Desarrollo de Aplicaciones WEB

CICLO FORMATIVO DE GRADO SUPERIOR

Unidad 1

Introducción a las Bases de Datos. Modelos de Datos

Módulo: Bases de Datos

Profesor del módulo: Juan Manuel Fernández Gutiérrez

Materiales registrados (ISBN: 978-84-692-3443-2 Deposito Legal: AS-3564-2009)

FORMACIÓN PROFESIONAL
Principado de Asturias

Introducción

En esta unidad se presenta una visión general de los sistemas de bases de datos y se desarrollan los conceptos fundamentales del modelo relacional.

Se empieza describiendo las características de los sistemas de base de datos y analizando sus ventajas con respecto a los sistemas tradicionales basados en ficheros. Se estudian las funciones de un Sistema Gestor de Bases de Datos (SGBD), teniendo en cuenta tanto la arquitectura a tres niveles, propuesta por ANSI/X3/SPARC, en que se estructuran los SGBD modernos, como las necesidades de los distintos tipos de usuarios del sistema. También, se presenta la arquitectura cliente/servidor y sus distintas formas de implementación en los sistemas actuales. Se trata el concepto de modelo de datos y se describen las características de los modelos convencionales en que se apoyan los sistemas de bases de datos dominantes en el mercado en las últimas décadas, es decir, los modelos CODASYL, jerárquico y relacional. Finalmente, se desarrolla con cierto detalle la estructura y restricciones del modelo relacional, así como los elementos fundamentales del álgebra relacional.

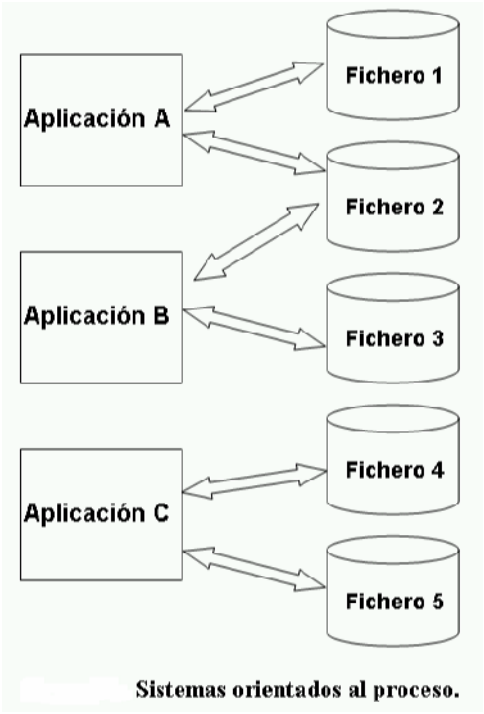
Objetivos

- Describir las características de las bases de datos.
- Diferenciar las estructuras de ficheros tradicionales y las estructuras de almacenamiento basadas en tecnologías de bases de datos.
- Conocer los aspectos fundamentales de la arquitectura cliente-servidor.
- Identificar las características de los modelos de bases de datos en red, jerárquico y relacional.
- Describir los elementos que componen la estructura del modelo relacional.
- Comprender el fundamento de las restricciones inherentes y de usuario.
- Utilizar el álgebra relacional para escribir expresiones que representen operaciones de recuperación o actualización de una base de datos relacional.

Contenidos Generales

1. *Concepto de base de datos.*

Una base de datos es **"una colección no redundante de datos compartibles entre diferentes sistemas de aplicación"**. Esta definición que aparece en la obra "*Data Análisis for Database Design*" de David R. Howe permite un primer acercamiento al concepto de base de datos.



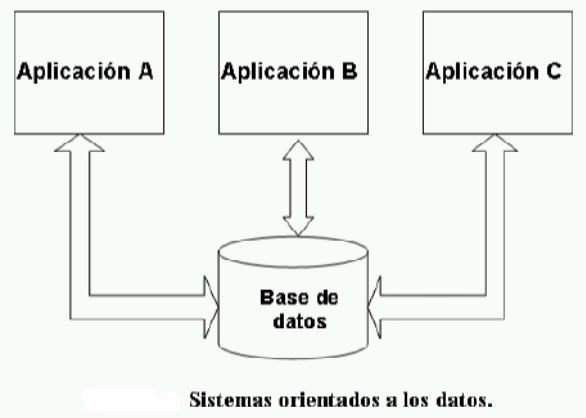
Sistemas orientados al proceso.

El almacenamiento de un conjunto de datos con el objetivo de que sirvan a distintas aplicaciones, evita su redundancia y todos los problemas asociados a la misma. Si fuera necesaria la repetición de datos por razones de eficiencia, esta redundancia será controlada por el sistema. Así, en caso de que una aplicación actualice un dato redundante, el sistema se encargará de propagar dicha actualización a todas sus ocurrencias. De esta forma, se consigue que los resultados obtenidos por las distintas aplicaciones sean coherentes, ya que no cabe la posibilidad de que utilicen distintos valores del mismo dato.

Los sistemas bases de datos se desarrollan en los años sesenta del pasado siglo como un enfoque alternativo a los sistemas basados en ficheros utilizados hasta entonces. En los sistemas informáticos de aquella época los datos se almacenan en ficheros diseñados específicamente para cada aplicación. Aunque varias aplicaciones manejen datos comunes, los ficheros en que se encuentran no suelen ser compartidos entre ellas, por lo que el mismo dato puede aparecer repetido en dos o más ficheros. Cuando una aplicación actualiza uno de estos datos redundante, lo hace sólo en su fichero, con lo que durante cierto tiempo pueden obtenerse distintos valores para el mismo dato dependiendo del fichero que se consulte. Problemas adicionales de la redundancia de información son la ocupación extra de memoria secundaria y el aumento del tiempo de proceso al tener que repetirse los mismos procedimientos de validación, actualización y almacenamiento de los datos en los distintos ficheros.

Los sistemas de ficheros son **sistemas orientados al proceso**, ya que en ellos los datos están subordinados a los procesos, cada aplicación dispone de sus propios ficheros de datos. La descripción de estos datos y sus interrelaciones aparecen en los programas que hacen uso

de ellos. Los cambios en la estructura de los datos o en su almacenamiento llevan a un rediseño de las aplicaciones que los utilizan.



Sistemas orientados a los datos.

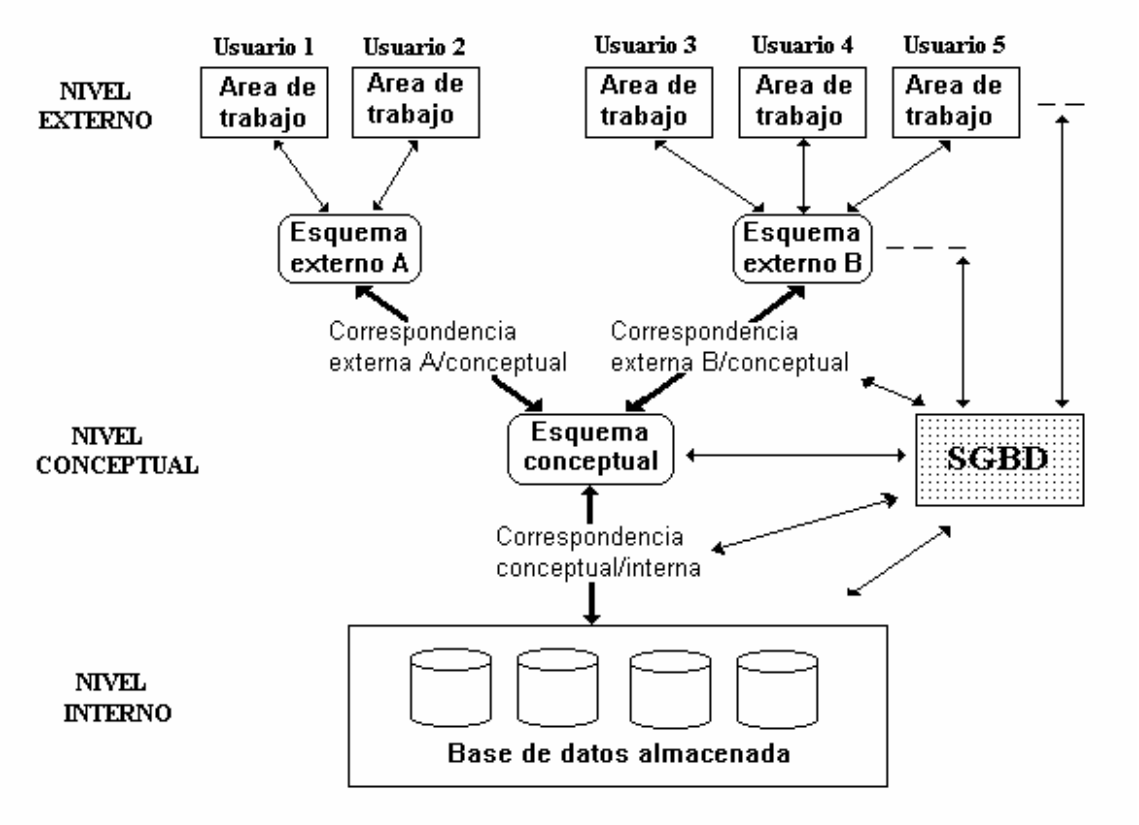
Las bases de datos son **sistemas orientados a los datos.** Éstos se almacenan independientemente de las aplicaciones. En la base de datos también se guardan la descripción de los datos y las interrelaciones que puedan existir entre ellos. Los datos se almacenan interrelacionados y estructurados de acuerdo con un determinado modelo de datos que permita recoger el máximo significado asociado a ellos. Los cambios en los datos, en su forma de almacenamiento o en los caminos de acceso a ellos no tienen por que llevar a una reprogramación de las aplicaciones.

2. El sistema gestor de la base de datos.

En un sistema de base de datos, los programas no acceden directamente a los datos, sino que es el software del sistema el encargado de proporcionar a las aplicaciones los datos que éstas demanden. Este software, conocido genéricamente como **sistema gestor de la base de datos (SGBD)**, no sólo tendrá a su cargo las funciones de recuperación y manipulación de los datos, sino que también deberá permitir su definición, e igualmente deberá implementar otras funciones encaminadas a mantener la integridad, seguridad y confidencialidad de los datos.

El sistema gestor de la base de datos puede ser definido como **un conjunto de programas que permiten que los usuarios describan, recuperen y manipulen eficazmente los datos almacenados en la base de datos, protegiendo dichos datos contra todas aquellas acciones, intencionadas o no, que los puedan corromper**.

Las arquitecturas de bases de datos han evolucionado mucho desde sus comienzos, aunque la considerada estándar hoy en día es la propuesta en 1975 por el comité ANSI/X3/SPARC (Standard Planning and Requirements Committee of the American National Standards Institute on Computers and Information Processing). Este comité propuso una arquitectura general para DBMSs (Database Management Systems) basada en tres niveles o esquemas: el nivel físico, o de máquina, el nivel externo, o de usuario, y el nivel conceptual. Así



Arquitectura de un SGBD a tres niveles

mismo describió las interacciones entre estos tres niveles y todos los elementos que conforman cada uno de ellos.

El **nivel externo** o **esquema externo** es la visión que tiene de la base de datos un usuario en particular y en él estarán solo reflejados aquellos datos e interrelaciones que necesite ese usuario. También se incluirán las restricciones de uso de esos datos y, en algunas implementaciones, los caminos de acceso a esos datos. Existirán distintos esquemas externos de la misma base de datos, ya que distintos usuarios y aplicaciones tienen distintas percepciones de ella. Un mismo esquema externo puede ser compartido por varios usuarios.

El **nivel conceptual** o **esquema conceptual** corresponde a la visión global de todos los datos de la base. En este nivel se incluirá la descripción de todos los datos e interrelaciones, así como las restricciones de integridad de los datos y las restricciones de acceso a ellos. El esquema conceptual representa la visión integrada de todos los usuarios.

El **nivel interno** o **esquema interno** corresponde a la estructura de almacenamiento que sostiene los datos. Es muy dependiente de la implementación concreta del SGBD. En el esquema interno se especifican aspectos relativos al almacenamiento de los datos en memoria secundaria y acceso a los mismos.

El SGBD es el encargado de posibilitar la definición y manipulación de la base de datos a estos tres niveles y de gestionar las interfaces entre niveles y con el usuario.

3. Descripción, manipulación y transformación de los datos.

El SGBD debe permitir la descripción de los distintos esquemas de la base de datos, así como que los usuarios de la base recuperen, inserten, modifiquen o eliminen datos de ella. También, se debe encargar de realizar la correspondencia o transformación (**mapping**, *en inglés*) entre los formatos de los datos correspondientes a los distintos esquemas vistos en el punto anterior.

El administrador de la base de datos es el encargado de especificar los elementos de datos que la forman, su estructura, interrelaciones, restricciones, etc. Es su responsabilidad la definición tanto del esquema conceptual como del interno como de los distintos esquemas externos, aunque a veces, puede dejar que los programadores realicen la descripción de los esquemas externos implicados en las aplicaciones que implementan.

Al hablar de administrador de la base de datos hay que entenderlo más como una función que como una persona concreta. La función de administración puede ser realizada por una sola persona o por un grupo de ellas.

El administrador especificará los distintos esquemas mediante un **lenguaje de definición de datos** (LDD, o DDL en inglés). Cada tipo de SGBD dispondrá de uno o más lenguajes LDD específicos. En unos casos existe un único lenguaje, con sentencias que permiten describir los elementos de los distintos esquemas, y en otros hay un lenguaje distinto para cada nivel de abstracción.

Las operaciones a realizar sobre la información contenida en la base se especifican mediante un **lenguaje de manipulación de datos** (LMD, o DML en inglés). Este lenguaje de manipulación puede estar formado por un conjunto de sentencias (**lenguaje huésped**) que son

admitidas dentro de un programa escrito en otro lenguaje (**lenguaje anfitrión**) de propósito general. En los SGBD actuales suele existir también un LMD **autocontenido,** que no necesita ir embebido en otro para especificar cualquier tarea de recuperación o actualización de información.

El SGBD llevará a cabo las operaciones de manipulación de datos cumpliendo siempre las restricciones de seguridad especificadas por el administrador.

En un SGBD que siga la propuesta ANSI/X3/SPARC, existirán dos niveles de transformación:

- la **transformación conceptual-interna** que permite hacer pasar los datos desde su representación conceptual a su representación interna. y viceversa, y
- **la transformación externa-conceptual** que permite hacer pasar los datos desde su representación en el nivel conceptual a la correspondiente a los distintos esquemas externos, y viceversa.

Para poder efectuar automáticamente la transformación de datos de un nivel a otro, el SGBD debe conocer las correspondencias existentes entre niveles que estarán almacenadas junto con los esquemas en el diccionario de la base de datos.

4. Protección de datos.

El SGBD debe disponer de mecanismos que eviten que la base de datos quede en un estado inconsistente a causa de fallos que den lugar a la alteración y corrupción de los datos. Estos fallos pueden ser fallos físicos (de memoria principal, de memoria secundaria, etc.), fallos lógicos (de programa, del sistema operativo, del SGBD, etc.) y fallos humanos, intencionados o no. También, el SGBD dispondrá de facilidades para que el administrador proteja los datos contra accesos no autorizados, manteniendo su privacidad.

Al tratar la protección de los datos de una base, hay que referirse a los conceptos de **seguridad**, **integridad** y **confidencialidad**.

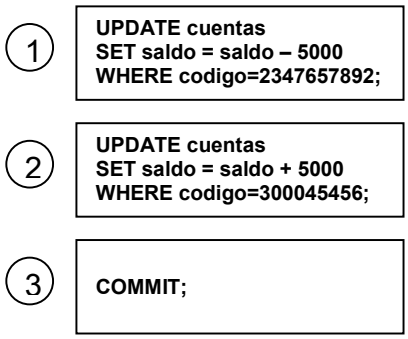
❑ Seguridad

La seguridad (o **recuperación**) de la base de datos se refiere a su protección contra fallos lógicos o físicos que destruyan los datos total o parcialmente.

El origen de estos fallos puede ser múltiple, y además de los instrumentos de protección facilitados por el SGBD, se emplearán otros ajenos a él como: sistemas contraincendios, sistemas tolerantes a fallos, sistemas de control de acceso físico a los equipos, etc.

El objetivo de la seguridad, cuando se produce cualquiera de estos fallos, es mantener la consistencia de la base de datos.

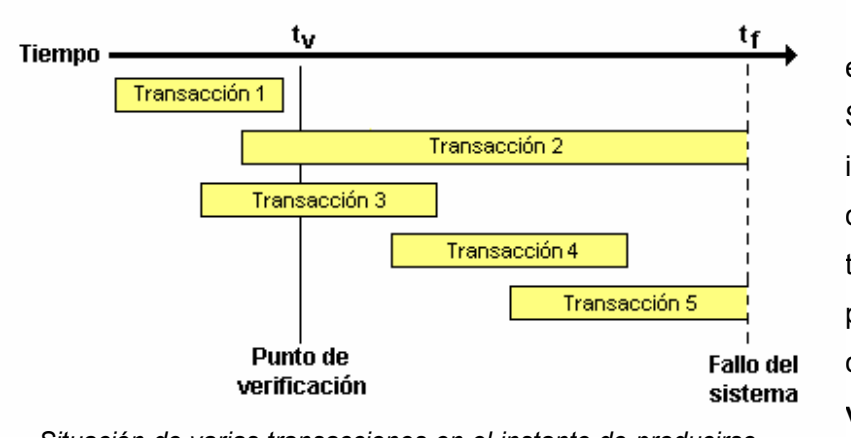
Veamos un ejemplo en que al producirse un fallo la base de datos queda en un estado inconsistente. Se supone un proceso en que se está realizando una transferencia de fondos entre dos cuentas bancarias, concretamente, se transfieren 5.000 euros de la cuenta **2347657892** a la cuenta **300045456**. En la primera operación se restan los 5.000 euros del saldo de la primera cuenta, y en la segunda operación se suma la misma cantidad al saldo de la segunda cuenta. Si se produjese un fallo a consecuencia del cual sólo se realizase la primera de las dos operaciones, la base de datos quedaría en un estado inconsistente: habría 5.000 euros perdidos, que no está ni en una cuenta ni en otra.



Transferencia bancaria.

La implementación por parte del SGBD del concepto de transacción hace que este tipo de problemas nunca se produzcan. Una **transacción es una secuencia de operaciones que han de ejecutarse de forma atómica,** es decir, o se realizan todas las operaciones que componen la transacción o ninguna. De esta forma, en el ejemplo, la transacción sería la operación de transferencia, que estaría compuesta de las dos operaciones de modificación de saldo. Si durante la segunda operación de modificación se produjera un fallo que impidiera su terminación, el SGBD automáticamente desharía la primera y así la base de datos quedaría en un estado consistente. La orden **COMMIT** de la figura, comunicaría al SGBD el fin de la transacción.

Un fallo que afecta a la seguridad de la base de datos es aquél que provoca la pérdida de memoria volátil (por ejemplo, fallo en el suministro eléctrico). Al producirse esta situación, pueden quedar transacciones sin finalizar y otras, que habiendo concluido satisfactoriamente, se encontraban en áreas de memoria intermedia y aun no se habían grabado en disco. La recuperación de estos fallos, conocida como **recuperación en caliente**, implica deshacer las operaciones correspondientes a transacciones inconclusas y repetir las transacciones finalizadas y no grabadas. Esta recuperación la realizará automáticamente el SGBD, y para ello el método más habitual es apoyarse en uno o más ficheros diarios.



Situación de varias transacciones en el instante de producirse un fallo de sistema.

Un fichero **diario** o **log** es un fichero en el que el SGBD va grabando toda la información necesaria para deshacer o rehacer una transacción. El SGBD ejecuta periódicamente la operación conocida como **punto de verificación** o **punto de recuperación** (**checkpoint**), consistente en grabar en memoria secundaria el contenido de las áreas de memoria intermedia y registrar tanto en el fichero diario como en un fichero de rearranque dicho punto de verificación. Cuando el SGBD debe realizar una recuperación de la base de datos después de una caída del sistema, o sea, cuando lleva a cabo una recuperación en caliente, obtiene la dirección del registro de recuperación más reciente del fichero de rearranque y recorre el fichero ***log*** desde ese punto hasta el final para obtener la información de qué transacciones debe deshacer o rehacer.

En la figura se ven varias situaciones de transacciones ante un supuesto de fallo que causa la pérdida de memoria volátil. La transacción 1 no es afectada por el fallo ya que finalizó con anterioridad al último punto de verificación y, por tanto, cuando se produce la pérdida de memoria estaba grabada en disco. Las transacciones 3 y 4 finalizan antes del fallo pero no hay seguridad de que estuviesen grabadas, pues finalizaron después del ***checkpoint***, por lo que deben ser rehechas. Finalmente, las transacciones 2 y 5 no habían concluido cuando se produjo el fallo y deben ser deshechas.

Para recuperarse de un fallo de memoria secundaria que afecte a la base de datos (**recuperación en frío**) debe utilizarse una copia de seguridad(***backup***) de la base de datos. La restauración del último ***backup*** realizado deja la base de datos recuperada a la fecha en que se hizo dicha copia de seguridad. Para poner la base de datos al día habrá que utilizar todos los ficheros diarios que se han generado desde esa fecha hasta la actualidad en orden cronológico.

- **Integridad**

La base de datos puede entrar también en un estado inconsistente a consecuencia de la realización de operaciones incorrectas. El SGBD debe detectar y corregir estas operaciones, asegurando en todo momento la integridad de la base de datos, es decir, que los datos contenidos en la base sean correctos.

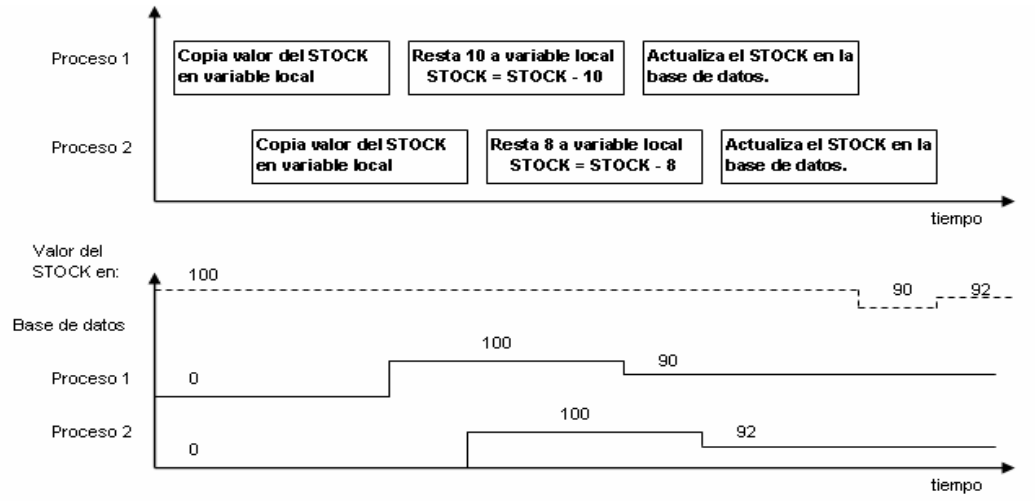
La integridad de los datos pude verse afectada por operaciones que atentan contra las restricciones de integridad y por las interferencias producidas entre accesos concurrentes.

Las restricciones de integridad son definidas en el momento del diseño de la base de datos y son una forma más de captar y almacenar el significado asociado a los datos. Por ejemplo, en la base de datos de una empresa de distribución una restricción de integridad puede ser que en la entidad LINEAS_PEDIDO nunca pueda existir una referencia a un artículo inexistente en la entidad ALMACEN. O dicho de otra forma, sólo pueden incluirse en un pedido artículos registrados en almacén.

El SGBD debe velar por que se cumplan las restricciones de integridad y, siguiendo el ejemplo, la operación de insertar una línea en LINEAS_PEDIDO con una referencia de artículo que no esté en ALMACEN será detectada como un intento de violar dicha restricción y, en consecuencia, será rechazada.

En general, una restricción de integridad irá acompañada de la especificación de la acción a realizar en caso de intento de violación de la misma (rechazo de la operación, en el ejemplo) y de la indicación del momento en que debe realizarse dicha acción.

En sistemas multiusuario el SGBD debe disponer de algún mecanismo de control de concurrencia para evitar las inconsistencias que se pueden producir en la base de datos por acceso concurrente. Un caso típico producido por accesos concurrentes es el conocido como **problema de la operación perdida,** que se ejemplifica en la figura.



El problema de la operación perdida

Una aplicación de gestión de pedidos está siendo ejecutada simultáneamente por dos usuarios, y en un instante dado ambos procesos intentan actualizar el stock en almacén del mismo artículo. Los distintos pasos de los dos procesos se desarrollan paralelamente en el tiempo como se indica en la figura. Al final, la base de datos queda en un estado inconsistente

puesto que el stock en almacén del artículo en cuestión es de 92 unidades cuando en realidad debería ser de sólo 82 unidades. El problema reside en que la actualización del stock que realiza el proceso 2 hace que se pierda la operación de actualización previamente realizada por el proceso 1.

Existen distintas técnicas para evitar los problemas derivados de los accesos concurrentes. Una de estas técnicas consiste en establecer un bloqueo sobre los datos que se están modificando, de forma que, hasta que no finalice dicha modificación, ningún otro proceso podrá acceder a esos datos. Los bloqueos serán gestionados automáticamente por el SGBD, aunque es posible que se permita también a los usuarios establecer bloqueos explícitos.

En el ejemplo anterior, se establecería un bloqueo al iniciarse la transacción del proceso 1 y se levantaría al terminar ésta, de forma que la transacción del proceso 2 actuaría sobre un stock de 90 unidades y no de 100, como ocurría sin bloqueo.

- **Confidencialidad**

Este concepto alude a la protección de los datos contra el acceso de usuarios no autorizados. El SGBD debe proteger los datos de los intentos de acceso de los usuarios que no cuenten con la autorización adecuada, pero en el sistema informático existirán también otros mecanismos ajenos al propio SGBD con esta misma función, como controles del sistema operativo, controles de tipo físico de acceso a las instalaciones, etc.

La implementación del esquema de confidencialidad es responsabilidad del administrador de la base de datos, pero en su diseño intervendrán otras instancias de la empresa u organismo. La decisión sobre qué información es pública y a qué niveles, puede depender de aspectos legales, sociales y éticos, cuestiones de política de la empresa u organismo, etc.

Por lo que respecta exclusivamente al SGBD, existentes dos tipos de gestión de la confidencialidad conocidos como **control de acceso discrecional** y **control de acceso obligatorio**. En ambos casos, la unidad de datos a proteger varía desde una base de datos entera hasta un ítem de datos específico. La diferencia entre ambos tipos estriba en lo siguiente:

- En el caso del **control discrecional**, cada usuario tiene diferentes derechos de acceso (también conocidos por **privilegios** o **autorizaciones**) sobre diferentes objetos. Normalmente, usuarios diferentes tendrán diferentes privilegios en el mismo objeto. Los esquemas discrecionales son muy flexibles.
- En el caso de **control obligatorio**, cada objeto es etiquetado con un cierto **nivel de clasificación** y a cada usuario se le da un **nivel de acreditación**. Un objeto

sólo puede ser accedido por aquellos usuarios con un nivel de acreditación apropiado. Los esquemas obligatorios son comparativamente rígidos.

Las decisiones sobre a qué usuarios se les debe permitir realizar qué operaciones sobre qué objetos son decisiones políticas, no técnicas, por lo que quedan fuera de la jurisdicción del SGBD en sí; lo único que puede hacer el SGBD es obligar al cumplimiento de estas decisiones una vez tomadas. Para que el SGBD pueda llevar a cabo esas funciones adecuadamente ha de cumplirse que:

- El resultado de esas decisiones políticas se dé a conocer al sistema por medio de sentencias en un lenguaje de definición apropiado, y sean recordadas por el sistema registrándolas en forma de reglas o restricciones de **autorización**.
- Exista una forma de verificar una solicitud de acceso dada contra las restricciones de autorización aplicables. (Por "solicitud de acceso" entendemos aquí la combinación de operación solicitada más objeto solicitado más usuario solicitante.)

Para poder decidir qué restricciones son aplicables a una solicitud dada, el sistema debe ser capaz de reconocer el origen de dicha solicitud, es decir, debe ser capaz de reconocer al usuario específico del cual proviene una solicitud. Por esa razón, cuando los usuarios obtienen acceso al sistema casi siempre se les pide proporcionar no sólo su identificador de usuario, sino también una contraseña.

Además de estas técnicas, en algunos SGBD se utiliza la criptografía como medio de protección de la confidencialidad. Esta técnica permite transformar el contenido de la base, haciendo que sea necesario conocer la correspondiente clave de descifrado para hacer uso de la información almacenada.

Complementando la protección de la confidencialidad, los SGBD suelen proporcionar facilidades de auditoria que permiten registrar las operaciones realizadas por los usuarios, pudiendo así detectar accesos no permitidos y mejorar el sistema de protección de la confidencialidad.

5. El diccionario de la base de datos.

La estructura de un SGBD está articulada alrededor del diccionario de datos. En él estará almacenada la definición de los distintos esquemas de la base de datos y de las correspondencias entre ellos.

La información almacenada en el diccionario de datos es imprescindible para realizar las funciones de manipulación y transformación. Igualmente, en el diccionario de datos se registrarán las informaciones necesarias para implementar la función de protección de datos. Un diccionario completo incluirá también información del uso que las distintas aplicaciones y usuarios están haciendo de la base de datos, información del estilo de qué usuarios están conectados a la base de datos, qué aplicaciones hacen uso de qué datos, etc.

Al referirse al diccionario de datos se utilizan con frecuencia otros términos como **directorio de datos** o **catálogo.** Existe una cierta confusión en la terminología utilizada, por lo que con ánimo de aclararla vamos a describir distintos tipos de almacenes de datos.

Un **diccionario de datos** contiene información sobre los datos almacenados en la base de datos desde el punto de vista del usuario. En el diccionario estará almacenado lo que los usuarios necesitan para comprender el significado de los datos y poder manejarlos adecuadamente, como descripciones, significado, estructuras, consideraciones de seguridad, uso por las aplicaciones, etc.

Un **directorio de datos** contiene, en un formato legible para la máquina, las especificaciones necesarias para pasar de la representación externa de los datos a su representación interna. El directorio es un instrumento del SGBD, y esta orientado a facilitarle la información que necesita para su funcionamiento.

A veces, se habla de **diccionario/directorio de datos**, en referencia a paquetes de soporte lógico que incluyen ambas funciones.

Cuando el diccionario de datos está implementado sin ninguna interfaz directa con el directorio, se habla de **diccionario pasivo**. Por contra, cuando el diccionario y el directorio están integrados en un diccionario/directorio que sirve tanto a las necesidades de los usuarios como del SGBD, se habla de **diccionario activo**.

Una **enciclopedia o repositorio** es un diccionario que almacena la información manejada por una herramienta CASE (***C**omputer **A**ided **S**oftware **E**ngineering*). En el repositorio se almacena información textual y gráfica relativa a las distintas fases del ciclo de vida de un sistema. Las enciclopedias no suelen ser activas, o sea, no están integradas con el directorio de datos, pero suelen proporcionar mecanismos que facilitan la tarea de generar las descripciones de los datos propias del directorio a partir de las descripciones lógicas de los datos, obtenidas en la fase de diseño.

Los SGBD relacionales almacenan información sobre los datos contenidos en la base de datos para ser utilizada tanto por el propio sistema como por el usuario. Este diccionario/directorio de datos que implementan los sistemas relacionales se conoce como

catálogo. No obstante, la descripción lógica de los datos almacenada en un catálogo es pobre, incorpora poco significado. Un aspecto importante del catálogo es que su sistema de gestión no es, como en otros tipos de almacenes, un sistema de gestión propio e independiente del SGBD, sino que es el propio SGBD. O dicho de otra forma, al catálogo se accede con las mismas herramientas que al resto de la base de datos. Se dice que el catálogo es una **metabase**, es decir, una base de datos con datos sobre los datos (sobre su estructura) de la base de datos. A los datos del catálogo, también se les denomina **metadatos**.

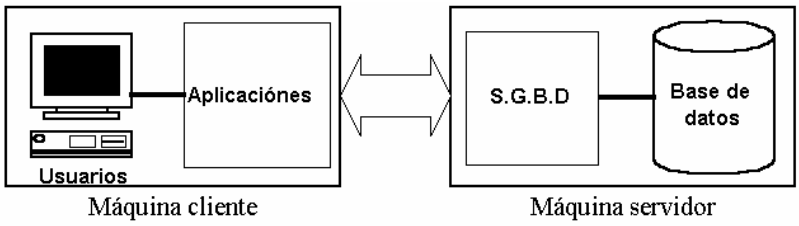
6. Arquitecturas de aplicación.

El objetivo último de un sistema de bases de datos es facilitar el desarrollo y ejecución de aplicaciones. Por tanto, desde un punto de vista amplio, un sistema de bases de datos posee una estructura compuesta de dos partes: un **servidor**, también denominado **backend**, y un conjunto de **clientes** o **frontends**.

Arquitectura cliente/servidor

El servidor permite llevar a cabo todas las funciones propias del SGBD indicadas en los apartados anteriores. Realmente, el servidor es el SGBD mismo.

Los clientes son las distintas aplicaciones ejecutadas entorno al SGBD, tanto las aplicaciones de usuario como las distintas utilidades proporcionadas por el proveedor del SGBD o por otros proveedores. Todas ellas se comunican con el servidor a través de los correspondientes esquemas externos.

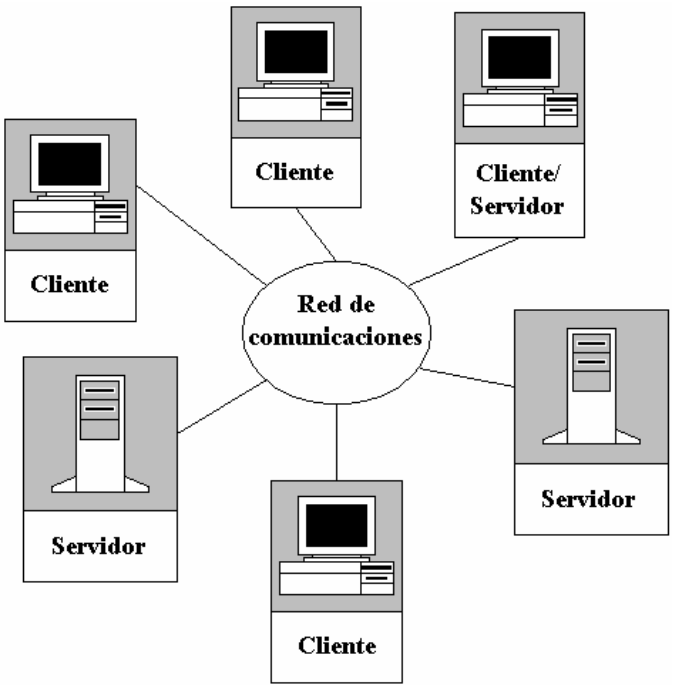


Sistema de B.D. con varias máquinas.

Estas dos partes en que se divide el sistema pueden ejecutarse en la misma máquina o en máquinas distintas interconectadas a través de algún sistema de comunicación. Esta segunda posibilidad, el servidor ejecutándose en una máquina y el cliente en otra, es a la que se suele aplicar el término cliente/servidor, hablándose de máquina cliente y máquina servidor o servidor de base de datos.

La arquitectura cliente/servidor nos lleva a lo que se conoce como **proceso distribuido**, esto es, al procesamiento de los datos distribuido sobre varias máquinas conectadas entre sí a través de una red de comunicaciones. El proceso distribuido exige la presencia de algún software que se encargue de gestionar las comunicaciones entre las distintas máquinas que participan en el proceso. Estos programas de comunicaciones harán de interfaz entre el software propio del sistema de bases de datos y el de gestión de la red.

También estaríamos ante un caso de proceso distribuido cuando una máquina con más de un procesador separa la ejecución de las aplicaciones y del S.G.B.D. en distintos procesadores. No obstante, esto no es lo más frecuente.



Sistema de B.D. con varias máquinas servidores y clientes

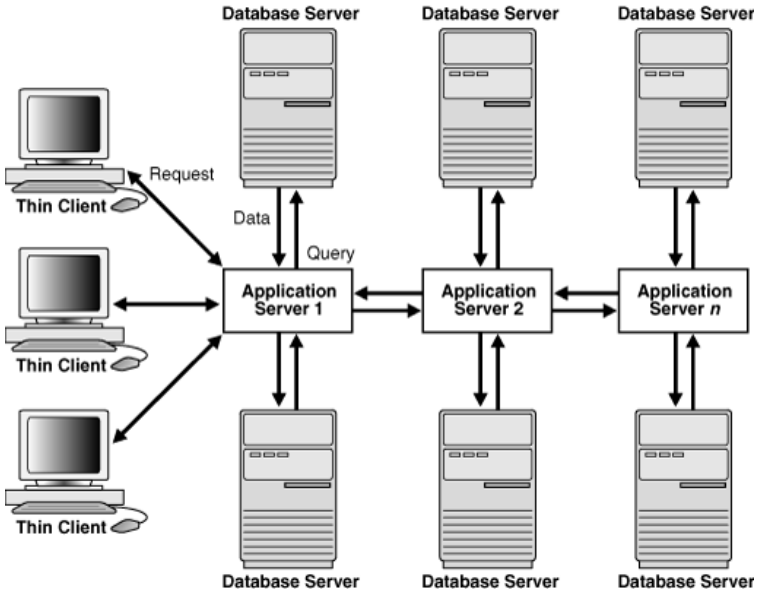
Existen distintas posibilidades de proceso distribuido según se repartan las partes cliente y servidor sobre las distintas máquinas. En la figura podemos ver un ejemplo en el que se conjugan las distintas posibilidades. Unidos por una red de comunicaciones, que en realidad podrían ser varias redes interconectadas, tenemos varios clientes y servidores. A cada servidor pueden tener acceso simultáneo varias máquinas clientes y también es posible que una máquina cliente pueda acceder a más de un servidor. Igualmente hemos representado un caso de máquina que actúa como cliente de un SGBD local, es decir, que ejecuta tanto la parte cliente como la parte servidor de un sistema de base de datos local a ella. Sería posible, que esta misma máquina actuase como cliente de otros servidores o que gestionase el acceso a sus datos por parte de otros clientes.

El caso de una máquina cliente con acceso a más de un servidor puede presentarse en dos variantes:

- El cliente sólo puede conectarse a un servidor a un tiempo. No es posible en una sola consulta obtener datos de más de un servidor.

- El cliente puede acceder simultáneamente a varios servidores. En una sola consulta se pueden combinar datos disponibles en distintos servidores. Desde el punto de vista del cliente, desde una perspectiva lógica, es como si sólo existiera un único servidor. Este caso se conoce como **sistema de base de datos distribuida**.

En un sistema de base de datos distribuida, una aplicación debe ser capaz de acceder a los datos sin tener que saber que éstos pueden estar distribuidos en varias bases de datos, soportadas por diferentes sistemas gestores, que se ejecutan en diferentes máquinas, con diversos sistemas operativos y conectadas por diferentes redes. Se dice, que esta distribución de los datos es **transparente** para las aplicaciones.



*Arquitectura **multitier**.*

Otra arquitectura ampliamente utilizada en la actualidad es la conocida como **multinivel** o **multicapa (***Multitier Architecture*), que se utiliza en aplicaciones WEB, tanto en intranets como en Internet. En esta arquitectura la parte servidor se sigue ejecutando en el servidor de base de datos, pero la parte cliente se reparte en dos o más niveles: uno o varios servidores de aplicaciones y las máquinas clientes. La mayor parte del proceso de la aplicación se realiza de una forma centralizada en el servidor de aplicación, quedando para las máquinas clientes la parte de la aplicación que actúa de interfaz con el usuario.

7. Tipos de usuarios de un sistema de bases de datos.

Veremos a continuación los distintos tipos de de usuarios que interactúan con un sistema de bases de datos, relacionándolos con las facilidades que el sistema les proporciona para cubrir sus necesidades.

Usuarios finales.- Se trata de usuarios, con poco o ningún conocimiento informático, que acceden a la base de datos a través de:

- Aplicaciones diseñadas específicamente para atender sus necesidades. Por ejemplo, un cliente de una empresa de distribución tiene acceso a una aplicación Web que le permite realizar pedidos, imprimir facturas, ver el estado de sus pedidos, etc. Todas estas operaciones dan lugar a la inserción y manipulación de información en la base de datos de una forma totalmente transparente al usuario.
- Un lenguaje de manipulación de datos autocontenido, que permite al usuario realizar operaciones de recuperación y actualización de información de la base de datos. Por ejemplo, si el sistema de base de datos de la empresa de distribución está montado sobre un servidor Oracle, el personal de administración de la empresa podrá saber el número de pedidos pendientes de proceso escribiendo una consulta en lenguaje SQL desde el entorno SQL*Plus. El uso de lenguajes de manipulación exige el conocimiento de dicho lenguaje y del esquema externo al que se accede, pero facilita el acceso inmediato a la base de datos sin necesidad de desarrollar una aplicación específica.
- Utilidades que permiten realizar consultas y modificaciones a la base de datos sin necesidad de conocer un lenguaje de manipulación. Estas utilidades, basadas en menús y formularios, permiten que los usuarios satisfagan necesidades de información no previstas de antemano y para las que no existen aplicaciones desarrolladas.

Los lenguajes de manipulación son facilidades suministradas por el propio SGBD. Las utilidades pueden formar parte del SGBD o ser aplicaciones externas a él.

Programadores de aplicaciones.- Los programadores que desarrollan aplicaciones en entornos de base de datos pueden llevar a cabo su trabajo de distintas formas:

- Escribiendo sus programas en un lenguaje de tercera generación en el que se intercalan sentencias en un lenguaje de manipulación de datos. Las sentencias del lenguaje LMD, denominado **lenguaje huésped**, se encargarán de las labores de acceso a la base de datos y las sentencias del lenguaje de tercera generación, o **lenguaje anfitrión**, se ocupan del resto de las tareas encomendadas al programa. La aplicación Web puesta a disposición de los usuarios del punto anterior podría estar desarrollada en lenguaje JAVA (lenguaje anfitrión) con sentencias en lenguaje SQL (lenguaje huésped) para acceder a la base de datos.
- Desarrollando sus aplicaciones por medio de generadores de pantallas, generadores de informes y lenguajes de cuarta generación. Los generadores de pantallas e informes permiten generar el código de la aplicación de una forma

rápida a partir del diseño realizado por el programador utilizando las facilidades gráficas que la herramienta pone a su disposición. También pueden escribir código utilizando un lenguaje de cuarta generación (L4G). Los lenguajes L4G incluyen entre sus sentencias aquellas que manejan el acceso a los datos de la base. Por ejemplo, nuestra empresa de distribución podría disponer de una aplicación para llevar a cabo todo el proceso de gestión, cuyos módulos se hayan desarrollado utilizando el generador de formas y menús de la firma Oracle (*Oracle Forms*) y el generador de informes de informes y gráficos de la misma empresa (*Oracle Reports*). También se habrá escrito código utilizando el lenguaje Oracle PL/SQL.

- Empleando para escribir sus programas el lenguaje LMD autocontenido, sin apoyo de un lenguaje anfitrión. Esta es una posibilidad real pero poco usada por los programadores, dado que se suele obtener una mayor eficiencia utilizando el lenguaje LMD como huésped de otro lenguaje o utilizando un lenguaje L4G.

El administrador de la base de datos (DBA o DataBase Administrador).- Es el usuario, o usuarios, responsable del diseño, creación y mantenimiento de la base de datos.

Para facilitarle la tarea de diseño, el sistema puede poner a su disposición herramientas de ayuda, como las herramientas CASE (*Computer Aided Software Engineering*).

El SGBD pondrá a disposición del DBA varios lenguajes de definición de datos (lenguajes LDD) mediante los que especificará los distintos esquemas (externos, conceptual e interno) de la base de datos para su compilación y almacenamiento en el directorio de datos. Es normal que exista un lenguaje para cada nivel de abstracción, pero también es posible que se utilice el mismo para más de un nivel. Mediante estos lenguajes también se especificarán las restricciones de integridad y de confidencialidad.

Dentro del mantenimiento de la base de datos está incluida la modificación de su estructura, a cualquier nivel, cuando sea necesario. Cuando haya que añadir nuevos objetos o modificar los existentes, se utilizarán las sentencias adecuadas de los lenguajes LDD.

También, los SGBD suelen disponer de utilidades o programas de servicio diseñados para ayudar al DBA en la explotación del sistema. Estos programas realizan funciones de modificación del tamaño de los ficheros, obtención de estadísticas de utilización, carga de ficheros, obtención de distintos tipos de copias de seguridad de la base de datos, recuperación tras una caída del sistema, etc. Algunas de estas utilidades, aunque suministradas por el mismo proveedor del SGBD, puede que trabajen en el nivel externo del sistema, por lo que son en realidad programas de aplicación especial. Otras trabajan directamente en el nivel interno, formando parte realmente del SGBD.

El administrador de la base de datos de la empresa de distribución que venimos presentando como ejemplo, dispondrá de todas las sentencias SQL implementadas en el SGBD Oracle para definición y modificación de todo tipo de objetos, tanto lógicos como físicos. Utilizando las sentencias de este sistema, el DBA podrá definir o modificar objetos del nivel lógico de la base de datos, como tablas y vistas, establecer restricciones de integridad o controlar el acceso a los objetos de la base. También, podrá definir y modificar objetos del nivel físico, como *tablespaces* o *rollback segments*. Si dispone de la utilidad *Oracle Enterprise Manager*, podrá realizar muchas de las tareas de administración sin necesidad de escribir líneas SQL. Para facilitar la carga en la base de datos contenidos en ficheros tendrá a su disposición la utilidad *SQL*Loader*. Las operaciones de exportación e importación de información entre bases de datos Oracle se pueden llevar a cabo con las utilidades *Export* e *Import.*

8. Modelos de datos.

En una base de datos se representa una parte del mundo real con el objetivo de que dicha representación pueda ser utilizada por distintas aplicaciones actuales y futuras. Esta representación de la parcela del mundo real objeto de interés se hará de acuerdo con unas reglas, que varían de unas bases de datos a otras y que constituyen el modelo de datos.

El **universo del discurso** es la parte o aspecto del mundo real que el diseñador toma en consideración y que tras el adecuado diseño se plasmará en una base de datos. Por ejemplo, en una empresa pueden tomarse en consideración distintos universos del discurso como el de gestión comercial, el de gestión de personal o el de gestión de suministros. Todos tienen el mismo mundo real de referencia, la empresa, pero cada uno contempla distintos aspecto del mismo.

El estudio y descripción del universo del discurso se lleva a cabo con el apoyo de un conjunto de conceptos, reglas y convenciones que reciben el nombre de **modelo de datos**.

Existen distintos modelos de datos, no todos implementados en sistemas de bases de datos. Todos ellos son una herramienta que facilita la interpretación y representación del universo del discurso. La representación del universo del discurso obtenida por aplicación de un modelo de datos será una estructura de datos denominada **esquema**.

Al aplicar una cierta sintaxis al modelo se obtiene un **lenguaje de datos**. Distintas sintaxis aplicadas al mismo modelo dan lugar a distintos lenguajes. Por ejemplo, el álgebra relacional y el cálculo relacional son dos lenguajes del mismo modelo de datos, el relacional, totalmente equivalentes, que sólo difieren en su sintaxis. El esquema estará descrito mediante un lenguaje de datos propio del modelo aplicado al universo del discurso.

En el universo del discurso es preciso distinguir entre las estructuras y los datos o valores que se almacenan en esas estructuras. Las primeras son propiedades estáticas o que varían poco con el tiempo, las segundas son propiedades que varían con el tiempo o dinámicas. Todo modelo de datos tendrá una parte estática que permitirá representar las estructuras y una parte dinámica que permitirá representar las propiedades dinámicas del universo del discurso. Asociado a la parte estática del modelo habrá un **lenguaje de definición de datos (LDD)** y asociado a la dinámica existirá un **lenguaje de manipulación de datos (LMD)**.

La estática del modelo constará de una serie de objetos con los que representar las estructuras del mundo real. Estos objetos varían de unos modelos a otros, varían sus nombres y varía su representación. Asociados a los objetos pueden existir restricciones que hagan que no sea posible representar, en ese modelo, ciertos objetos o asociaciones entre ellos. Se suele distinguir entre restricciones inherentes, que son las que vienen impuestas por la definición del modelo, y restricciones de usuario, que son las introducidas por el diseñador en el esquema como representación fiel del universo del discurso.

La representación de las propiedades estáticas del universo del discurso, obtenida aplicando la parte estática de un modelo de datos, constituye el esquema de dicho universo. El conjunto de los valores que toman los distintos objetos del esquema en un instante dado se denomina **ocurrencia del esquema** o base de datos. La ocurrencia del esquema varía en el tiempo, ya que, como consecuencia de operaciones de actualización sobre la base de datos, los objetos del esquema van tomando distintos valores a lo largo del tiempo.

La parte dinámica de un modelo de datos estará constituida por un conjunto de operaciones, definidas sobre los objetos de su parte estática, que permiten la transformación entre ocurrencias del esquema. Habrá operaciones que permiten localizar una ocurrencia o conjunto de ocurrencias de un objeto del esquema y otras que permitirán realizar una determinada acción (recuperación, modificación, inserción o borrado) sobre la ocurrencia u ocurrencias previamente localizadas. Las primeras se conocen genéricamente como **operaciones de selección** y las segundas como **operaciones de acción**.

9. Tipos de modelos de datos

De acuerdo con los niveles de abstracción de la arquitectura ANSI, se distinguen tres tipos de modelos:

- **Modelos externos**, que sirven para construir esquemas externos o de usuario.

- **Modelos conceptuales**, que permitirán la construcción de esquemas lógicos de todo el universo del discurso.
- **Modelos internos**, que se utilizarán para definir el esquema interno o físico de la base de datos.

Los dos primeros tipos de modelos se pueden considerar como **modelos lógicos**, ya que se ocupan de representar aspectos lógicos del universo del discurso omitiendo cualquier aspecto físico.

Algunos autores dividen los modelos lógicos en **conceptuales** y **convencionales**. No está clara la separación entre estos dos tipos de modelos. Los modelos convencionales son los implementados actualmente en sistemas gestores de bases de datos, como el modelo jerárquico, el modelo en red o el modelo relacional. Los modelos conceptuales poseen un mayor nivel de abstracción que los convencionales y como consecuencia una mayor flexibilidad y una mayor capacidad semántica. Ejemplos de modelos conceptuales serían el modelo de entidad/interrelación de Chen o el modelo Relacional/Tasmania (RM/T).

Los modelos convencionales sirven de conexión entre los esquemas externos y el interno, actuando como interfaz entre el informático y el sistema.

Dado el mayor nivel de abstracción de los modelos conceptuales, éstos sirven como interfaz entre el informático y el usuario final en las primeras fases del proceso de diseño de bases de datos. No obstante, dado que algunos modelos conceptuales, como el modelo E/R, están implementados en herramientas CASE y empiezan a formar parte de algunos SGBD, también pueden actuar como conexión entre los esquemas externos e interno.

10. Modelos convencionales.

Son los que tienen implementación en SGBD reales, por lo que también podríamos referirnos a ellos como modelos de bases de datos. A continuación, veremos las características generales del modelo de datos jerárquico y del modelo de datos en red, que son el fundamento de distintos sistemas de bases de datos comerciales que dominaron el mercado en los años setenta y ochenta del siglo pasado. El modelo de datos relacional, en el que se apoyan los productos utilizados en la actualidad, será objeto de un amplio estudio en los puntos que siguen hasta el final de la unidad.

El modelo de datos en red representa las entidades del universo del discurso como nodos de un grafo y las interrelaciones entre ellas como los arcos que unen los nodos. No existe ninguna restricción ni en el tipo ni en la cantidad de los arcos y, por tanto, es posible

representar estructuras de datos tan complejas como sea preciso. En este modelo se pueden representar directamente interrelaciones de cualquier cardinalidad y de cualquier grado: reflexivas, binarias, ternarias, etc.

La flexibilidad del modelo en red trae como consecuencia que su implementación sea difícil y poco eficiente, por lo que es preciso aplicarle ciertas restricciones a la hora de llevarlo a la práctica. Los modelos CODASYL y jerárquico pueden considerarse como derivados del modelo en red al establecer unas restricciones concretas.

❑ El modelo CODASYL.

El grupo DBTG (***Data Base Task Group***) del comité CODASYL (***COnference on DAta SYstem Languages***) presenta entre los años 1969 y 1981 distintos informes en los que se especifica un modelo de datos en red que seguirán con mayor o menor fidelidad distintos productos.

En el modelo CODASYL, los nodos son tipos de **registro** (*record*, en su nomenclatura) que están formados por **campos** (*data item*). Dentro de un registro pueden existir **agregados de datos** (*data aggregate*), que pueden ser grupos repetitivos o vectores. Los arcos o interrelaciones entre entidades reciben el nombre de **conjuntos** (*SET*).

El conjunto o SET de este modelo sólo permite representar interrelaciones **1:N** o **1:1**. Uno de los nodos del conjunto, el que participa con la cardinalidad uno, se denomina **propietario** (*owner*) y los demás son **miembros** (*members*). En un conjunto siempre existe un propietario y cero o más miembros. Los registros miembros de un conjunto pueden ser todos del mismo tipo o de tipos distintos. Un registro puede participar en más de un conjunto, y lo puede hacer con distinta categoría en cada uno, es decir, puede ser miembro en un conjunto y propietario en otro.

Como consecuencia de las restricciones inherentes, el modelo CODASYL no permite representar directamente interrelaciones reflexivas, ni interrelaciones N:1, ni tampoco de tipo N:M. Estas restricciones se salvan por el uso de ciertos mecanismos, como los registros de enlace o *links*, en los que no vamos a entrar. La limitación en la representación de las interrelaciones reflexivas fue eliminada en las especificaciones CODASYL de 1978.

El modelo presenta, además de los elementos lógicos ya citados, elementos de naturaleza física que el usuario debe definir y manejar.

Las especificaciones CODASYL plantean inicialmente una arquitectura de sistema gestor de base de datos a dos niveles (esquema y subesquemas), pasando posteriormente a tres niveles para mejorar la independencia físico/lógica.

Dispone de un **lenguaje de definición de datos del esquema** (*Schema DDL*) autocontenido que permite la definición de todos los elementos que forman parte de la base de datos.

De igual forma, CODASYL especifica un **lenguaje de definición de datos del subesquema** (*Subschema DDL*) huésped, inicialmente de COBOL y luego de otros lenguajes, que permite la descripción de subconjuntos del esquema para distintas aplicaciones que sólo necesitan trabajar sobre una parte de la base de datos.

En el informe de 1978 se introduce un nuevo nivel en la arquitectura del SGBD, **el esquema de almacenamiento**, con lo que se sacan del esquema muchos de los aspectos físicos de la base de datos. Para definición de este nuevo nivel aparece un nuevo lenguaje, el **lenguaje de definición del esquema de almacenamiento** (DSDL), que se va a ocupar de la definición de todos los aspectos físicos del almacenamiento de los datos.

El **lenguaje de manipulación de datos** (DML) definido por CODASYL implementa la parte dinámica del modelo. Este lenguaje, que es de tipo navegacional, puede ir embebido en distintos lenguajes de tercera generación (COBOL, C, FORTRAN, PL/1), y permite que el usuario pueda navegar por la base de datos seleccionando el registro sobre el que se hará una determinada operación de borrado, modificación, recuperación, etc.

Dos implementaciones populares del modelo CODASYL lo constituyen los sistemas de bases de datos Total e IDMS.

❑ El modelo jerárquico.

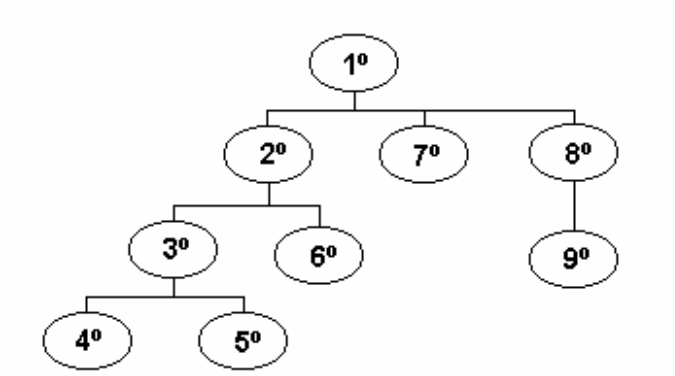
En este modelo de datos se apoyaron algunos de los primeros sistemas gestores de bases de datos comerciales. El modelo carece de estandarización, como el caso del CODASYL para el modelo en red, pero el sistema comercial jerárquico más importante, el IMS (***Information Management System***) de IBM, constituye su estándar de facto.

El esquema en el modelo jerárquico es una estructura en árbol formada por nodos, que representan entidades, y arcos, que representan las interrelaciones entre las entidades. Cada nodo será un registro con sus correspondientes campos, como en el modelo CODASYL.

Este modelo puede ser visto como un caso particular del modelo en red, al que se somete a una serie de restricciones inherentes para conseguir las características de árbol ordenado.

El modelo jerárquico presenta muchos problemas a la hora de ser utilizado para modelar el universo del discurso. Estos problemas vienen derivados fundamentalmente de la falta de capacidad de la estructura jerárquica para representar ciertas estructuras muy corrientes en la realidad, como las interrelaciones N:M, las interrelaciones reflexivas, las redes, etcétera. Esta falta de flexibilidad del modelo obliga a la introducción de redundancias cuando se quieren introducir en el esquema situaciones como las citadas.

La parte dinámica del modelo estará implementada en los sistemas de bases de datos jerárquicas por medio de un lenguaje de manipulación de datos huésped de un lenguaje de tercera generación. De estos lenguajes de manipulación, el más conocido es el DL/I del sistema IMS.



Orden de recorrido de un árbol en preorden

La manipulación de datos en el modelo jerárquico, igual que en todo modelo, necesita la selección previa de los datos sobre los que se realizará a continuación la acción de recuperación o actualización.

La operación de selección en el modelo jerárquico es de tipo navegacional, semejante a la del modelo CODASYL. El funcionamiento de algunos formatos de la operación de selección va unida al concepto de recorrido del árbol en un cierto orden, normalmente **preorden**. Recorrer un árbol o subárbol en preorden significa comenzar por la raíz, luego el subárbol izquierdo, y finalmente el derecho; cada subárbol se recorrerá también en preorden.

La recuperación en DL/I, que va asociada a la selección, consiste, como en cualquier otro lenguaje de datos, en llevar el registro marcado como activo a la correspondiente plantilla de entrada/salida. Cuando se inserta un nuevo registro en una estructura jerárquica, excepto para la raíz, queda conectado a un registro padre seleccionado mediante una sentencia de selección. Cuando se borra un registro de un árbol, se borran también todos sus hijos.

11. Características generales del modelo de datos relacional.

El modelo relacional es propuesto por Edgar Frank Codd en 1970 a través de un artículo (*"A Relational Model of Data for Large Shared Data Banks"*) publicado en el boletín de la ACM (*Communications of the Association for Computer Machinery*). Codd propone un modelo de

datos basado en la teoría matemática de las relaciones, en el que los datos se estructuran en el ámbito lógico en forma de relaciones. En el artículo se concede gran importancia al hecho de que la estructura lógica de los datos sea independiente de su almacenamiento interno, cosa que los modelos de la época no recogían y que constituye el objetivo fundamental del modelo relacional.

El modelo relacional es un modelo lógico que adopta como estructura fundamental la **relación**. Una relación puede ser vista y tratada como una tabla de datos siempre que se tomen ciertas precauciones que más adelante mencionaremos. Todos los datos de una base de datos relacional se representan en forma de relaciones cuyo contenido varía con el tiempo.

El modelo relacional, como todo modelo de datos, tiene una parte estática constituida por los objetos permitidos y las restricciones, y una parte dinámica o conjunto de operaciones definidas sobre la estructura, que permite la transición entre estados de la base de datos. La parte dinámica del modelo está constituida por el **álgebra relacional** y el **cálculo relacional**.

Se apoya en un sólido fundamento matemático, lo que facilita su estudio y desarrollo y, además, resulta fácil de comprender y usar debido a que permite presentar la información en forma de tablas.

Tabla	Relación
Fila	Tupla
Columna	Atributo

Tablas y relaciones

A partir de su definición original, el modelo ha ido evolucionando por aportación de muchos autores. Unas partes del modelo son más sólidas y se mantienen prácticamente sin cambios, mientras que otras son aún motivo de gran controversia entre distintos autores.

12. Estructura del modelo.

El elemento fundamental del modelo relacional, la **relación**, puede ser representada como una tabla de datos, de forma que cada fila de la tabla sería una **tupla** de la relación y cada columna representaría un **atributo** de la relación. El número de filas de la tabla constituye la **cardinalidad** de la relación y el número de columnas es el **grado** de la relación. Se puede decir que en una base de datos relacional el usuario percibe los datos estructurados en forma de tablas.

El valor que adopta en cada tupla un atributo es tomado de un conjunto de valores, conocido como **dominio,** relacionado con él. Varios atributos pueden tomar sus valores de un mismo dominio, pero cada atributo toma valores de un único dominio.

Para que una tabla represente con fidelidad a la correspondiente relación se deben tener en cuenta ciertas características que diferencian a una relación de una tabla:

- Una relación no tiene tuplas duplicadas; por tanto, en una tabla nunca puede haber filas duplicadas.
- Las tuplas de una relación no tienen orden, luego el orden de las filas de una tabla no es significativo.
- Cada atributo de una tupla toma un solo valor del dominio correspondiente, por lo que en una tabla no puede haber más de un valor en cada celda.

❑ Dominios.

Un **dominio** es un conjunto finito de valores homogéneos y atómicos caracterizados por un nombre.

Los atributos de una relación toman sus valores de dominios asociados a ellos. Por ejemplo, en la base de datos de nuestra empresa de distribución existirá una relación con los datos de sus clientes. En la figura mostramos las columnas que podría tener la tabla junto con sus dominios correspondientes.

Columnas/ Atributos	Dominios subyacentes
CODIGO	Cualquier combinación de cuatro dígitos.
CIF	Conjunto de todos los NIF's y CIF's posibles.
NOMBRE	Conjunto de todas las combinaciones de letras que den lugar a nombres de clientes.
DIRECCION	Conjunto de todas las combinaciones de letras que formen direcciones postales.
LOCALIDAD	Conjunto de todas los nombres de localidades españolas.
PROVINCIA	Conjunto de todos los nombres de provincias.
COD_POSTAL	Conjunto de los códigos postales.

Los valores de los dominios deben ser **homogéneos,** es decir, todos los valores de un dominio tienen que ser del mismo tipo. El dominio correspondiente a PROVINCIA no podría contener el valor "MIERES", o el valor "33600".

Los valores de los dominios también deben ser **atómicos,** lo que significa que serán indivisibles desde el punto de vista semántico, es decir, que si se descompusieran perderían el significado asociado a ellos en el modelo. Por ejemplo, en el dominio LOCALIDAD cada una de las tres palabras que componen el valor "Santiago de Compostela" tienen un significado distinto al conjunto.

Los dominios participan en las relaciones a través de los atributos. Se puede definir **atributo** diciendo que es el papel que un dominio toma en una relación. Todo atributo debe tomar sus valores de un único dominio, que se denomina **dominio subyacente**.

Las implementaciones comerciales del modelo relacional no incorporan, o lo hacen de una forma deficiente, el concepto de dominio, pese a ser un elemento muy importante del modelo.

Algunos autores hablan de **dominios compuestos** en contraposición al concepto de **dominio simple** definido aquí. Un dominio compuesto es una combinación de dominios simples que tiene un nombre y, quizá, ciertas restricciones de integridad asociadas. El ejemplo típico es un dominio FECHA, compuesto por tres dominios simples: DIA, MES y AÑO, que lleva asociadas las restricciones adecuadas para que, de las posibles combinaciones de valores de los tres dominios simples, no se admitan como validas aquellas que dan lugar a fechas inexistentes.

Otra forma de ver un dominio es considerarlo como un **tipo de datos definido por el usuario** de un lenguaje de programación. Desde este punto de vista, podemos decir que los sistemas gestores de bases de datos relacionales implementan los dominios de una forma primitiva, puesto que suelen soportar tipos de datos primarios como INTEGER, CHAR, etc. Estos tipos de datos son **tipos definidos por el sistema** y el concepto de dominio exigiría que el sistema permitiera al usuario definir sus propios tipos de datos, tales como “nombre de localidad”, “código de cliente”, etc.

Para algunos autores, como C.J. Date, el concepto de dominio compuesto está inmerso en otro más amplio. Según Date, no hay nada en el modelo relacional que exija que los valores de un dominio se limiten a tipos sencillos como números, cadenas, etc.; pueden ser tan complejos como se quiera, la única limitación es que su estructura sea invisible al SGBD (valores encapsulados). Incluso, podríamos tener un dominio compuesto de relaciones. Al definir uno de estos dominios es preciso definir los operadores, o funciones, que manejen los valores del dominio. Estos operadores son los únicos que pueden ver la estructura interna de los valores del dominio, pero el SGBD no la ve. En la definición de un dominio pueden entrar otros dominios definidos previamente.

- **Relaciones.**

Dados los conjuntos D_1, D_2, ..., D_n, no necesariamente distintos, R es una relación sobre esos n conjuntos si es un conjunto de n-tuplas, cada una de las cuales obtiene su primer elemento de D_1, su segundo elemento de D_2, y así sucesivamente. Se dice que D_j es el j-ésimo dominio de R. Se puede expresar este mismo concepto diciendo que R es un subconjunto del producto cartesiano D_1**X**D_2**X**....**X**D_n de sus dominios.

La definición matemática anterior, que es la definición original dada por Codd, presenta algunos aspectos criticables desde el punto de vista de las bases de datos, como es el que no se distinga claramente entre dominios y atributos, y que exija un orden en los elementos componentes de cada tupla. Una definición más adecuada de relación es la siguiente:

Una relación R sobre un conjunto de dominios D_1, D_2, ..., D_n, no necesariamente distintos, se compone de dos partes, una cabecera y un cuerpo.

- La **cabecera**, denominada también **intensión o esquema de la relación**, es un conjunto de n pares atributo-dominio subyacente, siendo n el **grado** de la relación.
- El cuerpo, extensión, instancia u ocurrencia de la relación es un conjunto de *m* n-tuplas (o, abreviadamente, tuplas) donde cada n-tupla es un conjunto de *n* pares atributo-valor. El valor asociado a cada atributo es tomado de su dominio subyacente. El número de n-tuplas, *m*, es la **cardinalidad** de la relación.

La cabecera es la parte estática de la relación; define a la relación, y un cambio en ella significa que tenemos otra relación distinta.

El cuerpo esta formado por el conjunto de tuplas que en un instante determinado satisfacen el correspondiente esquema de relación. La extensión de una relación, al contrario que su intensión, varía con el tiempo.

De la definición de relación se deducen las siguientes propiedades:

- En una relación no existen tuplas repetidas.
- Las tuplas de una relación no están ordenadas.
- Los atributos de una relación no tienen orden.
- Todos los valores de los atributos son atómicos.

13. Restricciones del modelo.

El modelo relacional, como todo modelo de datos, tiene unas restricciones, es decir, unas estructuras u ocurrencias no permitidas. Estas restricciones pueden ser inherentes y de usuario.

Las cuatro propiedades derivadas de la definición de relación, junto con la **regla de integridad de entidad** constituyen las **restricciones inherentes** del modelo.

Una **restricción de usuario** es un predicado definido sobre un conjunto de dominios, atributos o tuplas, que debe ser verificado por los correspondientes objetos para que constituyan una ocurrencia válida del esquema. Entre las restricciones de usuario se incluye la **regla de integridad referencial**.

Los datos almacenados en la base han de adaptarse a las estructuras impuestas por el modelo (restricciones inherentes), y han de cumplir las restricciones de usuario para ser ocurrencias válidas.

- **Claves.**

Algunas de las restricciones están definidas en torno a los conceptos de clave candidata, clave primaria o clave ajena.

Clave candidata de una relación es un subconjunto no vacío de atributos de esa relación, digamos K, que en todo instante de tiempo satisface las dos propiedades siguientes:

- Unicidad: **en cualquier momento no existen dos tuplas de la relación con el mismo valor de K.**
- Irreductibilidad: **No existe ningún subconjunto de atributos de K que cumpla la propiedad de unicidad.**

CLIENTES

CODIGO	CIF	NOMBRE	DIRECCION	LOCALIDAD	PROVINCIA	COD_POSTAL
1111	111111A	SISTEMAS INFORMATICOS S.L.	PEZ 13	MADRID	MADRID	28005
2222	2222222B	SISINFO S.A.	TRIBULETE, 14	MADRID	MADRID	28006
3333	3333333C	BYTES C.B.	MAYOR, 41	ALCOBENDAS	MADRID	28123
...	.....	.....	.....	.....	.....	.....

Esta relación tendría como claves candidatas: CODIGO y CIF.

El atributo NOMBRE podría ser clave candidata si aceptamos que no es posible tener dos clientes con el mismo nombre.

Probablemente, por simplicidad, escogeríamos CODIGO o CIF como clave primaria. Las claves candidatas no elegidas pasan a ser claves alternativas.

Claves candidatas, primaria y alternativas

Toda relación tiene por lo menos una clave candidata, ya que de la propiedad de que no pueden existir dos tuplas iguales se deriva que el conjunto de todos los atributos de la relación cumplirá siempre la propiedad de unicidad y dicho conjunto, o algún subconjunto de él, será clave candidata.

Clave primaria es una de las claves candidatas que se elige como tal según criterios ajenos al propio modelo. En principio es obligatorio elegir la clave primaria, aunque hay autores que consideran que el papel de la clave primaria debería ser desempeñado por las claves candidatas.

La clave primaria, o las claves candidatas siguiendo el criterio de los autores discrepantes, constituye el mecanismo básico de direccionamiento a nivel de tuplas en un sistema relacional.

Una vez elegida clave primaria, el resto de claves candidatas reciben el nombre de **claves alternativas.**

Clave ajena de una relación R_2 es un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de una relación R_1 (R_1 y R_2 pueden ser la misma relación). R_1 es la relación referenciada y R_2 la relación que referencia.

La clave ajena y la clave primaria a la que apunta deben estar definidas en el mismo dominio.

Si seguimos el criterio de que no es obligatoria la existencia de clave primaria, la clave ajena de una relación debería apuntar a una clave candidata.

ALMACEN

REFERENCIA	DENOMINACION	P_TARIFA	
0001	MONITOR 15" SONY	250,5	....
0002	MONITOR 17" SONY	280,25	....
0003	MONITOR 19" SONY	301,2	....
0004	MONITOR 21" SONY	332,2	....
0005	TECLADO GENIUS INT	57	....
...	.....	...	....

Clave primaria: REFERENCIA

PEDIDOS

NUM_PEDIDO	FECHA	CODIGO_CLIENTE
000010	14/11/2008	5555
000011	14/11/2008	7777
000012	16/11/2008	2222

Clave primaria: NUM_PEDIDO.

L_PEDIDOS

PEDIDO	REFERENCIA	CANTIDAD
000010	0005	5
000010	0003	10
000011	0006	10
000011	0005	2
000012	0005	4

Clave primaria: concatenación de PEDIDO y REFERENCIA.

Claves ajenas: PEDIDO, que apunta a NUM_PEDIDO de PEDIDOS y REFERENCIA que apunta a REFERENCIA de ALMACEN.

Claves primarias y ajenas

- **Regla de integridad de entidad.**

La **regla de integridad de entidad** dice que "ningún componente de la clave primaria de una relación puede ser nulo". Esta regla debería aplicarse también a las claves candidatas, pero el modelo no lo exige.

Surge aquí el concepto de **nulo**, muy controvertido en el modelo relacional. Un nulo no es un valor, es más bien una marca que se aplica a un atributo indicando que éste no tiene valor. El que un atributo no tenga valor puede obedecer a causas diversas, por ejemplo, a que no conocemos su valor, o a que en esa tupla concreta el atributo no es aplicable.

- **Regla de integridad referencial.**

La segunda regla general de integridad del modelo relacional, la de **integridad referencial**, dice que "la base de datos no debe contener valores de clave ajena sin concordancia". Es decir, que una clave ajena debe coincidir con un valor de clave primaria de la relación a la que apunta o tener valor nulo.

Se puede observar que los conceptos de clave ajena e integridad referencial se han definido uno en términos de otro. No es posible hablar de clave ajena sin mencionar el concepto de integridad referencial, ni definir integridad referencial sin hablar de claves ajenas.

Cualquier estado de la base de datos que no satisfaga la regla de integridad referencial, así como cualquier otra restricción, será incorrecto por definición. Ahora bien, la regla no dice nada de cómo evitar estos estados incorrectos. Una solución sería rechazar cualquier operación que intente llevar a la base de datos a un estado que viole la regla de integridad referencial. Otra opción menos drástica sería que el sistema aceptara la operación pero llevara a cabo, si fuese necesario, operaciones adicionales que dejasen la base de datos en un estado válido. Por tanto, al diseñar una base de datos, además de definir las claves ajenas, es preciso especificar qué debe hacer el sistema en caso de que se presenten operaciones de borrado o modificación que atenten contra la integridad referencial.

- **Otras restricciones de usuario.**

Algunos autores distinguen una tercera regla denominada **regla de integridad de atributo**; que dice que "todo atributo debe satisfacer la restricción de tomar los valores de su dominio subyacente". Aunque no enunciada hasta ahora, esta regla iba implícita en la definición de la relación.

Una restricción de usuario en general está constituida por un predicado definido sobre un conjunto de dominios, atributos o tuplas, que debe ser verificado por los correspondientes objetos para que constituyan una ocurrencia válida del esquema.

En las distintas implementaciones del modelo relacional, las restricciones pueden tener un soporte declarativo (se declaran en un cierto lenguaje, se almacenan en el catálogo y el sistema las verifica) o ser soportadas por procedimientos almacenados o disparadores.

Una restricción, en general, constará de un nombre, el predicado que debe ser satisfecho y una respuesta a la operación que intenta violar la restricción. También puede ser necesario

indicar el momento en que se produce la verificación de la restricción dentro de la transacción, si se hace al finalizar cada sentencia o al final de la transacción.

14. El tratamiento de nulos.

Cuando intentamos modelar y representar en un esquema relacional un determinado universo del discurso, sucede a veces que ciertas propiedades de una determinada entidad carecen de valor. Esta falta de valor puede ser debida a que no se conoce, caso del número de teléfono en una ocurrencia de la entidad CLIENTE, o a que esa propiedad no es aplicable a cierta ocurrencia de la entidad, como sucede con el NIF del cónyuge de un empleado soltero en la entidad EMPLEADOS.

AND	C	Q	F
C	C	Q	F
Q	Q	Q	F
F	F	F	F

NOT	
C	F
Q	Q
F	C

OR	C	Q	F
C	C	C	C
Q	C	Q	Q
F	C	Q	F

IS NULL	
C	F
Q	C
F	F

Operadores lógicos en lógica trivaluada y tabla de verdad del operador IS NULL.

Para resolver el problema, se introdujo en el modelo relacional el concepto de nulo (*null*). Un nulo no es realmente un valor, se trata de una marca que indica que el atributo correspondiente no tiene asignado valor.

El uso de nulos exige definir operaciones y funciones específicas para nulos, como las operaciones aritméticas, las de comparación, los operadores lógicos, operaciones del álgebra relacional o funciones estadísticas.

Sin valores nulos solo existen dos valores lógicos: **cierto** y **falso**. Con la aparición de los nulos surge un tercer valor el **quizás** (NULL). Surge así la lógica trivaluada, distinta de la lógica habitual.

En la figura se muestran las tablas de verdad de las operaciones AND, OR y NOT en lógica trivaluada.

La lógica trivaluada hace necesarios nuevos operadores como el operador unario IS NULL, que se evalúa como cierto sólo en el caso de que el operando sea nulo.

Las operaciones aritméticas en que aparece algún operador nulo dan como resultado nulo; lo mismo ocurre en las operaciones de comparación. Más adelante veremos como afectan los nulos al álgebra relacional.

La introducción de valores nulos en el modelo relacional da lugar a una serie de problemas y anomalías, por lo que algunos autores argumentan que la solución dada al problema de la información faltante no es la adecuada, y que, mientras se estudia mejor el problema, se deberían utilizar valores por defecto en lugar de nulos.

La implementación de los nulos en los productos relacionales no es uniforme, sino que varía de unos a otros, por lo que, antes de usar nulos, debemos analizar perfectamente cómo son tratados por el producto concreto que vayamos a utilizar.

Terminaremos este apartado indicando que algunos autores propugnan utilizar dos tipos de nulos, que diferencien entre los valores **inaplicables** y los valores **desconocidos aplicables**. El uso de dos tipos de nulos lleva a la aparición de la lógica cuatrivaluada, es decir, en lugar de tres valores lógicos cuatro.

15. El modelo relacional en la arquitectura ANSI.

Todos los elementos del modelo vistos hasta ahora (dominios, relaciones, claves y restricciones) constituyen el esquema conceptual de la arquitectura ANSI.

Hasta ahora hemos hablado de relaciones sin más, pero existen distintos tipos. Las **relaciones base o relaciones reales** son relaciones autónomas con nombre y tienen representación directa en el almacenamiento interno. Estas relaciones base son las que forman parte del esquema conceptual de la base de datos.

Existen otras relaciones denominadas **vistas o relaciones virtuales**. Una vista es una relación derivada con nombre, que se representa dentro del sistema exclusivamente por su definición en términos de otras relaciones con nombre; no poseen datos almacenados propios, separados y distinguibles, como las relaciones base. Las vistas, que son como ventanas sobre relaciones reales, equivalen al esquema externo de la arquitectura ANSI, aunque dicha equivalencia presenta la salvedad de que no todas las vistas relacionales son actualizables.

En la arquitectura ANSI el usuario solo tiene acceso a su esquema externo, pero en el modelo relacional tiene acceso, si dispone de las correspondientes autorizaciones, tanto a las relaciones base como a las vistas.

El modelo relacional no dice nada del esquema interno ya que se trata de un modelo lógico. Cada producto relacional implementa el esquema interno de la forma que considera más oportuna para conseguir un mejor rendimiento del sistema.

16. El álgebra relacional.

El álgebra relacional fue introducida por E.F.Codd como una colección de operaciones formales que actúan sobre las relaciones produciendo como resultado otras relaciones. Por medio de estos operadores se pueden construir expresiones que indiquen las transformaciones a realizar en una base de datos para pasar de un estado origen a otro estado objetivo. El álgebra, como todos los lenguajes relacionales, es un lenguaje de especificación que opera sobre conjuntos de tuplas.

El álgebra relacional, que veremos más adelante, constituye la parte dinámica del modelo relacional.

El álgebra original presentada por Codd estaba formada por ocho operadores, de los cuales cuatro son adaptaciones al mundo relacional de los correspondientes de teoría de conjuntos y otros cuatro son nuevos. El álgebra, como otras partes del modelo relacional, ha sufrido una evolución considerable, surgiendo nuevos operadores y modificaciones de algunos de los originales a partir de propuestas de distintos autores.

Operadores de conjuntos	Operadores especiales
Unión	Restricción
Intersección	Proyección
Diferencia	División
Producto cartesiano	Reunión natural

Origen de los operadores originales de álgebra relacional.

Los ocho operadores originales del álgebra son**: unión, producto cartesiano generalizado, diferencia, restricción, proyección, intersección, combinación natural (*natural join*) y división**. De estos ocho, los cinco primeros se suelen considerar como **operadores primitivos**, siendo la intersección, división y reunión natural **operadores derivados**, ya que se pueden deducir de los primeros.

El hecho de que el resultado de cualquier operación del álgebra relacional sea una relación, lo que se conoce como **propiedad de cierre,** hace posible escribir expresiones algebraicas anidadas, es decir, que una expresión siempre puede ser un operando de cualquier otra expresión.

17. Operadores tradicionales de conjuntos.

Son operadores definidos en la teoría de conjuntos a los que se les aplicaron ciertas restricciones para adaptarlos al hecho de que una relación es un conjunto en que sus elementos son tuplas.

Así, para que la propiedad de cierre siga siendo cierta, la unión incluida en el álgebra relacional es una versión limitada de la unión matemática, en la que se obliga a que las dos relaciones que actúan de operandos tengan cabeceras idénticas o, expresado de otra forma, a que sean **compatibles respecto de la unión**. El que dos relaciones tengan cabeceras idénticas significa que:

a) Las dos tienen el mismo conjunto de nombres de atributo; y
b) los atributos correspondientes, es decir, los que tienen el mismo nombre en ambas relaciones, se definen sobre el mismo dominio.

Hay autores (C. J. Date) que hablan de relaciones de **tipos compatibles** en lugar de compatibles respecto a la unión. Los dos conceptos son equivalentes, salvo en que en el caso de relaciones de tipos compatibles la condición ***b*** anterior diría: "los atributos correspondientes deben ser del mismo tipo (definidos en el mismo dominio) o existir alguna función de conversión de tipo (dominio) conocida por el sistema, de forma que puedan ser convertidos al mismo tipo".

La exigencia de que los nombres de atributos de las dos relaciones compatibles sean iguales, es juzgada por algunos autores como un aspecto sintáctico poco relevante y demasiado restrictivo. No obstante, aunque se mantenga el requisito, siempre es posible disponer de un operador unario, que aplicado a una relación dé como resultado otra con los nombres de atributo renombrados.

Artículos suministrados por el proveedor A
ARTPROVA

REFERENCIA	DENOMINACION
0001	MONITOR 15" SONY
0002	MONITOR 17" SONY
0003	MONITOR 19" SONY
0004	MONITOR 21" SONY

Artículos suministrados por el proveedor B
ARTPROVB

REFERENCIA	DENOMINACION
0002	MONITOR 17" SONY
0003	MONITOR 19" SONY
0005	TECLADO GENIUS INT
0010	TECLADO LOGITECH A23

Relaciones ARTPROVA y ARTPROVB.

Al igual que la unión, la diferencia y la intersección exigen relaciones de tipos compatibles. El producto cartesiano, en cambio, no tiene esta exigencia, aunque sí tiene otra que veremos más adelante.

En la figura tenemos dos relaciones correspondientes al nuestro ejemplo de la empresa de distribución. En cada una de ellas están los artículos suministrados por dos proveedores, que llamamos A y B, a la empresa.

La **unión** de dos relaciones A y B compatibles respecto de la unión es otra relación, cuya cabecera es idéntica a la de A o B y cuyo cuerpo esta formado por todas las tuplas pertenecientes a A o a B o a ambas.

En el resultado de una unión, aquellas tuplas que existen en las dos relaciones, lógicamente, aparecen una sola vez, es decir, que se eliminan las tuplas repetidas.

Artículos suministra por cualquiera de los proveedores

ARTPROVA ∪ ARTPROVB

REFERENCIA	DENOMINACION
0001	MONITOR 15" SONY
0002	MONITOR 17" SONY
0003	MONITOR 19" SONY
0004	MONITOR 21" SONY
0005	TECLADO GENIUS INT
0010	TECLADO LOGITECH A23

Unión **de ARTPROVB y ARTPROVA**

La **intersección** de dos relaciones, compatibles respecto de la unión, A y B es otra relación cuya cabecera es idéntica a la de A o B, y cuyo cuerpo está formado por las tuplas *t* pertenecientes tanto a A como a B.

Artículos suministrados por ambos proveedores

ARTPROVA ∩ ARTPROVB

REFERENCIA	DENOMINACION
0002	MONITOR 17" SONY
0003	MONITOR 19" SONY

ARTPROVA intersección ARTPROVB

Intersección de **ARTPROVA y ARTPROVB.**

Artículos suministrados sólo por el proveedor B

ARTPROVB - ARTPROVA

REFERENCIA	DENOMINACION
0005	TECLADO GENIUS INT
0010	TECLADO LOGITECH A23

ARTPROVB menos ARTPROVA

Diferencia de **ARTPROVB** menos **ARTPROVA.**

La **diferencia** entre dos relaciones, compatibles respecto de la unión, A y B es otra relación cuya cabecera es idéntica a la de A o B, y cuyo cuerpo está formado por todas las tuplas *t* pertenecientes a A pero no a B. (la imagen de la derecha las referencias debe ser 1 y 4)

Artículos suministrados sólo por el proveedor A

ARTPROVA - ARTPROVB

REFERENCIA	DENOMINACION
0002	MONITOR 15" SONY
0003	MONITOR 21" SONY

ARTPROVA menos ARTPROVB

Diferencia de **ARTPROVA** menos **ARTPROVB.**

El **producto cartesiano** de dos relaciones, A y B, compatibles respecto al producto, es otra relación cuya cabecera es la combinación de las cabeceras de A y B y cuyo cuerpo esta

formado por el conjunto de todas las tuplas *t,* tales que *t* es la combinación de una tupla *a* perteneciente a A y una tupla *b* perteneciente a B.

El producto cartesiano generalizado o ampliado, definido en el párrafo anterior, parte del concepto de producto cartesiano de dos conjuntos adaptándolo para tener en cuenta que los conjuntos operando son relaciones.

En matemáticas, el producto cartesiano de dos conjuntos es el conjunto de todos los pares ordenados de elementos tales que el primer elemento de cada par pertenece al primer conjunto y el segundo elemento de cada par pertenece al segundo conjunto. Si se aplica esta operación a dos relaciones obtendríamos un conjunto de pares ordenados de tuplas, es decir, que no obtendríamos una relación con lo que se estaría atentando contra la propiedad de cierre del algebra relacional. El producto cartesiano de relaciones, sustituye el par ordenado de tuplas del producto cartesiano matemático por una tupla, que contiene todos los pares atributo-valor de dicho par de tuplas.

Puesto que la cabecera de un producto cartesiano generalizado estará formado por la combinación de las cabeceras de las dos relaciones iniciales, éstas no pueden tener nombres de atributos en común. El que dos relaciones tengan algún nombre de atributo en común, sólo obligaría a renombrar uno de ellos con el operador adecuado.

PEDIDOS

NUM_PEDIDO	FECHA	CODIGO_CLIENTE
000010	14/11/2008	5555
000011	14/11/2008	7777
...	...	...

L_PEDIDOS

PEDIDO	REFERENCIA	CANTIDAD
000010	0005	5
000010	0003	10
000011	0006	10
...	...	...

PEDIDOS X L_PEDIDOS

NUM_PEDIDO	FECHA	CODIGO_CLIENTE	PEDIDO	REFERENCIA	CANTIDAD
000010	14/11/2008	5555	000010	0003	10
000010	14/11/2008	5555	000010	0005	5
000010	14/11/2008	5555	000011	0006	10
000011	14/11/2008	7777	000010	0003	10
000011	14/11/2008	7777	000010	0005	5
000011	14/11/2008	7777	000011	0006	10
...	...	...	...	...	...

Producto cartesiano generalizado.

Diremos que dos relaciones son **compatibles con respecto al producto** si y, sólo si, sus cabeceras son disjuntas, es decir no tienen nombres de atributo en común.

La operación de producto cartesiano generalizado se incluye en el álgebra por razones conceptuales y no por su importancia práctica.

Los operadores unión, intersección y producto tienen las propiedades conmutativa y asociativa, mientras que la diferencia no las posee.

18. Operaciones relacionales especiales.

La restricción es un operador unario que permite seleccionar las tuplas de una relación que satisfacen una determinada condición o predicado de selección. Una definición formal de la **restricción**, también conocida como **selección**, es la siguiente:

Sea *theta* cualquier operador de comparación escalar simple (por ejemplo =, <>, >, >=, etc); la restricción *theta* de la relación A, según los atributos X e Y**,** denotada $\sigma_{X\Theta Y}$ **(A)**, es una relación con la misma cabecera de A y con un cuerpo formado por el conjunto de todas las tuplas *t* de A para las que la condición "X *theta* Y" se evalúa afirmativamente. Los atributos X e Y deben estar definidos sobre el mismo dominio y la operación **theta** debe ser aplicable a ese dominio.

Se puede especificar un literal en lugar de cualquiera de los dos atributos de la condición. También se pueden utilizar condiciones compuestas formadas por condiciones simples unidas por los operadores AND, OR y NOT.

La **proyección** de la relación A, según los atributos X, Y, ..., Z, denotada $\Pi_{X, Y, ..., Z}$ **(A)**, es una relación con (X, Y, ..., Z) como cabecera y cuyo cuerpo está formado por el conjunto de todas las tuplas de A definidas sobre los atributos (X, Y, ...,Z), eliminando las que resulten

$\sigma_{PEDIDO = NUM\_PEDIDO}$ (PEDIDOS X L_PEDIDOS)

NUM_PEDIDO	FECHA	CODIGO_ CLIENTE	PEDIDO	REFERENCIA	CANTIDAD
000010	14/11/2008	5555	000010	0003	10
000010	14/11/2008	5555	000010	0005	5
000011	14/11/2008	7777	000011	0006	10
...	...	...	...	...	...

Se ha restringido la relación resultado del producto cartesiano a aquellas tuplas en que los valores de PEDIDO y NUM_PEDIDO son iguales:

*Ejemplo de **RESTRICCIÓN.***

duplicadas.

$\Pi_{\text{CODIGO\_CLIENTE, PEDIDO, REFERENCIA, CANTIDAD}}(\sigma_{\text{PEDIDO = NUM\_PEDIDO}}(\text{PEDIDOS X L\_PEDIDOS}))$

CODIGO_CLIENTE	PEDIDO	REFERENCIA	CANTIDAD
5555	000010	0003	10
5555	000010	0005	5
7777	000011	0006	10
...	...	...	...

Se ha proyectado la relación resultado de la restricción sobre los atributos CODIGO_CLIENTE, PEDIDO, REFERENCIA y CANTIDAD.

*Ejemplo de **PROYECCIÓN.***

Podríamos decir que la proyección de una relación sobre ciertos atributos se obtiene cortando verticalmente las columnas correspondientes a los atributos y pegándolas para formar la tabla resultado (eliminando aquellas filas duplicadas). De igual forma, la restricción de una relación se construiría cortando horizontalmente aquellas filas que cumplen el predicado de selección y pegándolas.

Existen distintos tipos de reunión, siendo el más general la **reunión *theta***. En ella participan dos relaciones compatibles con respecto al producto, es decir, que no tienen atributos en común, y el resultado consta de las tuplas que se obtienen concatenando cada tupla de una relación con todas las tuplas de la segunda y eliminando aquellas que no cumplan una cierta condición.

Una definición sería la siguiente:

Sean las relaciones A y B compatibles con respecto al producto y sea *theta* un operador como el definido en la operación de restricción, la **reunión *theta*** de la relación A, según el atributo X, con la relación B, según el atributo Y**,** denotada $\mathbf{(A * B)_{X\ \Theta Y}}$, se define como el resultado de evaluar la expresión $\mathbf{\sigma_{X\ \Theta\ Y}\ (A \times B)}$**.**

Dicho de otra forma, el resultado es una relación con la misma cabecera que el producto cartesiano de A y B y con un cuerpo formado por el conjunto de todas las tuplas *t*, tales que *t* pertenece a ese producto cartesiano y la evaluación de la condición “X *theta* Y” resulta cierta para esa tupla t. Los atributos X e Y deberán estar definidos sobre el mismo dominio.

(PEDIDOS * L_PEDIDOS)$_{NUM\_PEDIDO = PEDIDO}$

NUM_PEDIDO	FECHA	CODIGO_CLIENTE	REFERENCIA	CANTIDAD
000010	14/11/2008	5555	0003	10
000010	14/11/2008	5555	0005	5
000011	14/11/2008	7777	0006	10
...	...	...	...	...

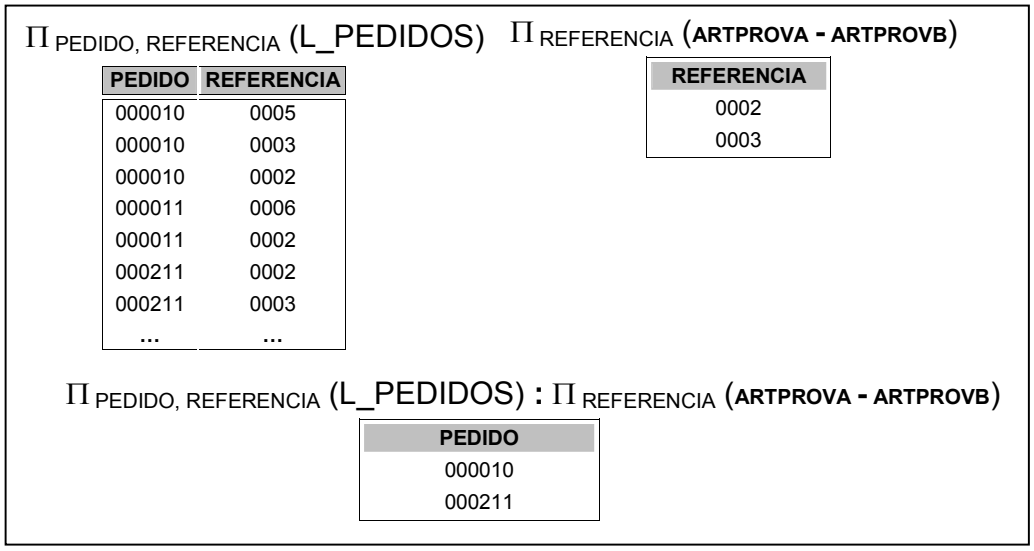
La reunión natural de PEDIDOS con L_PEDIDOS da lugar a una relación en que se ha eliminado una de las columnas donde aparecía el número de pedido.

*Ejemplo de **REUNIÓN NATURAL.***

Si el operador theta es "igual a", la reunión theta se llama **equirreunión**. Una equirreunión en que se elimina uno de los atributos que participa en la condición de igualdad se denomina **reunión natural**, y es el tipo de reunión más usado en la práctica. Una reunión natural es el resultado de un producto cartesiano seguido de una restricción por igualdad y finalmente una proyección que elimina uno de los atributos X o Y.

La **división** de dos relaciones es otra relación constituida por las tuplas que, al completarse con las tuplas (con todas) de la segunda relación, permiten obtener tuplas de la primera.

Si la relación A, relación dividendo, tiene su cabecera formada por los atributos X_1, X_2, ..., X_m, Y_1, Y_2, ..., Y_n y la relación B, relación divisor, tiene la cabecera formada por los atributos Y_1, Y_2, ...,



*Ejemplo de **DIVISIÓN.***

Y_n, la división de A entre B, denotada **A : B**, es una relación con la cabecera formada por los atributos X_1, X_2, ..., X_m, y el cuerpo formado por el conjunto de todas las tuplas tales que concatenadas con cada una de las tuplas de B dan lugar a tuplas incluidas en A.

Todos los atributos del divisor coinciden con atributos del dividendo y están definidos en los mismos dominios.

19. Expresiones algebraicas.

La propiedad de cierre del álgebra relacional permite escribir expresiones algebraicas. Estas expresiones sirven a distintos propósitos, entre los que están los siguientes:

- Definir el alcance de una operación de recuperación; es decir, definir qué datos se van a extraer de una base de datos en una consulta.
- Definir el alcance de una actualización; es decir, definir los datos que se van a insertar, modificar o eliminar en una actualización.
- Definir relaciones virtuales o vistas.
- Definir derechos de acceso; es decir, definir los datos incluidos en algún tipo de autorización.
- Definir restricciones de integridad; es decir, definir alguna restricción de usuario que deba satisfacer la base de datos.

Mediante una expresión algebraica el usuario especificará al sistema qué quiere hacer. Antes de ejecutar la expresión, el sistema la transformará en otra más eficiente desde el punto

de vista del rendimiento. Este proceso de optimización se lleva a cabo utilizando reglas de transformación del álgebra relacional. Por tanto, el álgebra constituye la base del proceso de optimización necesario en todo sistema relacional.

20. Operadores adicionales .

Con el fin de mejorar el poder expresivo de álgebra relacional, distintos autores han propuestos operadores adicionales. Algunos de esos operadores, que se especifican a continuación, son los de: renombrado, ampliación, resumen y reunión externa.

El operador de **renombrado** permite cambiar el nombre de los atributos de una relación. El cambio de nombre de un atributo puede ser necesario, como ya se indicó, para realizar algunas operaciones algebraicas. Este operador toma una relación especificada y devuelve una copia de dicha relación con algunos de los atributos originales cambiados de nombre.

L_PEDIDOS **RENOMBRAR** REFERENCIA **COMO** CODART, PEDIDO **COMO** NUM_PEDIDO

En este ejemplo, se cambia el nombre de los atributos REFERENCIA y PEDIDOS de la relación L_PEDIDOS a CODART y NUM_PEDIDO respectivamente.

El operador de **ampliación** dota al álgebra relacional de capacidad de cálculo escalar.

El operador de **ampliación** toma como operando una relación y devuelve otra semejante a la original pero con atributos añadidos. Estos atributos nuevos se obtienen de la evaluación de expresiones de cálculo escalar especificadas. Por ejemplo,

ALMACEN **AMPLIAR** (P_TARIFA * EXISTENCIAS) **COMO** VALOR_STOCK

La relación resultado contiene las tuplas de ALMACEN ampliadas en el producto precio de tarifa por existencias, es decir, que aparecen los artículos en almacén con sus existencias valoradas a precio de tarifa.

El operador **resumen**, o **agrupación**, permite agrupar en subconjuntos tuplas con valores comunes de ciertos atributos y aplicar a estos subconjuntos funciones de grupo como SUMA, CUENTA, PROMEDIO, MÁXIMO, MÍNIMO, etc. Por ejemplo, en la siguiente expresión

L_PEDIDOS **AGRUPAR POR** PEDIDO, **CUENTA** (REFERENCIA) **COMO** LINEAS_PEDIDO

el operador de agrupación (**AGRUPAR POR)** se aplica a la relación L_PEDIDOS, formándose grupos de tuplas con igual valor en el atributo PEDIDO, y se cuentan el número de valores del atributo REFERENCIA; cada tupla de la relación resultado tendrá un código de pedido y el número de líneas que contiene ese pedido

- **Reunión externa**

La reunión externa es una versión ampliada de la operación normal o interna, en cuyo resultado aparecen las tuplas de una relación que no tienen contraparte en la otra, con nulos en las posiciones de los demás atributos. No es una operación primitiva pero se utiliza bastante en la práctica.

La reunión natural de la relación

$\Pi_{\text{REFERENCIA, DENOMINACION}}$ **(ALMACEN)**

con la relación

$\Pi_{\text{PEDIDO, REFERENCIA}}$ **(L_PEDIDOS)**

no incorpora las tuplas de la primera que no tienen contrapartida en la segunda, es decir, las tuplas de referencias 0001, 0002 y 0004. En cambio, cuando se hace la reunión externa estas tuplas se incorporan al resultado.

$\Pi_{\text{PEDIDO, REFERENCIA}}$ **(L_PEDIDOS)**

PEDIDO	REFERENCIA
000010	0005
000010	0003
000011	0006
000011	0005
000012	0005

$\Pi_{\text{REFERENCIA, DENOMINACION}}$ **(ALMACEN)**

REFERENCIA	DENOMINACION
0001	MONITOR 15" SONY
0002	MONITOR 17" SONY
0003	MONITOR 19" SONY
0004	MONITOR 21" SONY
0005	TECLADO GENIUS INT
...	.....

Reunión natural

REFERENCIA	DENOMINACION	PEDIDO
0003	MONITOR 19" SONY	000010
0005	TECLADO GENIUS INT	000010
0005	TECLADO GENIUS INT	000011
0005	TECLADO GENIUS INT	000012
0006	RATON OPT. GENIUS G-34	000011

Reunión natural externa

REFERENCIA	DENOMINACION	PEDIDO
0001	MONITOR 15" SONY	null
0002	MONITOR 17" SONY	null
0003	MONITOR 19" SONY	000010
0004	MONITOR 21" SONY	null
0005	TECLADO GENIUS INT	000010
0005	TECLADO GENIUS INT	000011
0005	TECLADO GENIUS INT	000012
0006	RATON OPT. GENIUS G-34	000011
...	....	...

LA REUNIÓN EXTERNA.

Se distingue entre reunión externa izquierda (left outer join) y reunión externa derecha (right outer join) según esté como primer operando o segundo operando la relación que tiene tuplas sin contrapartida. También se habla de reunión externa simétrica cuando hay tuplas sin contrapartida en las dos relaciones.