuperación en QBE

De forma predeterminada, todas las consultas son de recuperación. Una consulta de actualización requiere la especificación de U bajo el nombre de la relación actualizada o en la columna actualizada. La recuperación La consulta correspondiente al Ejemplo 2.12 se muestra en la Figura 2.12 y la consulta de actualización El ejemplo 2.13 se muestra en la Figura 2.13. Para distinguir los ejemplos de las constantes, Los ejemplos están subrayados.

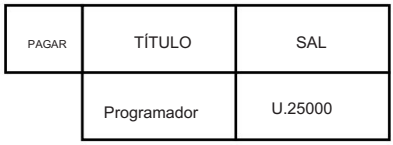

Fig. 2.13 Consulta de actualización en QBE

2.2 Revisión de redes de computadoras

En esta sección analizamos conceptos de redes informáticas relevantes para redes distribuidas. sistemas de bases de datos. Omitimos la mayoría de los detalles tecnológicos y técnicos. cuestiones a favor de discutir los conceptos principales.

Definimos una red de computadoras como una colección interconectada de sistemas autónomos. computadoras que son capaces de intercambiar información entre sí (Figura 2.14) Las palabras clave de esta definición están interconectadas y son autónomas. Queremos Las computadoras deben ser autónomas para que cada computadora pueda ejecutar programas en su propio sistema. propio. También queremos que las computadoras estén interconectadas para que sean capaces de Intercambio de información. Las computadoras en una red se denominan nodos, hosts y terminales. sistemas o sitios. Tenga en cuenta que a veces se utilizan los términos host y sistema final para referirse

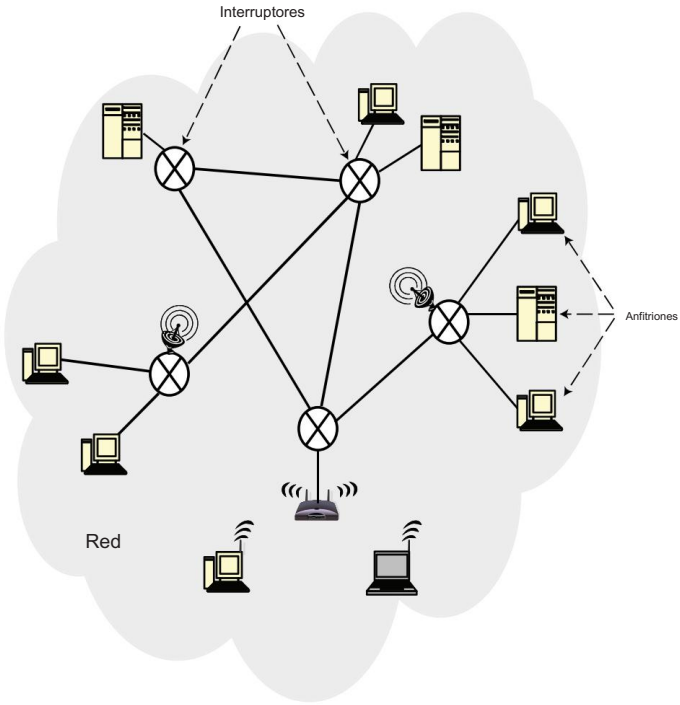

Figura 2.14 Una red de computadoras

Simplemente se refiere al equipo, mientras que el sitio se reserva para el equipo y el software que se ejecuta en él. De igual forma, el nodo se usa generalmente como referencia genérica para las computadoras o los conmutadores de una red. Forman uno de los componentes de hardware fundamentales de una red. El otro componente fundamental son los dispositivos y enlaces de propósito especial que forman la ruta de comunicación que interconecta los nodos. Como se muestra en la Figura 2.14, los hosts se conectan a la red mediante conmutadores (representados como círculos con una X), que son equipos de propósito especial que enrutan mensajes a través de la red. Algunos hosts pueden conectarse a los conmutadores directamente (mediante fibra óptica, cable coaxial o cable de cobre) y otros mediante estaciones base inalámbricas. Los conmutadores se conectan entre sí mediante enlaces de comunicación que pueden ser fibra óptica, cable coaxial, enlaces satelitales, conexiones de microondas, etc.[2]

La red informática más utilizada actualmente es Internet. Es difícil definir Internet, ya que el término se utiliza con diferentes significados, pero quizás la mejor definición sea que es una red de redes (Figura 2.15). Cada una de estas...

[2] Tenga en cuenta que los términos «conmutador» y «enrutador» a veces se usan indistintamente (incluso dentro del mismo texto). Sin embargo, en otras ocasiones tienen significados ligeramente distintos: «conmutador» se refiere a los dispositivos dentro de una red, mientras que «enrutador» se refiere a uno que se encuentra en el borde de la red, conectándola a la red troncal. Los usamos indistintamente, como en las figuras 2.14 y 2.15.

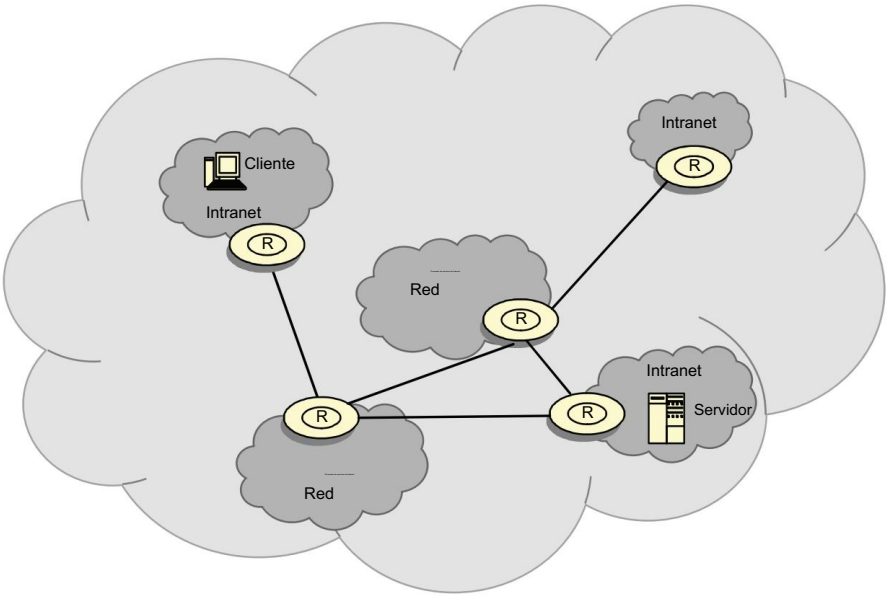

Figura 2.15 Internet

Las redes se denominan intranets para destacar su carácter interno a una organización. Una intranet, por lo tanto, consiste en un conjunto de enlaces y enrutadores (mostrados como "R" en la Figura 2.15) administrados por una sola entidad administrativa o por sus delegados. Por ejemplo, los enrutadores y enlaces de una universidad constituyen un único dominio administrativo. Dichos dominios pueden estar ubicados dentro de una única área geográfica (como la red universitaria mencionada anteriormente) o, como en el caso de grandes empresas o redes de proveedores de servicios de Internet (ISP), abarcar múltiples áreas geográficas. Cada intranet está conectada a otras mediante enlaces proporcionados por los ISP. Estos enlaces suelen ser medios de transmisión de datos dúplex de alta velocidad y larga distancia (definiremos estos términos en breve), como un cable de fibra óptica o un enlace satelital. Estos enlaces conforman lo que se denomina la red troncal de Internet. Cada intranet tiene una interfaz de enrutador que la conecta a la red troncal, como se muestra en la Figura 2.15. Por lo tanto, cada enlace conecta un enrutador de intranet al enrutador de un ISP. Los enrutadores de los ISP están conectados mediante enlaces similares a los enrutadores de otros ISP. Esto permite que los servidores y clientes de una intranet se comuniquen con servidores y clientes de otras intranets.

2.2.1 Tipos de redes

Existen varios criterios para clasificar las redes informáticas. Un criterio es la distribución geográfica (también llamada escala [Tanenbaum, 2003]), un segundo criterio es la distribución geográfica (también llamada escala [Tanenbaum, 2003]).

El primer criterio es la estructura de interconexión de los nodos (también llamada topología), y el tercero es el modo de transmisión.

2.2.1.1 Escala

En términos de distribución geográfica, las redes se clasifican en redes de área extensa, redes de área metropolitana y redes de área local. Las distinciones entre estas son algo difusas, pero a continuación se ofrecen algunas pautas generales que identifican cada una de ellas. La principal distinción entre ellas probablemente radica en el retardo de propagación, el control administrativo y los protocolos utilizados para su gestión.

Una red de área extensa (WAN) es aquella en la que la distancia de enlace entre dos nodos cualesquiera es superior a aproximadamente 20 kilómetros (km) y puede alcanzar miles de kilómetros. El uso de conmutadores permite la agregación de comunicaciones en áreas más extensas como esta. Debido a las distancias que deben recorrerse, la transmisión de datos en áreas extensas conlleva grandes retrasos. Por ejemplo, vía satélite, existe un retraso mínimo de medio segundo para la transmisión y el reconocimiento de los datos desde el origen hasta el destino. Esto se debe a que la velocidad de transmisión de las señales está limitada a la velocidad de la luz y a las grandes distancias que deben cubrirse (unos 31 000 km desde una estación terrestre hasta un satélite).

Las WAN se caracterizan típicamente por la heterogeneidad de los medios de transmisión, los ordenadores y la comunidad de usuarios. Las primeras WAN tenían una capacidad limitada de menos de unos pocos megabits por segundo (Mbps). Sin embargo, la mayoría de las actuales son WAN de banda ancha que ofrecen capacidades de 150 Mbps o superiores. Estos canales individuales se agrupan en los enlaces troncales; los enlaces troncales actuales suelen ser OC48 a 2,4 Gbps u OC192 a 10 Gbps. Estas redes pueden transportar múltiples flujos de datos con características variables (p. ej., flujos de datos, así como de audio/vídeo), y ofrecen la posibilidad de negociar un nivel de calidad de servicio (QoS) y reservar recursos de red suficientes para alcanzar dicho nivel.

Las redes de área local (LAN) suelen tener un alcance geográfico limitado (normalmente inferior a 2 km). Ofrecen comunicaciones de mayor capacidad a través de medios de transmisión económicos. Las capacidades suelen estar entre 10 y 1000 Mbps por conexión. Una mayor capacidad y las distancias más cortas entre hosts resultan en retardos muy cortos . Además, los entornos mejor controlados en los que se establecen los enlaces de comunicación (por ejemplo, dentro de edificios) reducen el ruido y las interferencias, y la heterogeneidad entre los ordenadores conectados es más fácil de gestionar, ya que se utiliza un medio de transmisión común.

Las redes de área metropolitana (MAN) tienen una escala intermedia entre las LAN y las WAN y cubren una ciudad o parte de ella. Las distancias entre nodos suelen ser del orden de 10 km.

2.2.1.2 Topología

Como su nombre indica, la estructura o topología de interconexión se refiere a la forma en que se interconectan los nodos de una red. La red de la Figura 2.14 es lo que se denomina una red irregular, donde las interconexiones entre nodos no siguen ningún patrón. Es posible encontrar un nodo conectado solo a otro nodo, así como nodos conectados a varios. Internet es una red irregular típica.

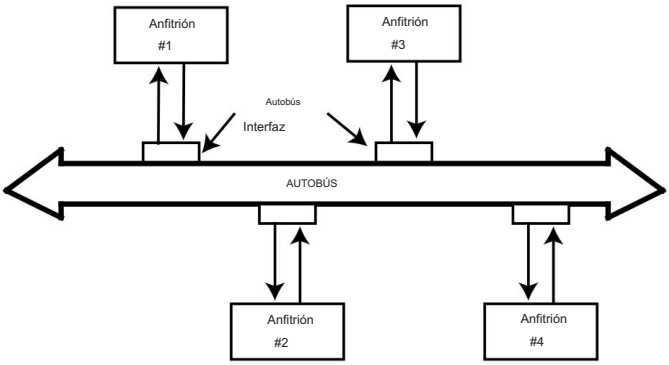

Fig. 2.16 Red de autobuses

Otra topología popular es el bus, donde todos los ordenadores están conectados a un canal común (Figura 2.16). Este tipo de red se utiliza principalmente en redes LAN. El control del enlace se realiza típicamente mediante el protocolo de acceso al medio con detección de portadora y detección de colisiones (CSMA/CD). El mecanismo de control de bus CSMA/CD se puede describir mejor como un esquema de "escuchar antes y durante la transmisión". El punto fundamental es que cada host escucha continuamente lo que ocurre en el bus. Cuando se detecta la transmisión de un mensaje, el host comprueba si el mensaje está dirigido a él y toma la acción correspondiente. Si desea transmitir, espera hasta que no detecta más actividad en el bus y luego coloca su mensaje en la red y continúa escuchando la actividad del bus. Si detecta otra transmisión mientras está transmitiendo un mensaje, se ha producido una "colisión". En tal caso, y al detectarse la colisión, los hosts transmisores abortan la transmisión, cada uno espera un tiempo aleatorio y luego retransmite el mensaje. El esquema básico CSMA/CD se utiliza en la red de área local Ethernet[3] .

Otras alternativas comunes son las redes en estrella, anillo, bus y malla.

---

[3] En la mayoría de las implementaciones actuales de Ethernet, varios buses están vinculados a través de uno o más conmutadores (llamados concentradores conmutados) para lograr una cobertura ampliada y controlar mejor la carga en cada segmento de bus. En estos sistemas, también se pueden conectar computadoras individuales directamente al conmutador. Se conocen como Ethernet conmutada.

Las redes en estrella conectan todos los hosts a un nodo central que coordina la transmisión en la red. Por lo tanto, si dos hosts desean comunicarse, deben hacerlo a través del nodo central. Dado que existe un enlace independiente entre el nodo central y cada uno de los demás, existe una negociación entre los hosts y el nodo central cuando desean comunicarse.

Las redes en anillo interconectan los hosts en forma de bucle. Este tipo de red se propuso originalmente para redes LAN, pero su uso en estas redes prácticamente ha desaparecido. Actualmente se utilizan principalmente en redes MAN (p. ej., anillos SONET). En su forma actual, la transmisión de datos a través del anillo suele ser bidireccional (los anillos originales eran unidireccionales), y cada estación (en realidad, la interfaz a la que está conectada cada estación) actúa como un repetidor activo que recibe un mensaje, verifica la dirección, copia el mensaje si está dirigido a esa estación y lo retransmite.

El control de la comunicación en redes de tipo anillo se controla generalmente mediante un token de control. En el tipo más simple de redes de anillo con token, un token, que tiene un patrón de bits para indicar que la red está libre y otro patrón de bits para indicar que está en uso, circula por la red. Cualquier sitio que desee transmitir un mensaje espera el token. Cuando llega, el sitio verifica el patrón de bits del token para ver si la red está libre o en uso. Si está libre, el sitio cambia el patrón de bits para indicar que la red está en uso y luego coloca los mensajes en el anillo. El mensaje circula por el anillo y regresa al remitente, quien cambia el patrón de bits a libre y envía el token al siguiente ordenador en la línea. • La

interconexión completa (o en malla) es aquella en la que cada nodo está interconectado con todos los demás nodos. Esta estructura de interconexión obviamente proporciona mayor confiabilidad y la posibilidad de un mejor rendimiento que las estructuras mencionadas anteriormente. Sin embargo, también es la más costosa. Por ejemplo, una conexión completa de 10.000 computadoras req 2^{200} [4]

2.2.2 Esquemas de comunicación

En términos de los esquemas de comunicación física empleados, las redes pueden ser redes punto a punto (también llamadas unicast) o redes de difusión (a veces también llamadas multipunto).

En las redes punto a punto, existen uno o más enlaces (directos o indirectos) entre cada par de nodos. La comunicación siempre se realiza entre dos nodos, y el receptor y el emisor se identifican mediante sus direcciones, incluidas en el encabezado del mensaje.
La transmisión de datos del emisor al receptor sigue uno de los muchos enlaces posibles entre ellos, algunos de los cuales pueden implicar la visita a otros nodos intermedios. Un nodo intermedio verifica la dirección de destino en el encabezado del mensaje y, si no está dirigida a él, la pasa al siguiente nodo intermedio. Este es el

[4] La forma general de la ecuación es n(n−1)/2, donde n es el número de nodos en la red.

Proceso de conmutación o enrutamiento. La selección de los enlaces a través de los cuales se envían los mensajes se determina mediante algoritmos de enrutamiento generalmente elaborados que escapan a nuestro alcance. Discutimos los detalles del cambio en la Sección 2.2.3.

Los medios de transmisión fundamentales para las redes punto a punto son los cables de par trenzado, coaxial o fibra óptica. Cada uno de estos medios tiene diferentes capacidades: par trenzado de 300 bps a 10 Mbps, coaxial hasta 200 Mbps y fibra óptica de 10 Gbps e incluso superior.

En las redes de difusión, existe un canal de comunicación común utilizado por todos los nodos de la red. Los mensajes se transmiten por este canal común y son recibidos por todos los nodos. Cada nodo verifica la dirección del receptor y, si el mensaje no está dirigido a él, lo ignora.

Un caso especial de difusión es la multidifusión, en la que el mensaje se envía a un subconjunto de nodos de la red. La dirección del receptor se codifica de alguna manera para indicar qué nodos son los destinatarios.

Las redes de transmisión generalmente se basan en radio o satélite. En el caso de la transmisión satelital, cada sitio envía su señal a un satélite, que a su vez la devuelve en una frecuencia diferente. Cada sitio de la red escucha la frecuencia de recepción y debe ignorar el mensaje si no está dirigido a ese sitio. Una red que utiliza esta técnica es HughesNet™.

La transmisión por microondas es otro modo de comunicación de datos y puede realizarse por satélite o vía terrestre. Los enlaces de microondas terrestres constituían una parte importante de las redes telefónicas de la mayoría de los países, aunque muchas de ellas se han convertido a fibra óptica. Además de los operadores públicos, algunas empresas utilizan enlaces de microondas terrestres privados. De hecho, las grandes metrópolis se enfrentan al problema de la interferencia de microondas entre los enlaces de operadores públicos y privados.

Un ejemplo muy temprano que suele identificarse como pionero en el uso de la transmisión de microondas por satélite es ALOHA [Abramson, 1973].

Las redes satelitales y de microondas son ejemplos de redes inalámbricas. Estos tipos de redes inalámbricas se conocen comúnmente como redes inalámbricas de banda ancha.

Otro tipo de red inalámbrica se basa en redes celulares. Una estación de control de red celular es responsable de un área geográfica denominada celda y coordina la comunicación de los hosts móviles en su celda. Estas estaciones de control pueden estar conectadas a una red troncal cableada y, por lo tanto, proporcionar acceso desde/hacia hosts móviles a otros hosts móviles o fijos en la red cableada.

Un tercer tipo de red inalámbrica con el que la mayoría de nosotros quizás estemos más familiarizados son las redes LAN inalámbricas (comúnmente conocidas como Wi-LAN o WiLan). En este caso, varias "estaciones base" se conectan a una red cableada y sirven como puntos de conexión para hosts móviles (similares a las estaciones de control en las redes celulares). Estas redes pueden proporcionar un ancho de banda de hasta 54 Mbps.

Una última observación sobre las topologías de difusión es que ofrecen la ventaja de facilitar la detección de errores y el envío de mensajes a más de un sitio que en las topologías punto a punto. Por otro lado, dado que todos escuchan, las redes de difusión no son tan seguras como las redes punto a punto.

2.2.3 Conceptos de comunicación de datos

Lo que denominamos comunicación de datos es el conjunto de tecnologías que permiten la comunicación entre dos hosts. No entraremos en demasiados detalles en este análisis, ya que, a nivel de DBMS distribuido, podemos asumir que existe la tecnología para transferir bits entre hosts. En su lugar, nos centraremos en algunos aspectos importantes para comprender los conceptos de retardo y enrutamiento.

Como se indicó anteriormente, los hosts están conectados por enlaces, cada uno de los cuales puede transportar uno o más canales. El enlace es una entidad física, mientras que el canal es una entidad lógica. Los enlaces de comunicación pueden transportar señales en formato digital o analógico. Las líneas telefónicas, por ejemplo, pueden transportar datos en formato analógico entre el hogar y la oficina central; el resto de la red telefónica ahora es digital e incluso el enlace del hogar a la oficina central se está volviendo digital con la tecnología de voz sobre IP (VoIP). Cada canal de comunicación tiene una capacidad, que puede definirse como la cantidad de información que se puede transmitir por el canal en una unidad de tiempo determinada. Esta capacidad se conoce comúnmente como el ancho de banda del canal. En los canales de transmisión analógicos, el ancho de banda se define como la diferencia (en hercios) entre las frecuencias más bajas y más altas que se pueden transmitir por segundo por el canal. En los enlaces digitales, el ancho de banda se refiere (de manera menos formal y con abuso de terminología) a la cantidad de bits que se pueden transmitir por segundo (bps).

Con respecto a los retrasos en la realización del trabajo del usuario, el ancho de banda de un canal de transmisión es un factor significativo, pero no es necesariamente el único. El otro factor en el tiempo de transmisión es el software empleado. Generalmente existen costos generales involucrados en la transmisión de datos debido a las redundancias dentro del propio mensaje, necesarias para la detección y corrección de errores. Además, el software de red agrega encabezados y tráilers a cualquier mensaje, por ejemplo, para especificar el destino o para verificar errores en todo el mensaje. Todas estas actividades contribuyen a los retrasos en la transmisión de datos. La velocidad real a la que se transmiten los datos a través de la red se conoce como la velocidad de transferencia de datos y esta velocidad suele ser menor que el ancho de banda real del canal de transmisión. Los problemas de software, que generalmente se denominan protocolos de red, se analizan en la siguiente sección.

En la comunicación entre computadoras, los datos suelen transmitirse en paquetes, como se mencionó anteriormente. Normalmente, se establecen límites máximos de tamaño de trama para cada red, y cada uno contiene datos, así como información de control, como las direcciones de destino y origen, códigos de verificación de errores de bloque, etc. (Figura 2.17). Si un mensaje que se envía desde un nodo de origen a un nodo de destino no cabe en una sola trama, se divide en varias. Esto se explica con más detalle en la Sección 2.2.4.

Existen diversas formas de conmutación/enrutamiento en redes punto a punto. Es posible establecer una conexión mediante un canal dedicado entre el emisor y el receptor. Esto se denomina conmutación de circuitos y se utiliza comúnmente en las conexiones telefónicas tradicionales. Cuando un abonado marca el número de otro, se establece un circuito entre ambos teléfonos mediante varios conmutadores. El circuito se mantiene durante la conversación y se interrumpe cuando uno de los dos interlocutores cuelga. Una configuración similar es posible en las redes informáticas.

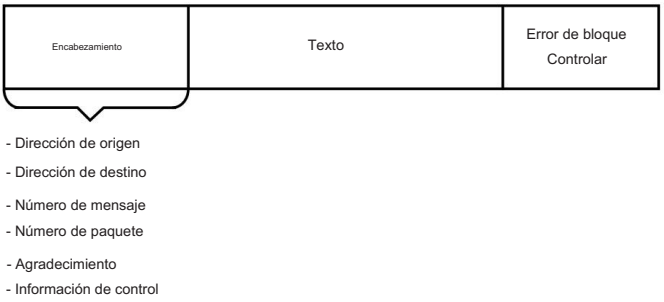

Fig. 2.17 Formato de trama típico

Otra forma de conmutación utilizada en la comunicación informática es la conmutación de paquetes, donde un mensaje se divide en paquetes y cada paquete se transmite individualmente. En nuestra discusión anterior sobre el protocolo TCP/IP, nos referimos a la transmisión de mensajes; de hecho, el protocolo TCP (o cualquier otro protocolo de la capa de transporte) divide cada paquete de aplicación en paquetes de tamaño fijo. Por lo tanto, cada mensaje de aplicación puede enviarse al destino en múltiples paquetes.

Los paquetes de un mismo mensaje pueden viajar de forma independiente y, de hecho, tomar rutas diferentes. El enrutamiento de paquetes por enlaces posiblemente diferentes en la red puede provocar que lleguen a su destino desordenados. Por lo tanto, el software de la capa de transporte en el sitio de destino debería poder ordenarlos a su orden original para reconstruir el mensaje. En consecuencia, son los paquetes individuales los que se enrutan a través de la red, lo que puede provocar que los paquetes lleguen a su destino en momentos diferentes e incluso desordenados. El protocolo de la capa de transporte en el destino es responsable de cotejar y ordenar los paquetes y de generar correctamente el mensaje de aplicación.

Las ventajas de la conmutación de paquetes son muchas. En primer lugar, las redes de conmutación de paquetes proporcionan una mayor utilización del enlace, ya que cada enlace no está dedicado a un par de equipos de comunicación y puede ser compartido por muchos. Esto es especialmente útil en la comunicación informática debido a su naturaleza en ráfagas: hay una ráfaga de transmisión y luego una interrupción antes de que comience otra ráfaga de transmisión. El enlace puede usarse para otras transmisiones cuando está inactivo. Otra razón es que la paquetización puede permitir la transmisión paralela de datos. Por lo general, no es necesario que varios paquetes que pertenecen al mismo mensaje recorran la misma ruta a través de la red. En tal caso, pueden enviarse en paralelo a través de diferentes rutas para mejorar el tiempo total de transmisión de datos. Como se mencionó anteriormente, el resultado de enrutar tramas de esta manera es que no se puede garantizar su entrega en orden.

Por otro lado, la conmutación de circuitos proporciona un canal dedicado entre el receptor y el emisor. Si se debe transmitir una cantidad considerable de datos entre ambos, o si compartir el canal en redes de conmutación de paquetes introduce demasiado retardo o varianza de retardo, o pérdida de paquetes (que son importantes en aplicaciones multimedia), el canal dedicado facilita esto significativamente. Por lo tanto, esquemas similares a la conmutación de circuitos (es decir, esquemas basados en reserva) han ganado popularidad en

las redes de banda ancha que soportan aplicaciones como multimedia con cargas de transmisión de datos muy elevadas.

2.2.4 Protocolos de comunicación

Establecer una conexión física entre dos hosts no es suficiente para que se comuniquen. Una comunicación fluida, fiable y eficiente entre hosts requiere la implementación de complejos sistemas de software, generalmente denominados protocolos. Los protocolos de red son "en capas", es decir, la funcionalidad de la red se divide en capas, cada una de las cuales realiza una función bien definida, basándose en los servicios proporcionados por la capa inferior y proporcionando un servicio a la capa superior. Un protocolo define los servicios que se prestan en una capa. El conjunto de protocolos en capas resultante se denomina pila de protocolos o suite de protocolos.

Existen diferentes pilas de protocolos para distintos tipos de redes; sin embargo, para la comunicación a través de Internet, el estándar es TCP/IP, que significa "Protocolo de Control de Transporte/Protocolo de Internet". En esta sección, nos centraremos principalmente en TCP/IP, así como en algunos de los protocolos LAN más comunes.

Antes de profundizar en los detalles de la pila de protocolos TCP/IP, analicemos cómo se transmite un mensaje de un proceso en el host C (Figura 2.15) a un proceso en el servidor S, suponiendo que ambos hosts implementan el protocolo TCP/IP. El proceso se muestra en la Figura 2.18.

El protocolo de capa de aplicación correspondiente toma el mensaje del proceso en el host C y crea un mensaje de capa de aplicación añadiendo información de encabezado de capa de aplicación (parte sombreada oblicuamente en la Figura 2.18), cuyos detalles no son relevantes para nosotros. El mensaje de aplicación se transfiere al protocolo TCP, que repite el proceso añadiendo su propia información de encabezado. El encabezado TCP incluye la información necesaria para facilitar la prestación de los servicios TCP que analizaremos a continuación. La capa de Internet toma el mensaje TCP generado y forma un mensaje de Internet, como se explica más adelante. Este mensaje se transmite físicamente desde el host C a su enrutador utilizando el protocolo de su propia red. Posteriormente, a través de una serie de enrutadores, llega al enrutador de la red que contiene el servidor S. Allí, el proceso se invierte hasta que se recupera el mensaje original y se entrega al proceso correspondiente en S. Los protocolos TCP en los hosts C y S se comunican para garantizar la comunicación de extremo a extremo que mencionamos.

2.2.4.1 Pila de protocolos TCP/IP

Lo que se conoce como TCP/IP es, de hecho, una familia de protocolos, comúnmente denominada pila de protocolos. Consta de dos conjuntos de protocolos: uno en la capa de transporte y otro en la capa de red (Internet) (Figura 2.19).

La capa de transporte define los tipos de servicios que la red proporciona a las aplicaciones. Los protocolos de esta capa abordan problemas como la pérdida de datos (¿puede la...

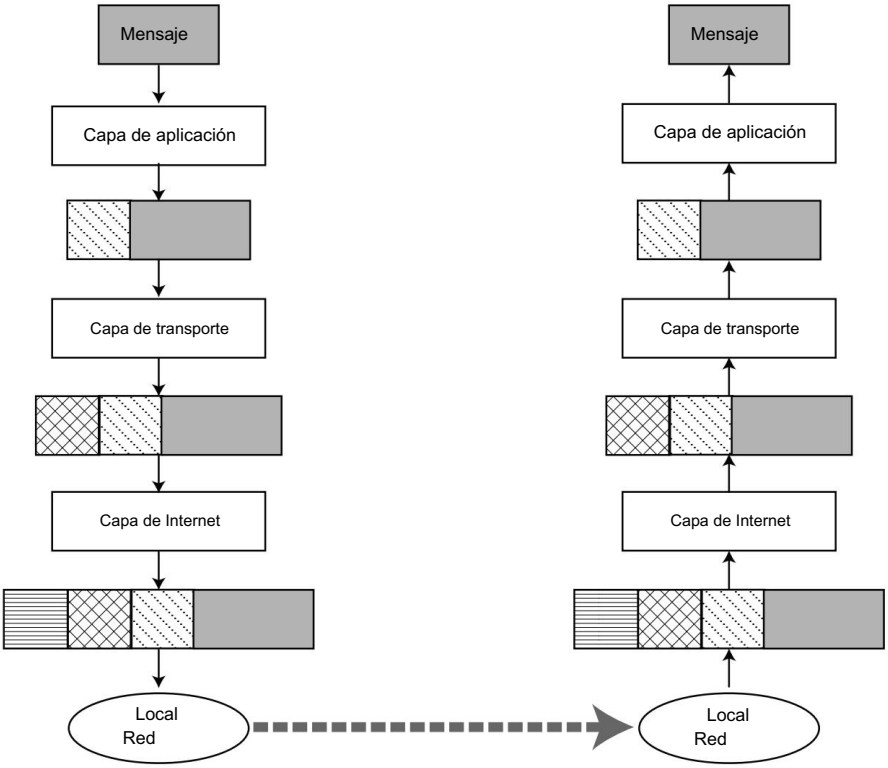

Fig. 2.18 Transmisión de mensajes mediante TCP/IP

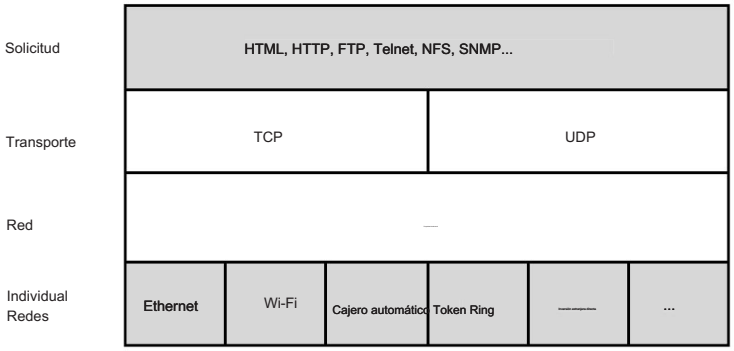

Figura 2.19 Protocolo TCP/IP

¿La aplicación puede tolerar la pérdida de algunos de los datos durante la transmisión?), ancho de banda (algunas aplicaciones tienen requisitos mínimos de ancho de banda mientras que otras pueden ser más elásticas en sus requisitos) y tiempo (¿qué tipo de retraso pueden tolerar las aplicaciones?). Por ejemplo, una aplicación de transferencia de archivos no tolera ninguna pérdida de datos, puede ser flexible en el uso del ancho de banda (funcionará independientemente de si la conexión es de alta o baja capacidad, aunque el rendimiento puede variar) y no tiene requisitos de tiempo estrictos (aunque no queramos que una transferencia de archivos tarde varios días, seguirá funcionando). Por el contrario, una aplicación de transmisión de audio/video en tiempo real puede tolerar una cantidad limitada de pérdida de datos (esto puede causar fluctuaciones y otros problemas, pero la comunicación seguirá siendo "comprensible"), tiene un requisito mínimo de ancho de banda (5-128 Kbps para audio y 5 Kbps-20 Mbps para video) y es sensible al tiempo (los datos de audio y video deben estar sincronizados).

Para abordar estos requisitos variables (al menos algunos de ellos), en la capa de transporte se proporcionan dos protocolos: TCP y UDP. TCP está orientado a la conexión, lo que significa que se requiere una configuración previa entre el emisor y el receptor antes de que pueda comenzar la transmisión real del mensaje; proporciona una transmisión confiable entre el emisor y el receptor al garantizar que los mensajes se reciban correctamente en el receptor (lo que se conoce como "confiabilidad de extremo a extremo"); garantiza el control de flujo para que el emisor no sature al receptor si el proceso del receptor no puede seguir el ritmo de los mensajes entrantes, y garantiza el control de la congestión para que el emisor se vea limitado cuando la red esté sobrecargada. Tenga en cuenta que TCP no aborda las garantías de temporización y ancho de banda mínimo, dejándolas a la capa de aplicació

UDP, por otro lado, es un servicio sin conexión que no ofrece las garantías de fiabilidad, control de flujo y control de congestión que ofrece TCP. Tampoco establece una conexión previa entre el emisor y el receptor. Por lo tanto, cada mensaje se transmite con la esperanza de llegar a su destino, pero no se ofrecen garantías de extremo a extremo. Por lo tanto, UDP tiene una sobrecarga significativamente menor que TCP y es la opción preferida por las aplicaciones que prefieren gestionar estos requisitos por sí mismas, en lugar de que el protocolo de red los gestione.

La capa de red implementa el Protocolo de Internet (IP), que permite empaquetar un mensaje en un formato estándar de Internet para su transmisión a través de la red. Cada mensaje de Internet puede tener una longitud de hasta 64 KB y consta de un encabezado que contiene, entre otros datos, las direcciones IP del remitente y del destinatario (números como 129.97.79.58, que quizás haya visto asociados a sus propios equipos), y el cuerpo del mensaje. El formato de cada red que compone Internet puede ser diferente, pero cada uno de estos mensajes se codifica en un mensaje de Internet mediante el Protocolo de Internet antes de su transmisión.

La importancia de TCP/IP radica en lo siguiente. Cada intranet que forma parte de Internet puede usar su propio protocolo preferido, por lo que los ordenadores de esa red implementan dicho protocolo (p. ej., el mecanismo Token Ring y la técnica CSMA/CS descrita anteriormente son ejemplos de estos tipos de protocolos). Sin embargo, para conectarse a Internet, necesitan poder comunicarse mediante TCP/IP, que se implementa sobre estos protocolos de red específicos (Figura 2.19).

[5] Hoy en día, muchas de las Intranets también utilizan TCP/IP, en cuyo caso la encapsulación IP puede no ser necesaria.

2.2.4.2 Otras capas de protocolo

Consideremos brevemente las otras dos capas que se muestran en la Figura 2.19. Aunque no forman parte de la pila de protocolos TCP/IP, son necesarias para crear aplicaciones distribuidas. Estas conforman las capas superior e inferior de la pila de protocolos.

La capa de Protocolo de Aplicación proporciona las especificaciones que deben seguir las aplicaciones distribuidas. Por ejemplo, si se crea una aplicación web, los documentos que se publicarán en la web deben escribirse según el protocolo HTML (tenga en cuenta que HTML no es un protocolo de red, sino un protocolo de codificación de documentos) y la comunicación entre el navegador del cliente y el servidor web debe seguir el protocolo HTTP. En esta capa se definen protocolos similares para otras aplicaciones, como se indica en la figura.

La capa inferior representa la red específica que se puede utilizar. Cada red tiene sus propios formatos de mensaje y protocolos, y proporciona los mecanismos para la transmisión de datos dentro de ella.

La estandarización de las redes LAN está liderada por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), específicamente por su Comité n.° 802; por lo tanto, el estándar desarrollado se conoce como el estándar IEEE 802. Las tres capas del estándar de red de área local IEEE 802 son la capa física, la capa de control de acceso al medio y la capa de control de enlace lógico.

La capa física se ocupa de cuestiones de transmisión de datos físicos, como la señalización. La capa de control de acceso al medio define protocolos que controlan quién puede acceder al medio de transmisión y cuándo. La capa de control de enlace lógico implementa protocolos que garantizan la transmisión fiable de paquetes entre dos ordenadores adyacentes (no de extremo a extremo). En la mayoría de las LAN, los protocolos de capa TCP e IP se implementan sobre estas tres capas, lo que permite que cada computadora pueda comunicarse directamente en Internet.

Para abarcar diversas arquitecturas LAN, el estándar de red de área local 802 se compone de varios estándares, no de uno solo. Originalmente, se especificó para admitir tres mecanismos a nivel de control de acceso al medio: el mecanismo CSMA/CD, Token Ring y el mecanismo de acceso por token para redes de bus.

2.3 Notas bibliográficas

Este capítulo abordó los aspectos básicos de los sistemas de bases de datos relacionales y las redes informáticas. Estos conceptos se analizan con mayor profundidad en varios libros de texto de calidad. En cuanto a la tecnología de bases de datos, podemos citar a [Ramakrishnan y Gehrke, 2003; Elmasri y Navathe, 2011; Silberschatz et al., 2002; Garcia- Molina et al., 2002; Kifer et al., 2006] y [Date, 2004]. Para redes informáticas, se puede consultar a [Tanenbaum, 2003; Kurose y Ross, 2010; Leon-Garcia y Widjaja, 2004; Comer, 2009]. Para un análisis más detallado de los problemas de comunicación de datos, se puede consultar [Stallings, 2011].

Capítulo 3

Diseño de bases de datos distribuidas

El diseño de un sistema informático distribuido implica la toma de decisiones sobre la ubicación de datos y programas en los sitios de una red informática, así como, posiblemente, el diseño de la propia red. En el caso de los SGBD distribuidos, la distribución de aplicaciones implica dos aspectos: la distribución del software del SGBD distribuido y la distribución de los programas de aplicación que se ejecutan en él. Los diferentes modelos arquitectónicos analizados en el Capítulo 1 abordan la cuestión de la distribución de aplicaciones. En este capítulo nos centramos en la distribución de datos.

Se ha sugerido que la organización de sistemas distribuidos puede ser investigada. cerrada a lo largo de tres dimensiones ortogonales [Levin y Morgan, 1975] (Figura 3.1):

1. Nivel de compartición 2.

Comportamiento de los patrones de acceso

3. Nivel de conocimiento sobre el comportamiento de los patrones de acceso

En términos del nivel de compartición, hay tres posibilidades. Primero, no hay compartición : cada aplicación y sus datos se ejecutan en un sitio, y no hay comunicación con ningún otro programa ni acceso a ningún archivo de datos en otros sitios. Esto caracteriza los primeros días de las redes y probablemente no sea muy común hoy en día. Luego encontramos el nivel de compartición de datos; todos los programas se replican en todos los sitios, pero los archivos de datos no. En consecuencia, las solicitudes de los usuarios se manejan en el sitio donde se originan y los archivos de datos necesarios se mueven por la red. Finalmente, en la compartición de datos más programas, tanto los datos como los programas pueden compartirse, lo que significa que un programa en un sitio determinado puede solicitar un servicio de otro programa en un segundo sitio, que, a su vez, puede tener que acceder a un archivo de datos ubicado en un tercer sitio.

Levin y Morgan distinguen entre compartir datos y compartir datos más programas para ilustrar las diferencias entre sistemas informáticos distribuidos homogéneos y heterogéneos. Indican, correctamente, que en un entorno heterogéneo suele ser muy difícil, y a veces imposible, ejecutar un programa determinado en un hardware diferente y con un sistema operativo distinto. Sin embargo, podría ser posible transferir datos con relativa facilidad.

MT Özsu y P. Valduriez, Principios de sistemas de bases de datos distribuidas: Tercera edición,
DOI 10.1007/978-1-4419-8834-8_3, © Springer Science+Business Media, LLC 2011

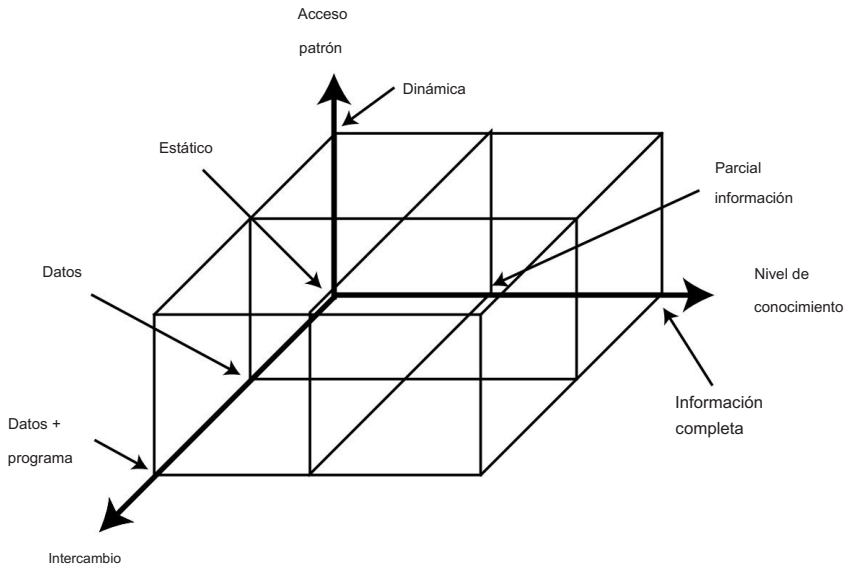

Figura 3.1 Marco de distribución

En la segunda dimensión del comportamiento de los patrones de acceso, se pueden identificar dos alternativas. Los patrones de acceso de las solicitudes de usuario pueden ser estáticos, es decir, inmutables , o dinámicos. Obviamente, es mucho más fácil planificar y gestionar entornos estáticos que sistemas distribuidos dinámicos. Desafortunadamente, es difícil encontrar muchas aplicaciones distribuidas reales que se clasifiquen como estáticas. La pregunta clave, entonces, no es si un sistema es estático o dinámico, sino cuán dinámico es. Cabe mencionar que es en esta dimensión donde se establece la relación entre el diseño de la base de datos distribuida y el procesamiento de consultas (véase la Figura 1.7).

La tercera dimensión de la clasificación es el nivel de conocimiento sobre el comportamiento de los patrones de acceso. Una posibilidad, por supuesto, es que los diseñadores desconozcan cómo accederán los usuarios a la base de datos. Esta es una posibilidad teórica, pero es muy difícil, si no imposible, diseñar un SGBD distribuido que pueda abordar esta situación eficazmente. Las alternativas más prácticas son que los diseñadores dispongan de información completa, donde los patrones de acceso puedan predecirse razonablemente y no se desvíen significativamente de estas predicciones, o información parcial, donde existan desviaciones de las predicciones.

El problema del diseño de bases de datos distribuidas debe considerarse dentro de este marco general. En todos los casos analizados, excepto en la alternativa de no compartir, se introducen nuevos problemas en el entorno distribuido que no son relevantes en un entorno centralizado. En este capítulo, nuestro objetivo es centrarnos en estos problemas específicos.

Se han identificado dos estrategias principales para el diseño de bases de datos distribuidas : el enfoque descendente y el ascendente [Ceri et al., 1987]. Como sus nombres indican, constituyen enfoques muy diferentes para el proceso de diseño. El enfoque descendente es más adecuado para SGBD distribuidos, homogéneos y estrechamente integrados , mientras que el diseño ascendente es más adecuado para múltiples bases de datos (véase la clasificación en el capítulo 1). En este capítulo, nos centramos en el diseño descendente y dejamos el diseño ascendente para el siguiente capítulo.

3.1 Proceso de diseño de arriba hacia abajo

En la Figura 3.2 se muestra un marco para el proceso de diseño descendente . La actividad comienza con un análisis de requisitos que define el entorno del sistema y determina las necesidades de datos y procesamiento de todos los usuarios potenciales de la base de datos [Yao et al., 1982a]. El estudio de requisitos también especifica dónde se espera que se ubique el sistema final con respecto a los objetivos de un DBMS distribuido como se identifica en la Sección 1.4. Estos objetivos se definen con respecto al rendimiento, la confiabilidad, la disponibilidad, la economía y la capacidad de expansión (flexibilidad).

El documento de requisitos es la base de dos actividades paralelas: el diseño de la vista y el diseño conceptual. El diseño de la vista define las interfaces para los usuarios finales. El diseño conceptual, por otro lado, examina la empresa para determinar los tipos de entidades y las relaciones entre ellas. Este proceso se puede dividir en dos grupos de actividades relacionadas [Davenport, 1981]: análisis de entidades y análisis funcional. El análisis de entidades se encarga de determinar las entidades, sus atributos y las relaciones entre ellas. El análisis funcional, por otro lado, se encarga de determinar las funciones fundamentales que intervienen en la empresa modelada. Es necesario contrastar los resultados de estos dos pasos para comprender mejor qué funciones gestionan qué entidades.

Existe una relación entre el diseño conceptual y el diseño de la vista. En cierto sentido, el diseño conceptual puede interpretarse como una integración de las vistas del usuario. Si bien esta actividad de integración de vistas es fundamental, el modelo conceptual debe ser compatible no solo con las aplicaciones existentes, sino también con las futuras. La integración de vistas debe utilizarse para garantizar que los requisitos de entidad y relación de todas las vistas se cubran en el esquema conceptual.

En las actividades de diseño conceptual y de diseño de vistas, el usuario debe especificar las entidades de datos y determinar las aplicaciones que se ejecutarán en la base de datos, así como la información estadística sobre estas aplicaciones. La información estadística incluye la especificación de la frecuencia de las aplicaciones del usuario, el volumen de información diversa, etc. Cabe destacar que la definición del esquema conceptual global, que se describe en la Sección 1.7, proviene del paso de diseño conceptual. Aún no se han considerado las implicaciones del entorno distribuido; de hecho, hasta este punto, el proceso es idéntico al de un diseño de base de datos centralizada.

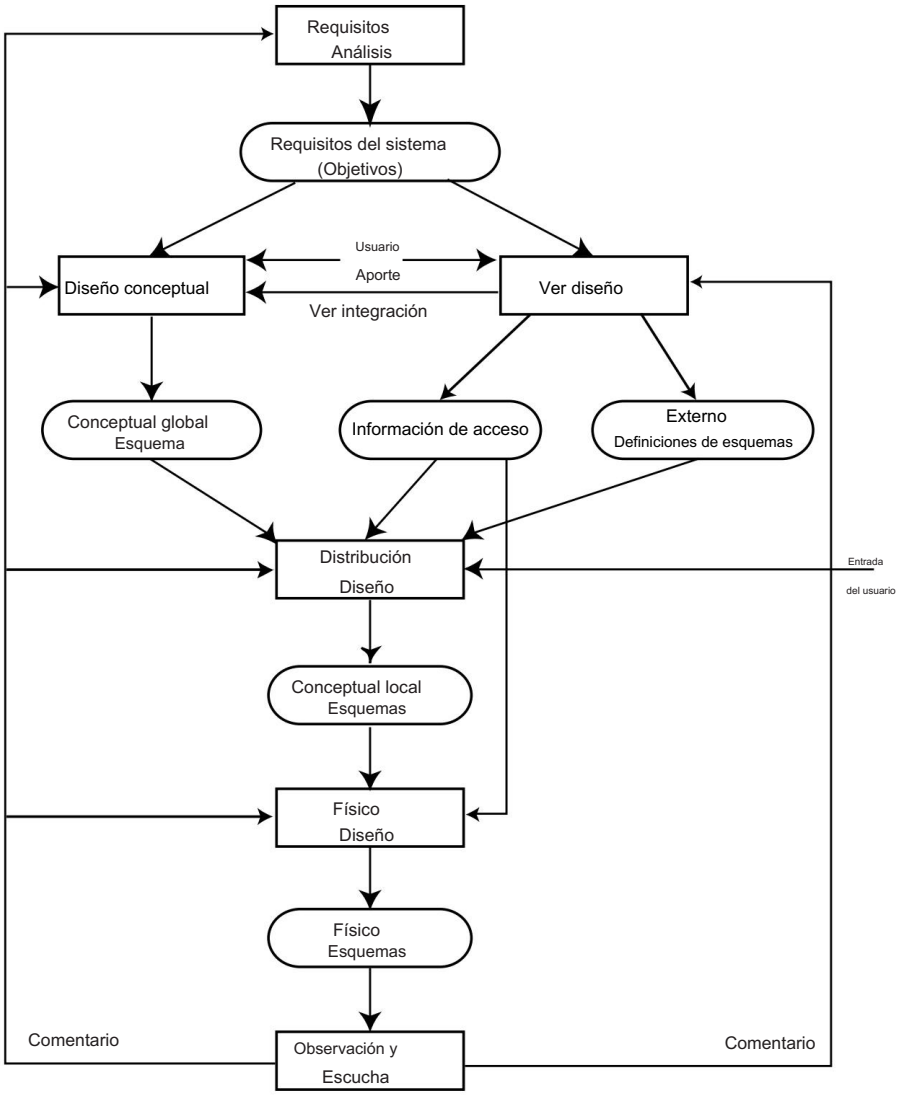

Fig. 3.2 Proceso de diseño de arriba hacia abajo

El esquema conceptual global (ECG) y la información de patrones de acceso recopilada como resultado del diseño de la vista son insumos para el paso de diseño de la distribución. El objetivo en esta etapa, que constituye el enfoque de este capítulo, es diseñar los esquemas conceptuales locales (ECL) distribuyendo las entidades en los sitios del sistema distribuido. Es posible, por supuesto, tratar cada entidad como una unidad de distribución. Dado que utilizamos

El modelo relacional como base de la discusión en este libro, las entidades corresponden a relaciones.

En lugar de distribuir relaciones, es bastante común dividirlas en subrelaciones , llamadas fragmentos, que posteriormente se distribuyen. Por lo tanto, el diseño de la distribución consta de dos pasos: fragmentación y asignación. La razón para separar el diseño de la distribución en dos pasos es abordar mejor la complejidad del problema. Sin embargo, esto plantea otras inquietudes, como se explica al final del capítulo.

El último paso del proceso de diseño es el diseño físico, que asigna los esquemas conceptuales locales a los dispositivos de almacenamiento físico disponibles en los sitios correspondientes. Las entradas de este proceso son el esquema conceptual local y la información del patrón de acceso a los fragmentos que lo componen.

Es bien sabido que cualquier actividad de diseño y desarrollo es un proceso continuo que requiere una monitorización constante y ajustes periódicos. Por ello, hemos incluido la observación y la monitorización como una actividad fundamental en este proceso. Tenga en cuenta que no solo se supervisa el comportamiento de la implementación de la base de datos, sino también la idoneidad de las vistas de usuario. El resultado es una especie de retroalimentación, que puede obligar a retroceder a uno de los pasos anteriores del diseño.

3.2 Problemas de diseño de distribución

En la sección anterior, indicamos que las relaciones en un esquema de base de datos suelen descomponerse en fragmentos más pequeños, pero no ofrecimos justificación ni detalles de este proceso. El objetivo de esta sección es completar estos detalles.

El siguiente conjunto de preguntas interrelacionadas abarca toda la cuestión. Por lo tanto, intentaremos responderlas en el resto de esta sección.

1. ¿Por qué fragmentar?
2. ¿Cómo debemos fragmentar?
3. ¿Cuánto debemos fragmentar?
4. ¿Hay alguna forma de comprobar la exactitud de la descomposición?
5. ¿Cómo debemos asignar?
6. ¿Cuál es la información necesaria para la fragmentación y asignación?

3.2.1 Razones de la fragmentación

Desde la perspectiva de la distribución de datos, no hay razón para fragmentarlos. Al fin y al cabo, en los sistemas de archivos distribuidos, la distribución se realiza sobre la base de archivos completos. De hecho, los primeros trabajos se centraron específicamente en la asignación de archivos a los nodos de una red informática. Consideramos modelos anteriores en la Sección 3.4.

Con respecto a la fragmentación, la cuestión clave es la unidad de distribución adecuada . Una relación no es una unidad adecuada por varias razones. En primer lugar, las vistas de aplicación suelen ser subconjuntos de relaciones. Por lo tanto, la localidad de los accesos de las aplicaciones no se define en las relaciones completas, sino en sus subconjuntos. Por lo tanto, es natural considerar los subconjuntos de relaciones como unidades de distribución.

En segundo lugar, si las aplicaciones con vistas definidas en una relación dada residen en sitios diferentes, se pueden considerar dos alternativas, con la relación completa como unidad de distribución. O bien, la relación no se replica y se almacena en un solo sitio, o bien se replica en todos o algunos de los sitios donde residen las aplicaciones. La primera opción genera un volumen innecesariamente alto de accesos remotos a datos. La segunda, por otro lado, presenta una replicación innecesaria, lo que causa problemas en la ejecución de actualizaciones (que se analizarán más adelante) y puede no ser conveniente si el almacenamiento es limitado.

Finalmente, la descomposición de una relación en fragmentos, cada uno tratado como una unidad, permite la ejecución simultánea de varias transacciones. Además, la fragmentación de relaciones suele resultar en la ejecución paralela de una sola consulta al dividirla en un conjunto de subconsultas que operan sobre fragmentos. Por lo tanto, la fragmentación suele aumentar el nivel de concurrencia y, por consiguiente, el rendimiento del sistema.
Esta forma de concurrencia, a la que llamamos concurrencia intraconsulta, se trata principalmente en los Capítulos 7 y 8, en el apartado de procesamiento de consultas.

La fragmentación también plantea dificultades. Si las aplicaciones tienen requisitos contradictorios que impiden la descomposición de la relación en fragmentos mutuamente excluyentes, las aplicaciones cuyas vistas se definen en más de un fragmento podrían sufrir una degradación del rendimiento. Por ejemplo, podría ser necesario recuperar datos de dos fragmentos y luego realizar su unión, lo cual resulta costoso. Minimizar las uniones distribuidas es un problema fundamental de la fragmentación.

El segundo problema se relaciona con el control semántico de datos, específicamente con la comprobación de integridad. Como resultado de la fragmentación, los atributos que participan en una dependencia pueden descomponerse en diferentes fragmentos que podrían asignarse a diferentes sitios. En este caso, incluso la tarea más sencilla de comprobar dependencias resultaría en la búsqueda de datos en varios sitios. En el capítulo 5 , retomamos el tema del control semántico de datos.

3.2.2 Alternativas de fragmentación

Las instancias de relación son esencialmente tablas, por lo que la cuestión radica en encontrar alternativas para dividir una tabla en tablas más pequeñas. Claramente, existen dos alternativas: dividirla horizontalmente o dividirla verticalmente.

Ejemplo 3.1. En este capítulo, utilizamos una versión modificada del esquema de base de datos relacional desarrollado en la Sección 2.1. Hemos añadido a la relación PROJ un nuevo atributo (LOC) que indica la ubicación de cada proyecto. La Figura 3.3 muestra la instancia de base de datos que utilizaremos. La Figura 3.4 muestra la relación PROJ de la Figura 3.3 dividida horizontalmente en dos relaciones. La subrelación PROJ1 contiene información sobre los proyectos cuyos...

Pulso electromagnético

ENO	ENAME	TÍTULO
E1	J. Doe	Ingeniero Electo
E2	M. Smith	Syst. Anal.
E3	A. Lee	Ingeniero Mecánico
	Programador E4 J. Miller	
E5	B. Casey	Anal Sist.
E6	L. Chu	Ing. Elect.
E7	R. Davis	Ingeniero Mecánico
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
Gerente E1	P1		12
Analista E2	P1		24
Analista E2	P2		6
Consultor E3	P3		10
Ingeniero E3	P4		48
Programador E4	P2		18
Gerente E5	P2		24
Gerente E6	P4		48
Ingeniero E7	P3		36
Gerente E8	P3		40

PROYECTO

PNO	Nombre de usuario	PRESUPUESTO	LOCALIDAD
	Instrumentación P1	150000	Montreal
	Desarrollo de base de datos P2.	135000	Nueva York
	P3 CAD/CAM	250000	Nueva York
	Mantenimiento P4	310000	París

PAGAR

TÍTULO	SAL
Ing. Elect.	40000
Anal sistémico.	34000
Ingeniería mecánica	27000
Programador	24000

Fig. 3.3 Base de datos de ejemplo modificada

Los presupuestos son inferiores a 200.000 dólares, mientras que PROJ2 almacena información sobre los proyectos. con presupuestos más grandes.

Ejemplo 3.2. La figura 3.5 muestra la relación PROJ de la figura 3.3 dividida verticalmente. en dos subrelaciones, PROJ1 y PROJ2. PROJ1 contiene solo la información sobre presupuestos de proyectos, mientras que PROJ2 contiene los nombres y ubicaciones de los proyectos. Es importante observar que la clave principal de la relación (PNO) está incluida en ambos fragmentos.

La fragmentación puede, por supuesto, estar anidada. Si las anidaciones son de diferentes tipos, Se obtiene una fragmentación híbrida. Aunque no tratamos la fragmentación híbrida como... una estrategia de fragmentación primitiva, muchas particiones de la vida real pueden ser híbridas.

3.2.3 Grado de fragmentación

El grado en que se debe fragmentar la base de datos es una decisión importante que afecta el rendimiento de la ejecución de consultas. De hecho, los problemas de la Sección 3.2.1... Las preguntas sobre las razones de la fragmentación constituyen un subconjunto de las respuestas a la La pregunta que abordamos aquí es: el grado de fragmentación va de un extremo a otro. es decir, no fragmentar en absoluto, al otro extremo, fragmentar al nivel de

PROY1

PNO	Nombre de usuario	PRESUPUESTO	LOCALIDAD
P1	Instrumentación	150000	Montreal
P2	Desarrollo de bases de datos.	135000	Nueva York

PROY2

PNO	Nombre de usuario	PRESUPUESTO	LOCALIDAD
P3	CAD/CAM	255000	Nueva York
P4	Mantenimiento	310000	París

Fig. 3.4 Ejemplo de partición horizontal

PROY1

PNO	PRESUPUESTO
P1	150000
P2	135000
P3	250000
P4	310000

PROY2

PNO	PNAME	LOCALIDAD
P1	Instrumentación	Montreal
P2	Desarrollo de bases de datos.	Nueva York
P3	CAD/CAM	Nueva York
P4	Mantenimiento	París

Fig. 3.5 Ejemplo de partición vertical

tuplas individuales (en el caso de fragmentación horizontal) o al nivel de atributos individuales (en el caso de fragmentación vertical).

Ya hemos abordado los efectos adversos de unidades de fragmentación muy grandes y muy pequeñas. Lo que necesitamos, entonces, es encontrar un nivel adecuado de fragmentación que sea un equilibrio entre ambos extremos. Dicho nivel solo puede definirse con respecto a las aplicaciones que se ejecutarán en la base de datos. La pregunta es, ¿cómo? En general, las aplicaciones deben caracterizarse con respecto a una serie de parámetros.

Según los valores de estos parámetros se pueden identificar fragmentos individuales. En la Sección 3.3 describimos cómo se puede realizar esta caracterización para fragmentaciones alternativas.

3.2.4 Reglas de corrección de la fragmentación

Aplicaremos las siguientes tres reglas durante la fragmentación, que, en conjunto, garantizan que la base de datos no sufra cambios semánticos durante la fragmentación.

1. Completitud. Si una instancia de relación R se descompone en fragmentos FR = {R1, R2, ..., Rn}, cada elemento de datos que se puede encontrar en R también se puede encontrar en uno o más de Ri. Esta propiedad, que es idéntica a la propiedad de descomposición sin pérdida de la normalización (Sección 2.1), también es importante en la fragmentación, ya que garantiza que los datos en una relación global se asignen a fragmentos sin ninguna pérdida [Grant, 1984]. Nótese que en el caso de la fragmentación horizontal, el "elemento" generalmente se refiere a una tupla, mientras que en el caso de la fragmentación vertical, se refiere a un atributo.
2. Reconstrucción. Si una relación R se descompone en fragmentos FR = {R1,R2, ...,Rn}, debería ser posible definir un operador relacional tal que

   $$R = R_i \quad , \qquad R_i \quad FR$$

   El operador será diferente para las distintas formas de fragmentación; sin embargo, es importante que pueda identificarse. La reconstructabilidad de la relación a partir de sus fragmentos garantiza que se conserven las restricciones definidas en los datos en forma de dependencias.
3. Disjunción. Si una relación R se descompone horizontalmente en fragmentos FR = {R1, R2, ..., Rn} y el dato di se encuentra en Rj , no se encuentra en ningún otro fragmento Rk (k = j). Este criterio garantiza la disjunción de los fragmentos horizontales. Si la relación R se descompone verticalmente, sus atributos de clave primaria se suelen repetir en todos sus fragmentos (para su reconstrucción). Por lo tanto, en caso de partición vertical, la disjunción se define únicamente en los atributos de clave no primaria de una relación.

3.2.5 Alternativas de asignación

Suponiendo que la base de datos esté fragmentada correctamente, es necesario decidir la asignación de los fragmentos a varios sitios de la red. Una vez asignados los datos, estos pueden replicarse o mantenerse como una sola copia. La replicación se basa en la fiabilidad y la eficiencia de las consultas de solo lectura. Si existen varias copias de un dato , es muy probable que alguna copia sea accesible en algún lugar, incluso si se producen fallos del sistema. Además, las consultas de solo lectura que acceden a los mismos datos pueden ejecutarse en paralelo, ya que existen copias en varios sitios. Por otro lado, la ejecución de consultas de actualización causa problemas, ya que el sistema debe garantizar que todas las copias de los datos se actualicen correctamente. Por lo tanto, la decisión sobre la replicación es un equilibrio que depende de la proporción de consultas de solo lectura con respecto a...