BASES DE DATOS I

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA
UNAD 2.005

JOHN ALEJANDRO FIGUEREDO LUNA
PRIMER EDICION

CONTENIDO

UNIDAD 1. FUNDAMENTOS DE BASES DE DATOS

CAPITULO 1. INTRODUCCIÓN A LAS BASES DE DATOS

La información hoy en día es el motor de toma de decisiones y el tesoro mas preciado de las organizaciones, ya que al conocer de una manera adecuada y de una forma acertada los procesos internos y externos se puede en determinado momento ayudar a tomar decisiones, esto con fin de lograr una mejor producción o posición en el mercado.

El manejo de gran cantidad de datos es la consecuencia del aumento de información que se maneja en el transcurso de nuestras vida y mas aun en el mundo empresarial, al incrementar todo este volumen de información que diariamente se puede acumular en el manejo de una empresa u organización se hace necesario organizarla para poder encontrar resultados rápidos y óptimos en el momento de utilizarla.

Debido a esta necesidad en los años setenta, para manejar toda esta información surgen las bases de datos, en la cual se integran archivos individuales para poder ser compartidos por todos los usuarios de la empresa.

Para el desarrollo de esta técnica de almacenamiento y ordenamiento de información es necesario conocer la causa que llevo el desenvolvimiento de dicho proceso, uno de ellos fue la transmisión de la información o los datos, es decir que el usuario tenga capacidad y facilidad de acceder a los datos de una forma remota.

Que el proceso de dialogo que se genere entre el usuario y la maquina sea el mas amigable y compatible en donde se pueda consultar y borrar, modificar e insertar datos en cualquier momento que se necesite en el manejo de la información.

El diseño de la base de datos es de gran importancia en el manejo de la información, ya que tiene como principal objetivo que los datos almacenados se puedan utilizar por una gran numero de aplicaciones.

Los predecesores de los sistemas de bases de datos fueron los sistemas de ficheros o archivos. No hay un momento concreto en que los sistemas de archivos hayan cesado y hayan dado comienzo los sistemas de bases de datos. De hecho, todavía existen sistemas de archivos en uso.

Los sistemas jerárquico y de red constituyen la primera generación de los SGBD. Pero estos sistemas presentan algunos inconvenientes:

- Es necesario escribir complejos programas de aplicación para responder a cualquier tipo de consulta de datos, por simple que ésta sea.
- La independencia de datos es mínima.
- No tienen un fundamento teórico.

En 1970 Codd, de los laboratorios de investigación de IBM, escribió un artículo presentando el modelo relacional. En este artículo, presentaba también los inconvenientes de los sistemas previos, el jerárquico y el de red. Entonces, se comenzaron a desarrollar muchos sistemas relacionales, apareciendo los primeros a finales de los setenta y principios de los ochenta. Uno de los primeros es System R, de IBM, que se desarrolló para probar la funcionalidad del modelo relacional, proporcionando una implementación de sus estructuras de datos y sus operaciones. Esto condujo a dos grandes desarrollos:

- El desarrollo de un lenguaje de consultas estructurado denominado SQL, que se ha convertido en el lenguaje estándar de los sistemas relaciónales.
- La producción de varios SGBD relacionales durante los años ochenta, como DB2 y SLQ/DS de IBM, y ORACLE de ORACLE Corporation.

Hoy en día, existen cientos de SGBD relacionales, tanto para microordenadores como para sistemas multiusuario, aunque muchos no son completamente fieles al modelo relacional.

Como respuesta a la creciente complejidad de las aplicaciones que requieren bases de datos, han surgido dos nuevos modelos: el modelo de datos orientado a objetos y el modelo relacional extendido. Sin embargo, a diferencia de los modelos que los preceden, la composición de estos modelos no está clara. Esta evolución representa la tercera generación de los SGBD

1.1 Propósito de los sistemas

Hoy en día, son muchas las aplicaciones que requieren acceder a datos. Bien sea un sencillo programa doméstico, bien una suite para la gestión empresarial. Estos datos se deben almacenar en algún soporte permanente, y las aplicaciones deben disponer de un medio para acceder a ellos. Normalmente, la forma en que un programa accede a un fichero es a través del Sistema operativo. Este provee de funciones como abrir archivo, leer información del archivo, guardar información, etc. No obstante, este procedimiento de acceso a ficheros es altamente ineficaz cuando se trata con un volumen elevado de información. Es aquí donde aparecen los Sistemas Gestores de Bases de Datos: proporcionan un interfaz entre aplicaciones y sistema operativo, consiguiendo, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, más fácil de implementar, y, sobre todo, más segura.

Un sistema gestor de bases de datos (SGDB) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGDB es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**. Los usuarios de los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- **Independencia**. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Redundancia mínima**. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

- **Consistencia**. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad**. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- **Integridad**. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- **Respaldo y recuperación**. Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Control de la concurrencia**. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden

a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

- **Tiempo de respuesta**. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

1.2 Visión de los datos

Korth define, un sistema de bases de datos como una colección de archivos interrelacionados y un conjunto de programas que permitan a los usuarios acceder y modificar estos archivos. Uno de los propósitos principales de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos.

Abstracción de datos: para que el sistema sea útil, si debe recuperar los datos eficientemente. Los tres niveles de abstracción de datos son:

- **nivel físico:** describe cómo se almacena realmente los datos. Se describen en detalle las estructuras de datos complejas de bajo nivel.
- **nivel lógico:** describe que datos se almacena en la base de datos y qué relaciones existen entre esos datos.
- **Nivel de vistas:** describe solo parte de la base de datos completa.

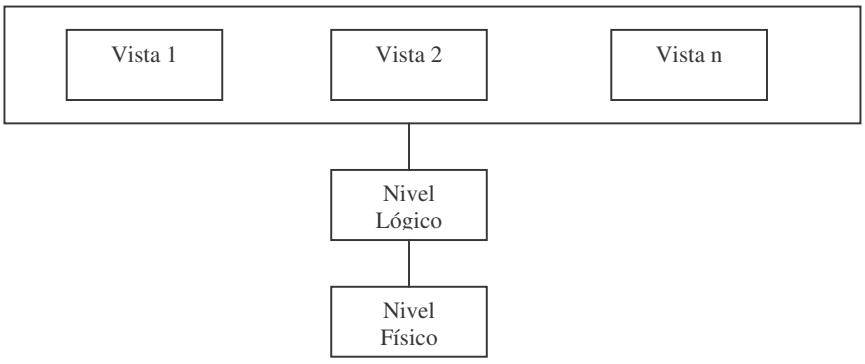


Figura 1.1
Los tres niveles de abstracción de datos

1.3 Modelos de los datos

Bajo la estructura de la base de datos se encuentra el modelo de datos: una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Los diferentes modelos de datos que sean propuesto se clasifican en tres grupos diferentes: modelos lógicos basados en los objetos, modelos lógicos basados en registros y modelos físicos.

- **Modelo entidad-relación:** esta o en una percepción del mundo real que consta de una colección de éstos básicos, llamados entidades, y niveles relaciones entre esos objetos. Una entidad es una cosa hubo gesto en el mundo real que distinguible de otros objetos.

- **Modelo relacional:** se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. La tabla está compuesta por varias columnas, y cada columna tiene un hombre único.

- **Otros modelos de datos:** el **modelo de datos orientado objetos** es otro modelo de datos que se puede observar como una extensión del modelo E-R con las nociones de en cápsula nación, métodos e identidad de objeto. El **modelo de datos relacional orientado objetos** combinan las características del modelo de datos orientado objetos y el modelo de datos relacional. El **modelo de datos de red** y el **modelo de datos jerárquico**, precedieron al modelo de datos relacional.

1.4 Usuarios y administradores de la Base de Datos

Las personas que trabajan con una base de datos se pueden catalogar como usuarios de bases de datos o como administradores de bases de datos.

1.4.1 Usuarios de bases de datos e interfaces de usuarios

Hay cuatro tipos diferentes de usuarios de un sistema de bases de datos, diferenciados por la forma en que ellos esperan interactuar con el sistema.

- Usuarios normales
- programadores de aplicaciones
- usuarios sofisticados
- usuarios especializados

1.4.2 Administrador de la base de datos

Una de las principales razones de usar SGBDs es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos la persona que tiene este control central sobre el sistema se llama administrador de la base de datos (ABD). Las funciones del ABD incluyen las siguientes:

- definición del esquema.
- Definición de la estructura y del método de acceso.
- Modificación del esquema y de la organización física.
- Concesión de autorización para el acceso a los datos.
- Mantenimiento rutinario.

1.5 Gestión de transacciones y de Almacenamiento

Una transacción es una colección de operaciones que se lleva a cabo como una única función lógica. Es responsabilidad del programador definir adecuadamente las diferentes transacciones, de tal manera que cada una preserve la consistencia de la base de datos. Asegurar la consistencia de las propiedades es responsabilidad del propio sistema de bases de datos, específicamente del componente de gestión de transacciones.
El sistema de bases de datos debe realizar la recuperación de fallos, es decir, detectar los fallos de sistema y restaurar la base de datos al estado que existía antes de que ocurriera el fallo.
Finalmente, cuando varias transacciones actualizan la base de datos concurrentemente, la consistencia de los datos puede no ser preservada, incluso aunque cada transacción individualmente sea correcta. Es responsabilidad del gestor de control de concurrencia controlar la interacción entre las transacciones concurrentes para asegurar la consistencia de la base de datos.

1.6 Estructura de un sistema de bases de datos

Un sistema de bases de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Los componentes funcionales de un sistema de bases de datos se pueden dividir a grandes rasgos en los componentes, gestor de almacenamiento y procesador de consultas.

1.6.1 Gestor de almacenamiento

Es un módulo de programas que proporciona la interfaces entre los datos de bajo nivel en la base de datos y los programas de aplicación y consultas emitidas al

sistema el gestor de almacenamiento es responsable de la interacción con el gestor de archivos. Los componentes del gestor de almacenamiento incluyen:

- Gestor de autorización e integridad, que comprueba que se satisfagan las restricciones de integridad y la autorización de los usuarios para acceder a los datos.
- Gestor de transacciones, que asegura que la base de datos quiere un estado consistente a pesar de los fallos del sistema, y que las ejecuciones de transacciones concurrentes ocurren sin conflictos.
- Gestor de archivos, que gestiona la reserva de espacio de almacenamiento de disco y las estructuras de datos usadas para representar la información almacenada en disco.
- Gestor de memoria intermedia, que es responsable de trae los datos del disco de almacenamiento a memoria principal y describir que datos tratar en memoria caché.

El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física el sistema:

- Archivos de datos.
- Diccionario de datos.
- Índices.

1.6.2 Procesador de consultas

Los componentes del procesador de consultas incluyen:

- Intérprete del LDD, que interpreta las instrucciones del LDD y registra las definiciones en el diccionario de datos.
- Compilador del LMD, que traduce las instrucciones del LMD en un lenguaje consultas a un plan de evaluación que consiste en instrucciones de bajo nivel que entiende el motor de evaluación de consultas.
- Motor de evaluación de consultas, que ejecutan las instrucciones de bajo nivel generadas por el compilador del LMD.

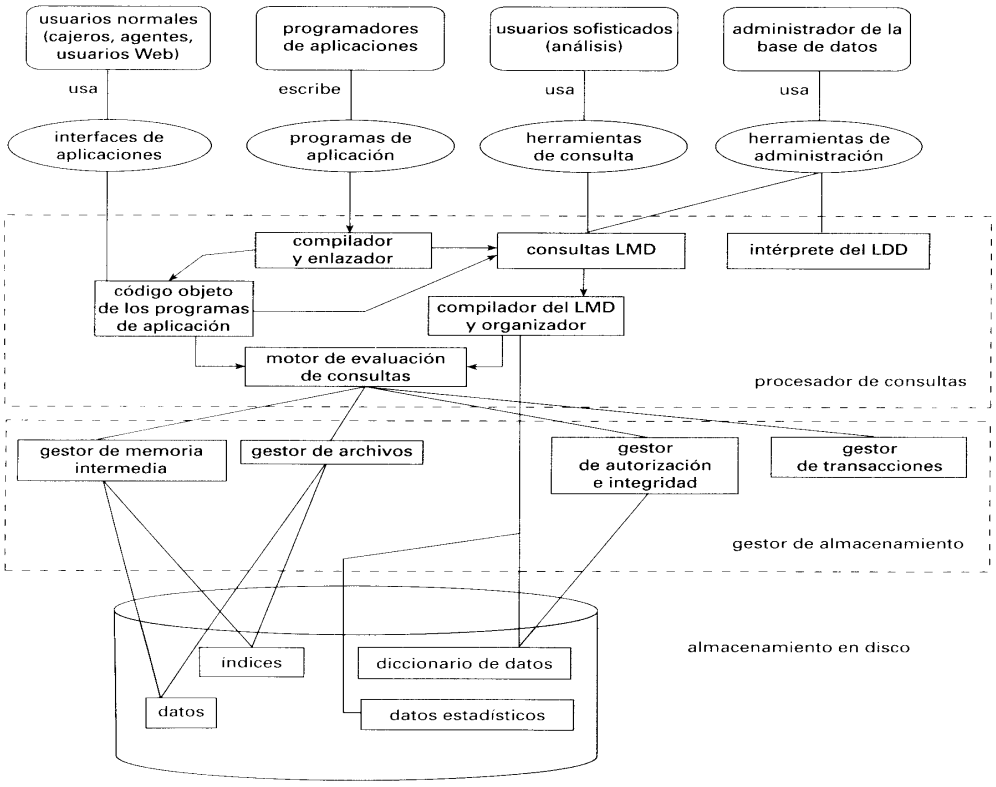


Figura 1.2
Estructura del sistema completo

1.7 Jerarquía de los datos campo, registro, archivos y Bases de datos

Los datos se organizan en una jerarquía que inician con una unidad mínima de información que es utilizada por los computadores denominado bit, y se desplaza a través de una escala hasta una base de datos. Un bit es un digito binario que representa un circuito que puede estar encendido o apagado. Los bits se pueden organizar en unidades denominadas bytes. Un byte esta compuesto por ocho bits, cada byte representa un carácter, que es la unidad de información básica, un carácter puede consistir de letras mayúsculas (A, B, C,..Z), letras minúsculas (a,b,c,…..z), dígitos numéricos (0,1,2,….9), o símbolos especiales(.![+]^*[-]/…).

Los caracteres se reúnen para formar un **campo**. Un campo por lo general es un nombre, un número o una combinación de caracteres que describen un aspecto de un objeto (por ejemplo, un empleado, una ubicación, etc.), o una actividad (Una venta). Un grupo de campos relacionados representan un **registro**. Al combinar descripciones de varios aspectos de un objeto o actividad, se obtiene una descripción mas completa de estos. Por ejemplo, el registro de un empleado es

una compilación de campos referentes a ese individuo. El nombre del empleado, su dirección, su número telefónico, sueldo, etc., representarían cada uno un campo. Un conjunto de registros relacionados representan un **archivo,** por ejemplo el archivo de empleados es la recopilación de todos los registros de los empleados de la compañía. De igual forma, un archivo de inventarios es la suma de todos los registros de inventarios de una compañía u organización en particular. Para el manejo de los archivos las aplicaciones se refieren a ellos como tablas.

En el nivel más alto de la jerarquía se encuentran las **bases de datos,** una recopilación de archivos integrados y relacionados. Bits, caracteres, campos, registros, archivos y bases de datos forman la jerarquía de datos.

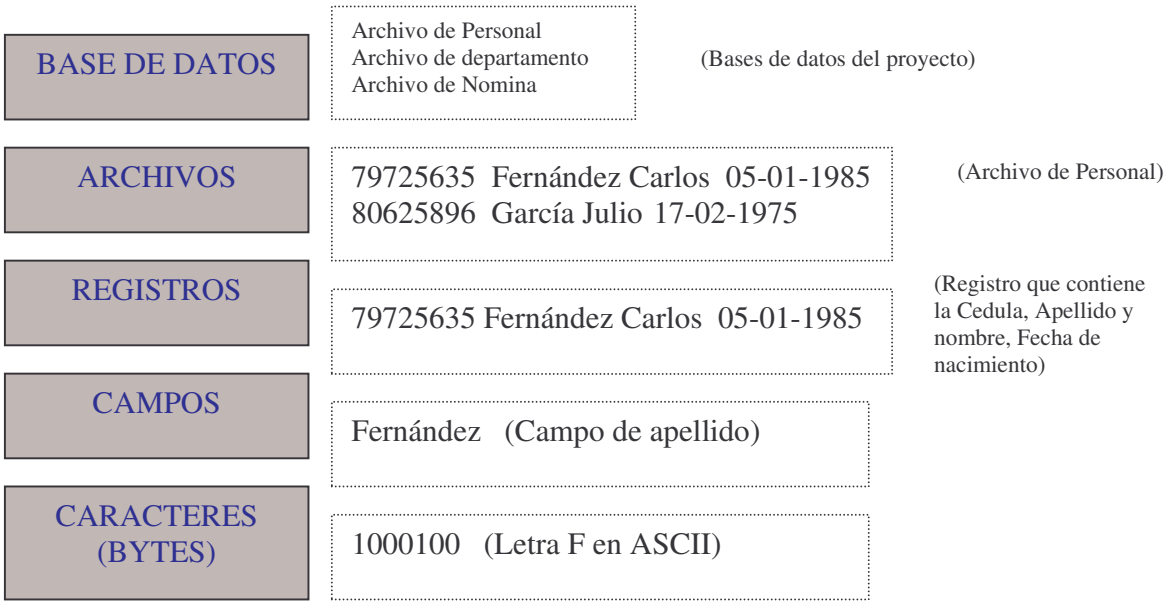


Figura 1.3.
La Jerarquía de los datos

1.8 Bases de datos distribuidas y centralizadas

1.8.1 Bases de datos centralizadas

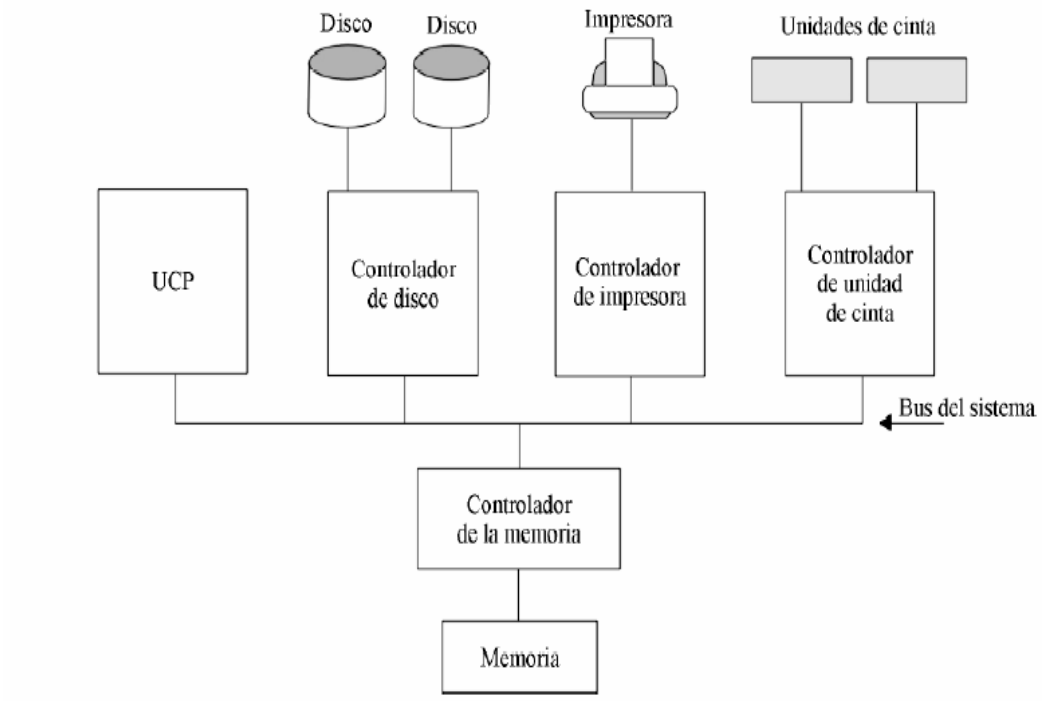


Figura 1.4.
Un sistema Informático centralizado

Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora. Tales sistemas comprenden el rango desde los sistemas de bases de datos monousuario ejecutándose en computadoras personales hasta los sistemas de bases de datos de alto rendimiento ejecutándose en grandes sistemas.

Una computadora moderna de propósito general consiste en una o unas pocas CPU's y un número determinado de controladores para los dispositivos que se encuentren conectados a través de un bus común, el cual proporciona acceso a la memoria compartida. Las CPU's poseen memorias caché locales donde se almacenan copias de ciertas partes de la memoria para acelerar el acceso a los datos. Cada controlador de dispositivo se encarga de un tipo específico de dispositivos (por ejemplo, una unidad de disco, una tarjeta de sonido o un monitor). La CPU y los controladores de dispositivo pueden ejecutarse concurrentemente, compitiendo así por el acceso a la memoria. La memoria caché reduce el acceso a la memoria, ya que la CPU necesita acceder a la memoria compartida un número de veces menor.
Se distinguen dos formas de utilizar las computadoras: como sistemas monousuario o como sistemas multiusuario. En la primera categoría están las computadoras personales y las estaciones de trabajo. Un sistema monousuario típico es una unidad de sobremesa utilizada por una única persona que dispone de una sola CPU, de uno o dos discos fijos y que trabaja con un sistema operativo

que sólo permite un único usuario. Por el contrario, un sistema multiusuario típico tiene más discos y más memoria, puede disponer de varias CPU y trabaja con un sistema operativo multiusuario. Se encarga de dar servicio a un gran número de usuarios que están conectados al sistema a través de terminales. Estos sistemas se denominan con frecuencia sistemas servidores.

Normalmente, los sistemas de bases de datos diseñados para funcionar sobre sistemas monousuario, como las computadoras personales, no suelen proporcionar muchas de las facilidades que ofrecen los sistemas multiusuario. En particular, no tienen control de concurrencia, que no es necesario cuando solamente un usuario puede generar modificaciones. Las facilidades de recuperación en estos sistemas, o no existen o son primitivas; por ejemplo, realizar una copia de seguridad de la base de datos antes de cualquier modificación. La mayoría de estos sistemas no admiten SQL y proporcionan un lenguaje de consulta muy simple, que en algunos casos es una variante de QBE (Query By Example).

Aunque hoy en día las computadoras de propósito general tienen varios procesadores, utilizan paralelismo de grano grueso, disponiendo de unos pocos procesadores (normalmente dos o cuatro) que comparten la misma memoria principal. Las bases de datos que se ejecutan en tales máquinas habitualmente no intentan dividir una consulta simple entre los distintos procesadores, sino que ejecutan cada consulta en un único procesador, posibilitando la concurrencia de varias consultas. Así, estos sistemas soportan una mayor productividad, es decir, permiten ejecutar un mayor número de transacciones por segundo, a pesar de que cada transacción individualmente no se ejecuta más rápido.

Las bases de datos diseñadas para las máquinas monoprocesador ya disponen de multitarea, permitiendo que varios procesos se ejecuten a la vez en el mismo procesador, usando tiempo compartido, mientras que de cara al usuario parece que los procesos se están ejecutando en paralelo.

1.8.2 Bases de datos Distribuidas

Los sistemas de bases de datos distribuidas son un caso particular de los sistemas de cómputo distribuido en los cuales un conjunto de elementos de procesamiento autónomos (no necesariamente homogéneos) se interconectan por una red de comunicaciones y cooperan entre ellos para realizar sus tareas asignadas. Entre los términos más comunes que se utilizan para referirse al cómputo distribuido podemos encontrar: funciones distribuidas, procesamiento distribuido de datos, multiprocesadores, multicomputadoras, procesamiento satelital, procesamiento tipo "backend", computadoras dedicadas y de propósito específico, sistemas de tiempo compartido, sistemas funcionalmente modulares.

En el cómputo distribuido existen muchos componentes para realizar una tarea. Los elementos que se pueden distribuir son:

- Control. Las actividades relacionadas con el manejo o administración del sistema.
- Datos. La información que maneja el sistema.
- Funciones. Las actividades que cada elemento del sistema realiza.
- Procesamiento lógico. Las tareas específicas involucradas en una actividad de procesamiento de información.

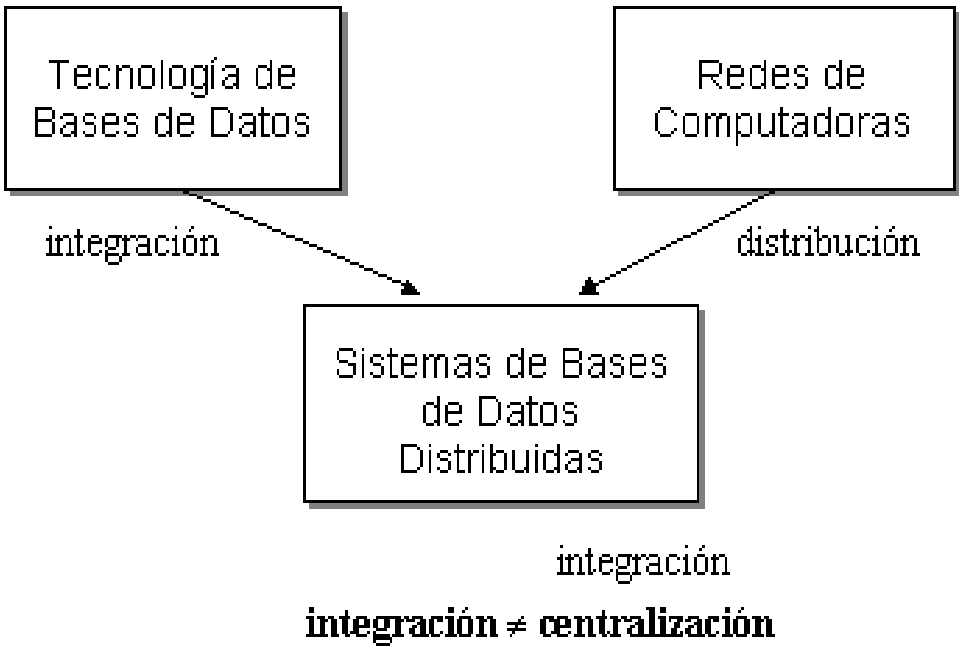


Figura 1.5.
Motivación de los sistemas de bases de datos distribuidos.

Una base de datos distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones.

Un sistema de bases de datos distribuida (SBDD) es un sistema en el cual múltiples sitios de bases de datos están ligados por un sistema de comunicaciones, de tal forma que, un usuario en cualquier sitio puede accesar los datos en cualquier parte de la red exactamente como si los datos estuvieran almacenados en su sitio propio.

Un sistema de manejo de bases de datos distribuidas (SMBDD) es aquel que se encarga del manejo de la BDD y proporciona un mecanismo de acceso que hace

que la distribución sea transparente a los usuarios. El término transparente significa que la aplicación trabajaría, desde un punto de vista lógico, como si un solo SMBD ejecutado en una sola máquina, administrara esos datos.

Un sistema de base de datos distribuida (SBDD) es entonces el resultado de la integración de una base de datos distribuida con un sistema para su manejo.

Dada la definición anterior, es claro que algunos sistemas no se pueden considerar como SBDD. Por ejemplo, un sistema de tiempo compartido no incluye necesariamente un sistema de manejo de bases de datos y, en caso de que lo haga, éste es controlado y administrado por una sola computadora.

Un sistema de multiprocesamiento puede administrar una base de datos pero lo hace usualmente a través de un solo sistema de manejo de base de datos; los procesadores se utilizan para distribuir la carga de trabajo del sistema completo o incluso del propio SMBD pero actuando sobre una sola base de datos. Finalmente, una base de datos la cual reside en un solo sitio de una red de computadoras y que es accesada por todos los nodos de la red no es una base de datos distribuida. Este caso se trata de una base de datos cuyo control y administración esta centralizada en un solo nodo pero se permite el acceso a ella a través de la red de computadoras.

El medio ambiente típico de un SMBDD consiste de un conjunto de sitios o nodos los cuales tiene un sistema de procesamiento de datos completo que incluye una base de datos local, un sistema de manejo de bases de datos y facilidades de comunicaciones. Si los diferentes sitios pueden estar geográficamente dispersos, entonces, ellos están interconectados por una red de tipo WAN. Por otro lado, si los sitios están localizados en diferentes edificios o departamentos de una misma organización pero geográficamente en la misma ubicación, entonces, están conectados por una red local (LAN).

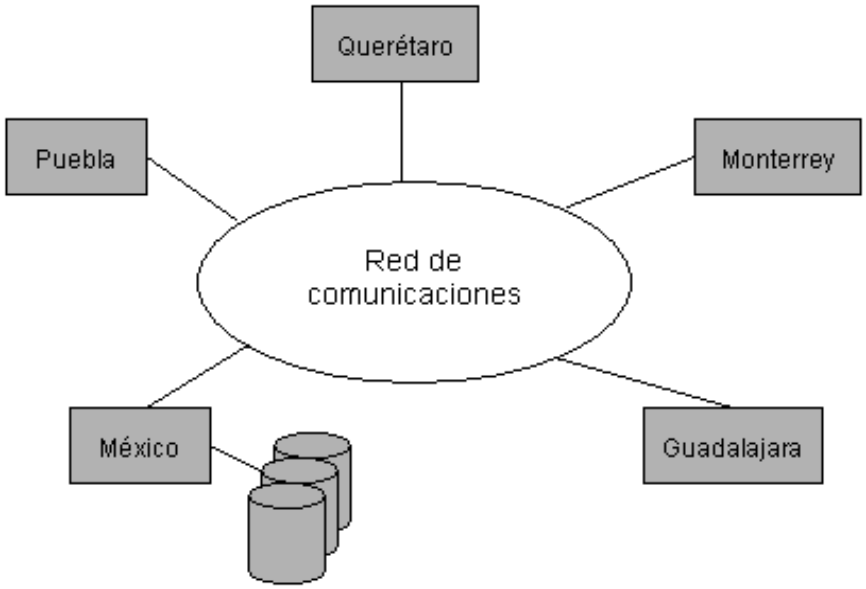


Figura 1.6
Un sistema centralizado sobre una red.

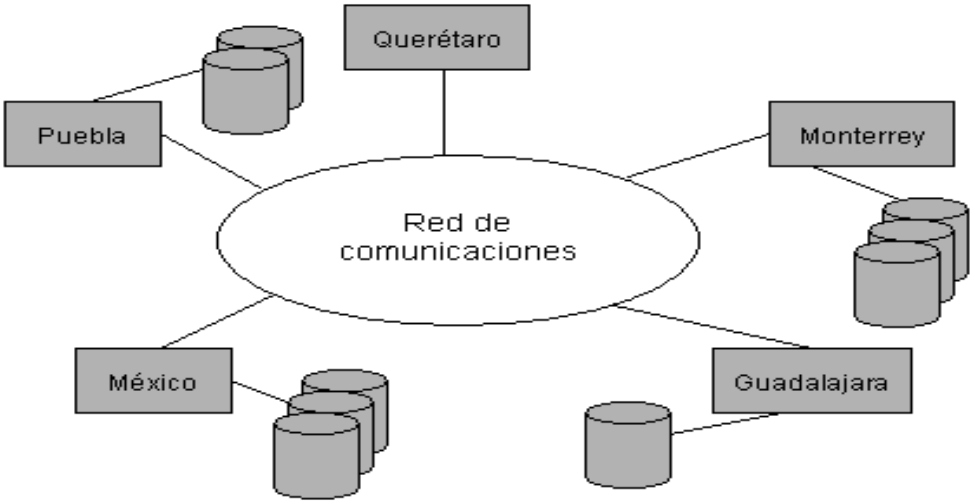


Figura 1.7
Un medio ambiente distribuido para bases de datos.

1.8.2.1 Aplicaciones

Los ambientes en los que se encuentra con mayor frecuencia el uso de las bases de datos distribuidas son:

- Cualquier organización que tiene una estructura descentralizada.
- Casos típicos de lo anterior son: organismos gubernamentales y/o de servicio público.

- La industria de la manufactura, particularmente, aquella con plantas múltiples. Por ejemplo, la industria automotriz.
- Aplicaciones de control y comando militar.
- Líneas de transportación aérea.
- Cadenas hoteleras.
- Servicios bancarios y financieros.

1.8.2.2 Ventajas

Los SMBDD tienen múltiples ventajas. En primer lugar los datos son localizados en lugar más cercano, por tanto, el acceso es más rápido, el procesamiento es rápido debido a que varios nodos intervienen en el procesamiento de una carga de trabajo, nuevos nodos se pueden agregar fácil y rápidamente. La comunicación entre nodos se mejora, los costos de operación se reducen, son amigables al usuario, la probabilidad de que una falla en un solo nodo afecte al sistema es baja y existe una autonomía e independencia entre los nodos.

Las razones por las que compañías y negocios migran hacia bases de datos distribuidas incluyen razones organizacionales y económicas, para obtener una interconexión confiable y flexible con las bases de datos existentes, y por un crecimiento futuro. El enfoque distribuido de las bases de datos se adapta más naturalmente a la estructura de las organizaciones. Además, la necesidad de desarrollar una aplicación global (que incluya a toda la organización), se resuelve fácilmente con bases de datos distribuidas. Si una organización crece por medio de la creación de unidades o departamentos nuevos, entonces, el enfoque de bases de datos distribuidas permite un crecimiento suave.

Los datos se pueden colocar físicamente en el lugar donde se accesan más frecuentemente, haciendo que los usuarios tengan control local de los datos con los que interactúan. Esto resulta en una autonomía local de datos permitiendo a los usuarios aplicar políticas locales respecto del tipo de accesos a sus datos.

Mediante la replicación de información, las bases de datos distribuidas pueden presentar cierto grado de tolerancia a fallas haciendo que el funcionamiento del sistema no dependa de un solo lugar como en el caso de las bases de datos centralizadas.

1.8.2.3 Desventajas

La principal desventaja se refiere al control y manejo de los datos. Dado que éstos residen en muchos nodos diferentes y se pueden consultar por nodos diversos de la red, la probabilidad de violaciones de seguridad es creciente si no se toman las precauciones debidas.

La habilidad para asegurar la integridad de la información en presencia de fallas no predecibles tanto de componentes de hardware como de software es compleja. La integridad se refiere a la consistencia, validez y exactitud de la información.

Dado que los datos pueden estar replicados, el control de concurrencia y los mecanismos de recuperación son mucho más complejos que en un sistema centralizado.

1.9 Componentes de sistemas de bases de datos

Los SGBD son paquetes de software muy complejos y sofisticados que deben proporcionar los servicios para el buen funcionamiento de la base de datos. No se puede generalizar sobre los elementos que componen un SGBD ya que varían mucho unos de otros. Sin embargo, es muy útil conocer sus componentes y cómo se relacionan cuando se trata de comprender lo que es un sistema de bases de datos.

Un SGBD tiene varios módulos, cada uno de los cuales realiza una función específica. El sistema operativo proporciona servicios básicos al SGBD, que es construido sobre él.

- El procesador de consultas es el componente principal de un SGBD. Transforma las consultas en un conjunto de instrucciones de bajo nivel que se dirigen al gestor de la base de datos.
- El gestor de la base de datos es el interface con los programas de aplicación y las consultas de los usuarios. El gestor de la base de datos acepta consultas y examina los esquemas externo y conceptual para determinar qué registros se requieren para satisfacer la petición. Entonces el gestor de la base de datos realiza una llamada al gestor de ficheros para ejecutar la petición.
- El gestor de ficheros maneja los ficheros en disco en donde se almacena la base de datos. Este gestor establece y mantiene la lista de estructuras e índices definidos en el esquema interno. Si se utilizan ficheros dispersos, llama a la función de dispersión para generar la dirección de los registros. Pero el gestor de ficheros no realiza directamente la entrada y salida de datos. Lo que hace es pasar la petición a los métodos de acceso del sistema operativo que se encargan de leer o escribir los datos en el buffer del sistema.
- El preprocesador del LMD convierte las sentencias del LMD embebidas en los programas de aplicación, en llamadas a funciones estándar escritas en el lenguaje anfitrión. El preprocesador del LMD debe trabajar con el procesador de consultas para generar el código apropiado.
- El compilador del LDD convierte las sentencias del LDD en un conjunto de tablas que contienen metadatos. Estas tablas se almacenan en el diccionario de datos.
- El gestor del diccionario controla los accesos al diccionario de datos y se encarga de mantenerlo. La mayoría de los componentes del SGBD acceden al diccionario de datos.

Los principales componentes del gestor de la base de datos son los siguientes:

- Control de autorización. Este módulo comprueba que el usuario tiene los permisos necesarios para llevar a cabo la operación que solicita.
- Procesador de comandos. Una vez que el sistema ha comprobado los permisos del usuario, se pasa el control al procesador de comandos.
- Control de la integridad. Cuando una operación cambia los datos de la base de datos, este módulo debe comprobar que la operación a realizar satisface todas las restricciones de integridad necesarias.
- Optimizador de consultas. Este módulo determina la estrategia óptima para la ejecución de las consultas.
- Gestor de transacciones. Este módulo realiza el procesamiento de las transacciones.
- Planificador (scheduler). Este módulo es el responsable de asegurar que las operaciones que se realizan concurrentemente sobre la base de datos tienen lugar sin conflictos.
- Gestor de recuperación. Este módulo garantiza que la base de datos permanece en un estado consistente en caso de que se produzca algún fallo.
- Gestor de buffers. Este módulo es el responsable de transferir los datos entre memoria principal y los dispositivos de almacenamiento secundario. A este módulo también se le denomina gestor de datos.

1.10 Funciones de sistemas manejadores de bases de datos

Codd, el creador del modelo relacional, ha establecido una lista con los ocho servicios que debe ofrecer todo SGBD.

1. Un SGBD debe proporcionar a los usuarios la capacidad de almacenar datos en la base de datos, acceder a ellos y actualizarlos. Esta es la función fundamental de un SGBD y por supuesto, el SGBD debe ocultar al usuario la estructura física interna (la organización de los ficheros y las estructuras de almacenamiento).
2. Un SGBD debe proporcionar un catálogo en el que se almacenen las descripciones de los datos y que sea accesible por los usuarios. Este catálogo es lo que se denomina diccionario de datos y contiene información que describe los datos de la base de datos (metadatos). Normalmente, un diccionario de datos almacena:

    - Nombre, tipo y tamaño de los datos.
    - Nombre de las relaciones entre los datos.
    - Restricciones de integridad sobre los datos.
    - Nombre de los usuarios autorizados a acceder a la base de datos.

- Esquema externo, conceptual e interno, y correspondencia entre los esquemas.
- Estadísticas de utilización, tales como la frecuencia de las transacciones y el número de accesos realizados a los objetos de la base de datos.

Algunos de los beneficios que reporta el diccionario de datos son los siguientes:

- La información sobre los datos se puede almacenar de un modo centralizado. Esto ayuda a mantener el control sobre los datos, como un recurso que son.
- El significado de los datos se puede definir, lo que ayudará a los usuarios a entender el propósito de los mismos.
- La comunicación se simplifica ya que se almacena el significado exacto. El diccionario de datos también puede identificar al usuario o usuarios que poseen los datos o que los acceden.
- Las redundancias y las inconsistencias se pueden identificar más fácilmente ya que los datos están centralizados.
- Se puede tener un historial de los cambios realizados sobre la base de datos.
- El impacto que puede producir un cambio se puede determinar antes de que sea implementado, ya que el diccionario de datos mantiene información sobre cada tipo de dato, todas sus relaciones y todos sus usuarios.
- Se puede hacer respetar la seguridad.
- Se puede garantizar la integridad.
- Se puede proporcionar información para auditorías.

3. Un SGBD debe proporcionar un mecanismo que garantice que todas las actualizaciones correspondientes a una determinada transacción se realicen, o que no se realice ninguna. Una transacción es un conjunto de acciones que cambian el contenido de la base de datos. Una transacción en el sistema informático de la empresa inmobiliaria sería dar de alta a un empleado o eliminar un inmueble. Una transacción un poco más complicada sería eliminar un empleado y reasignar sus inmuebles a otro empleado. En este caso hay que realizar varios cambios sobre la base de datos. Si la transacción falla durante su realización, por ejemplo porque falla el hardware, la base de datos quedará en un estado inconsistente. Algunos de los cambios se habrán hecho y otros no, por lo tanto, los cambios realizados deberán ser deshechos para devolver la base de datos a un estado consistente.
4. Un SGBD debe proporcionar un mecanismo que asegure que la base de datos se actualice correctamente cuando varios usuarios la están actualizando concurrentemente. Uno de los principales objetivos de los SGBD es el permitir que varios usuarios tengan acceso concurrente a los

datos que comparten. El acceso concurrente es relativamente fácil de gestionar si todos los usuarios se dedican a leer datos, ya que no pueden interferir unos con otros. Sin embargo, cuando dos o más usuarios están accediendo a la base de datos y al menos uno de ellos está actualizando datos, pueden interferir de modo que se produzcan inconsistencias en la base de datos. El SGBD se debe encargar de que estas interferencias no se produzcan en el acceso simultáneo.

5. Un SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos en caso de que ocurra algún suceso que la dañe. Como se ha comentado antes, cuando el sistema falla en medio de una transacción, la base de datos se debe devolver a un estado consistente. Este fallo puede ser a causa de un fallo en algún dispositivo hardware o un error del software, que hagan que el SGBD aborte, o puede ser a causa de que el usuario detecte un error durante la transacción y la aborte antes de que finalice. En todos estos casos, el SGBD debe proporcionar un mecanismo capaz de recuperar la base de datos llevándola a un estado consistente.
6. Un SGBD debe proporcionar un mecanismo que garantice que sólo los usuarios autorizados pueden acceder a la base de datos. La protección debe ser contra accesos no autorizados, tanto intencionados como accidentales.
7. Un SGBD debe ser capaz de integrarse con algún software de comunicación. Muchos usuarios acceden a la base de datos desde terminales. En ocasiones estos terminales se encuentran conectados directamente a la máquina sobre la que funciona el SGBD. En otras ocasiones los terminales están en lugares remotos, por lo que la comunicación con la máquina que alberga al SGBD se debe hacer a través de una red. En cualquiera de los dos casos, el SGBD recibe peticiones en forma de mensajes y responde de modo similar. Todas estas transmisiones de mensajes las maneja el gestor de comunicaciones de datos. Aunque este gestor no forma parte del SGBD, es necesario que el SGBD se pueda integrar con él para que el sistema sea comercialmente viable.
8. Un SGBD debe proporcionar los medios necesarios para garantizar que tanto los datos de la base de datos, como los cambios que se realizan sobre estos datos, sigan ciertas reglas. La integridad de la base de datos requiere la validez y consistencia de los datos almacenados. Se puede considerar como otro modo de proteger la base de datos, pero además de tener que ver con la seguridad, tiene otras implicaciones. La integridad se ocupa de la calidad de los datos. Normalmente se expresa mediante restricciones, que son una serie de reglas que la base de datos no puede violar. Por ejemplo, se puede establecer la restricción de que cada empleado no puede tener asignados más de diez inmuebles. En este caso sería deseable que el SGBD controlara que no se sobrepase este límite cada vez que se asigne un inmueble a un empleado.

Además, de estos ocho servicios, es razonable esperar que los SGBD proporcionen un par de servicios más:

1. Un SGBD debe permitir que se mantenga la independencia entre los programas y la estructura de la base de datos. La independencia de datos se alcanza mediante las vistas o subesquemas. La independencia de datos física es más fácil de alcanzar, de hecho hay varios tipos de cambios que se pueden realizar sobre la estructura física de la base de datos sin afectar a las vistas. Sin embargo, lograr una completa independencia de datos lógica es más difícil. Añadir una nueva entidad, un atributo o una relación puede ser sencillo, pero no es tan sencillo eliminarlos.
2. Un SGBD debe proporcionar una serie de herramientas que permitan administrar la base de datos de modo efectivo. Algunas herramientas trabajan a nivel externo, por lo que habrán sido producidas por el administrador de la base de datos. Las herramientas que trabajan a nivel interno deben ser proporcionadas por el distribuidor del SGBD. Algunas de ellas son:

    - Herramientas para importar y exportar datos.
    - Herramientas para monitorizar el uso y el funcionamiento de la base de datos.
    - Programas de análisis estadístico para examinar las prestaciones o las estadísticas de utilización.
    - Herramientas para reorganización de índices.
    - Herramientas para aprovechar el espacio dejado en el almacenamiento físico por los registros borrados y que consoliden el espacio liberado para reutilizarlo cuando sea necesario.

1.11 Arquitecturas de bases de datos

Hay tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. En esta arquitectura, el esquema de una base de datos se define en tres niveles de abstracción distintos:

1. En el nivel interno se describe la estructura física de la base de datos mediante un esquema interno. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.
2. En el nivel conceptual se describe la estructura de toda la base de datos para una comunidad de usuarios (todos los de una empresa u

organización), mediante un esquema conceptual. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.

3. En el nivel externo se describen varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinados y oculta a ese grupo el resto de la base de datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas.

La mayoría de los SGBD no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. En casi todos los SGBD que se manejan vistas de usuario, los esquemas externos se especifican con el mismo modelo de datos que describe la información a nivel conceptual, aunque en algunos se pueden utilizar diferentes modelos de datos en los niveles conceptual y externo.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada. Si la petición es de una obtención (consulta) de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada, para que coincida con la vista externa del usuario. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación. Estas correspondencias pueden requerir bastante tiempo, por lo que algunos SGBD no cuentan con vistas externas.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

- La independencia lógica es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.
- La independencia física es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de

mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

En los SGBD que tienen la arquitectura de varios niveles es necesario ampliar el catálogo o diccionario, de modo que incluya información sobre cómo establecer la correspondencia entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el catálogo. La independencia de datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios, sólo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Por lo tanto, la arquitectura de tres niveles puede facilitar la obtención de la verdadera independencia de datos, tanto física como lógica. Sin embargo, los dos niveles de correspondencia implican un gasto extra durante la ejecución de una consulta o de un programa, lo cual reduce la eficiencia del SGBD. Es por esto que muy pocos SGBD han implementado esta arquitectura completa.

1.12 ACTIVIDADES COMPLEMENTARIAS

1. ¿Cuáles son las cuatro diferencias principales entre un sistema de procesamiento de archivos y un SGDB?.

2. Investigue y determine cuáles son los inconvenientes de un sistema gestor de base de datos.

3. Elabore un cuadro y explique la diferencia entre independencia de datos física y lógica.

4. Explique las cinco responsabilidades del sistema gestor de la base de datos,¿Que ocurriría si no se realizara alguna de estas funciones?

5. ¿Cuáles son las cinco funciones principales del administrador de la base de datos?

6. Investigue y amplié la información relacionada con la arquitectura de dos y tres capas.

7. Explique mediante un cuadro comparativo las ventajas y desventajas entre una base de datos centralizada y una base de datos distribuida.

CONSULTAS WEB

- La página inicial del grupo especial de interés de la ACM en gestión de datos proporciona una gran cantidad de información sobre la investigación en bases de datos. http://www.acm.org/sigmodl

CAPITULO 2. PLAN ESTRATÉGICO DE DISEÑO DE BASES DE DATOS

Los cambios sustanciales en el plano económico, político y tecnológico que han tenido lugar en el ámbito internacional y el impacto directo de ellos en la economía han transformado el entorno y las condiciones en que operan nuestras organizaciones.

La estabilidad y funcionamiento de las organizaciones se reduce cada vez más y pasan a primer plano las situaciones de cambio, lo que exige una nueva mentalidad en los dirigentes.

Las organizaciones actuales deben ser conducidas sobre la base de cuatro ideas básicas que representan la guía para la elaboración de políticas claves que posibilitan a la organización un nivel de gestión de alta efectividad. Estas ideas son:

- Disposición permanente a dar flexibilidad a los sistemas de producción. Programas rígidos inflexibles, son incompatibles con los conceptos actuales y con la situación de cambio constante en las que están inmersas las organizaciones.

- Atención sistemática a la reducción de gastos, significando esto un elemento clave que no puede estar ausente de la mente de los directivos y sus subordinados por representar la base para la obtención de utilidades.

- Alto sentido de responsabilidad y atención al cliente como fuente esencial para la imagen y prestigio que llevan al éxito en el mercado a cualquier organización.

- Agresividad, visión amplia y rapidez con relación a la introducción sistemática y oportuna de innovaciones y cambios tecnológicos.

2.1 Etapas en la planeación

Cuando se desea hacer la planeación es necesario identificar el flujo y el orden de las etapas que guiaran los esfuerzos de la empresa en el logro de los objetivos planteados, en la siguiente grafica se observa el orden de estas etapas.

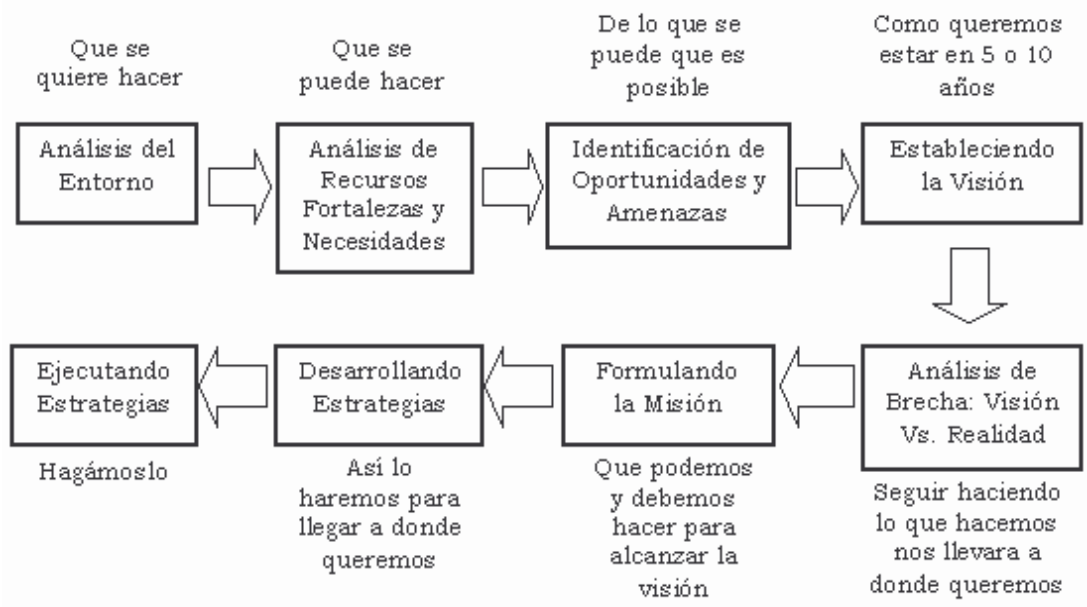


Figura 2.1
Etapas de la planeación

2.2 Importancia de la Planeación Estratégica

La planeación estratégica es importante porque ayudará a lograr una utilización y aplicación más efectiva de los escasos recursos con que cuenta una empresa, a nivel humano, financiero y tecnológico.

El mundo está plagado de documentos que contienen las declaraciones de misiones, valores, planes a corto y largo plazo, estrategias y programas. En ocasiones parece que las organizaciones compiten entre si para formular planes y declaraciones más complejas.

La experiencia ha demostrado que la complejidad de los planes y declaraciones en la mayoría de los casos esta en relación inversamente proporcional a su tamaño; por lo tanto un buen plan debe estar compuesto de 3 cualidades fundamentales que son:

- Sencillez
- Flexibilidad
- Credibilidad

Dando respuesta a los siguientes interrogantes:

- ¿Qué sé hacer?
- ¿Qué sé hacer mejor?
- ¿En qué soy mejor que mis competidores?

- ¿Qué podría aprender y desarrollar?
- ¿Con qué recursos cuento?

El plan debe especificar los asuntos más importantes que se van a considerar, debe identificar las diversas áreas operacionales claves y quienes serán sus responsables; el plan estratégico debe precisar los principios globales de trabajo de la organización, debe aclarar que los planes no se ejecutan por si mismos, por lo que es fundamental identificar con anticipación los problemas que pudieran presentarse en su ejecución.

2.3 Plan estratégico de diseño de bases de datos

Cuando se va a diseñar una base de datos, es importante considerar que los datos son el centro medular de cualquier sistema de información y en su diseño no se debe olvidar que las cualidades funcionales de una base de datos son:

- Disponibilidad de los datos
- Consistencia y seguridad de los datos
- Eficiencia en la actualización y recuperación de los datos

En el campo del desarrollo de sistemas de información, sobre ambientes de Bases de datos es muy variada entre las que se pueden mencionar CSF(Critical Success Factor) de interés general, BSP(Business System Planning) propuesta por IBM, o ISSP(Information System Strategic Planning) propuesta por Eliot.

De los anteriores tratamientos, se pueden destacar algunos factores de importancia que por lo general son comunes en estas temáticas:

- Misión, visión, Metas, Objetivos
- Descripción del bien o servicio
- Riesgos
- Factores claves de evaluación
- Planes de acción

La misión se enuncia de una manera única, distingue a la organización del resto, las metas organizacionales suministran un sentido básico de dirección, La Estrategia crea una dirección para la organización con base en sus diversos objetivos y orienta la movilización de recursos, empleados para encausar a la organización hacia consecución de las metas. (Stoner y Wankel).

El plan estratégico de diseño de bases de datos es producto de un acto creador, innovador, lógico y aplicable, que genera un grupo de acciones coherentes de asignación de recursos y decisiones tácticas encaminadas a lograr que la empresa alcance una posición competitiva ventajosa en el entorno socioeconómico donde

se desenvuelve y mejorar la eficacia de la gestión, mediante le uso de las bases de datos.

Aspectos generales sobre el plan: El Plan Estratégico es el documento más importante que se debe redactar en la empresa; en él se explica hacia dónde se quiere ir y lo más importante cómo se llegará; debe ser una herramienta creada por los directivos que utilizarán para esta tarea la información brindada por diferentes Departamentos o áreas de la organización.

Hay que tener en claro que cualquier Plan Estratégico, a medida que se va implementando necesita ajustes, por lo que debe estar sometido a revisiones constantes, fundamentalmente de manos del grupo que lo creó.

A continuación se propone un "borrador" sobre el cual usted podrá trabajar en el momento de desarrollar el Plan Estratégico para el desarrollo de la base de datos de su empresa, o del proyecto que se desee realizar:

a) Comience por contratar un coordinador: Deberá ser un profesional con importante experiencia en coordinar tareas de este tipo, por ejemplo un Consultor que no tenga ningún interés personal en la estructura final del proyecto. De esta manera la experiencia y la neutralidad sumarán múltiples beneficios en la tarea de redacción del documento.

b) Arme EL Equipo de Planificación de su empresa: Elegir los miembros no es tarea fácil, pero recuerde que los integrantes del grupo deben contar con toda la información; siempre es bueno que alguien permanezca como respaldo y apoyo en la oficina mientras se desarrolla el outside por si llegara a faltar algún dato o material importante.

c) Programe esta actividad fuera de su ambiente laboral: Lo ideal es retirarse un fin de semana con todo el grupo encargado de planificar el futuro de la empresa y trabajar con la mayor comodidad, fuera del ambiente de trabajo y de las interrupciones diarias que esto conlleva.

d) Obtenga los aportes de otros integrantes de su empresa: Aunque estos no participen directamente en la confección del Plan. Es importante envolver en este proyecto no sólo a los integrantes del grupo, sino también a los demás recursos humanos de la empresa, especialmente si cuentan con gente a cargo y con poder de decisión, cada dato que ellos aporten beneficiará el contenido del Plan y los comprometerá a la vez en el momento de su implementación.

e) Declare la Misión: Después de haber logrado lo anterior, estamos en condiciones de declarar la Misión del Plan a nivel empresa; todo proyecto debe tener su enunciado de Misión, que no difiera de la Misión institucional, preguntas como:

- Por qué existe la empresa.
- En qué sector comercial se desenvuelve.
- Qué produce, qué servicios ofrece.
- Qué mercados atiende.
- Qué es lo que requieren los clientes de la empresa y de sus directores o ejecutivos.
- De qué manera logramos reducir la ansiedad, los riesgos, los costos y fracasos e incrementar el placer, la seguridad, la satisfacción, la rentabilidad y el éxito de los clientes.

Ayudan en gran medida en la declaración Misión del proyecto, el enunciado debe ser muy breve y simple, pero debe reflejar todos estos puntos tan importantes.

f) Formule una declaración para la Visión del proyecto: Aquí se debe especificar qué se quiere para el futuro, use de guía preguntas como:

- Cómo quiere que sea considerado en su empresa.
- Qué sueños espera poder realizar.
- Cómo quiere ser reconocido en su entorno.
- Qué posición proyecta ocupar a nivel de empresa.
- Qué estándares de calidad piensa alcanzar.

g) Agregue a la declaración de la Visión sus valores y convicciones: Las aspiraciones proponen los resultados deseados; los valores y las convicciones establecen cómo hará la organización para alcanzarlos.

h) Lleve a cabo una evaluación de las políticas: Se realizan de acuerdo a lo que existe en relación al mercado potencial, tome como guía preguntas como:

- Cuáles son las principales tendencias (políticas, económicas, tecnológicas, religiosas, sociales, intelectuales, artísticas) que afectan a su negocio.
- Qué tendencias representan una amenaza para su organización.
- Cuáles de esas tendencias constituyen una oportunidad.

i) Desarrolle un análisis DOFA de su empresa: Haciendo énfasis en las Debilidades, Oportunidades, Fortalezas y Amenazas de cada área involucrada, tome como guía preguntas tales como:

- Qué es lo que la empresa hace bien.
- Qué es lo que no hace tan bien.
- Cuáles son los puntos fuertes globales y particulares en los que se destaca.
- Cuáles son, claramente, sus puntos débiles y sus deficiencias.
- Cuenta con alguna ventaja competitiva especial en el mercado.

j) Defina los objetivos del proyecto a corto plazo: Tenga en cuenta los resultados de la evaluación externas e internas. Piense en objetivos que den respuesta a la necesidad de cambio, que aprovechen las oportunidades externas y mejoren el proceso de toma de decisiones de la empresa.

k) Determine los Objetivos a largo plazo: Para tal fin ayúdese de preguntas como:

   - Qué se debería hacer de aquí a tres años.

   - Y qué en cinco años.

   Una vez que tenga este punto y el anterior definidos en su totalidad haga un listado por orden de importancia para luego definir los planes de acción.

l) Haga un análisis de los factores que inciden sobre cada Objetivo: Por ningún motivo debe suprimirse este punto, si usted no ha logrado determinar los objetivos, con seguridad hay razones que no se lo han permitido.

   - Cuáles son estas fuerzas, los factores o actitudes que actúan en forma negativa en el logro de ese objetivo específico. Descúbralas para eliminarlas.

   - Cuáles son las fuerzas, factores o actitudes que favorecen el logro de ese objetivo. Busque la manera de potenciar esas fuerzas positivas.

m) Desarrolle Planes de Acción: Para alcanzar los objetivos planteados se hace necesario de planes de acción que nos permitan lograrlos uno a uno, tome como base preguntas como las siguientes:

   - Cuáles serán las medidas que reducirán los aspectos negativos y maximizarán los positivos en el análisis hecho en el punto anterior.

   - Quién, dentro de su equipo, aceptará la responsabilidad de asegurar la implementación de cada una de esas medidas.

   - Cuándo y dónde serán implementadas.

   - Qué recursos se necesitarán.

n) Redacte un memorando con la asignación de funciones: Registre cada medida propuesta en orden cronológico y distribúyalo a cada uno de los colaboradores involucrados en el proceso. A través de reuniones semanales, controle el progreso que se va produciendo en las tareas que

debían iniciase la semana anterior y revise las que deberán implementarse en la semana próxima tomando las medidas correctivas que hicieran falta.

Una vez que el Plan esté definido y comience a ser aplicado en su empresa, tenga en cuenta que será muy importante auditarlo al cabo de algunos meses, junto con el Consultor que trabajó en su formulación; se deben controlar los logros generales alcanzados dentro del Plan y eventualmente tomado como base los resultados se podrán aplicar los cambios que sean necesarios.

2.4 Áreas funcionales de la organización

Cuando lo que se desea es desarrollar una base de datos para que almacene los datos de la empresa y para que nos ayude a mejorar los proceso de manejo de información y toma de decisiones es imprescindible que todos los componentes funcionales de la empresa se vean involucrados en su desarrollo, no debemos olvidar que las áreas administrativas toman decisiones con la información proveniente de los niveles operativos, razón por la cual el desconocimiento de esta regla trivial de gestión hará que todo proceso de innovación fracase.

Las organizaciones obedecen funcionalmente a un tipo de estructura funcional que depende básicamente del tipo de empresa que se este tratando, las empresas de prestación de servicios presentan una estructura diferente a la estructura de las empresas de manufactura o industriales, igualmente que a la estructura de las empresas comerciales y así sucesivamente con todos y cada uno de los tipos de empresas que existan.

Un proyecto de diseño de bases de datos debe ser liderado por un directivo del área informática avalado por un directivo de alto rango, esto con el fin de asegurar que el desarrollo llegue a feliz término y que la asignación de recursos sea la apropiada para este tipo de proyectos.

La resistencia al cambio es uno de los principales riesgos a los que se expone un proyecto de sistematización, debido al falso concepto que tiene el empleado del desarrollo tecnológico, pues se sigue creyendo que “Las maquinas desplazan al hombre”, lo cual no es cierto en su totalidad debido a que las maquinas solo contribuyen en la ejecución de tareas rutinarias y sirven de apoyo para la realización eficiente del trabajo humano.

El plan estratégico debe ser formulado por la alta administración con el fin de supervisar los intereses y las operaciones de organizaciones que cuentan con más de una línea de negocios.

La formulación del plan deben responder a preguntas como:

- Tipo y alcance de la base de datos a desarrollar.

- Metas y las expectativas para cada área involucrada.
- Cómo se deben asignar los recursos para que se puedan alcanzar las metas.
- Qué tipos de datos de deberán almacenar.
- Quienes serán los usuarios de la BD.
- Quien deberá administrar el recurso.
- Que áreas se verán involucradas (Producción, Mercadeo, Finanzas, Administración).

2.5 Procesos propios de cada área funcional

Tomando como base las respuesta obtenida del numeral anterior, se podrá identificar con claridad que áreas estarán comprometidas en este plan proceso y de acuerdo al tipo de información que se desea almacenar en la base de datos, es que se determinan los procesos del área que inciden directa o indirectamente con el desarrollo propuesto.

Cada proceso deberá estar lo suficientemente documentado para obtener el mayor cúmulo de información que permita traducirla en objetos de una base de datos, entre mas información se tenga de un proceso mayor será el grado de utilidad que presente la base de datos para quien posteriormente la use.

2.6 Actividades propias de cada proceso

Funcionalmente todo proceso se divide en actividades claramente diferenciadas la una de la otra y ordenadas en secuencia. Recordando lo anteriormente expresado referente a los tipos de organización así mismo es como podemos establecer todos y cada uno de los proceso y su composición por actividades que suelen ser únicas al entorno en el cual se desarrollen.

2.7 Entidades de información

Conociendo los procesos y las actividades que los conforman al interior de una organización es que se logra determinar que entidades de información soportan en su totalidad los requerimientos de datos para la normal ejecución de cada actividad, de cada proceso y así mismo de la misión institucional.

La definición de una entidad de información es el paso mas importante para el diseño de una base de datos, debido a que las entidades son abstracciones de objetos del mundo real que se encuentra constituidas por características llamadas atributos que son fácilmente identificables el las ares donde se ejecutan las actividades de un proceso empresarial.

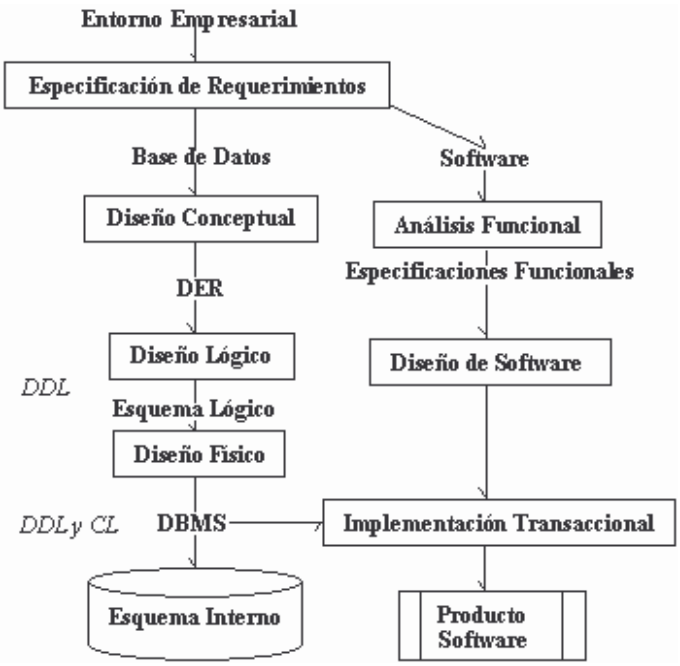
Las entidades de datos no son elementos aislados, sino que por el contrario son objetos de datos que presentan relaciones con directas e indirectas encargadas de ampliar los ámbitos de los datos para los usuarios que las utilizan.

En bases de datos una entidad es lo que se expuso anteriormente en lo referente a archivos. Los campos allí señalados son lo que se acaba de expresar como atributo en la entidad, razón por la cual no es descabellado expresar que la estructura de una base de datos se basa fundamentalmente en entidades y relaciones.

2.8 Proceso de desarrollo

El plan estratégico para el desarrollo de bases de datos, deberá estar enmarcado dentro de las políticas empresariales, y deberá ser consistente con la evolución y desarrollo empresarial, por tal razón el diseño de bases de datos será un proceso detallado y consistente que se ajuste a un proceder metodológico, cada fase deberá ser la base para el inicio de la siguiente; un ejemplo de este comportamiento puede observarse con detalle en el siguiente diagrama:

Figura 2.2 Proceso de desarrollo



Aquí se muestra como el desarrollo de la base de datos es paralelo al desarrollo del software de aplicación que va ha hacer uso de la BD, esto permitirá que algunas especificaciones procedí mentales del software puedan ser involucradas en los detalles lógicos y físicos de la base de datos de tal forma que responda con suficiencia a las aplicaciones de usuario final. En el diagrama se desatacan las fases por las cuales atraviesa una base de datos antes de ponerse al servicio del usuario, si en el plan se han tenido en cuanta estas fases con seguridad la ejecución del plan dará como resultado un instrumento corporativo que apoye el proceso de toma de decisiones.

2.9 Actividades Complementarias

1. Tomando como base la empresa donde usted trabaja, se solicita, un documento donde se establezcan los siguientes elementos de la planeación:

   - Visión
   - Misión
   - Metas
   - Objetivos
   - Estrategias para cada fase del desarrollo

Para lo anterior, haga uso de los datos obtenidos por usted en el análisis de la alternativa de solución propuesta y en la documentación lograda en sus visitas hechas al cliente en su trabajo de campo, realice y aporte como herramienta la matriz DOFA que se ajuste a la realidad de la entidad escogida.

2. ¿Cuál es la diferencia entre formalización, especialización y estandarización? ¿Cree que una organización que es alta en una de estas tres dimensiones también podría serlo en las restantes? Analícelo.

3. ¿Cuáles son los cinco subsistemas en las organizaciones? Si una organización tuviera que prescindir de uno de ellos, ¿sin cuál podría sobrevivir más tiempo?, Explique.

4. Visite tres empresas de la región y consulte con el administrador de la base de datos los siguiente:

   - ¿Que entidades y atributos existen en cada una de las bases de datos que administra?
   - ¿De que manera acceden a la base de datos para realizar posibles consultas y análisis de la información?
   - ¿Qué capacitación recibieron en el manejo de las herramientas de administración de bases de datos que manejan?
   - ¿Qué aspectos de la base de datos mejoraría o volvería a replantear y porque?
   - ¿Qué SMBD maneja y porque?

   Compare el resultado de las tres visitas y realice un cuadro con las diferencias, ventajas y desventajas encontradas en cada una de ellas.

UNIDAD 2. MODELADO DE DATOS

CAPITULO 3. CONCEPTO DE MODELO DE DATOS

3.1 Introducción

Entre las consideraciones fundamentales al organizar los datos en una base de datos se incluye, la determinación de los datos que deben recopilar, quienes tendrá acceso a la información y como podrían desear utilizar los datos almacenados. Con base en estas determinaciones se puede crear una base de datos. Su creación requiere de dos tipos de diseño diferentes: un diseño lógico y uno físico.

El diseño lógico de una base de datos muestra un modelo abstracto de cómo se deben estructurar y ordenar los datos para cumplir con las necesidades de información en una organización. El diseño lógico de una base de datos incluye la identificación de las relaciones entre las diferentes sesiones de datos y su agrupamiento en forma ordenada.

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los modelos de datos son el instrumento principal para ofrecer dicha abstracción.

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas (lecturas) y actualizaciones de datos. Además, los modelos de datos más modernos incluyen conceptos para especificar comportamiento, permitiendo especificar un conjunto de operaciones definidas por el usuario.

Los modelos de datos se pueden clasificar dependiendo de los tipos de conceptos que ofrecen para describir la estructura de la base de datos. Los modelos de datos de alto nivel, o modelos conceptuales, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos, mientras que los modelos de datos de bajo nivel, o modelos físicos, proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el ordenador.

Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Entre estos dos extremos se encuentran los modelos lógicos, cuyos conceptos pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles de cómo se almacenan los datos, pero pueden implementarse de manera directa en un ordenador.

Los modelos conceptuales utilizan conceptos como entidades, atributos y relaciones. Una entidad representa un objeto o concepto del mundo real como, por ejemplo, un empleado de la empresa inmobiliaria o una oficina. Un atributo representa alguna propiedad de interés de una entidad como, por ejemplo, el nombre o el salario del empleado. Una relación describe una interacción entre dos o más entidades, por ejemplo, la relación de trabajo entre un empleado y su oficina.

Cada SGBD soporta un modelo lógico, siendo los más comunes el relacional, el de red y el jerárquico. Estos modelos representan los datos valiéndose de estructuras de registros, por lo que también se denominan modelos orientados a registros. Hay una nueva familia de modelos lógicos, son los modelos orientados a objetos, que están más próximos a los modelos conceptuales.

Los modelos físicos describen cómo se almacenan los datos en el ordenador: el formato de los registros, la estructura de los ficheros (desordenados, ordenados, etc.) y los métodos de acceso utilizados (índices, etc.).

A la descripción de una base de datos mediante un modelo de datos se le denomina esquema de la base de datos. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo. Sin embargo, los datos que se almacenan en la base de datos pueden cambiar con mucha frecuencia: se insertan datos, se actualizan, etc. Los datos que la base de datos contiene en un determinado momento se denominan estado de la base de datos u ocurrencia de la base de datos.

La distinción entre el esquema y el estado de la base de datos es muy importante. Cuando definimos una nueva base de datos, sólo especificamos su esquema al SGBD. En ese momento, el estado de la base de datos es el ``estado vacío", sin datos. Cuando se cargan datos por primera vez, la base datos pasa al ``estado inicial". De ahí en adelante, siempre que se realice una operación de actualización de la base de datos, se tendrá un nuevo estado. El SGBD se encarga, en parte, de garantizar que todos los estados de la base de datos sean estados válidos que satisfagan la estructura y las restricciones especificadas en el esquema.

Por lo tanto, es muy importante que el esquema que se especifique al SGBD sea correcto y se debe tener muchísimo cuidado al diseñarlo. El SGBD almacena el esquema en su catálogo o diccionario de datos, de modo que se pueda consultar siempre que sea necesario.

3.2 Definición de Modelos de datos.

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información.

Con respecto al diseño de bases de datos, el modelado de datos puede ser descrito así (Brodie 1984:20): "dados los requerimientos de información y proceso de una aplicación de uso intensivo de datos (por ejemplo, un sistema de información), construir una representación de la aplicación que capture las propiedades estáticas y dinámicas requeridas para dar soporte a los procesos deseados (por ejemplo, transacciones y consultas). Además de capturar las necesidades dadas en el momento de la etapa de diseño, la representación debe ser capaz de dar cabida a eventuales futuros requerimientos".

Un modelo de datos es por tanto una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. Conceptualmente, una aplicación puede ser caracterizada por:

- Propiedades estáticas: entidades (u objetos), propiedades (o atributos) de esas entidades, y relaciones entre esas entidades.
- Propiedades dinámicas: operaciones sobre entidades, sobre propiedades o relaciones entre operaciones.
- Reglas de integridad sobre las entidades y las operaciones (por ejemplo, transacciones).

Así, un modelo de datos se distingue de otro por el tratamiento que da a estas tres categorías. El resultado de un modelado de datos es una representación que tiene dos componentes: las propiedades estáticas se definen en un esquema y las propiedades dinámicas se definen como especificaciones de transacciones, consultas e informes.

Un esquema consiste en una definición de todos los tipos de objetos de la aplicación, incluyendo sus atributos, relaciones y restricciones estáticas. Correspondientemente, existirá una reposición de información, la base de datos, que es una instancia del esquema.

Un determinado tipo de procesos sólo necesita acceder a un subconjunto predeterminado de entidades definidas en un esquema, por lo que este tipo de procesos puede requerir sólo un subconjunto de las propiedades estáticas del esquema general. A este subconjunto de propiedades estáticas se le denomina subesquema.

Una transacción consiste en diversas operaciones o acciones sobre las entidades de esquema o subesquema. Una consulta se puede expresar como una expresión lógica sobre los objetos y relaciones definidos en el esquema; una consulta identifica un subconjunto de la base de datos. Las herramientas que se usan para realizar las operaciones de definición de las propiedades estáticas y dinámicas de la base de datos son los lenguajes de definición y manipulación de datos (DDL, DML), junto con los lenguajes de consulta (QL).

3.3 Clasificación de los Modelos de datos

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), es un método para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas; son abstracciones que permiten la implementación de un sistema eficiente de base de datos, por lo general se refieren a algoritmos, y conceptos matemáticos.

3.3.1 Modelos Lógicos Basados en Objetos

Los modelos lógicos basados en objetos se usan para describir los datos en los niveles conceptual y de visión. Se caracterizan porque proporcionan una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente.

3.3.1.2 Modelo Entidad-Relación

El modelo E-R (Entidad-Relación) es un modelo de datos conceptual de alto nivel y que se suele utilizar bastante en el diseño de bases de datos. Se basa en una percepción del mundo real que consiste en un conjunto de objetos básicos denominados entidades y relaciones, y se desarrolló para facilitar el diseño de bases de datos.

El modelo E-R crea un modelo de la realidad que se asimila a la realidad que queremos modelar, y lo hace de forma que es independiente de la implementación posterior, ofreciendo un alto nivel de abstracción, y siendo una herramienta gráfica fácil de comprender.

El resultado del modelado E-R es un diagrama E-R que representa una estructura lógica general de la base de datos.

3.3.1.3 Modelo Orientado a Objetos

Este modelo también se basa en la percepción de una colección de objetos. Un objeto se caracteriza por tener un estado y un comportamiento. El estado corresponde a los valores que toman un conjunto de propiedades o variables de instancia, y el comportamiento es llevado a cabo mediante una serie de operaciones o funciones que operan sobre el objeto, y que se denominan métodos. Los objetos que tienen el mismo tipo de propiedades y el mismo comportamiento son agrupados en clases. Dichas clases se organizan en un diagrama o jerarquía de clases, en el que las clases pueden estar relacionadas mediante relaciones de asociación o mediante relaciones de herencia. La herencia permite la definición de clases a partir de clases existentes heredándose a las

nuevas clases las propiedades y el comportamiento de las clases existentes, cumpliéndose también que todos los objetos de una subclase también es objeto de su superclase.

La única forma en la que un objeto puede acceder a los datos de otro objeto es a través de los métodos de este objeto. Esto se denomina envío de mensajes al objeto. De esta forma, la interfaz de llamada mediante los métodos de un objeto define la parte visible, mientras que la parte interna del objeto (variables y código de los métodos) no es visible externamente. De esta forma se tienen dos niveles de abstracción.

Por ejemplo, sea un objeto que representa a una cuenta corriente, y que dicho objeto contiene las variables de instancia numeroDeCuenta y saldo. Este objeto puede tener un método denominado Ingresar que añade una cantidad al saldo.

A diferencia del modelo E-R, en el modelo OO, cada objeto tiene su propia entidad que viene dado por un OID (identificador del objeto) asignado por el sistema.

Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada ha objetos es una base de datos que incorpora todos los conceptos importantes de la programación orientada ha objetos:

Encapsulación: Ocultar datos del resto de los datos, impidiendo así accesos incorrectos o conflictos.

Herencia: Reusabilidad del código.

Polimorfismo: Sobrecarga de operadores o de métodos.

3.3.2 Modelos Lógicos Basados en Registros

Los modelos lógicos basados en registros se utilizan para describir datos en los modelos conceptual y de visión. Los modelos basados en registros se llaman así porque la base de datos está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos o atributos, y cada campo normalmente es de longitud fija. Los tres modelos de datos más ampliamente extendidos son el modelo relacional, el modelo en red y el modelo jerárquico

3.3.2.1 Modelo Relacional

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas su bases en 1970 por

Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que esta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que esta compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos. La información puede ser recuperada o almacenada por medio de "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relaciónales.

Las bases de datos relacionales pasan por un proceso al que se le conoce como normalización de una base de datos.

3.3.2.2 Modelo de Red

Éste es un modelo ligeramente distinto del jerárquico, en donde su diferencia fundamental es la modificación del concepto de un nodo, permitiendo que un mismo nodo tenga varios padres (algo no permitido en el modelo jerárquico).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos, pero aun así, la dificultad que significa administrar la información en una base de datos de red, ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.

Colecciones de registros y las relaciones entre datos se representan mediante enlaces dirigidos

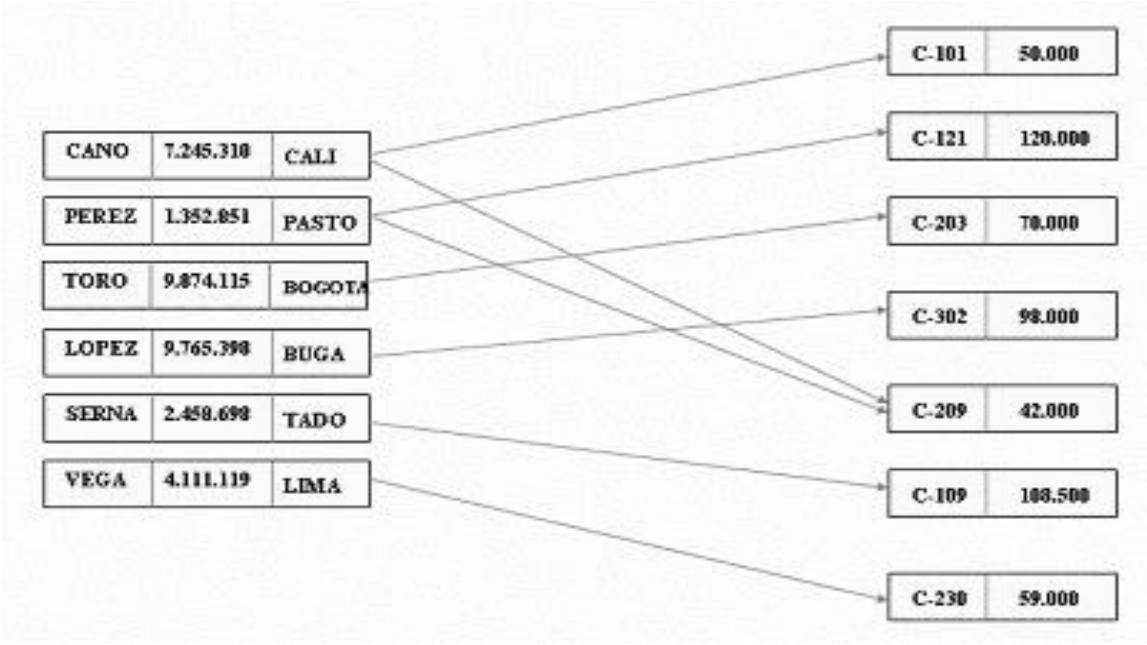


Figura 3.1 Modelo de red

3.3.2.3 Modelo Jerárquico

Estas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres se le conoce como raíz, y a los nodos que no tienen hijos se les conoce como hojas.

Una de las principales limitaciones de este modelo, es su incapacidad de representar eficientemente la redundancia de datos.

Los registros se organizan como colecciones de árboles, en lugar de grafos dirigidos

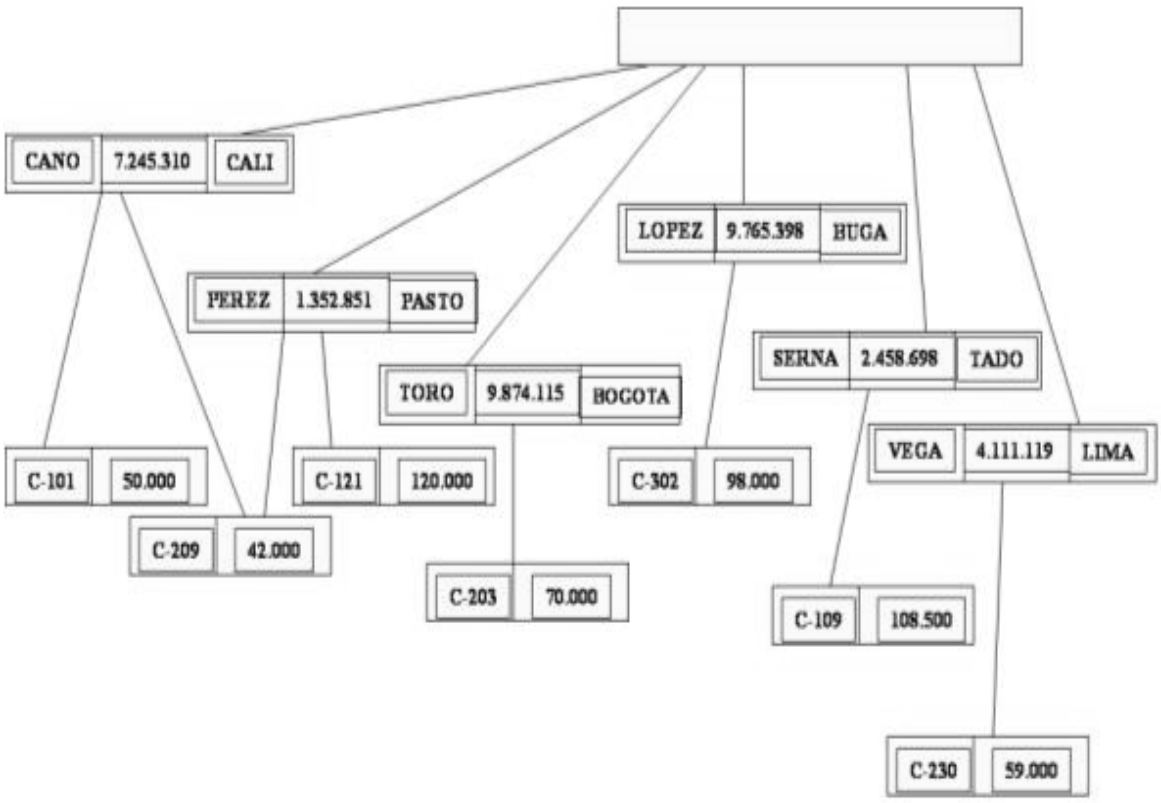


Figura 3.2 Modelo Jerárquico

3.4 Lenguajes e interfaces de bases de datos

Como los usuarios de un SGBD pueden tener distintos privilegios y distintos conocimientos informáticos, es necesario disponer de diferentes lenguajes e interfaces para cada tipo de usuarios.

3.4.1 Lenguaje de definición de datos

Una vez que se ha finalizado la tarea de diseño de la base de datos, y que se ha seleccionado un SGBD para su implementación, el primer pasó consiste en la especificación del esquema conceptual de la base de datos.

El esquema conceptual de la base de datos se especifica mediante una serie de definiciones expresadas en un Lenguaje de definición de datos (DDL, Data Definition Language). El SGBD contará con un compilador de DDL cuya función será procesar las sentencias en DDL para identificar las descripciones de los

Elementos de los esquemas y guardar la descripción del esquema en un diccionario de datos.

El diccionario de datos es un archivo que contiene metadatos, es decir, datos acerca de los datos. Este archivo se consulta cada vez que se leen o modifican los datos del Sistema de base de datos.

3.4.2 Lenguaje de manipulación de datos

Una vez que se han compilado los esquemas de la base de datos, y que ya se han introducido datos en la base de datos, los usuarios necesitarán algún mecanismo para obtener información de la base de datos. Las operaciones más comunes de manipulación son la consulta, inserción, eliminación y modificación de datos. Para ello, el SGBD ofrece un Lenguaje de manipulación de datos (DML, Data Manipulation Language).

En general existen dos tipos de DML:

- Procedímentales. Requieren que el usuario especifique qué datos desea y cómo hay que obtenerlos.
- No procedimentales. Requieren que el usuario especifique qué datos desea sin tener que especificar cómo obtenerlos.

Los DML no procedimentales suelen ser más fáciles de utilizar para los usuarios, ya que no hay que especificar la forma en que se tienen que obtener los datos, pero esta ventaja se convierte en un inconveniente, puesto que el código que se genere puede que no sea tan eficiente como el producido por los lenguajes de consulta procedimentales.

Una consulta es una sentencia que solicita información de la base de datos, y un lenguaje de consulta es el subconjunto de un DML que se utiliza para la recuperación y actualización de información de la base de datos, pero nosotros obviaremos esta diferencia.

Por último, siempre que las sentencias del lenguaje de consulta se incluyan en un lenguaje de programación de propósito general, a este lenguaje se le denomina lenguaje anfitrión.

3.5 Interfaces para Sistemas de gestión de bases de datos

Normalmente, los usuarios de un Sistema de base de datos, utilizan un lenguaje de consulta de alto nivel, mientras que los programadores utilizan el DML para la creación de consultas. Para la mayoría de los usuarios se suelen definir interfaces de usuario amigables para la interacción con la base de datos. A continuación vamos a ver los tipos de interfaces que hay.

- Interfaces basadas en menús. Presentan al usuario una lista de opciones en forma de menús que guían al usuario en la petición de consultas. De esta forma no es necesario conocer la sintaxis de un lenguaje de consulta, pues permiten la creación de la consulta eligiendo las opciones que presenta la interfaz.
- Interfaces gráficas. Suelen presentar al usuario los esquemas en forma de diagrama, y las consultas se especifican manipulando el diagrama con el ratón.
- Interfaces basadas en formularios. Estas interfaces presentan un formulario al usuario en el que se rellenan los huecos del formulario para la modificación de los datos, o bien para especificar los parámetros de la consulta.
- Interfaces de lenguaje natural. Estas interfaces aceptan la especificación de una consulta descrita en términos de un idioma concreto y construyen expresiones DML a partir de dicha especificación.
- Interfaces parametrizadas. Se trata de interfaces para usuarios que siempre suelen realizar el mismo conjunto reducido de operaciones, reduciendo el número de pulsaciones para la creación de la consulta.

3.5.1 Usuarios y administradores de la base de datos

Uno de los objetivos primordiales de un Sistema de bases de datos es el proporcionar un entorno de recuperación de información y de almacenamiento de datos en la base de datos.

Podemos hacer una clasificación de los tipos de usuarios de una base de datos en función de la forma en que interaccionan con el sistema.

- Administradores de la base de datos. Persona que tiene centralizado el control del sistema.
- Programadores de aplicaciones. Se trata de los profesionales que interactúan con el sistema a través de llamadas en DML, las cuales están incorporadas en un lenguaje anfitrión. A estos programas se les denominan programas de aplicación, como por ejemplo, los programas para la generación de cargos, abonos, transferencias de un sistema bancario. Como la sintaxis DML suele ser diferente de la sintaxis del lenguaje anfitrión, las llamadas en DML suelen ir precedidas de un carácter especial, de forma que se genere el código apropiado en el lenguaje anfitrión, lo cual se hace mediante un precompilador de DML, que convierte las sentencias DML en sentencias del lenguaje anfitrión. Una vez precompilado el programa, se compilaría mediante el compilador del lenguaje anfitrión, que generaría el código objeto apropiado.

- Usuarios sofisticados. Son los que interactúan con el sistema sin escribir programas, escribiendo las consultas en el lenguaje de consulta de la base de datos.
- Usuarios especializados. Se trata de usuarios sofisticados que crean aplicaciones de bases de datos especializadas para el procesamiento de la información.
- Usuarios ingenuos. Son los usuarios que interactúan con el sistema llamando a uno de los programas desarrollados por los programadores de aplicaciones.

Como primer tipo de usuario hemos descrito la figura del administrador, un usuario vital en el enfoque de bases de datos, y que tiene unas funciones que merecen ser estudiadas más detalladamente. Estas son:

- Definición del esquema conceptual. El esquema original de la base de datos se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de metadatos que se guardan en el diccionario de datos.
- Definición del esquema físico. Se trata de definir las estructuras de almacenamiento y los métodos de acceso adecuados (especificación de los tipos de índices)
- Modificación del esquema y de la organización física. Si bien las modificaciones tanto del esquema de la base de datos como de la organización física no son demasiado habituales, éstas se realizan modificando el esquema conceptual y físico.
- Creación de permisos para el acceso a los datos. El administrador de la base de datos es el encargado de definir los permisos que autorizan a los usuarios a acceder a ciertas partes de la base de datos.
- Especificación de las restricciones de integridad. Estas restricciones se guardan en el diccionario de datos para ser consultado cada vez que se realice una actualización.

3.5.2 Gestión de transacciones

Una transacción es un conjunto de operaciones sobre una base de datos que forman una unidad lógica de trabajo, como por ejemplo una transferencia de fondos de una cuenta a otra. Es indispensable que se produzca el cargo en la cuenta origen y el abono en la cuenta destino, o que no se produzca ninguna operación. A esto se le llama atomicidad. Otro de los requisitos es que se mantenga la consistencia de la base de datos (que la suma de las dos cuentas sea constante), y un tercer requisito es que se garanticé la durabilidad (que persistan los nuevos valores a pesar de la posibilidad de fallo del sistema).

La definición de las transacciones para que mantengan la consistencia es responsabilidad del programador de aplicaciones. El módulo de gestión de

transacciones del SGBD es responsable de garantizar la atomicidad y la durabilidad. Necesita también un mecanismo de recuperación de fallos, para restaurar la base de datos al estado anterior al inicio de una transacción interrumpida por un fallo del sistema.

Cuando hay varias transacciones que actualizan la base de datos de forma concurrente, el módulo de control de concurrencia controla la interacción entre ellas para garantizar la consistencia.

3.6 Actividades Complementarias

1. Describa los siguientes tres tipos de bases de datos modelo jerárquico, modelo de red y modelo relacional.

2. Cuales son los propósitos del lenguaje de definición de datos (DDL) y de un diccionario de datos?

3. Haga un cuadro comparativo en donde resalte las funciones y responsabilidades de los usuarios y administradores frente al manejo y administración de las bases de datos.

4. Explique cada uno de los elementos que intervienen en un modelo de bases de datos orientado a objetos.

5. ¿Qué es y para que sirve el nivel externo de una base de datos?

6. ¿Explique cuál es el nivel conceptual de un sistema de base de datos?

7. ¿Para usted que significado tiene base de datos y que importancia tiene en el ámbito empresarial?

CAPITULO 4. MODELO ENTIDAD – RELACIÓN

El modelo entidad relación, esta basado en una percepción del mundo real consistente en objetos básicos llamados entidades y en relaciones entre esos objetos.

Figura 4.1 Consultar libro

Una persona Consulta un libro ⟶ Entidad persona que se relaciona con entidad libro mediante una consulta

El aspecto semántica del modelo reposa en la representación del significado de los datos.

4.1 Conceptos básicos

Hay tres nociones básicas que emplea el modelo entidad relación conjunto de entidades, conjunto de relaciones y atributos.

4.1.12 Entidad

Una entidad es una “cosa” u “objeto” en el mundo real que es distinguible de todos los demás objetos.

Una entidad es todo elemento participante en un proceso del cual se requiere mantener o almacenar información. Es todo aquel componente de un proyecto que es importante para que su objetivo se cumpla.

Una entidad tiene un conjunto de propiedades, y los valores para algún subconjunto de propiedades pueden identificar una entidad de forma unívoca es

decir se tiene alguna identificación única y no puede existir duplicidad de identificación.

Figura 4.2 Entidades

4.1.13 Conjunto de entidades

Es un conjunto de entidades del mismo tipo que comparten las mismas propiedades o atributos (no sus valores).

Por ejemplo: El conjunto de todas las personas que son clientes de un banco.
El conjunto de todas las personas que son cliente de una tienda de videos.

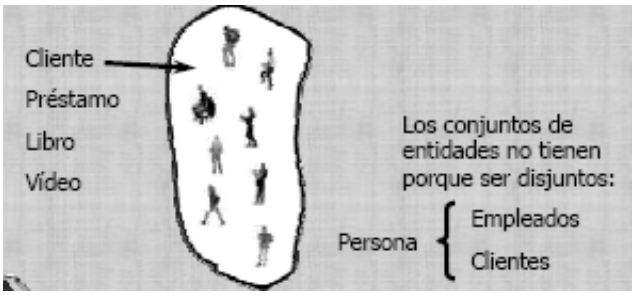


Figura 4.3 Conjunto de entidades

4.1.14 Atributos

Una entidad se representa mediante un conjunto de atributos. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades.

Un atributo es toda propiedad empleada para identificar, describir o expresar el estado de una entidad o una relación.

Cliente: identificación del cliente, nombre del cliente, dirección donde vive, ciudad donde vive el cliente,…..

Libro: Identificación del libro, autor del libro, editorial del libro, idioma en que esta escrito el libro,……

4.1.15 Valor de Atributo

Cada entidad tiene un valor para cada uno de sus atributos.

Identificador	Nombre	Dirección	Ciudad
78.523.365	Carlos	Cl. 28 No. 30-33	Acacias
52.654.289	Diana	Cl. 33 No. 20-54	Bogotá
17.254.258	Andrés	Cl. 54 No. 50-20	Medellín

Figura 4.4 Conjunto de valores de atributos

4.1.16 Dominio

El dominio (Conjunto de valores) de un atributo, es el conjunto de valores permitidos.

Todo atributo de una entidad debe pertenecer a uno y solo un tipo de dato que determina su representación física y las operaciones de manipulación aplicables sobre el. El concepto de dominio esta ligado al tipo de datos correspondiendo en principio a las nociones familiares de tipos de datos, que bien puede ser numérico, alfanuméricos y de fecha.

Autor del libro: cadenas de caracteres de una cierta longitud.
Año de nacimiento del cliente: números de cuatro cifras.

4.1.17 Los atributos compuestos

Son aquellos que se pueden dividir en atributos simples.

Nombre de cliente: Nombre propio del cliente, primer apellido del cliente, segundo apellido del cliente

Dirección del cliente: Calle del cliente, número de carrera del cliente, casa del cliente.

Los atributos compuestos hacen más claro el modelo agrupando atributos relacionados.

4.1.7 Un atributo multivalorado

Es aquel que puede tener un conjunto de valores para una entidad específica.

Teléfono de cliente: Algunos clientes no tienen teléfono y otros pueden tener varios (fijo casa, fijo oficina, móvil,…).

4.1.8 Los atributos derivados

Son aquellos cuyo valor se puede derivar del valor de otros atributos o entidades.

Edad de cliente: Año actual menos año de nacimiento
Número de libros prestados: Contar el número de préstamos activos

4.1.9 Un atributo toma valor nulo

Cuando una entidad no tiene valor para un atributo.

No Aplicable (segundo nombre)
Perdido (existe pero no se tiene)
Desconocido (no se conoce si existe o no)

Se debe tener mucho cuidado con los valores nulos y con su tratamiento

Ejemplo 1: Banco

Conjuntos de Entidades

Cliente (nombre_cliente, apellido1_cliente, apellido2_cliente, direccion_cliente, codigopostal_cliente, ciudad_cliente)
Sucursal (nombre_sucursal, direccion_sucursal, ciudad_sucursal, activos)
Cuenta (numero_cuenta, saldo)
Prestamo (numero_prestamo, cantidad_inicial, saldo)

Ejemplo 2: Biblioteca

Conjuntos de Entidades

Usuario (nombre, apellido1, apellido2, direccion, codigopostal, ciudad)
Libro (titulo, autor, editorial, año)

Ejemplo 3: Videoclub

Conjuntos de Entidades

Socio (nombre_socio, apellidos_socio, direccion_socio, telefono_socio, ciudad_socio, fechaalta)

Pelicula (titulo, género, duracion, clasificación, año, país, precioalquiler)

4.1.10 Una relación

Es una asociación entre diferentes entidades

Préstamo de un libro

Pedir un préstamo

Figura 4.4 Relaciones

4.1.11 Conjunto de Relaciones

Un conjunto de relaciones es un grupo de relaciones del mismo tipo. Se dice que las entidades participan en la relación.

Formalmente: Relación matemática, con n>=2, de n conjunto de entidades Sean E1, E2,…, En conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de (e1, e2, …, en) | e1 € E1, e2 € E2, …, en€En

Conjunto de Relaciones

La concesión de un préstamo a un cliente
La pertenencia de un préstamo a una sucursal
La apertura de una cuenta por un cliente en una sucursal
El préstamo de un libro a un usuario de la biblioteca
El alquiler de una película a un socio en el videoclub

El papel de una entidad en una relación especifica la función que desempeña esa entidad en esa relación. En la mayor parte de los casos, cuando las entidades son disjuntas, los papeles están implícitos Son útiles cuando la relación necesita aclaración de relaciones recursivas
Relación “trabaja para” que especifica quién es el jefe de quién en el banco (entidades: empleado y empleado)

Una relación puede tener atributos descriptivos, que describan aspectos propios de la relación (no pertenecen a las entidades implicadas). Para describir el ingreso

en cuenta, por parte del cliente, de una cantidad: fecha de imposición, persona que realiza la imposición, importe, …

Una relación debe estar identificada unívocamente a partir de sus entidades participantes, sin usar atributos descriptivos.

Un conjunto de relaciones binario es un conjunto de relaciones que implica dos conjuntos de entidades. La mayoría de los conjuntos de relaciones en un sistema de bases de datos son binarios (grado 2) La relación “trabaja de en” entre los conjuntos de entidades empleado, sucursal y puesto es una relación ternaria (grado 3)

Ejemplo 1: Banco

Conjuntos de Relaciones
Impositor (Cliente, Cuenta)
Prestatario (Cliente, Prestamo)
SucursalCuenta (Sucursal, Cuenta)
SucursalPrestamo (Sucursal, Prestamo)
Cliente (nombre_cliente, apellido1_cliente, apellido2_cliente, direccion_cliente, codigopostal_cliente, ciudad_cliente)
Sucursal (nombre_sucursal, direccion_sucursal, ciudad_sucursal, activos)
Cuenta (numero_cuenta, saldo)
Prestamo (numero _ préstamo, importe _ inicial, resto)

Ejemplo 2: Biblioteca

Conjuntos de Relaciones
Préstamo (Usuario, Libro)
Usuario (nombre, apellido1, apellido2, dirección, codigopostal, ciudad)
Libro (titulo, autor, editorial, año)

Ejemplo 3: Videoclub

Conjuntos de Relaciones
Alquiler (Socio, Película)
Socio (nombre_socio, apellidos_socio, dirección _ socio, telefono_socio, ciudad_socio, fechaalta)
Película (titulo, género, duración, clasificación, año, país, precioalquiler)

4.2 Restricciones

Las restricciones expresan limitaciones a las que se deben adaptar los contenidos de la base de datos.

- Correspondencia de cardinalidades

- Restricciones de participación

4.2.1 La correspondencia de cardinalidades

La razón de cardinalidad, expresa el número de entidades de un conjunto de entidades a las que una entidad de otro conjunto de entidades puede estar asociada vía un conjunto de relaciones.

Es decir Indica el número de entidades del conjunto de entidades E2 que se relacionan con una entidad del conjunto de entidades E1 y viceversa.

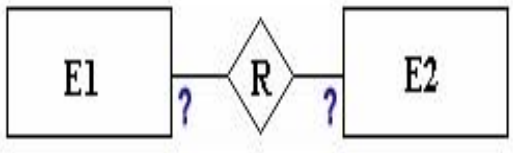


Figura 4.5 Cardinalidad

¿Cuántas cuentas puede tener un cliente?
¿Puede un usuario llevarse prestados varios libros? conjunto de relaciones binarias R entre los conjuntos de entidades A y B , E1 y E2.

4.2.2 Tipos de Cardinalidades

- Uno a uno. 1:1. Una entidad del conjunto de entidades E1 se relaciona con una única entidad del conjunto de entidades E2 y viceversa.

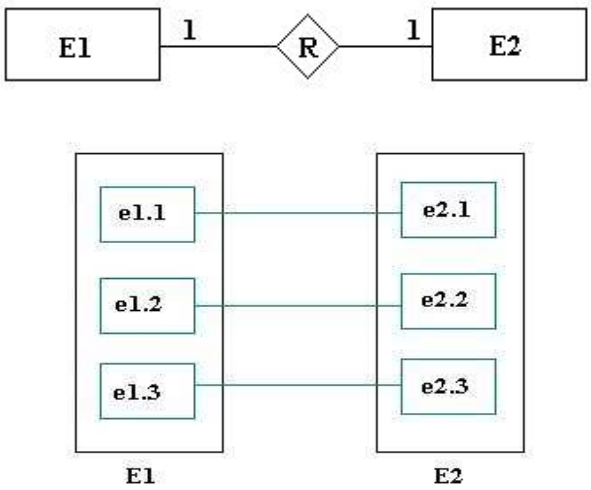


Figura 4.6 Relación uno a uno 1:1

- Uno a muchos. 1:n. Una entidad del conjunto de entidades E1 se relaciona con muchas entidades del conjunto de entidades E2 y una entidad del conjunto de entidades E2 solo puede estar relacionada con una entidad del conjunto de entidades E1.

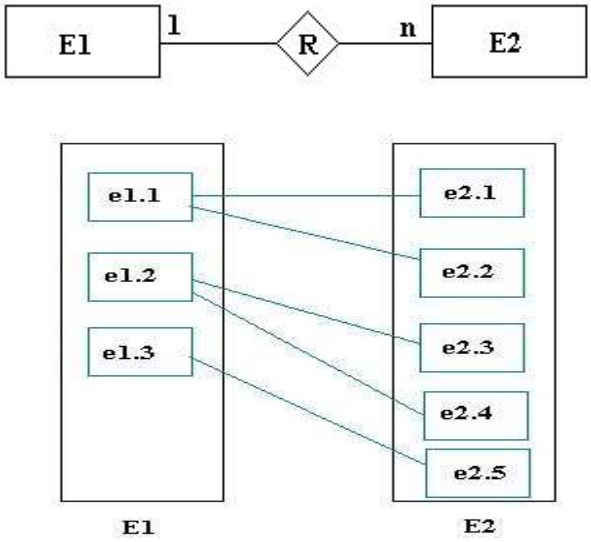


Figura 4.7 Relación uno a muchos 1:n

Muchos a uno. n:1. Una entidad en E1 esta asociada con una única entidad del conjunto de entidades E2 y una entidad del conjunto de entidades en E2 esta relacionada con muchas entidades del conjunto de entidades E1

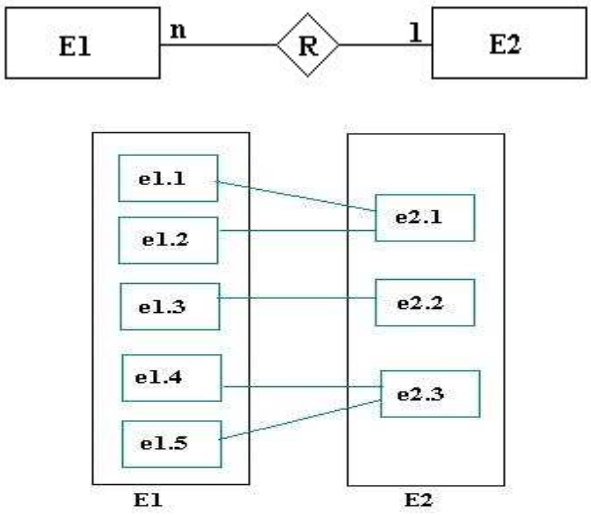


Figura 4.8 Relación muchos a uno n:1

Muchos a muchos. n:n. Una entidad del conjunto de entidades E1 esta relacionada con muchas entidades del conjunto de entidades E2 y viceversa.

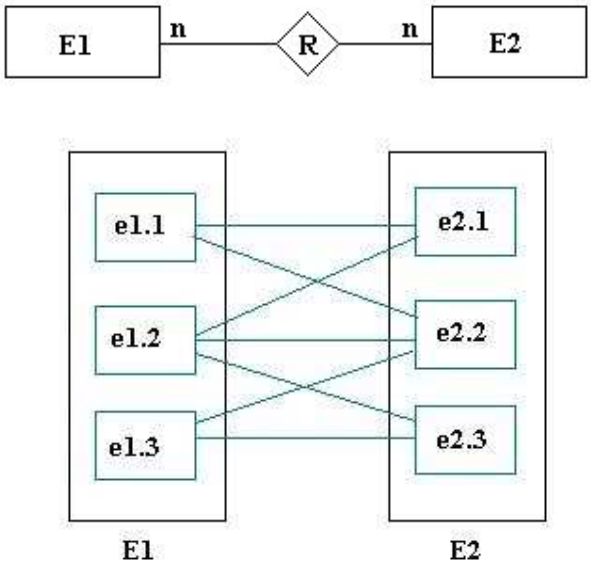


Figura 4.9 Relación muchos a muchos n:n

4.2.2.1Correspondencia de Cardinalidades Adecuada

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende, obviamente, de la situación del mundo real que modela el conjunto de relaciones.

Impositor (Cliente, Cuenta)
Cliente (nombre_cliente, apellido1_cliente, apellido2_cliente, direccion_cliente, codigopostal_cliente, ciudad_cliente)
Cuenta (numero_cuenta, saldo)
N-N N-1 1-N 1-1 ?

4.2.2.1 Participación

La participación de un conjunto de entidades en un conjunto de relaciones se dice que es total si cada entidad participa al menos en una relación Si sólo participan algunas entidades se dice que la participación del conjunto de entidades en el conjunto de relaciones es parcial.

Ejemplo 1: Banco
Conjuntos de Relaciones

Impositor (Cliente, Cuenta) = N-N
Prestatario (Cliente, Prestamo) = N-N
SucursalCuenta (Sucursal, Cuenta) = 1-N
SucursalPrestamo (Sucursal, Prestamo) = 1-N
Cliente (nombre_cliente, apellido1_cliente, apellido2_cliente, direccion_cliente, codigopostal_cliente, ciudad_cliente)
Sucursal (nombre_sucursal, dirección _ sucursal, ciudad_sucursal, activos)
Cuenta (numero_cuenta, saldo)
Prestamo (numero _ préstamo, importe _ inicial, resto)

Ejemplo 2: Biblioteca

Conjuntos de Relaciones
Préstamo (Usuario, Libro) = 1-N
Usuario (nombre, apellido1, apellido2, dirección, codigopostal, ciudad)
Libro (titulo, autor, editorial, año)

Ejemplo 3: Videoclub

Conjuntos de Relaciones
Alquiler (Socio, Película) = 1-N
Socio (nombre _ socio, apellidos_socio, dirección _ socio, telefono_socio, ciudad_socio, fechaalta)
Pelicula (titulo, género, duración, clasificación, año, país, precioalquiler)

4.2.3 Restricciones de integridad

Para completar los aspectos del mundo real representados en un modelo, necesitamos poder expresar restricciones de los datos que van más allá de las limitaciones impuestas por las definiciones de entidades, atributos y relaciones. Estas restricciones pueden ser:

- Restricciones de clave. Las claves son atributos o conjuntos de atributos que identifican una entidad dentro de su conjunto.
- Restricciones de valor único. Exigen que a un valor de un atributo o conjunto de atributos se le asocie un valor único de otros atributos.
- Restricciones de integridad referencial. Exigen que un valor referenciado por alguna entidad exista realmente en la base de datos.
- Restricciones de dominio. Exigen que el valor de un atributo pertenezca a un conjunto específico de valores.
- Restricciones generales. Son condiciones arbitrarias que deben cumplirse en la base de datos.

4.5 Claves

Es necesario disponer de una forma de especificar cómo, las entidades dentro de un conjunto de entidades dado y las relaciones dentro de un conjunto de relaciones dado, son distinguibles Una entidad es una “cosa” u “objeto” en el mundo real que es distinguible de todos los demás objetos Una clave es un conjunto suficiente de atributos capaces de distinguir las entidades (relaciones) de un conjunto de entidades (conjunto de relaciones) entre si (no sólo conceptualmente sino desde una perspectiva de bases de datos)

4.5.1 Superclave

Una superclave es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en un conjunto de entidades Si un conjunto de atributos es una superclave, entonces también lo es cualquier superconjunto de ese conjunto inicial de atributos.

nombre_cliente
nombre_cliente, apellido1_cliente, apellido2_cliente
nombre_cliente, apellido1_cliente, apellido2_cliente , codigopostal_cliente

4.5.2 Claves Candidatas

Una clave candidata es una superclave tal que un subconjunto de ella no es, también, una superclave.

nombre_cliente, codigopostal_cliente
apellido1_cliente, apellido2_cliente

4.5.3 Clave Primaria

Una clave primaria es una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Cualesquiera dos entidades individuales en el conjunto de entidades no pueden tener, al mismo tiempo, el mismo valor en sus atributos clave.

La elección de una clave representa una restricción en el desarrollo del mundo real que se modela. La clave primaria se debe elegir de forma que los atributos que la forman nunca, o al menos muy raramente, cambien.

4.4 Diagrama Entidad – Relación

El diagrama Entidad-Relación permite expresar gráficamente la estructura lógica general de una base de datos (su simpleza y claridad son las responsables de su uso generalizado)

Existen diversas maneras de representar la cardinalidad, la más sencilla es la presentada anteriormente, donde solo se especifica el número máximo de relaciones que puede tener una entidad con entidades del otro conjunto de entidades con que se asocia.

Korth distingue en el diagrama la cardinalidad a partir de líneas dirigidas o no. La línea dirigida indica que la relación es a uno. Por ejemplo una relación de 1: n se graficaría así:

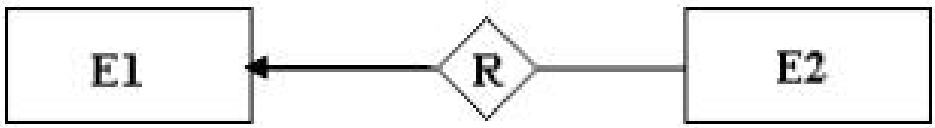
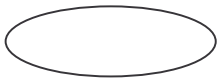
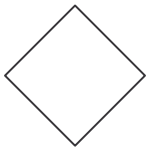


Figura 4.10 Representación de la cardinalidad

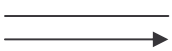
Rectángulos: Representan **conjuntos de entidades**



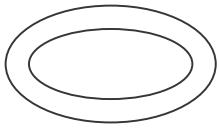
Elipses: Representan **atributos**



Rombos: Representan **conjuntos de relaciones**

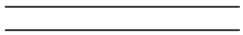


Líneas: Unen atributos a conjuntos de entidades o a conjuntos de relaciones y conjuntos de entidades a conjuntos de relaciones (uno; varios)

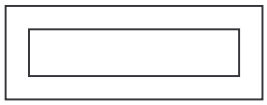


Elipses Dobles: Representan **atributos multivalorados.**

Elipses Discontinuas: Representan **atributos derivados.**



Líneas Dobles: Indican **participación total** de un conjunto de entidades en un conjuntos de relaciones.



Rectángulos Dobles: Representan **conjuntos de entidades débiles**

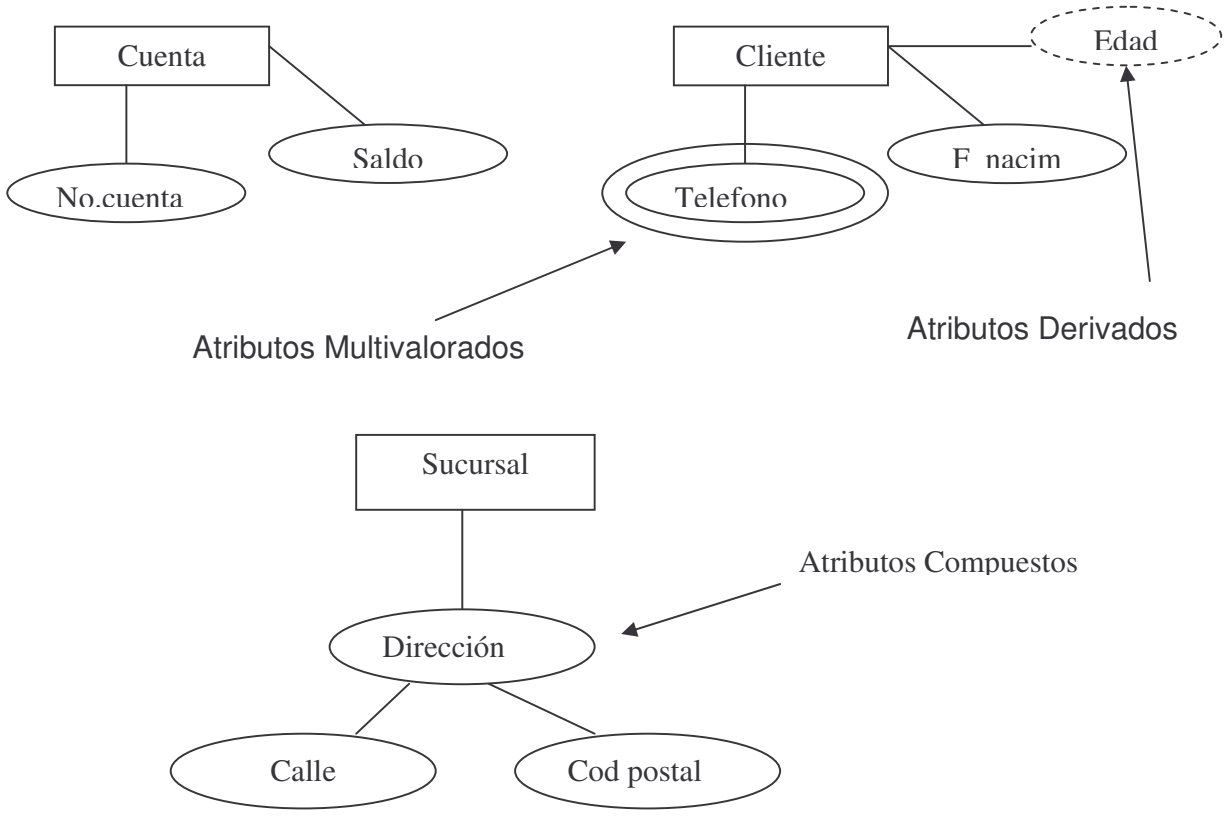
En los modelos conceptuales es suficiente con los valores máximos, pero para el análisis detallado se requieren de cuatro (4) puntos para expresar las reglas del negocio que se necesitan hacer cumplir en la estructura de la base de datos. En estos casos la cardinalidad de las relaciones se expresa con un valor mínimo y un máximo y se declara gráficamente en el diagrama entidad relación, dado que las relaciones se indican en ambos direcciones entre las entidades, la cardinalidad máxima y mínima debe indicarse igualmente

Como ejemplo para el desarrollo de un diagrama entidad relación podemos tener las siguientes entidades.

Cuenta	Sucursal	Cliente	Préstamo

Y con ellas podemos establecer los siguientes atributos

Figura 4.11 Representación de atributos



Ejemplo Banco: A Continuación se realiza el diagrama de cada una de las entidades y sus atributos para el manejo de cuentas y prestamos en un Banco.

Figura 4.12 Ejemplo Banco Diagrama entidades y atributos

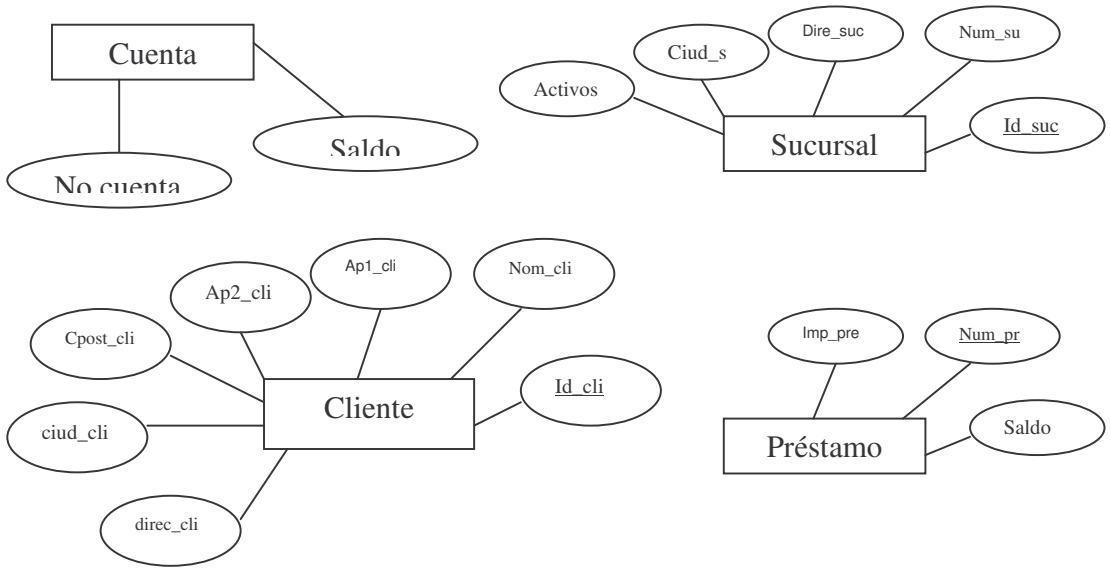


Figura 4.13 Relaciones

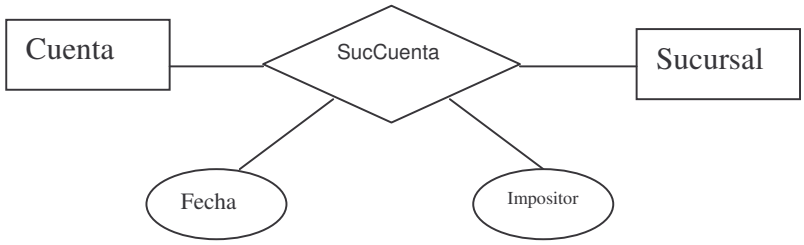
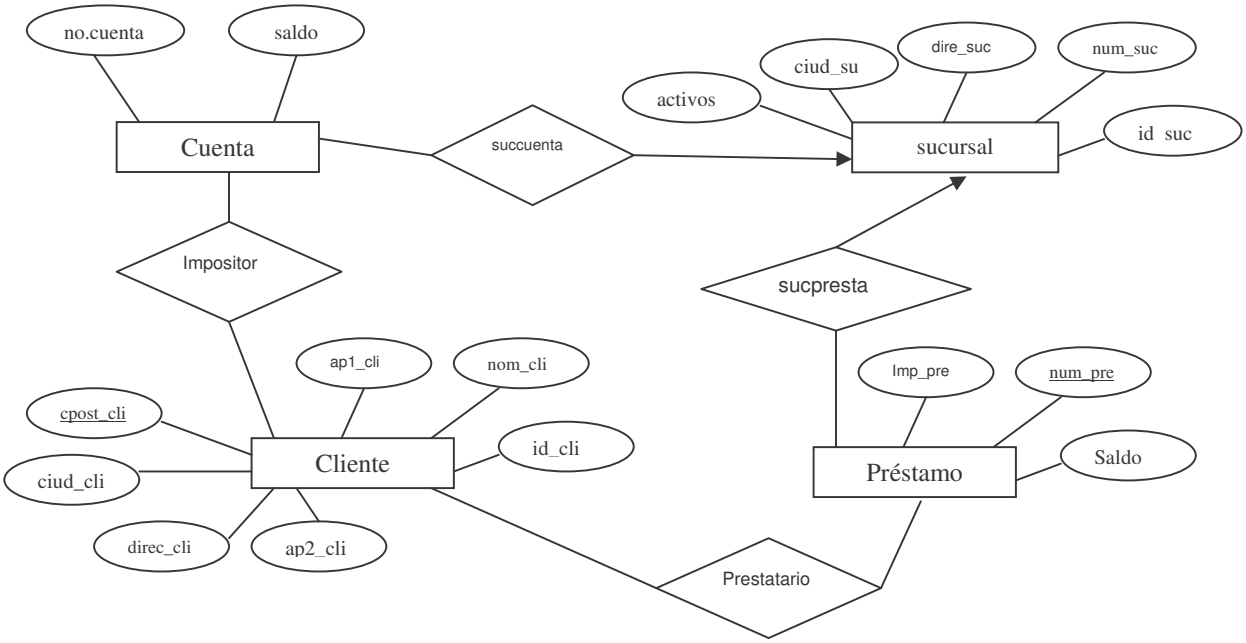


Diagrama E-R con atributos unidos a un conjunto de relaciones que hacen referencia al manejo de un Banco.

Figura 4.13 Diagrama E-R Banco



Para distinguir entre los diferentes tipos de relación que se generan es necesario se utilizan los siguientes tipos de líneas:

_____ Línea no dirigida

——————→ Línea dirigida

Estas se ubican entre el conjunto de relaciones y el conjunto de entidades involucradas.

Podemos observar que se generan diferentes tipos de relaciones entre todas las entidades que pueden ser varios a varios, uno a varios, varios a uno, uno a uno.

En el anterior ejemplo encontramos las siguientes relaciones identificadas de la siguiente forma:

- Succuenta: la relacione que se establece entre la entidad sucursal y la entidad cuenta es de varios a uno, ya que los une una línea dirigida entre las entidades sucursal y cuenta.
- Impositor: la relación que se establece entre la entidad cliente y la entidad cuenta es de varias a varios, ya que los une una línea no dirigida entre las entidades.
- Sucpres: la relacione que se establece entre la entidad sucursal y la entidad préstamo es de varios a uno, ya que los une una línea dirigida entre las entidades sucursal y cuenta.
- Prestatario: la relación que se establece entre la entidad cliente y la entidad préstamo es de varias a varios, ya que los une una línea no dirigida entre las entidades.

Los atributos que son claves deben aparecer subrayados en el diagrama.

4.4.1 Conjunto de entidades Débiles y Fuertes

Es aquel conjunto de entidades que no tiene atributos que puedan identificar una entidad en forma única, o sea que no poseen atributos para conformar la llave primaria; por lo tanto dependen de una entidad fuerte.

Gráficamente se representa así:

Figura 4.14 Representación de Entidades Débiles

Conjunto de entidades Fuerte. Conjunto de entidades que posee una clave primaria.

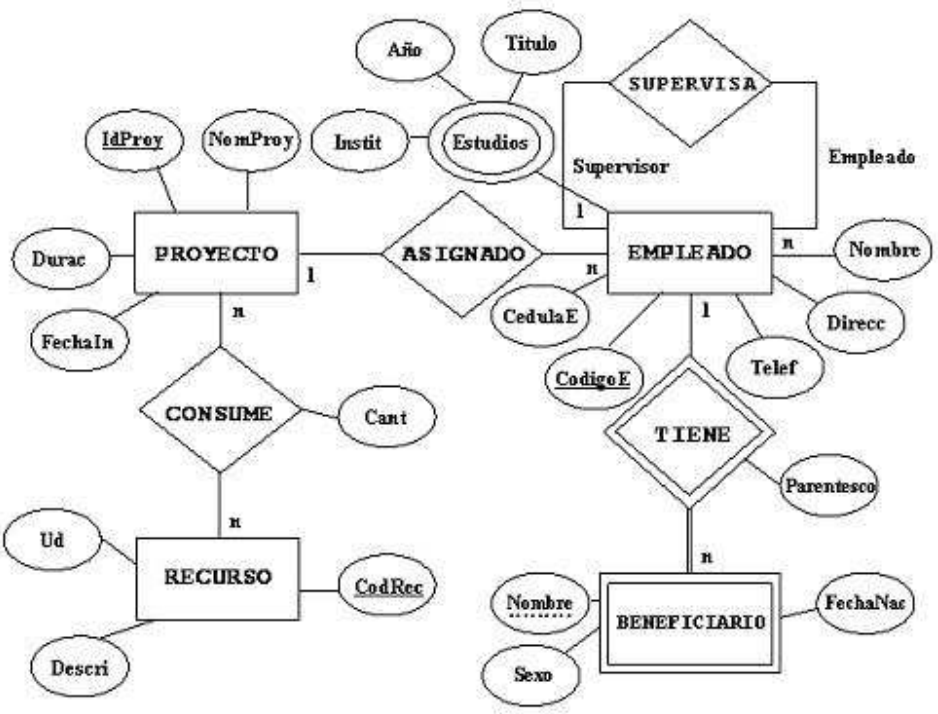
Las entidades débiles no pueden ser conocidas por sí solas; con el objeto de diferenciarlas se seleccionan algunos de sus atributos para formar un discriminador. Este discriminador se asocia con las llaves primarias de las

entidades fuertes a las que se encuentre subordinada para formar así su llave primaria propia. Los conjuntos de relaciones también tienen llaves primarias. Estas se conforman por las llaves primarias de los conjuntos de entidades que se asocian en la relación y todos los atributos descriptivos de la relación.

Se tiene una empresa desarrollando varios proyectos, a los que son asignados varios empleados, pero cada empleado solo esta vinculado a un proyecto, en un momento dado. Cada proyecto consume diferentes recursos en cantidades determinadas: los empleados están a cargo de un supervisor, que es un empleado también. Los empleados pueden tener personas beneficiarias (hijos, esposas, padres, etc.).

El diagrama entidad relación correspondiente sería:

Figura 4.13 Diagrama E-R Ejemplo Proyectos



En la gráfica se aprecia la forma en que se representan las entidades, las relaciones, los atributos y la cardinalidad de las relaciones.

Como ejercicio clasificar e identificar las entidades y los atributos. Expresar con sus palabras las relaciones e identificar la cardinalidad, expresando su significado.

4.5 Metodología de diseño conceptual

El primer paso en el diseño de una base de datos es la producción del esquema conceptual. Normalmente, se construyen varios esquemas conceptuales, cada uno para representar las distintas visiones que los usuarios tienen de la información. Cada una de estas visiones suelen corresponder a las diferentes áreas funcionales de la empresa como, por ejemplo, producción, ventas, recursos humanos, etc.

Estas visiones de la información, denominadas vistas, se pueden identificar de varias formas. Una opción consiste en examinar los diagramas de flujo de datos, que se pueden haber producido previamente, para identificar cada una de las áreas funcionales. La otra opción consiste en entrevistar a los usuarios, examinar los procedimientos, los informes y los formularios, y también observar el funcionamiento de la empresa.

A los esquemas conceptuales correspondientes a cada vista de usuario se les denomina esquemas conceptuales locales. Cada uno de estos esquemas se compone de entidades, relaciones, atributos, dominios de atributos e identificadores. El esquema conceptual también tendrá una documentación, que se irá produciendo durante su desarrollo. Las tareas a realizar en el diseño conceptual son las siguientes:

1. Identificar las entidades.

2. Identificar las relaciones.

3. Identificar los atributos y asociarlos a entidades y relaciones.

4. Determinar los dominios de los atributos.

5. Determinar los identificadores.

6. Determinar las jerarquías de generalización (si las hay).

7. Dibujar el diagrama entidad-relación.

8. Revisar el esquema conceptual local con el usuario.

4.5.1. Identificar las entidades

En primer lugar hay que definir los principales objetos que interesan al usuario. Estos objetos serán las entidades. Una forma de identificar las entidades es examinar las especificaciones de requisitos de usuario. En estas especificaciones se buscan los nombres o los sintagmas nominales que se mencionan (por

ejemplo: número de empleado, nombre de empleado, número de inmueble, dirección del inmueble, alquiler, número de habitaciones). También se buscan objetos importantes como personas, lugares o conceptos de interés, excluyendo aquellos nombres que sólo son propiedades de otros objetos. Por ejemplo, se pueden agrupar el número de empleado y el nombre de empleado en una entidad denominada empleado, y agrupar número de inmueble, dirección del inmueble, alquiler y número de habitaciones en otra entidad denominada inmueble.

Otra forma de identificar las entidades es buscar aquellos objetos que existen por sí mismos. Por ejemplo, empleado es una entidad porque los empleados existen, sepamos o no sus nombres, direcciones y teléfonos. Siempre que sea posible, el usuario debe colaborar en la identificación de las entidades.

A veces, es difícil identificar las entidades por la forma en que aparecen en las especificaciones de requisitos. Los usuarios, a veces, hablan utilizando ejemplos o analogías. En lugar de hablar de empleados en general, hablan de personas concretas, o bien, hablan de los puestos que ocupan esas personas.

Para liarlo aún más, los usuarios usan, muchas veces, sinónimos y homónimos. Dos palabras son sinónimos cuando tienen el mismo significado. Los homónimos ocurren cuando la misma palabra puede tener distintos significados dependiendo del contexto.

No siempre es obvio saber si un objeto es una entidad, una relación o un atributo. Por ejemplo ¿cómo se podría clasificar matrimonio? Pues de cualquiera de las tres formas. El análisis es subjetivo, por lo que distintos diseñadores pueden hacer distintas interpretaciones, aunque todas igualmente válidas. Todo depende de la opinión y la experiencia de cada uno. Los diseñadores de bases de datos deben tener una visión selectiva y clasificar las cosas que observan dentro del contexto de la empresa u organización. A partir de unas especificaciones de usuario es posible que no se pueda deducir un conjunto único de entidades, pero después de varias iteraciones del proceso de análisis, se llegará a obtener un conjunto de entidades que sean adecuadas para el sistema que se ha de construir.

Conforme se van identificando las entidades, se les dan nombres que tengan un significado y que sean obvias para el usuario. Los nombres de las entidades y sus descripciones se anotan en el diccionario de datos. Cuando sea posible, se debe anotar también el número aproximado de ocurrencias de cada entidad. Si una entidad se conoce por varios nombres, éstos se deben anotar en el diccionario de datos como alias o sinónimos.

4.5.2. Identificar las relaciones

Una vez definidas las entidades, se deben definir las relaciones existentes entre ellas. Del mismo modo que para identificar las entidades se buscaban nombres en las especificaciones de requisitos, para identificar las relaciones se suelen buscar las expresiones verbales (por ejemplo: oficina tiene empleados, empleado

gestiona inmueble, cliente visita inmueble). Si las especificaciones de requisitos reflejan estas relaciones es porque son importantes para la empresa y, por lo tanto, se deben reflejar en el esquema conceptual.

Pero sólo interesan las relaciones que son necesarias. En el ejemplo anterior, se han identificado las relaciones empleado gestiona inmueble y cliente visita inmueble. Se podría pensar en incluir una relación entre empleado y cliente: empleado atiende a cliente, pero observando las especificaciones de requisitos no parece que haya interés en modelar tal relación.

La mayoría de las relaciones son binarias (entre dos entidades), pero no hay que olvidar que también puede haber relaciones en las que participen más de dos entidades, así como relaciones recursivas.

Es muy importante repasar las especificaciones para comprobar que todas las relaciones, explícitas o implícitas, se han encontrado. Si se tienen pocas entidades, se puede comprobar por parejas si hay alguna relación entre ellas. De todos modos, las relaciones que no se identifican ahora se suelen encontrar cuando se valida el esquema con las transacciones que debe soportar.

Una vez identificadas todas las relaciones, hay que determinar la cardinalidad mínima y máxima con la que participa cada entidad en cada una de ellas. De este modo, el esquema representa de un modo más explícito la semántica de las relaciones. La cardinalidad es un tipo de restricción que se utiliza para comprobar y mantener la calidad de los datos. Estas restricciones son aserciones sobre las entidades que se pueden aplicar cuando se actualiza la base de datos para determinar si las actualizaciones violan o no las reglas establecidas sobre la semántica de los datos.

Conforme se van identificando las relaciones, se les van asignando nombres que tengan significado para el usuario. En el diccionario de datos se anotan los nombres de las relaciones, su descripción y las cardinalidades con las que participan las entidades en ellas.

4.5.3. Identificar los atributos y asociarlos a entidades y relaciones

Al igual que con las entidades, se buscan nombres en las especificaciones de requisitos. Son atributos los nombres que identifican propiedades, cualidades, identificadores o características de entidades o relaciones.

Lo más sencillo es preguntarse, para cada entidad y cada relación, ¿qué información se quiere saber de ..? La respuesta a esta pregunta se debe encontrar en las especificaciones de requisitos. Pero, en ocasiones, será necesario preguntar a los usuarios para que aclaren los requisitos. Desgraciadamente, los usuarios pueden dar respuestas a esta pregunta que también contengan otros conceptos, por lo que hay que considerar sus respuestas con mucho cuidado.

Al identificar los atributos, hay que tener en cuenta si son simples o compuestos. Por ejemplo, el atributo dirección puede ser simple, teniendo la dirección completa como un solo valor: `Cl. 40 No 24-35, Buque´; o puede ser un atributo compuesto, formado por la calle (`40´), el número (`25-35´) y el barrio (`Buque´). El escoger entre atributo simple o compuesto depende de los requisitos del usuario. Si el usuario no necesita acceder a cada uno de los componentes de la dirección por separado, se puede representar como un atributo simple. Pero si el usuario quiere acceder a los componentes de forma individual, entonces se debe representar como un atributo compuesto.

También se deben identificar los atributos derivados o calculados, que son aquellos cuyo valor se puede calcular a partir de los valores de otros atributos. Por ejemplo, el número de empleados de cada oficina, la edad de los empleados o el número de inmuebles que gestiona cada empleado. Algunos diseñadores no representan los atributos derivados en los esquemas conceptuales. Si se hace, se debe indicar claramente que el atributo es derivado y a partir de qué atributos se obtiene su valor. Donde hay que considerar los atributos derivados es en el diseño físico.

Cuando se están identificando los atributos, se puede descubrir alguna entidad que no se ha identificado previamente, por lo que hay que volver al principio introduciendo esta entidad y viendo si se relaciona con otras entidades.

Es muy útil elaborar una lista de atributos e ir eliminándolos de la lista conforme se vayan asociando a una entidad o relación. De este modo, uno se puede asegurar de que cada atributo se asocia a una sola entidad o relación, y que cuando la lista se ha acabado, se han asociado todos los atributos.

Hay que tener mucho cuidado cuando parece que un mismo atributo se debe asociar a varias entidades. Esto puede ser por una de las siguientes causas:

Se han identificado varias entidades, como director, supervisor y administrativo, cuando, de hecho, pueden representarse como una sola entidad denominada empleado. En este caso, se puede escoger entre introducir una jerarquía de generalización, o dejar las entidades que representan cada uno de los puestos de empleado.

Se ha identificado una relación entre entidades. En este caso, se debe asociar el atributo a una sola de las entidades y hay que asegurarse de que la relación ya se había identificado previamente. Si no es así, se debe actualizar la documentación para recoger la nueva relación.

Conforme se van identificando los atributos, se les asignan nombres que tengan significado para el usuario. De cada atributo se debe anotar la siguiente información:

- Nombre y descripción del atributo.
- Alias o sinónimos por los que se conoce al atributo.
- Tipo de dato y longitud.
- Valores por defecto del atributo (si se especifican).
- Si el atributo siempre va a tener un valor (si admite o no nulos).
- Si el atributo es compuesto y, en su caso, qué atributos simples lo forman.
- Si el atributo es derivado y, en su caso, cómo se calcula su valor.
- Si el atributo es multievaluado.

4.5.4. Determinar los dominios de los atributos

El dominio de un atributo es el conjunto de valores que puede tomar el atributo. Por ejemplo el dominio de los números de oficina son las tiras de hasta tres caracteres en donde el primero es una letra y el siguiente o los dos siguientes son dígitos en el rango de 1 a 99; el dominio de los números de teléfono y los números de fax son los 9 dígitos.

Un esquema conceptual está completo si incluye los dominios de cada atributo: los valores permitidos para cada atributo, su tamaño y su formato. También se puede incluir información adicional sobre los dominios como, por ejemplo, las operaciones que se pueden realizar sobre cada atributo, qué atributos pueden compararse entre sí o qué atributos pueden combinarse con otros. Aunque sería muy interesante que el sistema final respetara todas estas indicaciones sobre los dominios, esto es todavía una línea abierta de investigación.

Toda la información sobre los dominios se debe anotar también en el diccionario de datos.

4.5.5. Determinar los identificadores

Cada entidad tiene al menos un identificador. En este paso, se trata de encontrar todos los identificadores de cada una de las entidades. Los identificadores pueden ser simples o compuestos. De cada entidad se escogerá uno de los identificadores como clave primaria en la fase del diseño lógico.

Cuando se determinan los identificadores es fácil darse cuenta de si una entidad es fuerte o débil. Si una entidad tiene al menos un identificador, es fuerte (otras denominaciones son padre, propietaria o dominante). Si una entidad no tiene

atributos que le sirvan de identificador, es débil (otras denominaciones son hijo, dependiente o subordinada).

Todos los identificadores de las entidades se deben anotar en el diccionario de datos.

4.5.6. Determinar las jerarquías de generalización

En este paso hay que observar las entidades que se han identificado hasta el momento. Hay que ver si es necesario reflejar las diferencias entre distintas ocurrencias de una entidad, con lo que surgirán nuevas subentidades de esta entidad genérica; o bien, si hay entidades que tienen características en común y que realmente son subentidades de una nueva entidad genérica.

En cada jerarquía hay que determinar si es total o parcial y exclusiva o superpuesta.

4.5.7. Dibujar el diagrama entidad-relación

Una vez identificados todos los conceptos, se puede dibujar el diagrama entidad-relación correspondiente a una de las vistas de los usuarios. Se obtiene así un esquema conceptual local.

4.5.8. Revisar el esquema conceptual local con el usuario

Antes de dar por finalizada la fase del diseño conceptual, se debe revisar el esquema conceptual local con el usuario. Este esquema está formado por el diagrama entidad-relación y toda la documentación que describe el esquema. Si se encuentra alguna anomalía, hay que corregirla haciendo los cambios oportunos, por lo que posiblemente haya que repetir alguno de los pasos anteriores. Este proceso debe repetirse hasta que se esté seguro de que el esquema conceptual es una fiel representación de la parte de la empresa que se está tratando de modelar.

4.6. Reducción del Diagrama Entidad-Relación a tablas

Una base de datos que se represente mediante un diagrama E-R puede representarse mediante un conjunto de tablas. De forma general suele existir una tabla para cada conjunto de entidades y una tabla para cada conjunto de relaciones, asignándole a cada tabla el nombre del conjunto de entidades o del conjunto de relaciones correspondiente, aunque esto es sólo de forma general, por lo que habrá que ver las particularidades de cada caso. Cada tabla consta de una serie de columnas con un nombre único.

Todo conjunto de entidades luego del mapeo se convierte en una tabla. Los atributos del conjunto de entidades serán los campos de la tabla y las entidades del conjunto de entidades serán las tuplas o registros.

Para el ejercicio que se viene desarrollando se tiene el siguiente modelo relacional, expresado a través de tablas:

Proyecto	Empleado	Recurso	Consumo	Beneficiario
*IdProy	*CodigoE	*CodRec	*IdProy	*CodigoE
NomProy	CedulaE	Descri	*CodRec	*Nombre
Durac	Nombre	ud	cant	parentesco
FechaInic	Direcc			fechaNac
	Telef			sexo
	Estudios			
	CodigoSu			

Figura 4.14 Modelo Relacional expresado en tablas

Se aprecia que el atributo estudios en la tabla empleado es compuesto (contiene otros atributos), por lo tanto debe llevarse a la tabla los atributos últimos:

Empleado
*CodigoE
CedulaE
Nombre
Direcc
Telef
Titulo
Instit
año
CodigoSu

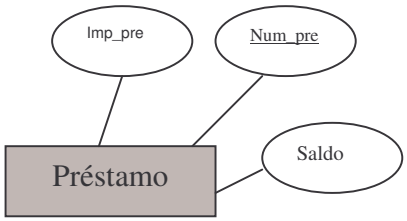
Figura 4.15 Atributos compuestos

Existen reglas bien definidas para la conversión de los elementos de un diagrama E-R a tablas:

4.6.1 Entidades Fuertes

Sea E un conjunto de entidades fuertes con los atributos descriptivos a1, a2, …, an Se representa mediante una tabla llamada E con n columnas distintas, cada una de las cuales correspondientes a cada uno de los atributos de E

Se crea una tabla con una columna para cada atributo del conjunto de entidades.



Préstamo

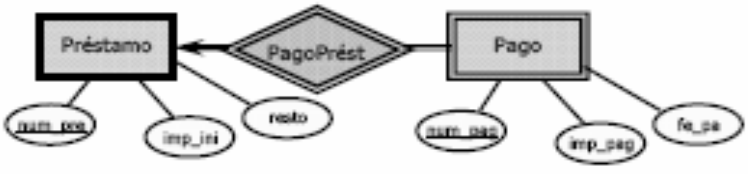
numero_prestamo	importe_inicial	resto
P-11	900	125
P-14	1200	1100
P-15	2000	565

Figura 4.16 Entidades Fuertes

4.6.2 Entidades Débiles

Sea A un conjunto de entidades débiles con los atributos a1, a2, …, am Sea B el conjunto de entidades fuerte del que A depende Sea la clave primaria de B el conjunto de atributos b1, b2, …, bn Se representa el conjunto de entidades A mediante una tabla llamada A con una columna por cada uno de los siguientes atributos: {a1, a2, …, am} U {b1, b2, …, bn}

Se crea una tabla que contiene una columna para los atributos que forman la llave primaria de la entidad fuerte a la que se encuentra subordinada.



Pago

numero_prestamo	numero_pago	importe_pago	Fecha_pago
P-11	53	50	07/06/2002
P-14	16	35	12/12/2000
P-15	11	25	01/03/1999

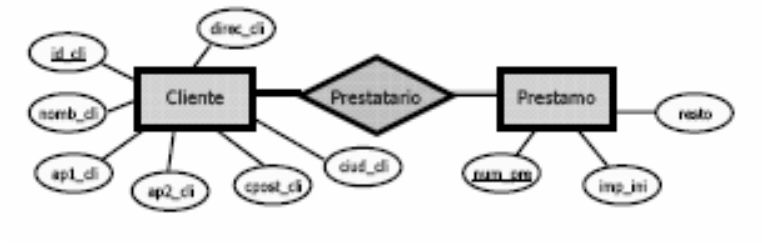
Figura 4.17 Entidades Débiles

4.6.3 Relación

Sea R un conjunto de relaciones, sean a1, a2, …, am el conjunto de atributos formados por la unión de las claves primarias de cada uno de los conjuntos de entidades que participan en R y sean b1, b2, …, bn los atributos descriptivos de R (si los hay) Se representa el conjunto de relaciones R mediante una tabla llamada R con una columna por cada uno de los siguientes atributos:

{a1, a2, …, am} U {b1, b2, …, bn}

Se crea una tabla que contiene una columna para cada atributo descriptivo de la relación y para cada atributo que conforma la llave primaria de las entidades que están relacionadas.



Prestatario

id_cliente	numero_prestamo
514862	P-11
486251	P-14
485926	P-15

Figura 4.18 Relación

4.6.4 Redundancia De Tablas

Un conjunto de relaciones uniendo un conjunto de entidades débiles con el correspondiente conjunto de entidades fuertes es un caso especial En general, la tabla para el conjunto de relaciones que une un conjunto de entidades débiles con su correspondiente conjunto de entidades fuertes es redundante y, por tanto, no es necesaria.

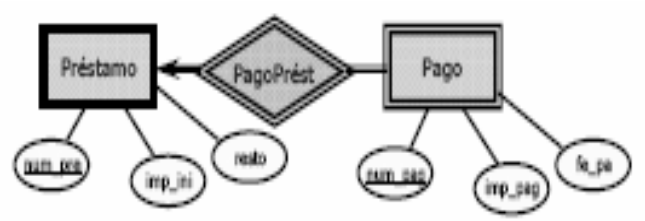


Figura 4.19 Entidades débiles y fuertes

4.6.5 Combinación De Tablas

Considerar un conjunto de relaciones R, varios a uno, del conjunto de entidades A al conjunto de entidades B.

⟶ Tablas A, B y R

Suponer que la participación de A es total

Entonces, se pueden combinar las tablas A y R para formar una única tabla consistente en la unión de las columnas de ambas tablas. En el caso de relaciones uno a uno, la tabla del conjunto de relaciones se puede combinar con cualquiera de las tablas de los conjuntos de entidades. Las tablas se pueden combinar incluso si la participación es parcial (usando valores nulos)

4.6.6 Atributos Compuestos

Los atributos compuestos se manejan creando un atributo separado para cada uno de los atributos componentes; no se crea una columna separada para el atributo compuesto.

4.6.7 Atributos Multivalorados

Para los atributos multivalorados se crean nuevas tablas; esa nueva tabla contiene la clave primaria del conjunto de entidades al cual pertenece el atributo y los diferentes valores del atributo generan diferentes tuplas

4.6.8 Generalización- Especialización

Se crea una tabla para el conjunto de entidades de más alto nivel; Para cada conjunto de entidades de nivel más bajo se crea una tabla que incluya una columna para cada uno de los atributos de ese conjunto de entidades más una columna por cada atributo de la clave principal del conjunto de entidades de alto nivel.

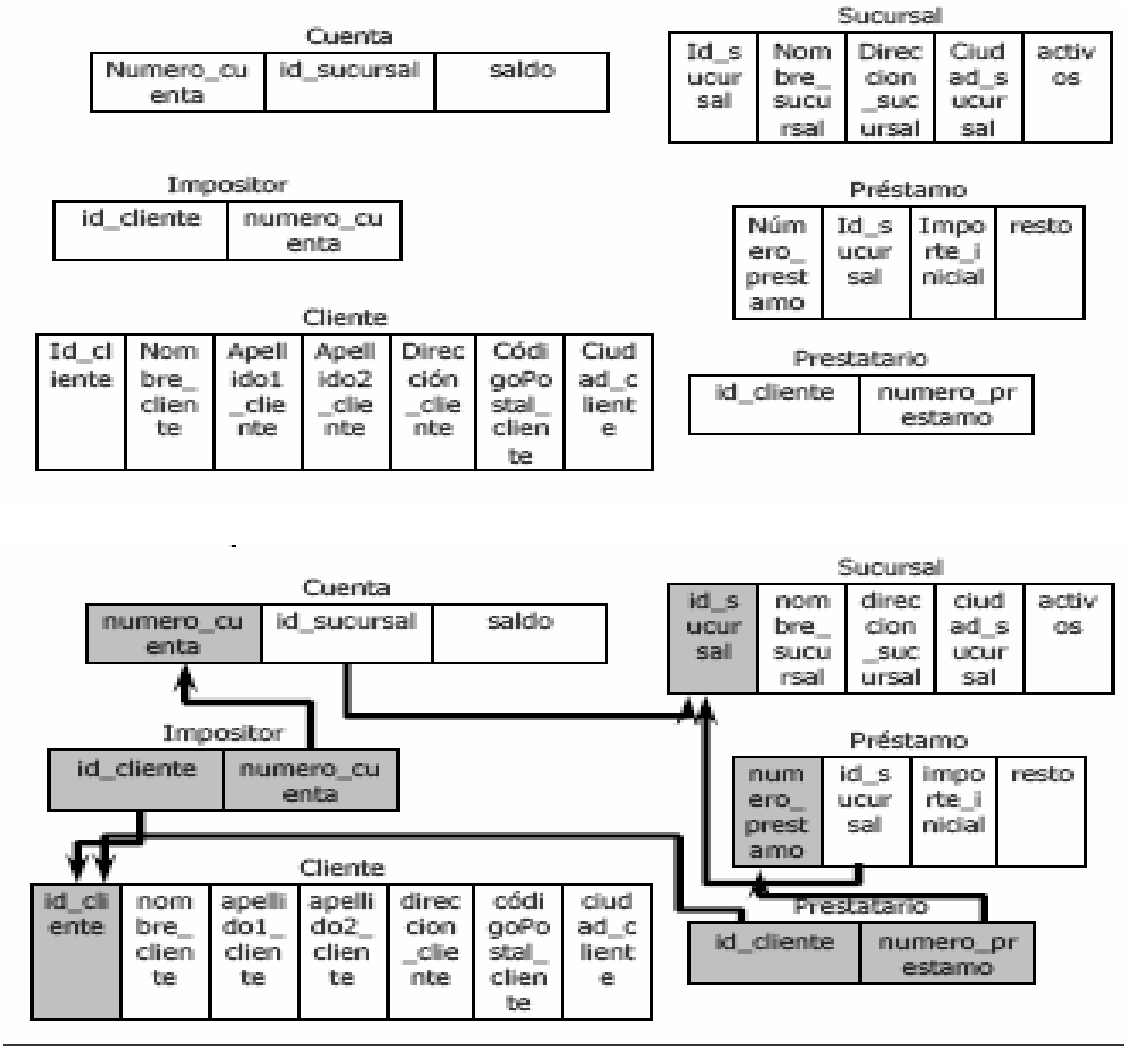


Figura 4.20 Ejemplo Banco

4.7 Actividades Complementarias

1. Construya un diagrama E-R para un hospital con un conjunto de pacientes y un conjunto de médicos. Asóciese con cada paciente un registro de las diferentes pruebas y exámenes realizados.

2. Una oficina de registro de una universidad mantiene datos acerca de las siguientes entidades:

    - asignaturas: incluyendo el número, titulo, programa y prerrequisitos
    - ofertas de asignaturas: incluyendo número de asignatura, año, semestre, número de sección, profesor(es), horarios y aulas
    - estudiantes: incluyendo id-estudiante, nombre y programa
    - profesores: incluyendo número de identificación, nombre, departamento y titulo.

    Además, la matricula de los estudiantes en asignaturas y las notas concedidas a estudiantes en cada asignatura en la que están matriculados se deben modelar adecuadamente.

3. Diseñe un diagrama E-R para almacenar los logros de su equipo deportivo favorito. Se deberían almacenar los partidos jugados, los resultados de cada partido, los jugadores de cada partido y las estadísticas individuales de cada jugador para cada partido. Las estadísticas de resumen se deberían modelar como atributos derivados.

4. Construya un esquema conceptual para la siguiente descripción a través del MER (diagrama distinguiendo entidades fuertes y débiles + cardinalidad + claves primarias + claves foráneas) con su correspondiente estructura de datos

    Diseñe un sistema de base de datos para controlar la información sobre rutas de una compañía de buses. Cada ruta cubierta por la compañía tiene un lugar de inicio y uno de término, pero puede pasar por varias paradas intermedias. La compañía está distribuida en varias sucursales. No todas las ciudades donde paran los autobuses tienen una sucursal; sin embargo, toda sucursal debe estar en una ciudad situada en las rutas de autobuses. Pueden existir múltiples sucursales en una misma ciudad y también múltiples paradas en la misma ciudad. La compañía asigna un autobús a cada ruta; algunas rutas pueden tener varios autobuses. Cada autobús tiene un conductor y un asistente, asignados por el día.

    Quizá necesite hacer ciertas suposiciones sobre los requerimientos de la aplicación; haga suposiciones RAZONABLES conforme avance.

5. Explique cual es la diferencia entre una entidad débil y una entidad fuerte.

CONSULTAS WEB

- Existen algunas herramientas herramientas de modelado de datos independientes que soportan diagramas E-R y diagramas de clase UML. Entre ellas están Rational Rose (www.rational.com/products/rose). Visio Enterprise (www.visio.com) y Erwin (www.cai.com/products)

CAPITULO 5. MODELO RELACIONAL

Un modelo de datos permite crear una representación de la realidad. Uno de estos modelos es el modelo Entidad-Relación, que permite crear una representación abstracta de la realidad. Dado que la representación de la realidad que obtiene es una representación abstracta, necesitamos un modelo de datos que sea directamente implementable. Uno de estos modelos es el modelo relacional, el cual será el objeto de estudio de este tema. El modelo relacional, además de diferenciarse del modelo Entidad-Relación en que es un modelo de implementación, se diferencia en que es un modelo lógico basado en registros en lugar de ser un modelo lógico basado en objetos. Así, la percepción de la realidad se realiza modelando los conceptos de la realidad como registros. Estos registros son agrupados en tablas, denominadas relaciones, lo que da lugar al nombre de modelo relacional. En este tema estudiaremos el modelo relacional, así como los dos lenguajes formales desarrollados para este modelo, como son el álgebra relacional y el cálculo relacional.

5.1. Orígenes del modelo relacional

Desde el punto de vista histórico, el modelo relacional fue introducido por Codd en 1970 sobre una base teórica bastante sólida y utiliza una estructura de datos sencilla y uniforme: la relación.

El modelo de datos relacional es relativamente nuevo y se ha establecido como el principal modelo de datos para aplicaciones comerciales de procesamiento de datos, debido fundamentalmente a la existencia en el mercado de muchos SGBD relacionales comerciales. Otros modelos de datos son más expresivos, como es el caso del modelo orientado a objetos, pero los SGBD para estos modelos no están tan extendidos.

No obstante, y debido a que el modelo relacional presenta algunas carencias para el modelado de objetos complejos, se ha desarrollado un modelo de datos que combina el modelo relacional con algunas de las características del modelo orientado a objetos. Este modelo mixto es conocido como el modelo objeto-relacional, el cual parece ser el modelo de datos que domine el mercado en los próximos años. Sin embargo, debido a que la mayor parte de las aplicaciones que se siguen desarrollando son para SGBD relacionales, y a que los conceptos explicados en este tema son aplicables tanto al modelo relacional como al objeto-relacional, no trataremos este último en este curso.

1.3 Estructuras de las Bases de datos relacionales (características y propiedades)

El modelo relacional representa a una base de datos como una colección de relaciones, es decir, un conjunto de tablas formadas por filas y columnas. Cada fila representa un conjunto de datos relacionados entre sí. Dichos valores pueden referirse a un conjunto de hechos que describen a una entidad o bien a un vínculo entre entidades. Por ejemplo, podemos tener una fila de una tabla que incorpore datos de los empleados para cada uno de los empleados de una empresa.

Las columnas representan las propiedades de cada una de las filas de la tabla. Por ejemplo, en una tabla que contenga información de empleados podemos tener columnas como DNI y Nombre para describir distintas características o propiedades de los empleados. El nombre de la tabla y los nombres de las columnas ayudan a interpretar el significado de los valores que están en cada una de las filas de la tabla. Por tanto, una base de datos relacional sería un conjunto de tablas con nombres únicos, en los que las filas representan hechos y las columnas representan propiedades.

En la terminología del modelo relacional, una fila se denomina tupla, una cabecera de columna es un atributo y una tabla es una relación.

Una Base de Datos Relacional consiste en un conjunto de tablas, a cada una de las cuales se le asigna un nombre exclusivo. Cada fila de la tabla representa una relación entre un conjunto de valores

Una tabla es un conjunto de relaciones → existe una fuerte correspondencia entre el concepto de tabla y el concepto matemático de relación

Informático	Matemático
Tabla	Relación
Registro o fila	Tupla
Campo o columna	Atributo

Figura 5.1 Estructura de las bases de datos relaciónales

5.2.1. Dominios, atributos tuplas y relaciones

La siguiente tabla tiene tres atributos: nombreCli, dniCli, y domicilio.

nombreCli	dniCli	Domicilio
Aranda	1	La Reina nº7
García	2	Fragata azul nº8
Hayes	3	Gibraltar español nº14
Turner	4	Gibraltar español nº17
Vilches	5	Diamante S/N
Lara	6	Gato negro nº13
Guerrero	7	Perro nº1

Figura 5.2 Atributos

Formalmente: Un dominio D es un conjunto de valores atómicos (indivisibles). Puede especificarse con el tipo de datos al que pertenecen los valores, y también con un nombre significativo que ayude a interpretarlos, así como otra información adicional. Por ejemplo, el dominio Peso_personas es un dominio numérico, especificado en kilogramos.

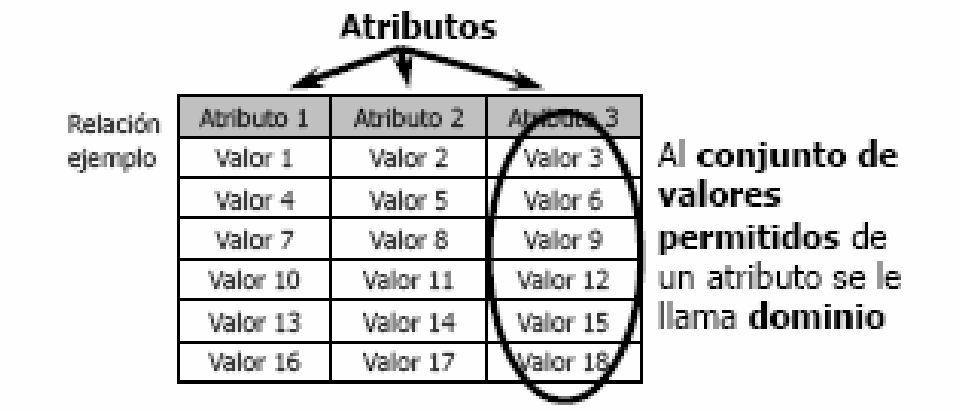


Figura 5.3 Estructura Básica de una tabla

Una tabla de n atributos es un subconjunto de D1 x D2 x … x Di x … x Dn-1 x Dn (matemáticamente una relación es un subconjunto del producto cartesiano de la lista de dominios)

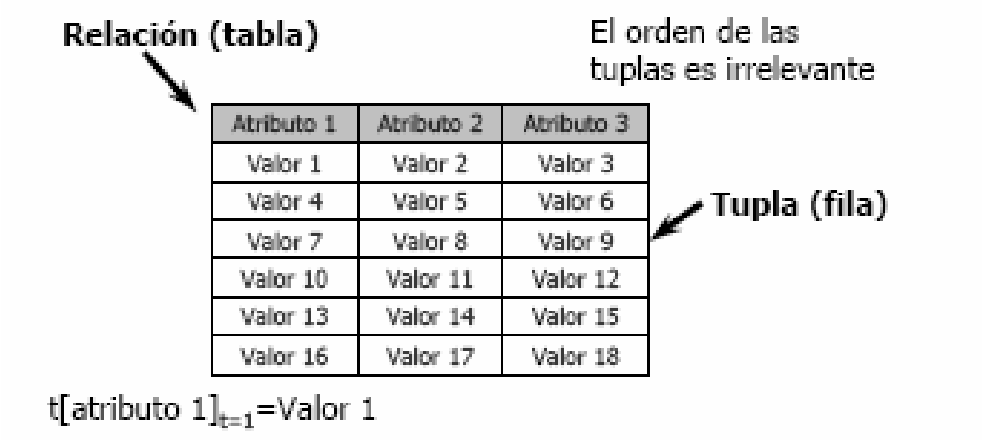


Figura 5.4 Nomenclatura

Los dominios deben ser atómicos pero no tiene por que ser disjuntos y el valor nulo pertenece a todos los dominios.

- Esquema de Relación: Similar al concepto de definición de tipos en los lenguajes de programación Esquema-abc=(atributo1(dominio1), atributo2(dominio2), …, atributoN(dominioN))

- Relación: Similar al concepto de variable en los lenguajes de programación nombre(Esquema-abc).

- Ejemplar de Relación: Similar al concepto de valor de variable en los lenguajes programación variable conforme se actualiza

5.2.2 Características de las relaciones

La definición de relación implica características que la distinguen de un fichero o tabla, que vamos a describir a continuación.

5.2.2.1. Orden de las tuplas en una relación

Como una relación es un conjunto de tuplas, y los elementos de un conjunto no están ordenados, las tuplas de una relación no tienen un orden específico. En cambio, los registros de un archivo se almacenan físicamente, de modo que siempre existe un orden entre ellos.

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

Según la definición de relación, cada tupla es una lista ordenada de n valores, por lo que el orden de los valores y el orden de los atributos en la definición de relación es importante. Sin embargo, a nivel lógico esto no es importante, siempre que se mantenga la correspondencia entre atributos y valores.

Hay una definición alternativa de relación que no necesita la ordenación de los valores de una tupla, que consiste en definir cada tupla como un conjunto de pares (<atributo><valor>). Esto hace que el orden de los atributos no sea importante, ya que el nombre del atributo aparece junto a su valor.

En nuestro caso, usaremos la primera definición de relación, donde los atributos y los valores de las tuplas están ordenados, ya que esto simplifica la notación.

Cada valor de una tupla es un valor indivisible en el modelo relacional básico, por lo que no se permiten atributos compuestos ni multivaluados. Sin embargo, recientemente se han intentado eliminar estas limitaciones, mediante el concepto de relaciones anidadas.

Cuando el valor de un atributo de una tupla es desconocido, o no se le puede aplicar, se usa un valor especial llamado valor nulo, aunque se debe intentar evitar su uso, ya que estos valores posteriormente son difíciles de manejar.

5.2.2.3. Interpretación de una relación

El esquema de una relación se puede interpretar como una declaración. Por ejemplo, el esquema de la relación clientes establece un cliente tiene nombreCli, dniCli y ciudadCli. Cada tupla de la relación se puede interpretar como un hecho.

Un esquema de relación también puede interpretarse como un predicado, y los valores de cada tuplas serán valores que satisfacen el predicado. Esta interpretación es útil en el contexto de la programación lógica.

5.2.3. Notación del modelo relacional

En este tema usaremos esta notación:

- Un esquema de relación R de grado n se denotará con $R(A_1, A_2, ..., A_n)$.
- Una n-tupla de una relación r(R) se denotará con $t=<v_1,v_2, ..., v_n>$.
- Para representar el valor v_i de t para el atributo A_i usaremos $t[A_i]$ o $t.A_i$.
- $t[A_1, A_2, ..., A_z]$ es una subtupla de valores $<v_1, v_2, ..., v_z>$ de t que se corresponden con los atributos especificados en la lista.
- Las letras mayúsculas denotan nombres de relaciones.
- Las letras q, r, s denotan estados de relaciones.
- Las letras t, u, v denotan tuplas.
- Un atributo 85A puede especificarse con el nombre de la relación R a la que pertenece usando la notación R.A. Esto es útil cuando se usa el mismo nombre para dos atributos en relaciones distintas, ya que dentro de una relación todos los atributos deben tener nombres distintos.

5.2.4 Claves

Los conceptos de superclave, de clave candidata y de clave primaria son aplicables en el modelo relacional de la misma forma que en el modelo entidad-relación

5.3 Restricciones relacionales y esquemas de bases de datos relacionales

Vamos a describir algunos tipos de restricciones sobre los datos que se pueden expresar en el esquema de una base de datos relacional, como son las restricciones de dominio, de clave, de integridad de entidades y de integridad referencial.

5.3.1. Restricciones de dominio

Las restricciones de dominio especifican que el valor de cada atributo debe ser un valor atómico de su dominio. Los tipos de datos asociados a los dominios suelen incluir los tipos numéricos estándar para enteros y reales, así como caracteres y cadenas de longitud fija y variable, fecha, hora, fecha y hora y moneda. También pueden especificarse intervalos o conjuntos explícitos de valores permitidos.

5.3.2. Restricciones de clave y restricciones sobre nulos

Todas la relaciones que forman el esquema de una base de datos relacional tienen que tener definida una clave primaria. Los conceptos de superclave, clave candidata y clave primaria ya vistos para el modelo Entidad-Relación también se puede aplicar al modelo relacional.

Por ejemplo, en el esquema de las sucursales, {nombreSuc} y {nombreSuc, ciudadSuc} son superclaves, pero {nombreSuc, ciudadSuc} no es clave candidata, puesto que si le quitamos el atributo ciudadSuc sigue siendo superclave, o lo que es lo mismo {nombreSuc} {nombreSuc, ciudadSuc} y {nombreSuc} es en sí una superclave.

Por tanto, se tiene que cumplir que en una relación R, si t_1 y t_2 son dos tuplas distintas de R, y K es el conjunto de atributos que forman la superclave, $t_1[K] \neq t_2[K]$.

La clave de una relación se determina a partir del significado de los atributos, y es una propiedad que no varía con el tiempo. Usaremos como convenio subrayar los atributos que forman la clave primaria de un esquema de relación.

Otra de las restricciones sobre los atributos especifica si se permiten o no los valores nulos. Si toda tupla de una relación debe tener un valor válido y no nulo para un atributo, entonces tendrá la restricción de ser no nulo.

5.3.3. Bases de datos relacionales y esquemas de bases de datos

En esta sección vamos a definir una base de datos relacional y un esquema de base de datos relacional. Un esquema de base de datos relacional S es un conjunto de esquemas de relaciones S = {R_1, R_2, ..., R_n} y un conjunto de restricciones de integridad RI. Un estado o instancia de una base de datos relacional BD de S es un conjunto de estados de relaciones BD = {r_1, r_2, ...,r_m}, en el que cada r_i es un estado de R_i y los estados de relaciones satisfacen las restricciones de integridad especificadas en RI.

De ahora en adelante, utilizaremos un ejemplo de una empresa bancaria, que tiene los siguientes esquemas de relaciones:

Sucursales = (nombreSuc, ciudadSuc, activo)

Empleados = (nombreEmp, dniEmp, teléfono, nombreSuc)

Cuentas = (numeroCta, saldo, nombreSuc)

Clientes = (nombreCli, dniCli, domicilio)

EsquemaCtaCli = (numeroCta, dniCli)

EsquemaTransacciones = (numeroCta, numeroTrans, fecha, importe)

Este conjunto de esquemas formaría el esquema conceptual de la base de datos del banco, y podría haberse obtenido mediante el paso a tablas de un diagrama entidad-relación como el que se muestra en la Figura

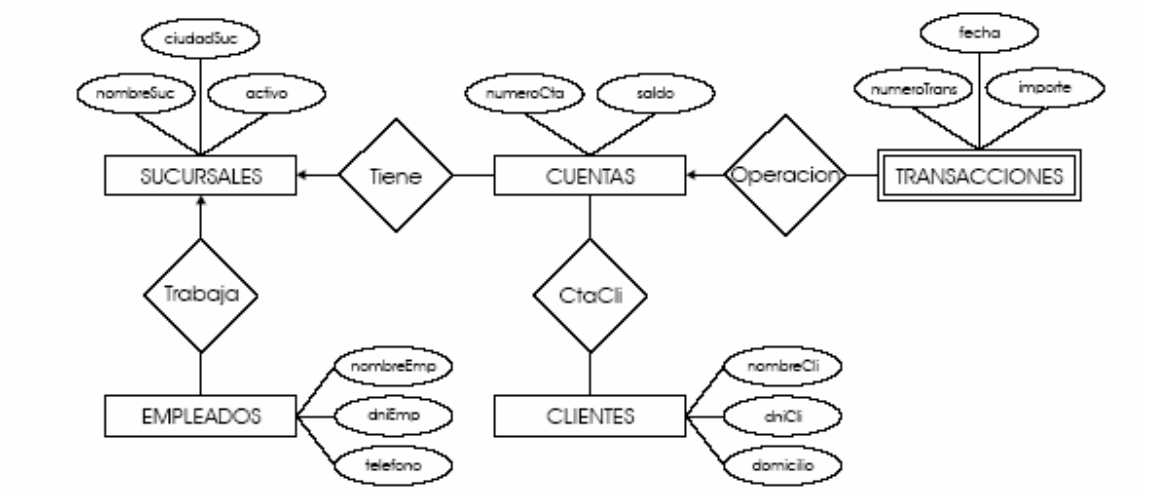


Figura 5.5 Diagrama E-R del sistema bancario

Los atributos que representan el mismo concepto del mundo real pueden tener o no el mismo nombre en relaciones distintas, y viceversa.

Las restricciones de integridad se especifican en el esquema de una base de datos, y además de las restricciones de dominio y clave, incluyen las restricciones de integridad de entidades e integridad referencial.

5.3.4. Integridad de entidades, integridad referencial y claves externas.

La restricción de integridad de entidades establece que ningún valor de clave primaria puede ser nulo. Esto se debe a que el valor de la clave primaria sirve para identificar las tuplas de una relación, y si la clave primaria puede tener valores nulos, no podríamos identificar algunas tuplas.

La restricción de integridad referencial se especifica entre dos relaciones, y establece que en una tupla de una relación que haga referencia a otra relación, deberá referirse a una tupla existente en dicha relación. Por ejemplo, el atributo nombreSuc de empleados indica la sucursal en que trabaja un empleado, y su valor deberá coincidir con el valor de nombreSuc en alguna tupla de la relación sucursales.

Para completar la definición de integridad referencial, debemos definir el concepto de clave externa. Un conjunto de atributos CE del esquema de la relación R_1 es una clave externa de R_1 si satisface estas condiciones:

1. Los atributos de CE tienen el mismo dominio que los atributos de la clave primaria CP de otro esquema de relación R_2; se dice que los atributos CE hacen referencia o se refieren a la relación R_2.
2. Un valor de CE en una tupla t_1 del estado actual de R_1 es el valor de CP en alguna tupla t_2 del estado actual de R_2, o bién es nulo. Si no es nulo, diremos que la tupla t_1 hace referencia a la tupla t_2. R_1 será la relación referenciante, y R_2 la relación referenciada.

En una base de datos con muchas relaciones, suele haber muchas restricciones de integridad referencial, ya que surgen de las relaciones representadas entre las entidades representadas por los esquemas de relación. Una clave externa también puede hacer referencia a su propia relación (caso de supervisor y empleado).

Las restricciones de integridad referencial pueden representarse gráficamente trazando un arco desde la clave externa hacia la relación a la que hace referencia, con la flecha apuntando hacia el atributo o atributos referenciados. Deben expresarse en el esquema de la base de datos, para que se mantengan automáticamente.

Las restricciones vistas no incluyen otra clase de restricciones llamadas restricciones de integridad semántica, tales como "un alumno no puede estar matriculado de más de 80 créditos", que necesitan un lenguaje de especificación de restricciones de propósito general. Para ello, se usan disparadores o triggers y aserciones. También hay otras restricciones llamadas restricciones de transición,

para tratar con cambios de estado de la base de datos ("el sueldo de un empleado sólo puede incrementarse"). Estas restricciones se especifican con reglas de actividad y disparadores. +

5.4. Operaciones de actualización y tratamiento de la violación de restricciones

Las operaciones del modelo relacional pueden clasificarse en recuperaciones y actualizaciones. En esta sección, vamos a ver las operaciones de actualización básicas, que son tres: Insertar (insertar una o más tuplas nuevas en una relación), Eliminar (elimina tuplas en una relación) y Actualizar (modifica los valores de algunos atributos de tuplas existentes). Cuando se realizan estas operaciones, no se deben violar las restricciones de integridad.

5.4.1. Insertar

Para insertar datos en una relación, bien especificamos la lista de valores de la tupla que se va a insertar o escribimos una consulta cuyo resultado sea el conjunto de tuplas que se va a insertar.

Para insertar una nueva transacción en la cuenta 6 de 700 € realizada el 7-11-2003, escribiríamos lo siguiente:

Insertar < "6", “2”, "7-11-2003", 700> en transacciones.

Al insertar, puede violarse una restricción de dominio si se proporciona un valor de atributo que no pertenece al dominio correspondiente. También puede violarse una restricción de clave si la clave de la nueva tupla ya existe en la relación. La integridad de entidades puede violarse si la clave primaria de la nueva tupla es nula. Y la integridad referencial puede violarse si el valor de cualquier clave externa de la nueva tupla hace referencia a una tupla que no existe en la relación referenciada.

Cuando una inserción viola una o más restricciones, la opción predeterminada es rechazar la inserción.

3.4.2. Eliminar

La operación Eliminar sólo puede violar restricciones de integridad referencial, cuando las claves externas de otras tuplas de la base de datos hacen referencia a la tupla que se va a eliminar. Para especificar la información a eliminar, se expresa una condición en términos de los atributos de la relación correspondiente.

Por ejemplo, para eliminar todas las transacciones de la cuenta número 1 escribiríamos:

Eliminar en transacciones las tuplas con numeroCta=”1”

Cuando la eliminación provoca una violación de restricciones de integridad referencial, tenemos tres opciones. La primera es rechazar la eliminación. La segunda es la eliminación en cascada, eliminando también las tuplas que hacen referencia a la tupla o tuplas que se desean eliminar. Una tercera opción es modificar los valores del atributo de referencia que provocan la violación: estos valores se pondrían a nulo o se modificarían para hacer referencia a otra tupla válida. La opción seleccionada en cada caso debe poder especificarse también en el esquema de la base de datos.

5.4.3. Actualizar

La operación Actualizar modifica los valores de uno o más atributos de una o más tuplas de una relación. Es necesario especificar una condición sobre los atributos de la relación para seleccionar la tupla o tuplas a modificar.

Por ejemplo, para cambiar el domicilio del cliente con dniCli 1 a C/Real nº 14, escribiríamos:

Actualizar domicilio de clientes con dniCli = "1"

La actualización de un atributo que no es clave primaria ni clave externa no suele dar problemas (excepto de dominio). Modificar un valor de la clave primaria es similar a eliminar una tupla e insertar otra en su lugar, por lo que pueden darse los problemas vistos para inserciones y eliminaciones. Si se modifica un atributo de una clave externa, hay que comprobar que el nuevo valor existe en la relación referenciada (o es nulo).

5.5 Lenguajes de consulta

Un lenguaje de consulta es un lenguaje en el que el usuario puede solicitar información de la base de datos. Estos lenguajes de consulta suelen ser de más alto nivel que los lenguajes de programación estándar, y pueden clasificarse en procedimentales y no procedimentales.

En un lenguaje procedimental el usuario da instrucciones al sistema para que realice una serie de operaciones sobre la base de datos con el fin de obtener el resultado deseado.

En un lenguaje no procedimental el usuario describe la información deseada sin dar un procedimiento concreto sobre cómo obtener esa información.

5.5.1 Álgebra Relacional

El álgebra relacional es un lenguaje de consulta procedimental Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

Operaciones fundamentales: selección, proyección, unión, diferencia de conjuntos, producto cartesiano y renombramiento Unarias / Binarias

5.5.1.1 Selección

La operación selección selecciona tuplas que satisfacen un predicado dado Para denotar la selección se utiliza la letra griega sigma minúscula (σ) El predicado aparece como subíndice de σ La relación sobre la que se aplica aparece entre paréntesis

σ predicado (relación)

Por tanto, para seleccionar las tuplas de la relación Sucursales que representan sucursales que están en la ciudad "Madrid", escribimos:

$\sigma_{ciudadSuc = "Madrid"}$ (sucursales)

y la relación resultante es la siguiente:

Nombresuc	Ciudadsuc	Activo
Castellana	Madrid	900000
Sol	Madrid	710000

Figura 5.6 Relación Resultante selección de tuplas

Podemos encontrar todas las tuplas en las que el activo de la sucursal sea mayor que 50.000 € escribiendo:

$\sigma_{activo > 50000}$ (sucursales)

con lo que obtendríamos

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000
Ronda	Almería	80000
Sol	Madrid	71000

Figura 5.7 Relación Resultante selección de tuplas

En general se permiten las comparaciones utilizando los operadores =, ≠, <, ≤, > y ≥ en el predicado de selección. Además, pueden combinarse predicados para formar un predicado compuesto utilizando los conectores lógicos y (Λ) y o (v). Por tanto, para encontrar las tuplas de la relación sucursales de las sucursales de "Madrid" con un activo superior a 80000, escribiríamos:

$\sigma_{ciudadSuc = "Madrid" \square activo > 80000}$ (sucursales)

Lo que nos devolvería la relación

nombreSuc	ciudadSuc	Activo
Castellana	Madrid	90000

Figura 5.8 Relación Resultante selección de tuplas

5.5.1.2. Proyección

La operación de proyección es una operación unaria que devuelve su relación argumento con ciertas columnas omitidas. Puesto que la relación es un conjunto, devolvería todas las filas duplicadas. La proyección se denota por la letra griega pi (Π), y los atributos que se desea obtener en el resultado se ponen como subíndices de Π.

Para conocer el nombre y teléfono de los empleados del banco escribiríamos:

$\Pi_{nombreEmp, telefono}$ (empleados)

y obtendríamos la siguiente relación:

nombreEmp	telefono
García	101010
Torres	111111
López	121212
Villegas	131313
Fernández	141414
Urrutia	151515

Figura 5.9 Relación Resultante Proyeccion de tuplas

5.5.1.3 Secuencias de operaciones

En general, es posible aplicar varias operaciones del álgebra relacional del mismo tipo una tras otra, pero también es posible agrupar varias operaciones diferentes en una sola expresión del álgebra relacional, y esto se hace anidándolas. Esta característica se debe a que el resultado de aplicar a una relación una operación del álgebra relacional es otra relación.

Para conocer el nombre y el teléfono de los empleados que trabajan en la sucursal de "Castellana" escribiríamos:

$\Pi_{nombreEmp, telefono}$ ($\sigma_{nombreSuc = "Castellana"}$(sucursales))

Lo que daría lugar a la siguiente relación:

nombreEmp	telefono
García	101010
Torres	111111

Figura 5.10 Relación Resultante secuencia de operaciones

Obsérvese como en lugar de pasar una relación "original" a la operación de proyección, se ha pasado una expresión, que se evalúa como relación.

5.5.1.4. Unión

La operación unión se limita a unir resultados de relaciones, y funciona de la misma forma que la unión de conjuntos, con la observación de que las relaciones que se unan han de tener los mismos atributos, y que se eliminan las tuplas repetidas.

Supongamos que queremos conocer todos los nombres de personas relacionadas con el sistema bancario, ya sea que trabajen en el banco, o que tengan una cuenta en el banco. Para esto necesitamos obtener información de la relación empleados y de la relación clientes.

Para conocer los nombres de los clientes, escribiríamos:

$\Pi_{nombreCli}(clientes)$

Para conocer los nombres de los empleados, escribiríamos:

$\Pi_{nombreEmp}(empleados)$

Luego para contestar a la consulta planteada de cuáles son los nombres de todas las personas relacionadas con el sistema bancario deberíamos unir las dos relaciones resultantes, mediante el operador unión, para que aparecieran los nombres de los clientes y los nombres de los empleados, por lo que tendríamos que escribir lo siguiente:

$\Pi_{nombreEmp}(empleados) \cup \Pi_{nombreCli}(clientes)$

El resultado de esta consulta sería el siguiente, teniendo en cuenta que las tuplas repetidas se eliminan.

NombreEmp
García
Torres
López
Villegas
Fernández

Figura 5.11 Relación Resultante

En general, debemos asegurarnos que las uniones se realizan entre relaciones compatibles, es decir, que tengan el mismo número de atributos y que los dominios de los atributos sean iguales dos a dos.

5.5.1.5. Diferencia de conjuntos

La operación de diferencia de conjuntos, representada por el símbolo -, nos permite encontrar las tuplas que estén en una relación pero no estén en otra. La expresión R - S da como resultado la relación que contiene las tuplas que están en R pero que no están en S.

Como ejemplo, la consulta que nos daría el nombre de los clientes que no son empleados sería:

$\Pi_{nombreCli}(clientes) - \Pi_{nombreEmp}(empleados)$

El resultado de la consulta sería la siguiente relación:

nombreCli
Aranda
Hayes
Turner
Vilches
Lara
Guerrero

Figura 5.12 Relación Resultante

5.5.1.6 Producto cartesiano

Las operaciones de selección y proyección vistas hasta ahora sólo permiten extraer información de una relación a la vez. Pues bien, para poder combinar varias relaciones se puede utilizar la operación de producto cartesiano, cuyo símbolo es una equis (x). El resultado de la operación de producto cartesiano es una nueva relación que tiene tantas columnas como la suma del número de columnas de cada relación, y las tuplas de esta nueva relación se obtienen mediante la combinación de todas las tuplas de las relaciones que participan en la operación de producto cartesiano.

A la hora de realizar la operación de producto cartesiano, hay que tener en cuenta que como se está haciendo el producto de las relaciones se puede generar un resultado bastante grande, por lo que siempre tendremos que tener claro, antes de escribir la operación, si existe un predicado que debamos considerar, especialmente

Cuando lo que queramos realizar sea relacionar las tablas. Para hacer referencia a los nombres de las columnas de forma que se eviten posibles ambigüedades, se han de especificar los nombres de las tablas seguidos de un punto y el nombre de los atributos, para determinar con exactitud a qué atributo de qué relación estamos haciendo referencia.

A continuación se ilustra la relación que resulta de hacer el producto cartesiano Préstamo x Prestatario Averiguar los nombres de todos los clientes que tienen concedido un préstamo en la Sucursal de Fingoi

Préstamo

numero_ prestamo	nombre_ sucursal	importe
P-13	Fingoi	10000
P-23	Fingoi	8700
P-18	Vite	15100

Prestatario

nombre_ cliente	numero_ prestamo
Fernández	P-13
Abril	P-13
Pérez	P-18
Fernández	P-23
Barreiro	P-13
Rodríguez	P-18

nombre_ cliente	prestatario. numero_ prestamo	prestamo. numero_ prestamo	nombre_ sucursal	importe
Fernández	P-13	P-13	Fingoi	10000
Fernández	P-13	P-23	Fingoi	8700
Fernández	P-13	P-18	Vite	15100
Abril	P-13	P-13	Fingoi	10000
Abril	P-13	P-23	Fingoi	8700
Abril	P-13	P-18	Vite	15100
Pérez	P-18	P-13	Fingoi	10000
Pérez	P-18	P-23	Fingoi	8700
Pérez	P-18	P-18	Vite	15100

Figura 5.13 Relación Resultante

Como se puede observar en la relación resultante del producto cartesiano anterior, el número de tuplas de la relación producto cartesiano se dispara, debido a que se realiza la combinación de todas las tuplas de una relación con todas las tuplas de las relaciones que también participen en el producto cartesiano. Sin embargo, no todas las tuplas que hay en dicha relación parecen tener un significado lógico. Este significado lógico se refiere a que cuando se quieren combinar varias relaciones, uno de los resultados buscados es el de obtener sólo las tuplas en las que coinciden los valores de los atributos que establecen la relación o relaciones. Para ello, deberíamos de aplicar una selección al resultado del producto cartesiano de forma que se seleccionasen únicamente esas tuplas. Ilustremos esto con un ejemplo.

Si en la relación anterior aplicamos el predicado que selecciona sólo las tuplas en las que coincide el nombre de sucursal, estaremos combinando para cada prestatario los datos de la sucursal en la que trabaja dicho, Seleccionamos en el resultado del producto cartesiano las tuplas cuyo valor en nombre_sucursal sea el que se busca.

σnombre_sucursal=“Fingoi” (prestatario x préstamo)

Dando lugar a este resultado.

nombre_ cliente	prestatario. numero_ prestamo	prestamo. numero_ prestamo	nombre_ sucursal	importe
Fernández	P-13	P-13	Fingoi	10000
Fernández	P-13	P-23	Fingoi	8700
Abril	P-13	P-13	Fingoi	10000
Abril	P-13	P-23	Fingoi	8700
Pérez	P-18	P-13	Fingoi	10000
Pérez	P-18	P-23	Fingoi	8700
Fernández	P-23	P-13	Fingoi	10000
Fernández	P-23	P-23	Fingoi	8700
Barreiro	P-13	P-13	Fingoi	10000
Barreiro	P-13	P-23	Fingoi	8700
Rodríguez	P-18	P-13	Fingoi	10000
Rodríguez	P-18	P-23	Fingoi	8700

Figura 5.14 Relación Resultante

Filtramos el resultado a través del número de préstamo σprestatario.numero_prestamo=prestamo.numero_prestamo

(σnombre_sucursal=“Fingoi” (prestatario x préstamo))

nombre_ cliente	prestatario. numero_ prestamo	prestamo. numero_ prestamo	nombre_ sucursal	importe
Fernández	P-13	P-13	Fingoi	10000
Abril	P-13	P-13	Fingoi	10000
Fernández	P-23	P-23	Fingoi	8700
Barreiro	P-13	P-13	Fingoi	10000

Figura 5.15 Relación Resultante

Y proyectamos el resultado buscado

Πnombre_cliente (σprestatario.numero_prestamo=prestamo.numero_prestamo (σnombre_sucursal=“Fingoi” (prestatario x préstamo)))

nombre_ cliente
Fernández
Abril
Barreiro

Figura 5.16 Relación Resultante

5.5.1.7 Renombramiento

La operación renombramiento permite poner nombre a una relación (obtenida a través de una expresión) que no lo tenga. Para denotar la proyección se utiliza la letra griega rho minúscula (ρ) (también la partícula as en una lista de argumentos) El nombre a dar aparece como subíndice de ρ La expresión sobre la que se aplica aparece entre paréntesis. **ρ nombre (expresión)**

5.5.1.8. Renombrar

Las consultas en las que aparece varias veces una misma relación pueden originar ambigüedades, que necesitamos resolver de algún modo.

Imaginemos que deseamos conocer los nombres de los empleados que trabajan en la misma sucursal que García.

Para obtener el nombre de la sucursal donde trabaja García, escribiríamos

$$\Pi_{nombreSuc}\,(\sigma_{nombreEmp = "García"}(empleados))$$

Sin embargo, para encontrar otros empleados que trabajen en esa misma sucursal debemos volver a hacer referencia a la relación empleados

$$\sigma_P \times \Pi_{nombreSuc}\,(\sigma_{nombreEmp = "García"}(empleados))$$

Donde P es un predicado de selección que hace que los valores de nombreSuc sean iguales a los de García. Para especificar a qué valor de nombreSuc estamos haciendo referencia, no podemos utilizar empleados.nombreSuc, ya que se estarían tomando de la relación empleados, y no el del resultado parcial.

Esto se soluciona mediante el operador renombrar (ρ).

La expresión $\rho_X(R)$

Devuelve la relación R con el nombre X. De esta forma podemos hacer referencia varias veces a la misma relación sin ambigüedad.

En la consulta que estamos intentando resolver, utilizaremos el nombre empleados2 para renombrar la relación empleados. Por tanto la consulta quedaría de la forma siguiente:

$$\Pi_{empleados.nombreEmp}(\sigma_{empleados2.nombreSuc = empleados.nombreSuc}\,(empleados \times$$

$$\rho_{empleados2}(\Pi_{nombreSuc}\,(\sigma_{nombreEmp = "García"}(empleados)))))$$

El resultado de la consulta sería

nombreEmp
García
Torres

Figura 5.17 Relación Resultante

5.5.2. Otras operaciones del álgebra relacional

5.5.2.1 Intersección de conjuntos

Se trata de una operación adicional del álgebra de conjuntos, puesto que la operación de intersección ∩, se puede expresar como dos diferencias de conjuntos de esta forma: R ∩ S = R - (R - S) Para conocer el nombre de los empleados que tienen cuenta en alguna de las sucursales del banco, escribiríamos lo siguiente: ΠnombreCli(clientes) ∩ ΠnombreEmp(empleados) El resultado de la consulta sería la siguiente relación:

nombreCli
García

Figura 5.18 Relación Resultante

5.5.2.2 Producto theta

Se trata de una operación para simplificar las consultas que utilizan la operación de producto cartesiano, y que permite especificar la condición de selección de registros como subíndice de dicha operación. Se representa mediante $x_{\Theta,}$ siendo Θ subíndice el predicado que selecciona las tuplas deseadas, que pueden ser las que tienen valores coincidentes de campos comunes o no.

Como ejemplo, supongamos que estamos interesados en obtener el nombre de los clientes que tienen una cuenta en la sucursal de "Castellana". Con lo que sabemos hasta ahora, haríamos el producto cartesiano de las relaciones clientes, cuentas y ctacli, luego seleccionaríamos de esta relación las tuplas en las que coincidiesen los DNI de los clientes y los números de cuenta, y por último haríamos una proyección para filtrar los atributos que deseamos. Todo esto lo podríamos escribir de la forma siguiente:

$\Pi_{nombreCli}$ ($\sigma_{cuentas.numeroCta = ctacli.numeroCta \wedge ctacli.dniCli = clientes.dniCli}$ ($\sigma_{cuentas.nombreSuc = "Castellana"}$ (clientes x ctacli x cuentas)))

Pues bien, con la operación de producto theta, la consulta se escribiría de la forma siguiente:

$\Pi_{nombreCli}$ (($\sigma_{cuentas.nombreSuc = "Castellana"}$ (clientes $x_{clientes.dniCliente=cuentacli.dniCliente}$ cuentacli $x_{cuentacli.numeroCta=cuentas.numeroCta}$ cuentas)

El resultado de la consulta sería el siguiente:

nombreCli
Aranda

Figura 5.19 Relación Resultante

5.5.2.3 Producto natural

Se trata de una operación para simplificar las consultas que utilizan la operación de producto cartesiano o producto theta, y que permite obviar la especificación de condición de selección de registros como subíndice de dicha operación. Se representa mediante |x|, y hace directamente el producto cartesiano y la selección de las tuplas deseadas, que serán las que tienen valores coincidentes de campos comunes.

Volvamos al ejemplo anterior, en el que deseamos obtener el nombre de los clientes que tienen una cuenta en la sucursal de "Castellana". Con lo que sabemos hasta ahora, haríamos el producto cartesiano de las relaciones clientes, cuentas y ctacli, y seleccionaríamos de esta relación las tuplas en las que coincidiesen los DNI de los clientes y los números de cuenta, y por último haríamos una proyección para filtrar los atributos que deseamos. Con el producto natural, lo podríamos escribir de la forma siguiente:

$\Pi_{nombreCli}$ (($\sigma_{cuentas.nombreSuc = "Castellana"}$ (clientes $|x|_{clientes.dniCliente=cuentacli.dniCliente}$ cuentacli $|x|_{cuentacli.numeroCta=cuentas.numeroCta}$ cuentas)

El resultado de la consulta sería el siguiente:

nombreCli
Aranda

Figura 5.20 Relación Resultante

5.5.2.4. División

La operación de división, cuyo símbolo es ÷, es adecuada para consultas que incluyen la expresión "todos". Suponga que desea saber cuáles son los clientes que tienen cuentas en todas las sucursales de Granada. En primer lugar, podemos obtener los nombres de todas las sucursales de Granada, con la expresión

$\Pi_{nombreSuc}(\sigma_{ciudadSuc = "Granada"}(sucursales))$

A continuación, podemos obtener los nombres de las sucursales en las que tienen cuenta todos los clientes:

$$\Pi_{nombreCli,nombreSuc}(clientes \mid x \mid cuentacli \mid x \mid cuentas)$$

Por último, necesitamos ver los clientes que aparecen en la segunda relación con todos los nombres de sucursales de la primera.

Formalmente, sean r(R) y s(S) dos relaciones, tales que (todos los atributos del esquema S están incluidos en el esquema R). La relación r ÷ s es una relación con el esquema R – S (atributos de R que no están en S). Una tupla t pertenece a r ÷ s si y sólo si se cumplen estas dos condiciones:

1. t pertenece a $\Pi_{R-S}(r)$

2. Para cada tupla t_S de s existe una tupla t_r de r que satisface que

$$t_r[S] = t_s[S] \text{ y } t_r[R - S] = t$$

5.5.2.5. Asignación

La operación de asignación, representada por ←, funciona de forma similar a la asignación en un lenguaje de programación.

La evaluación de una asignación no da como resultado una relación que se presenta al usuario, sino que el resultado de la expresión que está a la derecha de ← se asigna a la variable de relación situada a la izquierda de ←.

5.5.2.6. Extensiones del producto natural

A continuación estudiaremos algunas extensiones del producto natural y de la unión, dada su creciente aparición en los SGBD comerciales.

Recordemos que en la operación de producto natural R |x| S, se creaba una relación con las tuplas de R que tienen tuplas coincidentes en S y viceversa. Por tanto, si una tupla de una relación no encuentra otra "tupla relacionada" en la otra relación, no aparece en la relación resultante.

Para evitar esta discriminación de tuplas existen unas operaciones denominadas genéricamente uniones externas (outer joins), y aquí estudiaremos la Unión externa izquierda y la Unión externa.

Unión externa izquierda (Left Join)

Se denota por el símbolo =x. R =x S conserva todas las tuplas de la primera relación (la relación de la izquierda), de forma que si no se encuentran tuplas coincidentes en S se rellenan con valores nulos.

Imaginemos que deseamos saber los nombres de sucursal de la ciudad de "Almería", junto con los datos de los empleados que trabajan en dichas sucursales. Los datos de las sucursales son necesarios aunque no existan datos sobre los empleados. Para ello, haríamos una combinación de una operación de proyección, selección y un left join de esta forma.

$\Pi_{\text{sucursales.nombreSuc,empleados.nombreEmp}}($

$\sigma_{\text{ciudadSuc="Almería"}}(\text{sucursales}) =x_{\text{ sucursales.nombreSuc=empleados.nombreSuc}}$ empleados)

Así pues, el resultado sería el siguiente:

Sucursales.nombreSuc	Empleados.nombreEmp
Paseo	López
Paseo	Villegas
Zapillo	Fernández
Ronda	nulo

Figura 5.21 Relación Resultante

Análogamente, existe una operación denominada Unión externa derecha, denotada por x=, que incluye en la relación R x= S todas las tuplas de S (la de la derecha) aunque no tengan tuplas con valores coincidentes en la relación de la izquierda.

5.5.2.7 Unión externa

Se trata de una unión que une dos relaciones que no sean compatibles, es decir, que no se les pueda aplicar el operador de unión, ya que sólo algunos de los atributos son compatibles con la unión.

En el sistema bancario imaginemos que deseamos conocer los nombres, DNI, teléfonos y dirección de todas las personas implicadas en el sistema bancario (clientes y empleados). Para ello, haríamos lo siguiente:

$\Pi_{nombreEmp,\ dniEmp,\ telefono}$ (empleados) =x= $\Pi_{nombreCli,\ dniCli,\ domicilio}$ (clientes)

y el resultado sería

nombreEmp	dniEmp	telefono	domicilio
García	10	101010	nulo
Torres	11	111111	nulo
López	12	121212	nulo
Villegas	13	131313	nulo
Fernández	14	141414	nulo
Urrutia	15	151515	nulo
Guerrero	7	777777	Perro nº1
Aranda	1	nulo	La Reina nº7
García	2	nulo	Fragata azul nº8
Hayes	3	nulo	Gibraltar español nº14
Turner	4	nulo	Gibraltar español nº17
Vilches	5	nulo	Diamante S/N
Lara	6	nulo	Gato negro nº13

Figura 5.22 Relación Resultante

5.5.2 Cálculo Relacional De Tuplas

Cuando escribimos una expresión en álgebra relacional, damos una secuencia de procedimientos que genera la respuesta a nuestra consulta. El cálculo relacional de tuplas, en cambio, es un lenguaje de consultas no procedimental.

Describe la información deseada sin dar un procedimiento específico para obtener esa información.

Una consulta en el cálculo relacional de tuplas se expresa como {t|P(t)} es decir, el conjunto de todas las tuplas t, tal que el predicado P es verdadero para t. Usamos t[A] para representar el valor de la tupla t en el atributo A, y usamos tÎr para indicar que la tupla t está en la relación r.

5.5.2.1. Consultas ejemplo.

Encontrar el nombre_sucursal, número_préstamo, nombre_cliente y cantidad para préstamos mayores de 1200 dólares:

{t|tÎpréstamo Ù t[cantidad]>1200}

Supóngase que queremos únicamente el atributo nombre_cliente. En el cálculo relacional de tuplas necesitamos escribir una expresión para una relación sobre el esquema (nombre_cliente). Necesitamos aquellas tuplas en (nombre_cliente) tales

que exista una tupla en préstamo correspondiente a ese nombre_cliente con el atributo cantidad>1200.

Para expresar esto necesitamos la construcción existe ($)de la lógica matemática. $tÎr(Q(t)) que significa “existe una tupla t en la relación r tal que el predicado Q(t) es verdadero”.

Usando esta notación podemos escribir la consulta “encontrar a todos los cliente con un préstamo superior a 1200” como

{t|$sÎpréstamo (t[nombre_cliente] = s[nombre_cliente] Ù s[cantidad]>1200}

La expresión anterior se lee “el conjunto de todas las tuplas t tal que existe una tupla s en la relación préstamo para la cual los valores de t y s para el atributo nombre_cliente son iguales y el valor de s para el atributo cantidad es mayor de 1200 dólares”.
La variable de tupla t se define sólo en el atributo nombre_cliente, puesto que éste es el único atributo para el que se especifica una condición para t. Así, el resultado es una relación sobre (nombre_cliente). Considérese la consulta “encontrar a todos los clientes que tienen un préstamo de la sucursal Perrydge y las ciudades en las que viven”. Esta consulta implica dos relaciones, por lo que requiere que tengamos dos clausulas existe conectadas por un y (Ù).

{t|$sÎpréstamo (t[nombre_cliente] = s[nombre_cliente] Ù s[nombre_sucursal] = “Perrydge” Ù $uÎcliente
(u[nombre_cliente] = s[nombre_cliente] Ù t[ciudad_cliente] = u[ciudad_cliente]))}

Este es “el conjunto de todas las tuplas (nombre_cliente, ciudad_cliente) para las cuales nombre_cliente es un receptor de préstamo en la sucursal Perrydge y ciudad_cliente es la ciudad de nombre_cliente”. La variable de tupla s
asegura que el cliente tiene un préstamo en la sucursal Perrydge, y u tiene la restricción que debe pertenecer al mismo cliente que s, y u asegura que ciudad_cliente es la ciudad del cliente. Ahora considérese la consulta “encontrar a todos los clientes que tienen una cuenta en Perrydge pero no un préstamo en ella”.

{t|$uÎdepósito (t[nombre_cliente] = u[nombre_cliente] Ù u[nombre_sucursal] = “Perrydge” Ù
Ø$sÎpréstamo (t[nombre_cliente] = s[nombre_cliente] Ù s[nombre_sucursal] = “Perrydge”)}

Esta expresión usa la cláusula $uÎdepósito (…) para exigir que el cliente tenga una cuenta en Perrydge y la cláusula Ø$sÎpréstamo (…) para eliminar los que tengan también un préstamo.

La consulta que vamos a considerar a continuación usa la implicación (Þ). La fórmula PÞQ significa “P implica Q”; es decir, si P es verdadera, Q debe ser

verdadera, y es equivalente lógicamente a ØPÚQ. El uso de la implicación a menudo sugiere una interpretación más intuitiva de una consulta.

Considérese la consulta “encontrar a todos los clientes que tienen una cuenta en todas las sucursales de Brooklyn”. Para esta consulta introducimos la construcción “para todos”, representada por ". La notación "tÎr(Q(t)) significa “Q es verdadera para todas las tuplas t en la relación r”. La consulta sería:

{t|"uÎsucursal (u[ciudad_sucursal]=”Brooklyn” Þ $sÎdepósito t[nombre_cliente]=s[nombre_cliente] Ù
u[nombre_sucursal]=s[nombre_sucursal]))}

Interpretamos la expresión como el conjunto de todos los clientes (t[nombre_cliente]) tal que para todas las tuplas u en la relación sucursal, si el valor de u en el atributo ciudad_sucursal es Brooklyn entonces el cliente tiene una cuenta en la sucursal cuyo nombre aparece en el atributo nombre_sucursal de u.

5.5.2.2 Definición formal.

Ahora estamos preparados para la definición formal. Una expresión del cálculo relacional de tuplas es de la forma: {t|P(t)} donde P es una fórmula. En una fórmula pueden aparecer varias variables de tuplas. Se dice que una variable de tupla es una variable libre a menos que este cuantificada por un $ o por un ", que entonces se dice variable limite.

Una fórmula en el cálculo relacional de tuplas se compone de átomos. Un átomo tiene una de las siguientes

formas:

1.- sÎr, donde s es una variable de tupla y r es una relación. No se permite la operación Ï.
2.- s[x]Qu[y], donde s y u son variables de tuplas, x e y son atributos sobre los que están definidos s y u respectivamente, y Q es un operador de comparación (<, £, =, ³, >). Se requiere que los atributos x e y tengan dominios cuyos miembros puedan compararse por medio de Q.
3.- s[x] Qc, donde s es una variable de tupla, x es una atributo sobre el que s esta definida, Q es un operador de comparación, y c es una constante en el dominio del atributo x. Las fórmulas se construyen a partir de átomos usando las siguientes reglas:
1.- Un átomo es una fórmula.
2.- Si P1 es una fórmula, entonces también lo son ØP1 y (P1).
3.- Si P1 y P2 son fórmulas, entonces también lo son P1ÙP2, P1ÚP2 y P1ÞP2.
4.- Si P1(s) es una fórmula que contiene una variable de tupla libre, entonces $sÎr(P1(s)) y "sÎr(P1(s)) también

lo son. Como en el álgebra relacional, es posible escribir expresiones equivalentes que en apariencia no son idénticas, como estas tres:
1.- P1ÙP2 es equivalente a Ø(ØP1ÚØP2).
2.- "tÎr(P1(t)) es equivalente a Ø$tÎr(ØP1(t)).
3.- P1ÞP2 es equivalente a ØP1ÚP2

5.5.2.3. Seguridad de expresiones.

Una relación en el álgebra relacional de tuplas puede generar una relación infinita. Supóngase:
{t|Ø(tÎpréstamo)}
Existe un número infinito de tuplas que no están en préstamo. La mayoría contienen valores que ni siquiera están en la BD, por lo que no queremos permitir estas expresiones.
Para ayudarnos a definir una restricción introducimos el concepto de dominio de una fórmula relacional de tuplas, P. El dominio de P, representado dom(P) es el conjunto de todos los valores referenciados por P. así, como el conjunto de todos los valores que aparecen en una o más relaciones cuyos nombres aparecen en P. Por ejemplo, dom(tÎpréstamo Ù t[cantidad]>1200) es el conjunto de todos los valores que aparecen en préstamo mayores que 1200, y dom(Ø(tÎpréstamo)) es el conjunto de todos los valores que no aparecen en préstamo.
Decimos que una expresión {t|P(t)} es segura si todos los valores que aparecen en el resultado son valores de dom(P).

5.5.2.4. Poder expresivo de los lenguajes.

El cálculo relacional de tuplas restringido es equivalente en poder expresivo al álgebra relacional. Esto quiere decir que para cada expresión en el álgebra relacional existe una relación equivalente en el álgebra relacional de tuplas.

5.5.3. El cálculo relacional de dominios.

Existe una segunda forma de cálculo relacional llamada cálculo relacional de dominios. Esta forma usa variables de dominio que toman valores del dominio de un atributo más que valores de una tupla completa, y esta intimamente relacionado con el cálculo relacional de tuplas.

5.5.3.1 Definición formal.

Una expresión en el cálculo relacional de dominios es de la forma {<x1, x2, …, xn>|P(x1, x2, …, xn)}, donde x1, x2, …, xn reprsentan variables de dominio. P representa una fórmula compuesta por átomos, siendo un átomo una expresión con una de estas formas:

1.- <x1, x2, …, xn>Îr, donde r es una relación de n atributos, y x1, x2, …, xn son variables de dominio o constantes de dominio.

2.- xQy, donde x e y son variables de dominio, y Q es un operador de comparación (<, £, =, ³, >). Es requisito de los atributos x e y tengan dominios que puedan compararse.

3.- xQc, donde x es una variable de dominio, Q es un operador de comparación, y c es una constante en el dominio del atributo para el cual x es una variable de dominio.

Las fórmulas se construyen a partir de átomos usando las siguientes reglas:

1.- Un átomo es una fórmula.
2.- Si P1 es una fórmula, entonces también lo son ØP1 y (P1).
3.- Si P1 y P2 son fórmulas, entonces también lo son P1ÙP2, P1ÚP2 y P1ÞP2.
4.- Si P1(x) es una fórmula en x, donde x es una variable de dominio, entonces $x(P1(x)) y "x(P1(x)) también son fórmulas.
Para abreviar la notación escribimos $a,b,c(P(a,b,c)) en lugar de $a($b($c(P(a,b,c))))

5.5.3.2 . Consultas ejemplo.

Ahora damos consultas en el cálculo relacional de dominios para los ejemplos ya considerados. Obsérvese el parecido con las correspondientes del cálculo relacional de tuplas.
1.- Encontrar nombre_sucursal, número_préstamo, nombre_cliente y cantidad de préstamos > 1200.

{<b,l,c,a>|<b,l,c,a>Îpréstamo Ù a>1200}

2.- Encontrar a todos los clientes que tienen un préstamo en Perrydge y las ciudades en que viven.

{<c,x>|$b,l,a(<b,l,c,a>Îpréstamo Ù b=”Perrydge” Ù $y(<c,y,x>Îcliente))}

3.- Encontrar a todos los clientes que tienen una cuenta en todas las sucursales de Brooklyn.

{<c>|"x,y,z(Ø<x,y,z>Îsucursal Ú z¹”Brooklyn” Ú ($a,n(<x,a,c,n>Îdepósito)))}

Interpretamos esta expresión como el conjunto de todas las tuplas nombre_cliente <c> tal que para todas las tuplas (nombre_sucursal, activo, ciudad_sucursal) <x,y,z>, al menos una de las siguientes declaraciones es verdadera.

<x,y,z> no es una tupla de la relación sucursal, y por tanto no corresponde a una sucursal de Brooklyn.

z no tiene el valor Brooklyn.
El cliente c tiene una cuenta (con número de cuenta a y saldo n)en la sucursal x.

5.5.3.3 Seguridad de las expresiones.

Una expresión como {<b,c,l,a>|Ø(<b,c,l,a>Îpréstamo)} no es segura porque permite valores en el resultado que no están en el dominio de la expresión.
Para el cálculo relacional de dominios debemos preocuparnos también por la forma de las fórmulas dentro de las cláusulas existe y para todos.
En el cálculo relacional de tuplas restringimos cualquier variable cuantificada existencialmente a moverse sobre una relación específica. Ahora añadimos reglas a la definición de seguridad para tratar con casos como el ejemplo anterior. Decimos que una expresión

{<x1, x2, …, xn>|P(x1, x2, …, xn)}

Es segura si se cumplen todas las condiciones siguientes:

1.- Todos los valores que aparecen en tuplas de la expresión son valores de dom(P).

2.- Para cada subfórmula “existe” de la forma $x(P1(x)), la subfórmula es verdadera si, y sólo si, existe un valor x de dom(P1) tal que P1(x) es verdadero.

3.- Para cada subfórmula “para todos” de la forma "x(P1(x)), la subfórmula es verdadera si, y sólo si, P1(x) es verdadero para todos los valores de x de dom(P1).
El propósito de estas reglas adicionales es asegurar que podemos probar las subfórmulas “para todos” y “existe” sin tener que probar infinitas posibilidades.

5.5.4. Modificación de la base de datos.

Las modificaciones de la BD se expresan usando el operador asignación. Las asignaciones se hacen a relaciones ya existentes en la BD.

5.5.4.1. Eliminación.

Una solicitud de eliminación se expresa muy parecida a una consulta. Sin embargo, en vez de presentar tuplas al usuario, quitamos las tuplas seleccionadas de la BD. Sólo podemos eliminar tuplas completas, no únicamente valores de determinados atributos. En álgebra relacional, una eliminación se expresa:

r ß r - E

donde r es una relación y E es una consulta del álgebra relacional. Ejemplo, eliminar todas las cuentas de Smith:

depósito ß depoósito - snombre_cliente=”Smith” (depósito)

5.5.4.2. Inserción.

Para insertar datos en una relación, bien especificamos la tupla que se va a insertar o escribimos una consulta cuyo resultado sea un conjunto de tuplas que se va a insertar. En este último caso, los valores de los atributos para las tuplas insertadas deben ser miembros del dominio de los atributos, y las tuplas insertadas deben ser del mismo orden.

En álgebra relacional una inserción se representa:
r ß r È E

La inserción de una sola tupla se expresa haciendo que E sea una relación constante que contiene una tupla.

Supóngase que queremos insertar el hecho de que Smith tiene 1200 dólares en la cuenta 9732 de Perrydge.

depósito ß depósito È {(“Perryde”, 9732, “Smith”, 1200)}

Podríamos querer insertar tuplas basadas en el resultado de una consulta. Supóngase que queremos proporcionar a todos los clientes con préstamos en Perrydge una cuenta de ahorros de 200 dólares. EL número de préstamo servirá como número de cuenta.

r1 ß (snombre_sucursal=”Perrydge” (préstamo))
r2 ß Õnombre_sucursal,número_préstamo, nombre_cliente (r1)
depósito ß depósito È (r2 x {(200)})

5.5.4.3. Actualización.

En ciertas ocasiones podemos querer cambiar un valor en una tupla sin cambiar todos los valores de la tupla. Para ello usamos el operador actualización d que tiene la forma:
d AßE (r)
donde r es una relación con atributo A al cual se le asigna el valor de la expresión E. La expresión E es cualquier expresión aritmética que implica constantes y atributos de planificación de relación r.

Sopóngase que estamos pagando el 5% de interés a todas las cuentas:
d saldoßsaldo * 1,05 (depósito)

La sentencia anterior se aplica una vez a cada tulpa de depósito. Pero ahora supóngamos que queremos dar el 6% a cuentas de más de 10000 dólares y el 5% al resto:

d saldoßsaldo * 1,06 (ssaldo>10.000 (depósito)
d saldoßsaldo * 1,05 (ssaldo>10.000 (depósito)

Es importante el orden en que aplicamos las expresiones de actualizar. Si cambiamos el orden, una cuenta cuyo saldo estuviera justo por debajo de 10000 recibiría el 11,3% de interés.

5.5.4.4. Vistas.

En los ejemplos expuestos hemos operado en el nivel del modelo conceptual, es decir, hemos supuesto que la colección de relaciones que se nos dan son las relaciones reales almacenadas en la BD.
No es conveniente que todos los usuarios vean el modelo conceptual completo. Las consideraciones de seguridad pueden requerir que se escondan ciertos datos al usuario.
Cualquier relación que no es parte del modelo conceptual, pero se hace visible al usuario como una relación virtual se llama vista.
Puesto que las relaciones reales en el modelo conceptual pueden modificarse, generalmente no es posible almacenar una relación correspondiente a una vista. Una vista debe volverse a calcular para cada consulta que se refiere a ella.
Cuando se define una vista, el DBMS debe almacenar la definición de la vista. Así, la definición de vista no es una expresión del álgebra relacional. Más bien, hace que se guarde una expresión que se va a sustituir en consultas utilizando la vista.

5.5.4.5. Definición de vista.

Una vista se define usando la sentencia **create view** de la forma create view v as <expresión de consulta>
donde <expresión de consulta> es cualquier expresión legal de consulta en álgebra relacional. El nombre de la vista se representa por v.

Considerése la vista que consta de sucursales y sus clientes:

create view todos_clientes as Õnombre_sucursal, nombre_cliente (depósito) È Õnombre_sucursal, nombre_cliente (préstamo)

Una vez que se ha definido la vista, el nombre de la vista puede usarse para referirse a la relación virtual que genera la vista. Los nombres de vistas pueden aparecer en cualquier lugar que pueda aparecer el nombre de una relación.

5.5.4.6. Actualización por medio de vistas y valores nulos.

Aunque las vistas son una herramienta útil para las consultas, presentan valores importantes si las actualizaciones, inserciones o eliminaciones se expresan usando vistas. La dificultad es que una modificación de la BD expresada en

términos de vistas debe traducirse en una modificación de las relaciones reales en el modelo conceptual de la BD.
Para ilustrar el problema considérese la vista info_préstamo, definida create view info_préstamo as Õnombre_sucursal, número_préstamo, nombre_cliente (préstamo) Puesto que permitimos que un nombre de vista aparezca en cualquier lugar, podríamos escribir

info_préstamo ß info_préstamo È {(“Perrydge”, 3, “Ruth”)}

Esta inserción debe estar representada por una inserción en la relación préstamo, ya que préstamo es la relación real. Sin embargo, para insertar una tupla en préstamo, debemos tener algún valor para cantidad. Existen dos enfoques:

1.- Rechazar la inserción y devolver un mensaje de error al usuario.
2.- Insertar una tupla (“Perrydge”, 3, “Ruth”, null) en la relación préstamo.

El simbolo null representa un valor_nulo o valor_en_lugar_de. Significa que el valor es desconocido o no existe. Todas las comparaciones que implican null son falsas por definición.

El problema general de la modificación de BD por medio de vistas ha sido el tema de investigaciones sustanciales. Otro área de interés relacionado con vistas es el modelo de relación universal. En este modelo se da al usuario una vista que consta de una relación. Esta relación es el producto natural de todas las relaciones en la BD relacional real.

La principal ventaja de este modelo es que los usuarios no necesitan preocuparse de recordar que atributos están en la relación. Así, la mayor parte de las consultas son más fáciles de formular en un DBMS de relación universal que en uno de relación estándar.

Quedan cuestiones sin resolver referentes a modificaciones de BD de relación universal y todavía no se ha llegado a un consenso acerca de cuál es la mejor definición del significado de ciertos tipos complejos de consultas de relación universal.

5.6 Actividades Completarías

1. Realizar las siguientes consultas utilizando álgebra relacional con el siguiente esquema de bases de datos.

   Esquema de base de datos "**BANCO**"

   SUCURSAL(nomSuc, activo, ciudadSuc)
   CLIENTE(nomCli, calle, ciudadCli)
   BANQUERO_PERSONAL(nomCli, nomBanq)
   CUENTA(nomSuc, numCuenta, nomCli, saldo)
   PRESTAMO(nomSuc, numPrestamo, nomCli, cantidad)

Enunciados de consulta:

- Datos de los préstamos de la sucursal "Ronda Sur".
- Nombres de los clientes cuyo banquero personal es su tocayo.
- Nombres de los clientes que viven en la misma calle y ciudad que "Maldonado".
- Nombres de los clientes de la sucursal "Ronda Sur".
- Clientes del banquero "Palao" y ciudades en las que viven.
- Nombres de clientes con cuentas en la sucursal "Ronda Sur" pero sin préstamos allí.
- Nombres de los clientes con préstamo y cuenta en la sucursal "Ronda Sur".
- Nombre de los clientes que tienen algún préstamo en el banco y las ciudades en donde viven.
- Activo y nombre de todas las sucursales con cuentas de clientes que vivan en Murcia.
- Nombres de los clientes con cuenta en todas las sucursales que están en Murcia.

2. Diseñar una base de datos que recoja la organización de una Universidad. Se considera que:

   - Los departamentos pueden estar en una sola facultad o ser interfacultativos, agrupando en este caso cátedras que pertenecen a facultades distintas.
   - Una cátedra se encuentra en un único departamento.
   - Una cátedra pertenece a una sola facultad.
   - Un profesor está siempre asignado a un único departamento y adscrito a una o varias cátedras, pudiendo cambiar de cátedra, pero no de

departamento. Interesa la fecha en que un profesor es adscrito a una cátedra.
- Existen áreas de conocimiento, y todo departamento tendrá una única área de conocimiento.

3. Se desea diseñar una base de datos para una sucursal bancaria que contenga información sobre los clientes, las cuentas, las sucursales y las transacciones producidas. Construir el modelo E/R teniendo en cuenta las siguientes restricciones:

   - Una transacción viene determinada por su número de transacción, la fecha y la cantidad.
   - Un cliente puede tener muchas cuentes.
   - Una cuenta puede tener muchos clientes.
   - Una cuenta sólo puede estar en una sucursal.

4. Dadas las relaciones siguientes:

   HOMBRES(NOMH, EDAD)
   Significado: Cada fila representa a un hombre, cuyo nombre es NOMH y su edad en años es EDAD.

   MUJERES(NOMM, EDAD)
   Significado: Cada fila representa a una mujer, cuyo nombre es NOMM y su edad en años es EDAD.

   HSIM(NOMH, NOMM)
   Significado: El hombre NOMH cae simpático a la mujer NOMM

   MSIM(NOMH, NOMM)
   Significado: La mujer NOMM cae simpática al hombre NOMH

   MATRIM(NOMH, NOMM)
   Significado: La pareja NOMH y NOMM están casados

   Escribir las sentencias necesarias para responder a las preguntas siguientes
   a) Hallar las parejas de hombres y mujeres que se caen mutuamente simpáticos
   b) Hallar los matrimonios en los que ambos esposos se caen mutuamente simpáticos.
   c) Hallar las mujeres casadas a quienes no cae simpático su marido.

5. Sean las relaciones siguientes:

SOCIO (AFICIONADO, VIDEOCLUB)
Significado: AFICIONADO es SOCIO de VIDEOCLUB

GUSTA (AFICIONADO, PELÍCULA)
Significado:PELÍCULA GUSTA a AFICIONADO

VIDEOTECA (VIDEOCLUB, PELÍCULA)
Significado: VIDEOCLUB dispone en su VIDEOTECA de PELÍCULA

Escribir las sentencias necesarias para responder a las preguntas siguientes:

a) Videoclubes que disponen de alguna película que le guste a José Pérez
b) Aficionados que son socios al menos de un videoclub que dispone de alguna película de su gusto
c) Aficionados que no son socios de ningún videoclub donde tengan alguna película de su gusto.

6. Sean las relaciones siguientes:

PRO(NP, NOMP, CIUDADP)
Significado: Cada fila representa un proveedor, cuyo identificador es NP, su nombre NOMP y habita en la ciudad CIUDADP.

ART(NA, DESA, COLOR, TALLA)
Significado: Cada fila representa un artículo, cuyo identificador es NA y su descripción es DESA.

FAB(NF, NOMF, CIUDADF)
Significado: Cada fila representa un fábrica cuyo identificador es NF, su nombre es NOMF y esta situada en la ciudad CIUDADF.

PED(NP, NA, NF, CANTIDAD)
Significado: Cada fila representa un pedido del artículo NA al proveedor NP para la fábrica NF.

Escribir las sentencias necesarias para responder a las preguntas siguientes:

a) Hallar los nombres de las fábricas situadas en Madrid
b) Proveedores que suministran a la fábrica F1
c) Nombre de las fábricas a las que suministra el proveedor P1
d) Colores de los artículos suministrados por el proveedor P1
e) Artículos suministrados a las fábricas de Madrid
f) Artículos suministrados por proveedores en cuya ciudad hay alguna fábrica

g) Fábricas que usan al menos algún artículo suministrado por el proveedor P1
h) Parejas de ciudades tales que un proveedor de la primera abastece a una fábrica de la segunda
i) Proveedores que suministran un mismo artículo, al menos, a todas las fábricas.
j) Fábricas que usan, al menos, todos los artículos suministrados por el proveedor P1
k) Fábricas abastecidas por el proveedor P1 con todos los artículos que este suministra

CONSULTAS WEB

- Se puede obtener información sobre el Diseño de bases de datos relacionales en :
- http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x332.html
- http://www.tramullas.com/nautica/documatica/2-7.html
- http://atenea.udistrital.edu.co/profesores/jdimate/basedatos1/tema3_1.htm
- http://dis.um.es/~barzana/Informatica/IAGP/IAGP_BD_Relacional.html

UNIDAD 3. NORMALIZACION Y EL LENGUAJE SQL

CAPITULO 6. NORMALIZACION

6.1 Normalización

La normalización es una técnica, desarrollada inicialmente por E.F. Codd en 1972, para diseñar la estructura lógica de una base de datos en el modelo relacional La normalización es un proceso en el cual se va comprobando el cumplimiento de una serie de reglas, o restricciones, por parte de un esquema de relación; cada regla que se cumple aumenta el grado de normalización del esquema de relación; si una regla no se cumple, el esquema de relación se debe descomponer en varios esquemas de relación que sí la cumplan por separado

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.

6.2 Formas Normales

Un esquema de relación está en una determinada forma normal si satisface un cierto conjunto de restricciones.

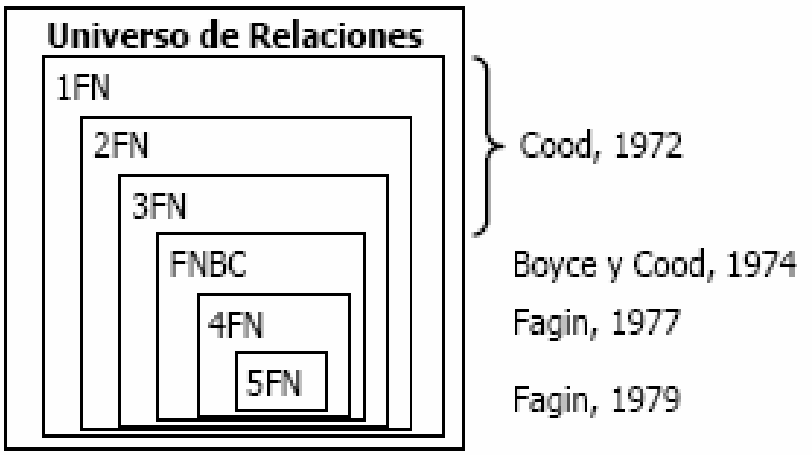


Figura 6.1 Formas Normales

El modelo relacional sólo requiere un conjunto de esquemas de relación en primera forma normal. Las restantes formas normales son opcionales, para evitar anomalías de actualización, es recomendable llegar al menos hasta la tercera forma normal o, mejor aún, hasta la forma normal de Boyce-Codd

6.2.1 Primera forma normal (1FN)

Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna.

Un dominio es atómico si se considera que los elementos del dominio son unidades indivisibles.

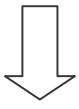
En muchos dominios en los que las entidades tienen una estructura compleja, la imposición de la representación en primera forma normal representa una carga innecesaria para el programador de las aplicaciones, que tiene que escribir código para convertir los datos a su forma atómica. Los sistemas modernos de bases de datos soportan muchos tipos de valores no atómicos

Si una relación no está en 1FN, hay que eliminar de ella los grupos repetitivos. Un grupo repetitivo será el atributo o grupo de atributos que tiene múltiples valores para cada tupla de la relación. Hay dos formas de eliminar los grupos repetitivos. En la primera, se repiten los atributos con un solo valor para cada valor del grupo repetitivo. De este modo, se introducen redundancias ya que se duplican valores, pero estas redundancias se eliminarán después mediante las restantes formas normales. La segunda forma de eliminar los grupos repetitivos consiste en poner cada uno de ellos en una relación aparte, heredando la clave primaria de la relación en la que se encontraban.

Un conjunto de relaciones se encuentra en 1FN si ninguna de ellas tiene grupos repetitivos.

Ejemplo Pedidos

Esquema_ordenes = (id_orden, fecha, id_cliente, nombre_cliente, provincia, numero_item, descripcion_item, cantidad, precio)

1 FN 

Esquema_ordenes = (id_orden, fecha, id_cliente, nombre_cliente, provincia)
Esquema_pedidos = (id_orden, numero_item, descripcion_item, cantidad, precio)

Vamos a usar un ejemplo para ilustrar el proceso de normalización. Supongamos que tenemos información sobre pedidos, con los siguientes atributos:

Num_Pedido, Fecha, Num_Proveedor, Nombre_Proveedor, Dirección_Proveedor, Num_Producto, Descripción, Precio, Cantidad, Precio Total_Producto, Precio_Total_Pedido

Considerando todos estos atributos como una relación universal que contiene toda la base de datos, hay que buscar una clave para dicha relación. En este caso, tenemos información de pedidos, con datos únicos para cada pedido y datos que se repiten dentro de un pedido. Parece lógico considerar que es una tabla de pedidos. Como para cada pedido hay un Num_Pedido único, ésta será nuestra clave inicial.

Una vez elegida la clave inicial, tenemos que ver si todos los atributos restantes tienen dominios que no sean conjuntos.
Los atributos Fecha, Num_Proveedor, Nombre_Proveedor, Direccion_Proveedor y Precio_Total_Pedido son únicos para cada pedido.

Los atributos Num_Producto, Descripcion, Precio, Cantidad y Precio_Total_Producto no son únicos para cada pedido, sino que para cada pedido tendrán tantos valores como líneas de productos haya incluidas en el pedido. Por lo tanto, la relación Pedidos que incluye a todos los atributos no está en 1FN, y es necesario realizar una transformación para obtener relaciones que estén en 1FN.
Tenemos dos métodos alternativos:

- Descomponer la relación que no cumple la 1FN en dos, pasando los atributos multivaluados a una nueva tabla, junto con la clave respecto a la que se repiten.
- Añadir a la clave de la relación que no cumple la 1FN los atributos necesarios para identificar realmente cada fila de la tabla, considerando que los atributos ya no son multivaluados.

En principio, es más aconsejable usar el primer método, puesto que avanza en el sentido de ir descomponiendo la base de datos en tablas.

Aplicando el caso a nuestro ejemplo, con el primer método tendríamos:

Pedidos (Num_Pedido, Fecha, Num_Proveedor, Nombre_Proveedor, Direccion_Proveedor, Precio_Total_Pedido)

Lineas (Num_Pedido, Num_Producto, Descripcion, Precio, Cantidad, Precio Total_Producto)

Observemos que la clave primaria de Lineas está compuesta por el número de pedido y el número de producto, porque se supone que en un pedido no aparece dos veces el mismo producto.

Con el método b), no se descompondría la tabla, sino que simplemente su clave primaria sería Num_Pedido y Num_Producto, y los atributos ya no serían multivaluados.

En cualquiera de ambos casos, las relaciones obtenidas sí cumplen la 1FN. Una relación puede tener varios grupos de atributos multivaluados, e incluso grupos anidados, que habrá que ir descomponiendo por niveles.

6.2.2 Segunda forma normal (2FN)

Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN. Las relaciones que no están en 2FN pueden sufrir anomalías cuando se realizan actualizaciones.

Para pasar una relación en 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).

Una vez que tenemos un esquema de relaciones en 1FN, observamos que seguimos teniendo posibilidad de anomalías. En nuestro ejemplo, tenemos repetida la información de proveedores y productos para cada pedido, y no se pueden introducir productos sin pedido.

La relación Pedidos está en 2FN, puesto que no hay claves candidatas distintas de la primaria, y mirando los atributos de la clave primaria, vemos que solo tiene un atributo, por lo que no puede haber dependencias parciales.

En cambio, la relación Líneas no está en 2FN: tampoco hay claves candidatas distintas de la primaria, y vemos que para un número de producto, hay una única descripción y precio del producto, luego Descripción y Precio dependen funcionalmente sólo de Num_Producto, que es una parte de la clave primaria de la tabla.

La transformación a realizar es la siguiente: Llevar los atributos no primos que dependen de parte de la clave primaria a una nueva tabla, y añadirles la parte de la clave primaria de la que dependen.

En nuestro caso, nos llevaremos Descripción y Precio de la tabla Líneas, y les añadiremos Num_Producto, creando una nueva tabla llamada Productos:

Lineas (Num_Pedido, Num_Producto, Cantidad, Precio_Total_Producto)

Productos (Num_Producto, Descripción, Precio)

Por tanto, el esquema de nuestra base de datos consta ya de tres relaciones: Pedidos, Líneas y Productos.

En otros casos, puede haber varios atributos o grupos de ellos que dependan de partes distintas de la clave primaria, creándose una tabla para cada grupo correspondiente a una parte de la clave primaria. No hay casos de tener que aplicar esta transformación de forma anidada.

6.2.3 Tercera forma normal (3FN)

Una relación está en tercera forma normal si, y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La *dependencia funcional transitiva* si existe un tercer conjunto de atributos *Z* disjunto de ellos para el que existe una dependencia funcional *X -> Z y Z -> Y*.

En una relación en tercera forma normal, ningún atributo no primo puede depender transitivamente de la clave primaria.

Aunque las relaciones en 2FN tienen menos redundancias que las relaciones en 1FN, todavía pueden sufrir anomalías frente a las actualizaciones. Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

Aplicado esto al ejemplo que estamos siguiendo en este tema, vemos que la relación Pedidos no está en tercera forma normal, ya que existe una dependencia funcional transitiva entre Nombre_Proveedor, Dirección_Proveedor y Num_Pedido a través de Num_Proveedor, ya que existe una dependencia funcional {Num_Pedido} -> {Num_Proveedor} y otra {Num_Proveedor} -> {Nombre_Proveedor, Dirección_Proveedor}. Así, el esquema Pedidos se transforma en los esquemas Pedidos y Proveedores siguientes.

Pedidos (Num_Pedido, Fecha, Num_Proveedor, Precio_Total_Pedido)

Proveedores (Num_Proveedor, Nombre_Proveedor, Direccion_Proveedor)

A continuación se muestra de forma resumida qué comprueba cada forma normal y qué proceso se debe llevar a cabo para hacer que un esquema esté en una forma normal determinada.

Primera forma normal (1FN): Una relación no debe contener atributos no atómicos o multivaluados ni debe contener relaciones anidadas. Si una relación no cumple

esta condición, es decir, no está en 1FN, se crean nuevas relaciones con los atributos multivaluados o relaciones anidadas.

Segunda forma normal (2FN): Una relación que tiene una clave primaria compuesta no debe contener atributos no primos que dependan funcionalmente de parte de la clave. Por tanto, todas las relaciones con claves simples están en 2FN. Su una relación no está en 2FN se crea una nueva relación con cada clave parcial y el conjunto de atributos que dependen funcionalmente de ella. No obstante, hay que tener en cuenta que debe seguir existiendo una relación con la clave primaria original y los atributos que dependen totalmente de ella.

Tercera forma normal (3FN): Una relación no debe tener atributos no primos que dependan funcionalmente de atributos no primos, es decir, no deberían existir dependencias funcionales transitivas por parte de los atributos no primos. Si una relación no está en 3FN se crea una nueva relación con los atributos no primos que determinan funcionalmente los otros atributos no primos.

Las definiciones de 2FN y 3FN pueden generalizarse para tener en cuenta las dependencias parciales o transitivas de todas las claves de un esquema de relación.

6.2.4 Forma normal de Boyce-Codd (BCFN)

Una relación está en la forma normal de Boyce-Codd si, y sólo si, todo determinante es una clave candidata.

La 2FN y la 3FN eliminan las dependencias parciales y las dependencias transitivas de la clave primaria. Pero este tipo de dependencias todavía pueden existir sobre otras claves candidatas, si éstas existen. La BCFN es más fuerte que la 3FN, por lo tanto, toda relación en BCFN está en 3FN.

La violación de la BCFN es poco frecuente ya que se da bajo ciertas condiciones que raramente se presentan. Se debe comprobar si una relación viola la BCFN si tiene dos o más claves candidatas compuestas que tienen al menos un atributo en común.

Consideremos este ejemplo. Tenemos la relación IMPARTE, cuyo esquema es (Alumno, Asignatura, Profesor). Cada Profesor imparte una sola asignatura, por lo que hay una DF Profesor -> Asignatura. Esta relación está en 1FN, 2FN y 3FN, pero es claramente redundante, ya que decimos quién es el profesor de cada asignatura para cada alumno. No está en FNBC, ya que hay un determinante (Profesor) que no es clave candidata.

La descomposición correcta en dos tablas es la que lleva a una nueva tabla el atributo del miembro izquierdo de la dependencia funcional, y le añade el miembro

derecho. En nuestro ejemplo, crearíamos la relación PROFESOR(Profesor, Asignatura), y la relación original quedaría como IMPARTE(Alumno, Profesor).

6.2.5 Otras formas normales

En esta sección presentaremos el concepto de dependencia multivaluada, en el que se basa la 4FN, y el concepto de dependencia de producto, en el que se basa la 5FN.

Una dependencia multivaluada (DMV) X ->> Y, especificada en el esquema de relación R, impone la siguiente restricción sobre cualquier estado r de R: si existen dos tuplas t_1 y t_2 en r tales que $t_1[X] = t_2[X]$, entonces deberán existir también dos tuplas t_3 y t_4 que verifiquen lo siguiente, donde Z representa el resto de los atributos de R:

- $t_1[X] = t_2[X] = t_3[X] = t_4[X]$
- $t_3[Y] = t_1[Y]$ y $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ y $t_4[Z] = t_1[Z]$

Si X ->> Y, entonces también se cumple que X ->> Z, lo que se denota como X ->> Y | Z.

Podemos ver un ejemplo con esta relación BIBLIOGRAFIA (Libro, Asignatura, Profesor), que contiene los libros recomendados por los profesores que imparten asignaturas. Resulta que todos los profesores que imparten una asignatura recomiendan el mismo conjunto de libros, un libro puede ser recomendado para varias asignaturas, y un profesor puede impartir varias asignaturas. Por lo tanto, si se asigna un nuevo profesor a una asignatura, hay que insertar tantas filas como libros recomendados para ella, y si se recomienda un nuevo libro, hay que insertar tantas filas como profesores de la asignatura.

Esta relación presenta dependencias multivaluadas, ya que dada una asignatura, aparece la combinación de todos los libros recomendados con todos los profesores.

Formalmente, diremos que una relación está en 4FN si para cada dependencia multivaluada no trivial, el determinante de la misma es clave candidata.

Para resolver el problema, podemos descomponer esta relación en dos:

LIBROS (Libro, Asignatura)

PROFESORES (Asignatura, Profesor)

La primera relación contiene los atributos implicados en los dos miembros de la dependencia multivaluada, y la segunda los atributos originales eliminando la parte derecha de la dependencia multivaluada. Esta descomposición es una descomposición sin pérdida, ya que si combinamos ambas tablas por su atributo común, recuperamos la información adicional, sin tuplas incorrectas.

Existen situaciones en las que una relación puede no presentar dependencias multivaluadas y seguir teniendo redundancia evitable. Este es el caso cuando existen dependencias de producto o reunión. Un esquema de relación tiene una dependencia de producto cuando es posible descomponerla sin pérdida en más de dos relaciones, pero no en dos.

Supongamos que tenemos la misma relación BIBLIOGRAFIA del caso anterior, pero que en este caso cada profesor recomienda libros distintos en la misma asignatura. En este caso, no existe una dependencia multivaluada, ya que si hacemos la descomposición y luego la combinación, tendremos filas incorrectas. Pero existe redundancia, ya que estamos repitiendo cada asignatura impartida por cada profesor para cada libro recomendado. Este es el tipo de redundancia que intenta evitar la 5FN.

Decimos que una relación está en 5FN está en 4FN y para cada dependencia de producto no trivial, todas las relaciones en las que se descompone son claves candidatas de la propia relación.

En el ejemplo anterior, BIBLIOGRAFIA presenta una dependencia de producto con tres relaciones, y ninguna de ellas es clave candidata, luego no está en 5FN. Para llegar a 5FN, necesitamos descomponer BIBLIOGRAFIA en tres relaciones:

LIBROS (Libro, Asignatura)

PROFESORES (Asignatura, Profesor)

RECOMENDADOS (Libro, Profesor)

Al combinar estas tres relaciones, obtendremos la relación original.

6.3 Actividades Complementarias

1. ¿Qué es la normalización y porque es importante su aplicación en el diseño de bases de datos?

2. ¿Qué características son importantes en el momento de elegir una clave primaria y una clave foránea en una base de datos?

3. ¿Qué significa integridad de la información?

4. Se desea diseñar una base de datos que recoja información sobre la Vuelta Ciclista a España. Los supuestos semánticos que se utilizaran son los siguientes (entre paréntesis aparecen los nombre de algunos campos que se desean tener en la bases de datos)

    - Cada edición de la vuelta viene identificada por un código (CE), además, interesa conservar el año (A) en que ha tenido lugar dicha edición.
    - Todas las ediciones tienen las mismas etapas (E), aunque puede variar su longitud. Se quiere guardar la longitud de cada etapa (KM) de una edición.
    - Cada equipo (EQ) tiene un nombre (N EQ), también queremos guardar su fecha de fundación (F EQ).
    - Todos los equipos tienen un único patrocinador (PA), el patrocinador puede variar de un año a otro, pero no dentro del mismo año.
    - Queremos tener los equipos que han participado en cada edición con el puesto (PE) conseguido por cada uno de ellos (dos equipos no pueden conseguir el mismo puesto en una misma edición).
    - Los corredores (C), interesa guardar también su nombre (N C) y fecha de nacimiento (F C), pueden cambiar de equipo de un año a otro, pero nunca dentro del mismo.
    - En cada etapa de una edición un corredor consigue un único puesto (P) y un puesto lo consigue un único corredor.
    - Nos interesa guardar los tiempos (T) conseguidos por cada corredor en cada etapa de una edición. En cada etapa de una edición, un puesto implica un tiempo determinado.

Se pide realizar el modelo entidad/relación de la base de datos y la normalización respectiva, utilizando los atributos que se dan en el enunciado.

CAPITULO 7 LENGUAJE DE CONSULTA SQL

7.1. Introducción

En el tema anterior hemos estudiado las operaciones del álgebra relacional y el cálculo relacional, imprescindibles para entender las consultas que podemos realizar en una base de datos relacional. En general, el álgebra relacional está clasificada como un lenguaje de consulta formal procedimental, en el que el usuario indica cómo (en qué orden) se deben especificar las operaciones de la consulta para obtener el resultado deseado, mientras que el cálculo relacional es no procedimental. En este tema vamos a estudiar dos lenguajes de consulta comerciales, como son el SQL, basado en álgebra relacional y cálculo relacional de tuplas, y el QBE, basado en cálculo relacional de dominios. Los SGBDRs comerciales cuentan con una interfaz de lenguaje declarativo de alto nivel en el que se especifica la consulta en SQL (Lenguaje de Consulta Estructurado, del inglés Structured Query Language) o QBE, y el propio SGBD es el que se encarga de realizar las optimizaciones necesarias para ejecutar la consulta.

7.2 SQL

SQL fue diseñado e implementado por IBM Research y se ha convertido en un estándar para las bases de datos relacionales. Además, es posible que los programadores de aplicaciones escriban sentencias SQL en las aplicaciones de bases de datos, dando lugar a lo que se conoce como *SQL embebido*. SQL es un lenguaje de consulta completo, y se puede utilizar como DDL o como DML, ya que cuenta con sentencias para • Definición de datos • Consulta de datos • Actualización de datos Además, SQL cuenta con mecanismos para la definición de vistas de la base de datos, creación y eliminación de índices y para la incorporación de sentencias SQL en lenguajes de programación de propósito general.

7.2.1. Definición de datos en SQL

SQL utiliza los términos de tabla (table), fila (row) y columna (column) en lugar de los términos relación, tupla y atributo del álgebra relacional. Las órdenes SQL para la definición de datos son CREATE (crear), ALTER (modificar) y DROP (eliminar). Antes de estudiar estas operaciones, analizaremos brevemente los tipos de dominios disponibles y los conceptos de esquema y catálogo.

7.2.1.1. Tipos de dominios

SQL tiene distintos tipos de datos disponibles, entre los que caben destacar aquellos para la definición de caracteres numéricos, cadenas, fecha y hora. Los tipos de datos numéricos permiten la definición de números enteros (INTEGER o INT, SMALLINT) o reales (FLOAT, REAL, DOUBLE PRECISION).

Incluso es posible definir números reales especificando la precisión en forma de número de decimales (DECIMAL(i,j), donde *i* especifica la precisión, y *j* el número de dígitos que hay detrás del punto decimal).

Los tipos de datos cadena pueden ser de longitud fija (CHAR(n) o CHARACTER(n), donde *n* representa el número de caracteres) o de longitud variable (VARCHAR(n), donde *n* es el número máximo de caracteres).

En cuanto a los tipos de fecha y hora, las fechas se definen mediante DATE y las horas mediante TIME. Las componentes de DATE son YEAR, MONTH y DAY, y suelen ir en la forma YYYY-MM-DD. Las componentes de TIME son HOUR, MINUTE y SECOND, y suele ir en la forma HH:MM:SS.

Si queremos definir un dominio explícitamente (como un tipo en un lenguaje de programación), podemos usar CREATE DOMAIN *dominio tipo_datos*. Posteriormente podremos usar este dominio al definir los atributos de las tablas.

7.2.1.2. Esquemas y catálogos

Las primeras versiones de SQL no contemplaban el concepto de esquema de base de datos relacional, y todas las tablas (relaciones) se consideraban parte del mismo esquema. El concepto de esquema SQL se incorporó en SQL2 con el fin de agrupar tablas y otros elementos pertenecientes a la misma aplicación.

Un esquema SQL se identifica con un nombre de esquema y consta de:

- Un *identificador de autorización* que indica al usuario o la cuenta que es propietario del esquema.

- Los *descriptores* de cada elemento del esquema. Entre estos elementos se encuentran las tablas, vistas, dominios y autorizaciones). Los esquemas se crean mediante la instrucción CREATE SCHEMA, que puede contener las definiciones de todos los elementos del esquema. También podemos asignar un nombre y autorización al esquema de forma que se puedan definir sus elementos más adelante.

Por ejemplo, esta instrucción crea un esquema denominado BANCO, cuyo propietario es el usuario con autorización MLOPEZ: CREATE SCHEMA BANCO AUTHORIZATION MLOPEZ; Además del concepto de esquema, SQL2 utiliza el concepto de catálogo, que es una colección con nombre de esquemas SQL.

7.2.1.3. Definición de tablas. Especificación de restricciones

Mediante la orden CREATE TABLE podemos crear cada una de las tablas de nuestra base de datos especificando su nombre, sus atributos y sus restricciones, donde el nombre se puede especificar sin más, o bien después del nombre del

esquema seguido de un punto. CREATE TABLE BANCO.CLIENTE... o CREATE TABLE CLIENTE...

En primer lugar se especifican los atributos, cada uno con su nombre, su dominio (un tipo de datos) y las restricciones en el caso de que las haya (p.e. ser un valor no nulo). A continuación se especifican las restricciones de clave, de integridad de entidades y de integridad referencial.

Aquí vemos un ejemplo de creación de una tabla en SQL:

CREATE TABLE CLIENTES (NOMBRECLI VARCHAR(50) NOT NULL, DNICLI VARCHAR(8) NOT NULL, DOMICILIO VARCHAR(50) NOT NULL, PRIMARY KEY (DNICLI) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE CUENTAS (NUMEROCTA VARCHAR(10) NOT NULL, SALDO DECIMAL(12,2) NOT NULL, NOMBRESUC VARCHAR(50) NOT NULL, PRIMARY KEY (NUMEROCTA) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE CTACLI (DNICLI VARCHAR(8) NOT NULL, NUMEROCTA VARCHAR(20) NOT NULL, PRIMARY KEY (DNICLI, CTACLI), FOREIGN KEY (DNICLI) REFERENCES CLIENTES(DNICLI), FOREIGN KEY (NUMEROCTA) REFERENCES CUENTAS(NUMEROCTA));

La forma más sencilla de especificar en la definición de una tabla que un atributo o un conjunto de atributos son clave es mediante una declaración PRIMARY KEY en la definición de la tabla. Los atributos que forman la clave irán encerrados entre paréntesis formando una lista separada por comas.

En el ejemplo anterior, en la tabla CLIENTES, DNICLI ha sido definido como clave primaria mediante PRIMARY KEY (DNICLI). Como una clave no puede tomar un valor nulo, los atributos que formen la clave deberán estar definidos como no nulos (NOT NULL). El proceso de definición de claves externas es parecido al de definición de claves primarias salvo que se utiliza FOREIGN KEY en lugar de PRIMARY KEY. No obstante, en la definición de una clave externa también hay que especificar a qué atributo se está haciendo referencia. Eso se hace escribiendo REFERENCES seguido del nombre de la tabla, y entre paréntesis, el nombre del atributo al que se hace referencia. En la tabla CTACLI del ejemplo anterior, DNICLI está definido como una clave externa y hace referencia al atributo DNICLI de CLIENTES. La definición de clave primarias y claves externas especifican restricciones. A estas restricciones se les puede asignar un nombre, el cual debe ser único en todo el esquema. La utilidad de asignar nombres a las restricciones radica en que quedan identificadas para cuando posteriormente se decida modificarlas. Para especificar restricciones basadas en valores, podemos incluir en la definición de un atributo la palabra reservada CHECK, seguida de una condición entre paréntesis, que deberá ser verificada por los valores del atributo. Si la restricción es a nivel de tupla, la cláusula CHECK se introduce en la definición de la tabla, como las restricciones de clave primaria o externa. Las

restricciones CHECK tienen el inconveniente de que cuando hacen referencia a relaciones mediante una subconsulta y hay cambios en las relaciones referenciadas pueden dejar de cumplirse. Para evitar este problema, disponemos de las *aserciones*, restricciones que siempre deben ser verdaderas, y se comprueban cada vez que se modifica cualquiera de las relaciones mencionadas en ellas.

7.2.1.4 Modificación de esquemas

La definición de una tabla se puede modificar mediante la orden ALTER TABLE, y las posibles acciones que se pueden realizar son: • Incorporación o eliminación de una columna (atributo) • Modificación de la definición de una columna • Incorporación o eliminación de las restricciones de una tabla Por ejemplo, si queremos añadir a la tabla CLIENTES la ciudad en la que viven, utilizaremos la orden ALTER TABLE BANCO.CLIENTES ADD CIUDAD VARCHAR(20);

Una vez modificada la definición de la tabla, se completan con los valores correspondientes utilizando la orden UPDATE que veremos más adelante. Para eliminar una columna tendremos que elegir CASCADE o RESTRICT para especificar la forma de eliminación. Si se elige CASCADE, todas las restricciones y vistas creadas a partir de dicha columna se eliminarán automáticamente del esquema. Por el contrario, si se elige RESTRICT sólo se eliminará la columna si no hay ninguna vista ni restricción sobre dicha columna.

Por ejemplo, esta orden elimina el campo que hemos añadido antes:

ALTER TABLE BANCO.CLIENTES DROP CIUDAD CASCADE;

Para modificar la definición de una columna de la tabla utilizaremos ALTER. Por ejemplo, para eliminar la restricción definida sobre la columna DOMICILIO de CLIENTES que impide que haya valores nulos, escribiríamos

ALTER TABLE BANCO.CLIENTES ALTER DOMICILIO DROP NOT NULL;

A la hora de eliminar algo, puede darse el caso de que lo que se quiere eliminar contenga elementos. En SQL es posible eliminar sólo en el caso de que no existan elementos, o bien forzar la eliminación, independientemente de si contiene elementos o no, y propagar los cambios en consecuencia. Para ello se utilizan RESTRICT y CASCADE, respectivamente.

La orden DROP TABLE CLIENTES RESTRICT; elimina la tabla CLIENTES si no hay ninguna restricción definida sobre ella en otro componente del esquema.

La orden DROP SCHEMA BANCO CASCADE; elimina el esquema BANCO de la base de datos y todas sus tablas y demás componentes.

7.2.1.5. El diccionario de datos

Todos los SGBD relacionales gestionan su propio *diccionario de datos* (aunque sólo sea la descripción de las tablas presentes en la base de datos) usando un esquema relacional. Las tablas que contienen información sobre las tablas contienen los *metadatos* y constituyen el *catálogo* de la base de datos. El diccionario de datos podría manipularse a nivel de actualización de datos como otra tabla, pero esto no sería adecuado. Lo que si podemos hacer es *consultarlo* igual que otras tablas de las base de datos. El estándar de diccionario de datos de SQL-2 no es seguido estrictamente en los SGBD relacionales, debido sobre todo a la información sobre cuestiones de estructuras de almacenamiento de la información, pero ofrece una plantilla que suele ser respetada por estos sistemas.

El estándar tiene dos niveles, DEFINITION_SCHEMA, que describe todas las estructuras de la base de datos, e INFORMATION_SCHEMA, que contiene vistas sobre el DEFINITION_SCHEMA y ofrece una interfaz con el diccionario de datos.

Como ejemplo de estas vistas, tenemos DOMAINS, TABLES, VIEWS, o COLUMNS. La vista COLUMNS incluye una fila para cada columna de la base de datos, con información sobre el nombre de la tabla a la que pertenece, su posición dentro de la misma, su dominio o tipo de datos, el valor predeterminado (si lo tiene), si admite nulos o no, y otra información adicional.

7.2.2 Consulta de datos en SQL

7.2.2.1. Estructura básica de una consulta SQL

La estructura básica de una orden SQL consta de tres cláusulas: SELECT, FROM y WHERE.

La cláusula SELECT se corresponde con la operación de proyección del álgebra relacional, y se utiliza para indicar cuáles son los atributos (separados por comas) que se desea que aparezcan en el resultado de la consulta.

La cláusula FROM corresponde a la operación de producto cartesiano del álgebra relacional, y se utiliza para indicar cuáles son las relaciones (separadas por comas) que participan en la consulta.

La cláusula WHERE se corresponde con los predicados de selección del álgebra relacional, y dichos predicados hacen referencia a atributos de relaciones especificadas en la cláusula FROM.

Una consulta típica en SQL tiene esta forma: SELECT A1, A2, ..., An FROM R1, R2, ..., Rm WHERE P donde cada Ai representa a un atributo, cada Ri representa a una relación y P es el predicado o condición de la consulta.

Esta consulta equivale a la siguiente expresión del álgebra relacional: ΠA1, A2, ..., An(σP (R1 x R2 x ... x Rm))

La lista de atributos puede sustituirse por un asterisco (*) para seleccionar todos los atributos de todas las relaciones que aparecen en la cláusula FROM, y la claúsula WHERE puede omitirse, por lo que el predicado siempre será verdadero, y se seleccionarán todas las filas de todas las tablas que aparezcan en la cláusula FROM.

Por ejemplo, para saber cuáles son los nombres de todos los clientes del banco escribiríamos la sentencia SQL siguiente:

SELECT nombreCli FROM clientes;

Y para saber cuáles son los nombres de los empleados que trabajan en la sucursal de "Sol", escribiríamos la sentencia SQL siguiente:

SELECT nombreEmp FROM empleados WHERE nombreSuc = "Sol";

En la cláusula SELECT también pueden aparecer expresiones con operadores aritméticos (+, -, *, /) o de otros tipos, por lo que para ver el saldo de las cuentas en dolares en lugar de pesos, escribiríamos

SELECT numeroCta, saldo * 2100 FROM cuentas;

7.2.2.2. Gestión de duplicados y valores nulos

Si hubiese dos empleados con el mismo nombre, aparecerían valores duplicados. Si estamos interesados en que no aparezcan duplicados se debe utilizar SELECT DISTINCT.
No obstante, si queremos que se conserven explícitamente todos los duplicados escribiremos SELECT ALL o SELECT. Por ejemplo, para saber cuáles son los nombres de los empleados que trabajan en la sucursal de "Sol" sin que aparezcan duplicados escribiríamos la sentencia SQL siguiente:

SELECT DISTINCT nombreEmp FROM empleados WHERE nombreSuc = "Sol";

Cuando en una consulta necesitamos manejar valores nulos, tenemos que usar las condiciones IS NULL e IS NOT NULL. Por ejemplo, para ver los nombres de los clientes que no tienen domicilio, usaríamos esta instrucción SQL:

SELECT nombreCli FROM clientes WHERE domicilio IS NULL;

7.2.2.3. Combinación de relaciones y otros aspectos básicos

SQL no dispone de una operación directa para combinar tablas mediante el producto natural, pero como éste es un producto cartesiano con una selección, es

relativamente sencillo escribir una orden SQL para realizar un producto natural. Recordemos que en el álgebra relacional, para realizar el producto natural a partir del producto cartesiano para obtener los nombres de todos los empleados que trabajan en todas las ciudades podríamos escribir:

Πempleados.nombreEmp (σsucursales.nombreSuc = empleados.nombreSuc (sucursales x empleados))

La sentencia SQL equivalente a esta expresión del álgebra relacional sería:

SELECT empleados.nombreEmp FROM sucursales, empleados WHERE sucursales.nombreSuc = empleados.nombreSuc;

Aunque se podría haber obtenido el mismo resultado a partir únicamente de la tabla *empleados*. Obsérvese que se ha seguido la notación *nombreRelacion. nombreAtributo* para eliminar posibles ambigüedades. Para poder utilizar más de un predicado en una cláusula WHERE se pueden utilizan los conectores lógicos AND, OR y NOT en lugar de los conectores del álgebra relacional ^, v, y ¬.

Por ejemplo, para saber cuáles son los nombres de los empleados que trabajan en alguna de las sucursales de la ciudad de "Acacias", escribiríamos:

SELECT empleados.nombreEmp FROM sucursales, empleados WHERE sucursales.nombreSuc = empleados.nombreSuc AND sucursales.ciudadSuc = "Acacias";

SQL incluye el operador BETWEEN para simplificar las cláusulas WHERE que se refieren a valores que sea mayor o igual que uno y menor o igual que otro, por lo que para saber cuáles son los números de las cuentas que tienen un saldo comprendido entre 20000 y 50000 podemos escribir
SELECT numeroCta FROM cuentas WHERE saldo BETWEEN 20000 AND 50000;

En lugar de SELECT numeroCta FROM cuentas WHERE saldo >= 20000 AND saldo <= 50000;

Análogamente, podemos utilizar el operador NOT BETWEEN.

Por último, si queremos realizar consultas SQL comparando ciertos atributos con valores de cadena que utilicen comodines utilizaremos el operador LIKE, teniendo en cuenta que el carácter de comodín para un carácter es el subrayado (_) y que el carácter de comodín para un conjunto de caracteres es el tanto por ciento (%).

De esta forma, para saber cuáles son los nombres de los clientes que viven en la calle "Gibraltar español" escribiríamos lo siguiente:

SELECT nombreCli FROM clientes WHERE domicilio LIKE "Gibraltar español%";
SQL permite la búsqueda de desigualdades con el operador NOT LIKE.

SQL permite al usuario especificar criterios para la presentación ordenada de resultados, y lo hace mediante la cláusula ORDER BY, que en el caso de que se utilice debe ir después de la cláusula WHERE (en caso de que exista).

Por ejemplo, para mostrar una lista ordenada alfabéticamente de los clientes que tienen alguna cuenta en la sucursal "Sol", escribiríamos lo siguiente: +

SELECT DISTINCT nombreCli FROM clientes, cuentas, ctacli WHERE clientes.dniCli = ctacli.dniCli AND cuentas.numeroCta = ctacli.numeroCta AND cuentas.nombreSuc = "Sol" ORDER BY nombreCli;

Por omisión, SQL lista las tuplas en orden ascendente (ASC), pero para especificar un orden descendente, escribiríamos DESC, de esta forma: ORDER BY *col1* [ASC|DESC] [, *col2* [ASC|DESC]], ...

En SQL, la combinación de relaciones puede realizarse también de otras formas ya vistas en el álgebra relacional.

A continuación veremos el INNER JOIN, NATURAL INNER JOIN y los OUTER JOIN. Para ilustrar estas operaciones nos basaremos en las tablas de Empleados y Sucursales. El INNER JOIN o reunión interna es una forma compacta de combinar relaciones atendiendo a una condición.

Su sintaxis es Tabla1 INNER JOIN Tabla2 ON condición Para ilustrar su uso, si queremos obtener el nombre de los empleados y la ciudad en la que está la sucursal en la que trabajan utilizando un INNER JOIN escribiremos
SELECT nombreEmp, ciudadSuc FROM Empleados INNER JOIN Sucursales ON Empleados.nombreSuc = Sucursales.nombreSuc;

En la reunión interna hay que especificar explícitamente la condición de *join* de la consulta, la cual no tiene por que ser siempre ligando los atributos que relacionan las tablas que participan en la consulta. Sin embargo, como gran parte de las consultas que se realizan sobre una base de datos suele combinar las tablas utilizando los atributos comunes SQL ofrece el NATURAL INNER JOIN para realizar el producto natural de dos tablas.

La consulta siguiente sería equivalente a la anterior, salvo que la que se muestra a continuación lo hace con NATURAL INNER JOIN.

SELECT nombreEmp, ciudadSuc FROM Empleados NATURAL INNER JOIN Sucursales

Tanto la reunión interna como la natural interna combinan dos relaciones de forma que en el resultado sólo aparecen tuplas que presentan valores si en cada tabla hay tuplas con valores coincidentes. En cambio, las reuniones externas relajan esta condición porque permiten que en el resultado haya tuplas que se

correspondan a tuplas de una tabla que no tienen tuplas relacionadas en la otra tabla.

Existen tres tipos de reuniones externas, que son la reunión externa izquierda (LEFT OUTER JOIN), la reunión externa derecha (RIGHT OUTER JOIN) y la reunión externa completa (FULL OUTER JOIN).

A continuación se muestra cada una de ellas y su sintaxis, similar a la de INNER JOIN. La reunión externa izquierda se calcula como la reunión interna, pero además, también se incluirán en el resultado las tuplas de la relación de la izquierda que no estén relacionadas con alguna tupla de la relación de la derecha, rellenándose los valores de los atributos correspondientes a la relación de la derecha con nulos. La consulta siguiente realiza la reunión externa izquierda de Sucursales de Horseneck con Empleados

SELECT Sucursales.nombreSuc, nombreEmp FROM Sucursales LEFT OUTER JOIN Empleados ON Sucursales.nombreSuc = Empleados.nombreSuc WHERE ciudadSuc = “Horseneck”;

El resultado de esta consulta sería

Sucursales.nombreSuc	nombreEmp
Perrydge	Hansen
Perrydge	Dubitzky
Mianus	Henson
Round Hill	Null

Análogamente, la reunión externa derecha (RIGHT OUTER JOIN) funciona de forma similar que la reunión externa izquierda salvo considerando las tuplas de la relación que aparece en la derecha.

Por último, la reunión externa completa es la operación más relajada de todas, ya devuelve las tuplas de ambas relaciones aunque no tengan tuplas relacionadas en la otra tabla. Por ejemplo, la consulta siguiente realiza la reunión externa completa de Sucursales y Empleados combinando las tuplas de Sucursales y Empleados. Si hay tuplas relacionadas las añade al resultado. Si no es así, añade la tupla no relacionada y completa los valores con nulos. (No obstante, en este caso y por la integridad referencial definida en Empleados no habrá tuplas en Empleados que no tengan tuplas relacionadas en Sucursales, por lo que la reunión externa completa devolverá el mismo resultado que una reunión externa izquierda.)

SELECT Sucursales.nombreSuc, nombreEmp

FROM Sucursales FULL OUTER JOIN Empleados ON

Sucursales.nombreSuc = Empleados.nombreSuc;

7.2.2.4. Agrupación y funciones de agregación

SQL permite la posibilidad de calcular funciones en grupos de tuplas utilizando la cláusula GROUP BY. Para formar estos grupos, se utilizan todos los atributos que aparecen en la cláusula GROUP BY, y cada una de las tuplas que tienen el mismo valor en cada uno de los atributos que se especifican en la cláusula GROUP BY se colocan en el mismo grupo.

SQL incluye funciones para calcular:

- Promedio: AVG
- Mínimo: MIN
- Máximo: MAX
- Total: SUM
- Cuenta: COUNT

A estas operaciones se les denomina funciones de agregación, ya que operan sobre grupos de tuplas, y el resultado de estas funciones es un valor único.

Por ejemplo, para conocer el activo medio por ciudades escribiríamos lo siguiente:

```
SELECT ciudadSuc, AVG(activo)
FROM sucursales
GROUP BY ciudadSuc;
```

Sobre el resultado de una agrupación es posible establecer una condición, pero ésta actuará a nivel de grupo, y no a nivel de tupla, como lo hace el WHERE. Las condiciones a nivel de grupo se definen mediante la cláusula HAVING, y se especifica detrás de GROUP BY. Así, si queremos saber cuáles son las medias de los activos de las sucursales por ciudades para aquellas en que la media sea superior a 20000, escribiríamos

```
SELECT ciudadSuc, AVG(activo)
FROM sucursales
GROUP BY ciudadSuc
HAVING AVG(activo) > 20000;
```

Cuando se utilizan grupos hay que tener en cuenta que en la cláusula SELECT sólo pueden aparecer los atributos por los que se está realizando la agrupación, o bien funciones de agregación.

Para saber el saldo máximo de las cuentas de las sucursales por ciudad agruparíamos por ciudad como muestra la consulta siguiente

```
SELECT ciudadSuc, MAX(saldo)
FROM Sucursales, Cuentas
WHERE Sucursales.nombreSuc = Cuentas.nombreSuc
GROUP BY ciudadSuc;
```

Para saber la ciudad en la que están las sucursales con más de una cuenta necesitamos conocer primero cuáles son las sucursales que tienen más de una cuenta, y luego utilizar esto para obtener en qué ciudad está

```
SELECT ciudadSuc
FROM Sucursales
WHERE nombreSuc IN
    (SELECT nombreSuc
    FROM Cuentas
    GROUP BY nombreSuc
    HAVING COUNT(numeroCta) > 1);
```

A veces, es necesario tratar toda la relación como si fuese un solo grupo, por lo que en estos casos no será necesario utilizar GROUP BY. Por ejemplo, para obtener el saldo medio, máximo y mínimo de las cuentas escribiríamos.

```
SELECT AVG(saldo), MAX(saldo), MIN(saldo)
FROM Cuentas;
```

Otra utilidad que se desprende de considerar toda la tabla como un grupo puede ser para conocer el número de tuplas de una relación. Para ello, podríamos escribir lo siguiente para conocer el número de clientes del banco.

```
SELECT COUNT(*)
FROM Clientes;
```

Para conocer el número de ciudades en los que el banco tiene sucursales escribiríamos.

```
SELECT COUNT (DISTINCT ciudadSuc)
FROM Sucursales
```

y se debe utilizar DISTINCT porque si no se obtendría simplemente el número de sucursales que tiene el banco, ya que se repetirían las ciudades en las que están las sucursales de una misma ciudad.

Por último, para conocer el nombre de las ciudades que tienen más de dos sucursales lo podríamos hacer con una consulta anidada

```
SELECT ciudadSuc
FROM Sucursales S
WHERE (SELECT COUNT (*)
        FROM Sucursales
        WHERE S.ciudadSuc = Sucursales.ciudadSuc) > 2;
```

7.2.2.5. Operaciones sobre conjuntos

SQL incorpora las operaciones UNION (unión), INTERSECT (intersección) y MINUS (diferencia) que operan sobre relaciones o tablas y se corresponden con las operaciones □, ∩ y - del álgebra relacional.

Veamos cómo podemos utilizar estas operaciones en SQL.

Por ejemplo, para saber cuáles son los nombres de todas las personas relacionadas con el sistema bancario escribiríamos:

```
SELECT nombreEmp
FROM empleados
UNION
SELECT nombreCli
FROM clientes;
```

Para encontrar todos los nombres que aparecen como empleado y como cliente escribiríamos:

```
SELECT nombreEmp
FROM empleados
INTERSECT
SELECT nombreCli
FROM clientes;
```

Por último para encontrar todos los nombres que aparecen como empleado pero no como cliente, escribiríamos:

```
SELECT nombreEmp
FROM empleados
MINUS
SELECT nombreCli
FROM clientes;
```

Por omisión, las operaciones UNION, INTERSECT y MINUS eliminan las tuplas duplicadas, pero si no se desea que se eliminen las tuplas duplicadas, escribiremos UNION ALL, INTERSECT ALL, y MINUS ALL.

7.2.2.6. Creación de alias y variables de tupla

Tal y como vimos en el tema anterior, a veces es necesario renombrar una relación de forma que se puedan comparar dos tuplas de la misma relación. Este era el caso de conocer cuáles son los nombres de los empleados que trabaja en la misma sucursal que "García".

Para crear alias sobre tablas en SQL, también llamados variables de tupla, basta con poner en la cláusula FROM el nombre del alias a continuación del nombre de la tabla, como se muestra en este ejemplo

```
SELECT E.nombreEmp
FROM Empleados E, Empleados Ebis
WHERE Ebis.nombreEmp = "García" AND
        Ebis.nombreSuc = E.nombreSuc;
```

No obstante, esta consulta también se podría haber realizado con una consulta anidada, las cuales se describen más adelante.

Otro ejemplo podría ser el de obtener las sucursales que tienen un activo superior a cualquier sucursal situada en Almería. Para ello, necesitamos hacer referencia dos veces a la misma tabla de forma que se podría hacer de esta forma.

```
SELECT E.nombreEmp
FROM Sucursales S1, Sucursales S2
WHERE S1.ciudadSuc = “Almería” AND
        S2.Activo > S1.Activo;
```

Si lo que queremos es cambiar los nombres de los atributos que resultan de una consulta, escribiremos en la cláusula SELECT el modificador AS entre el antiguo nombre del atributo y el nuevo nombre del atributo, tal y como se indica en este ejemplo

```
SELECT nombreSuc AS nombreDeSucursales
FROM sucursales;
```

7.2.2.7. Consultas anidadas y pertenencia a conjuntos

En el predicado de selección de tuplas (claúsula WHERE), además de poder comparar un atributo con un valor, podemos realizar la comparación con un conjunto de valores, es decir podemos especificar explícitamente un conjunto con los valores de la comparación, en lugar de realizar varias comparaciones OR. Para ello utilizaremos el operador de pertenencia a conjuntos IN, o bien el operador NOT IN, si lo que queremos comprobar es la no pertenencia al conjunto.

Por ejemplo, para saber cuáles son los nombres de los empleados que trabajan en las sucursales de "Sol" o "Castellana" podemos escribir

```
SELECT nombreEmp
FROM empleados
WHERE nombreSuc IN ("Sol", "Castellana");
```

en lugar de escribir

```
SELECT nombreEmp
FROM empleados
WHERE nombreSuc = "Sol" OR nombreSuc = "Castellana";
```

A esta forma de especificación de parámetros se le denomina conjuntos explícitos.

A la vista de este ejemplo, cabe pensar que se podría crear una consulta que utilizase como predicado de comparación otra consulta, es decir, anidar una consulta dentro de otra. A la consulta interna se le denomina consulta anidada, y a la que utiliza el resultado de la ejecución de la consulta anidada se le denomina consulta externa.

Para ello, veamos otra forma de escribir la consulta que devuelve cuáles son los nombres de los empleados que trabajan en las sucursales de la ciudad de "Madrid".

```
SELECT nombreEmp
FROM empleados
WHERE nombreSuc IN (SELECT nombreSuc
FROM sucursales
```

```
WHERE ciudadSuc = "Madrid");
```

En las consultas anidadas es posible que se puedan originar ambigüedades entre los nombres de los atributos si hay atributos con el mismo nombre en la consulta externa y en la consulta interna. En este caso, y dado que los atributos que no están precedidos por un nombre de relación se refieren a la consulta anidada más interna, se trata de calificar los nombres de los atributos con el nombre de las tablas correspondientes.

7.2.2.8. Consultas correlacionadas

Un caso especial de consultas anidadas son las consultas correlacionadas. Se trata de consultas en las que la consulta anidada se ejecuta una vez para cada tupla de consulta externa. Esto ocurre cuando en el WHERE de una subconsulta se hace referencia a un atributo que está en la consulta externa. Por ejemplo, supongamos que tenemos las siguientes tablas.

Personal

Nombre	Apellidos	NSS	AñoNacimiento	Sexo	Salario	NSSSuperv
Pedro	Márquez	1	1954	H	1200	2
Isabel	Fernández	2	1972	M	1150	3
María	Yagüe	3	1963	M	2200	null
Juan	Martín	4	1971	H	1050	3

Familia

Nombre	NSSP	AñoNacimiento	Sexo	Parentesco
Juana	1	1974	M	Hija
Manuel	1	1976	H	Hijo
Margarita	1	1958	M	Esposa
María	3	1993	M	Hija
Luis	3	1963	H	Esposo

Si queremos saber el nombre y apellidos de los empleados que tienen familiares con su mismo nombre y sexo podríamos escribir

```
SELECT Nombre, Apellidos
FROM Personal
WHERE Nombre IN
(SELECT Nombre
```

```
FROM Familia
WHERE NSS=NSSP AND
        Personal.Nombre = Familia.Nombre AND
        Personal.Sexo = Familia.Sexo);
```

o bien

```
SELECT Personal.Nombre, Personal.Apellidos
FROM Personal, Familia
WHERE NSS=NSSP AND
    Personal.Nombre = Familia.Nombre AND
    Personal.Sexo = Familia.Sexo);
```

En algunos casos es necesario que en las consultas anidadas se compruebe si el resultado de la subconsulta es vacío o no. Esto se realiza utilizando la función EXISTS en la cláusula WHERE. Por ejemplo, para el caso anterior se trataría de obtener registros de empleado para los que “existen” familiares con su mismo nombre y sexo.

```
SELECT Nombre, Apellidos
FROM Personal
WHERE EXISTS
(SELECT *
FROM Familia
WHERE NSS=NSSP AND
        Personal.Nombre = Familia.Nombre AND
        Personal.Sexo = Familia.Sexo);
```

Si ahora queremos saber el nombre de las personas que no tienen familiares escribiríamos.

```
SELECT Nombre, Apellidos
FROM Personal
WHERE NOT EXISTS
(SELECT *
FROM Familia
WHERE NSS=NSSP);
```

A veces es necesario que alguna de las condiciones del predicado de la consulta realicen comparaciones sobre conjuntos, como pueden ser cuáles son los nombres de las sucursales que tienen un activo superior a cualquiera de las sucursales de la ciudad de "Almería", o bien el nombre de las sucursales que tengan un activo mayor que el de todas las sucursales de la ciudad de "Almería".

Para ello utilizaremos SOME y ALL de esta forma.

En primer lugar, para conocer los nombres de las sucursales que tienen un activo superior a cualquiera de las sucursales de la ciudad de "Almería" utilizaremos > SOME de esta forma.

```
SELECT nombreSuc
FROM sucursales
WHERE activo > SOME (SELECT activo
FROM sucursales
WHERE ciudadSuc = "Almería");
```

No obstante, y tal y como vimos en la sección anterior, esto también se puede realizar mediante el uso de alias de tabla (variables de tupla).

Si ahora lo que quisiésemos fuese el nombre de las sucursales que tienen un activo superior al de todas las sucursales de la ciudad de "Almería" utilizaremos > ALL de esta forma

```
SELECT nombreSuc
FROM sucursales
WHERE activo > ALL (SELECT activo
FROM sucursales
WHERE ciudadSuc = "Almería");
```

SQL también permite las construcciones < SOME, <= SOME, >= SOME, <> SOME, = SOME, < ALL, <= ALL, >= ALL, = ALL y <> ALL.

Por tanto IN, SOME y ALL permiten comprobar un único valor con un conjunto de valores.

7.2.2.9. Vistas. Actualización de vistas

Una vista SQL es una tabla derivada a partir de otras. Estas tablas pueden ser tablas base (tablas de las definidas originalmente en los esquemas) o vistas. Una

vista no tiene por que estar almacenada físicamente, sino que puede recuperar la información bajo demanda. Así, las vistas son consideradas como tablas virtuales, a diferencia de las tablas base que sí almacenan realmente los registros. Este hecho limita la actualización de las vistas, sobre todo cuando se realiza la combinación de tablas o cuando se realizan operaciones de agregación. No obstante, y a efectos de consulta, una vista y una tabla no ofrecen diferencia alguna, y esto es lo que hace que las vistas sean un mecanismo muy utilizado, dada la utilidad que presentan para la personalización de información (p.e. ocultar información de las tablas base) o para la simplificación de consultas.

En SQL las vistas se crean mediante la orden

```
CREATE VIEW nombreVista AS ExpresionSELECT;
```

El ejemplo siguiente crearía una vista que devuelve el nombre, teléfono y la ciudad en la que trabajan cada uno de los empleados del banco.

```
CREATE VIEW EmpleCiudad
AS SELECT nombreEmp, telefono, ciudadSuc
FROM Empleados, Sucursales
WHERE Empleados.nombreSuc = Sucursales.nombreSuc;
```

La vista queda formada por las columnas que aparezcan en el SELECT que defina la vista, aunque también es posible especificar el nombre que van a tener las columnas de la vista indicándolas explícitamente, como muestra el ejemplo siguiente que crea una vista en la que se obtiene el DNI de cada cliente y el saldo total de sus cuentas.

```
CREATE VIEW ClientesSumaSaldo (DNICli, SaldoTotal)
AS SELECT DNICli, SUM(Saldo)
FROM Clientes, CtaCli, Cuentas
WHERE Clientes.DNICli = CtaCli.DNICli and
CtaCli.numeroCta = Cuentas.numeroCta
GROUP BY Clientes.DNICli;
```

Así, el esquema de la primera vista sería EmpleCiudad(nombreEmp, telefono, ciudadSuc) y el de la segunda sería ClientesSumaSaldo (DNICli, SaldoTotal).

La eliminación de vistas se hace mediante la orden

```
DROP VIEW nombreVista
```

Las órdenes DROP VIEW EmpleCiudad y DROP VIEW ClientesSumaSaldo eliminarían las dos vistas que hemos definido.

Hemos dicho que las vistas son realmente definiciones de consultas, y no se almacenan como conjuntos de filas en la base de datos. Sin embargo, hay condiciones en las que se pueden actualizar las vistas. Realmente, estas modificaciones se efectuarán sobre la tabla de la que procede la vista. Para saber si una vista es actualizable o no, debemos tener en cuenta lo siguiente:

- Una vista definida sobre una sóla tabla es actualizable si los atributos de la vista incluyen la clave primaria de dicha tabla y todos los atributos que no pueden ser nulos.
- Una vista definida sobre la combinación de dos o más tablas no es actualizable.
- Una vista definida usando agrupación y funciones de agregación no es actualizable.

7.2.3. Modificación de datos en SQL

Hasta ahora nos hemos limitado a recuperar información de la base de datos. SQL dispone de órdenes para añadir, eliminar y actualizar los datos de una base de datos mediante las instrucciones INSERT, DELETE y UPDATE, respectivamente. A continuación se estudia cada una de ellas.

7.2.3.1. Inserción

En SQL los datos son introducidos en las tablas mediante la orden INSERT, la cual se puede utilizar para insertar un solo registro en una tabla o bien para insertar en una tabla el resultado de una consulta.

Para insertar un registro en una tabla utilizaremos

INSERT INTO nombreTabla VALUES (valor1, valor2, ...);

Por ejemplo, la expresión siguiente crearía un nuevo empleado llamado Harry en la sucursal de "Los Pinos". Harry tiene el DNI 16 y su teléfono es 161616.

```
INSERT INTO Empleados
VALUES (“Harry”, “16”, “161616”, “Los Pinos”);
```

En el caso de que la tabla tuviese definida alguna columna que permitiese valores nulos o asignase valores predeterminados en el caso de no indicar ningún valor, se podrían especificar los nombres de columna a los que afectan los datos que se quieren insertar. Por ejemplo, si en la tabla de empleados se permitiese que el

teléfono fuese nulo podríamos escribir para un empleado de la sucursal de Los Pinos que se llamase Mukos y con DNI 17 lo siguiente.

```
INSERT INTO Empleados (nombreEmp, dniEmp, nombreSuc)
VALUES (“Mukos”, “17”, “Los Pinos”);
```

Por ultimo, una sentencia INSERT también puede insertar los valores que procedan de la ejecución de una consulta en lugar de especificar explícitamente los valores que se quieren introducir. En este caso la sintaxis es

```
INSERT INTO nombreTabla ExpresionSELECT;
```

Por ejemplo, podemos crear una tabla temporal para guardar el número de empleados para cada sucursal e introducir en ella el resultado de una consulta, como se muestra a continuación

```
CREATE TABLE NumeroEmple
(Sucursal VARCHAR(50),
Empleados INTEGER);
INSERT INTO NumeroEmple
SELECT NombreSuc, COUNT(nombreEmp)
FROM Empleados
GROUP BY NombreSuc;
```

7.2.3.2. Eliminación

La orden SQL para eliminar datos de una tabla es DELETE y elimina los registros de una tabla que satisfacen una condición. Su sintaxis es parecida al SELECT, salvo que no especifica atributos para la proyección.

```
DELETE
FROM Tabla
WHERE Condicion;
```

Por ejemplo, la orden siguiente eliminaría el registro del empleado Mukos

```
DELETE
FROM Empleados
WHERE nombreEmp = “Mukos”;
```

Como condición se puede establecer cualquier predicado por complejo que sea, o incluso se pueden especificar subconsultas. Por ejemplo, la expresión siguiente elimina todos los empleados de las sucursales de la ciudad de Madrid.

```
DELETE
FROM Empleados
WHERE nombreSuc IN (SELECT nombreSuc
FROM Sucursales
WHERE ciudadSuc = “Madrid”);
```

La eliminación de registros en una tabla puede desencadenar la eliminación de registros en otras tablas si se han definido eliminaciones en cascada. Por último, si no especifica la cláusula WHERE se eliminarán todos los registros de la tabla.

7.2.3.3. Actualización de la información de la base de datos

Los registros de una tabla se pueden modificar mediante la orden UPDATE de SQL. Esta orden actualiza el valor de las columnas de los registros que cumplan una condición. Al igual que en la orden DELETE, la condición se especifica en una cláusula WHERE. La sintaxis es la siguiente.

```
UPDATE tabla
SET Modificacion
WHERE Condicion
```

La consulta siguiente aumenta en 500 el saldo de las cuentas de la sucursal de Castellana.

```
UPDATE Cuentas
SET saldo = saldo + 500
WHERE nombreSuc = “Castellana”;
```

En la cláusula WHERE se pueden especificar subconsultas como hicimos con la instrucción DELETE. Por ejemplo, para incrementar en un 10 por ciento el saldo de las cuentas de sucursales de la ciudad de Almería escribiríamos

```
UPDATE Cuentas
SET saldo = saldo * 1.1
WHERE nombreSuc IN (SELECT nombreSuc
FROM Sucursales
WHERE ciudadSuc = “Almería”);
```

7.3 Actividades Complementarias

1. Realizar los ejercicios planteados en los capítulos 4, 5 y 6 utilizando las instrucciones del lenguaje de consulta SQL.

2. Se tiene la siguiente estructura relacional:

FACULTAD (cod_facultad, nombre, tot_estudiantes)
DEPARTAMENTOS (cod_dep, nombre, cod_facultad)
CARRERAS (cod_carrera, nombre, cod_dep)
MATERIAS (cod_materia, nombre, cod_carrera, semestre)
HORARIOS (cod_horario, dia hora_ini, hora_fin)
AULAS (aula, bloque, piso, caracteristicas)
PROFESORES (cedula, nombre, fecha_naci, tipo, categoría)
SEMESTRES_TRABAJADOS (cedula, semestre, fecha_ini, fecha_fin, salario, cod_materia)
PROGRAMACION_ACADEMICA (cod_programa, cod_materia, cod_horario, aula, bloque, fecha)
ALUMNOS (codigo, cedula, nombre, direccion, telefono, semestre_actual, total_semestres_cursados)
CURSOS (codigo, cod_programa, cedula_profe, nota_final)
PROMEDIOS_SEMESTRE (codigo, promedio)

CONSULTAS A REALIZAR:

- Liste el código y el nombre de los alumnos que cursan las materias del quinto semestre de la carrera de Ingeniería de Sistemas.
- Liste la cédula, el nombre y el tipo de los profesores que dictan materias en el departamento de Sistemas que tengan la máxima categoría.
- Muestre el código, nombre del alumno y código de la materia, de los alumnos que han cursado más de 2 veces una materia, durante su estadía en la universidad.
- Liste la ocupación de cada una de las aulas de cada uno de los bloques de la universidad en este año 2000. La ocupación se refiere a: código y nombre de la materia que se dicta, el profesor, que días y a que hora.

CAPITULO 8 POSTGRESQL

8 Introducción

En los últimos años, el software de bases de datos ha experimentado un auge extraordinario, a raíz de la progresiva informatización de casi la totalidad de las empresas de hoy día. No es extraño, que existan multitud de gestores de bases de datos, programas que permiten manejar la información de modo sencillo. De este modo tenemos Oracle, Microsoft SQL Server, Borland Interbase entre otras. Las soluciones software que hemos citado son comerciales. En el mundo del software libre, siempre que se necesita algo, tarde o temprano se implementa. Así tenemos MySQL, gestor muy usado en la web (combinado con php y apache) o PostgreSQL.

PostgreSQL es software libre. Concretamente está liberado bajo la licencia BSD, lo que significa que cualquiera puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente, PostgreSQL además de ser libre es gratuito y se puede descargar de su página web para multitud de plataformas.

La versión actual de PostgreSQL es la 8.1, y puede descargarse libremente desde su sitio oficial http://www.postgresql.org.

8.1. Instalación de PostgreSQL

El proceso de instalación de PostgreSQL para Linux, distribución RedHat 8.0. Utiliza un sistema de gestión de paquetes denominado RPM que permite instalar fácilmente el gestor de base de datos. Además PostgreSQL viene incluido en la distribución estándar por lo que simplemente hemos de instalar los paquetes que nos interesen:

```
[jav@cable213a078 jav]$ rpm -Uvh postgresql-7.2.2-1.i386.rpm
```

En caso de no usar distribuciones basadas en RPM o simplemente por gusto, se puede bajar el código fuente del sitio web de PostgreSQL. Para descomprimirlo.

```
[jav@cable213a078 jav]$ tar xvzf
postgresql-7.2.2-1.tar.gz
```

vamos al directorio resultante y ejecutamos el script de configuración.
[jav@cable213a078 postgresql]$
./configure

Para compilar se ejecuta

[jav@cable213a078 postgresql]$ **make**

y si no hay errores, se loguea como root y ejecuta

[root@cable213a078 postgresql]# **make install**

De esta forma ya se tiene PostgreSQL instalado en el equipo

Notas

Si algo va mal, o se tiene necesidades especiales (un directorio diferente del estándar, compilación para otras arquitecturas...) debería ejecutar

[jav@cable213a078 postgresql]$ **./configure –help**

y observar la salida.

8.2. Inicializar el servidor

Para hacer funcionar el servidor de base de datos. Lo primero es fijar la variable de entorno que almacenará la ruta hasta el directorio donde guardaremos la información de la base de datos:

[jav@cable213a078 postgresql]$ **export PGDATA=/el/path/deseado/data**

En nuestro caso:

```
[jav@cable213a078 postgresql]$ export PGDATA=/home/jav/db/data
```

El siguiente paso es crear las estructuras necesarias para iniciar el servidor de la base de datos. Esto se consigue con la orden **initdb**:

```
[jav@cable213a078 postgresql]$ initdb
The files belonging to this database system will be owned by user "jav".
This user must also own the server process.

Fixing permissions on existing directory /home/jav/db/data... ok
creating directory /home/jav/db/data/base... ok
creating directory /home/jav/db/data/global... ok
creating directory /home/jav/db/data/pg_xlog... ok
creating directory /home/jav/db/data/pg_clog... ok
creating template1 database in /home/jav/db/data/base/1... ok
creating configuration files... ok
initializing pg_shadow... ok
enabling unlimited row size for system tables... ok
creating system views... ok
loading pg_description... ok
vacuuming database template1... ok
copying template1 to template0... ok

Success. You can now start the database server using:

    /usr/bin/postmaster -D /home/jav/db/data
or
    /usr/bin/pg_ctl -D /home/jav/db/data -l logfile start
```

Siguiendo las instrucciones que amablemente nos da **initdb** iniciamos el servidor, añadiendo los parámetros **-i** (para admitir conexiones TCP/IP, ya lo explicaremos más adelante) y **-o** para obtener formato de fechas europeo en nuestra base de datos:

```
[jav@cable213a078 db]$ /usr/bin/postmaster -o -i -D /home/jav/db/data &
```

Tras algunos mensajes de depuración, el servidor estará corriendo en la máquina. Ahora crearemos una base de datos para ver el funcionamiento:

```
[jav@cable213a078 db]$createdb pepa
CREATE DATABASE
```

Ingresamos a la base de datos con la instrucción psql

```
[jav@cable213a078 db]$ psql pepa
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit

pepa=# create table provincias(prefijo integer, nombre varchar(20));
CREATE

pepa=# insert into provincias values(987, 'León');
INSERT 16559 1

pepa=# insert into provincias values(91, 'Madrid');
INSERT 16560 1

pepa=# insert into provincias values(923, 'Salamanca');
INSERT 16561 1

pepa=# select * from provincias;
 prefijo |  nombre
---------+-----------
     987 | León
      91 | Madrid
     923 | Salamanca
(3 rows)
```

De esta forma vemos el funcionamiento de la base de datos y la forma como se pueden ingresar datos a las tablas creadas.

8.3. Administración remota con OpenSSH

También podemos acceder a un máquina remota (donde está corriendo la base de datos) de modo seguro para administrarla. Para ello utilizaremos ssh que

permite establecer conexiones entre máquinas de modo que toda la información que intercambien vaya encriptada.

Lo primero es habilitar el servidor ssh en la máquina a la que pretendamos acceder, ssh utiliza el puerto 22 para las comunicaciones. Una vez cumplido este paso, ya estamos listos para aceptar conexiones remotas.

Ya hemos configurado lo necesario del lado del servidor. Veamos ahora los pasos necesarios del lado del cliente. Distinguiremos entre máquinas Linux y Windows .

- En Linux (o Unix) posiblemente ya tengamos el cliente ssh instalado. Si no es así nos dirigiremos al sitio oficial de OpenSSH http://www.openssh.com de donde bajaremos los fuentes, los compilaremos e instalaremos. Dependiendo de la distribución que usemos será más fácil o más difícil encontrar paquetes precompilados que nos ahorren el trabajo.

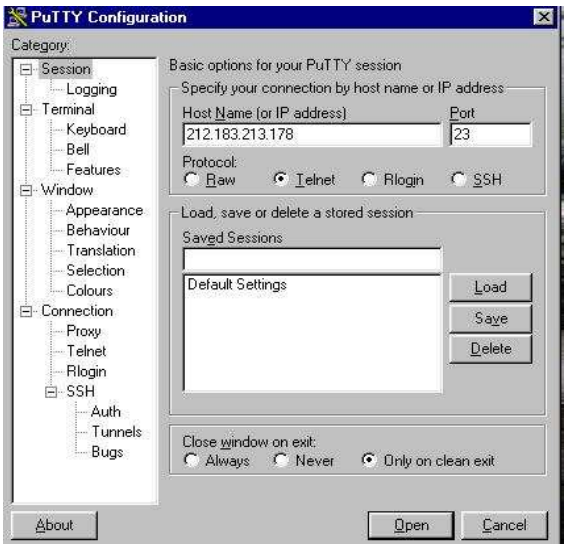
Suponiendo que ya tenemos el cliente instalado, es hora de acceder a la máquina remota. Sin entrar en detalles, diremos que basta con:

```
[jav@cable213a078 jav]$ ssh bbdd@212.183.213.178
bbdd@212.183.213.178's password:
Esta cuenta aloja el trabajo de BBDD
Directorios:
-datos: Contiene los ficheros de la base de datos
-docbd: Contiene los fuentes de la documentación y la documentación en sí. Ejecute make single-html para documentación en un único html o make multiple-html para documentación en varios html.
-programa: Contiene el código del programa que acceda a la BBDD

Cualquier duda puede remitirla a tjavier@usuarios.retecal.es o rtf@usuarios.retecal.es
[bbdd@cable213a078 bbdd]$
```

Genéricamente basta con **ssh login@IP** y se nos pedira el password. Una vez introducido, ya estaremos trabajando en la máquina remota y todo lo que hagamos se ejecutará en ella:

```
[bbdd@cable213a078 bbdd]$ ls

datos  docbd  programa
```

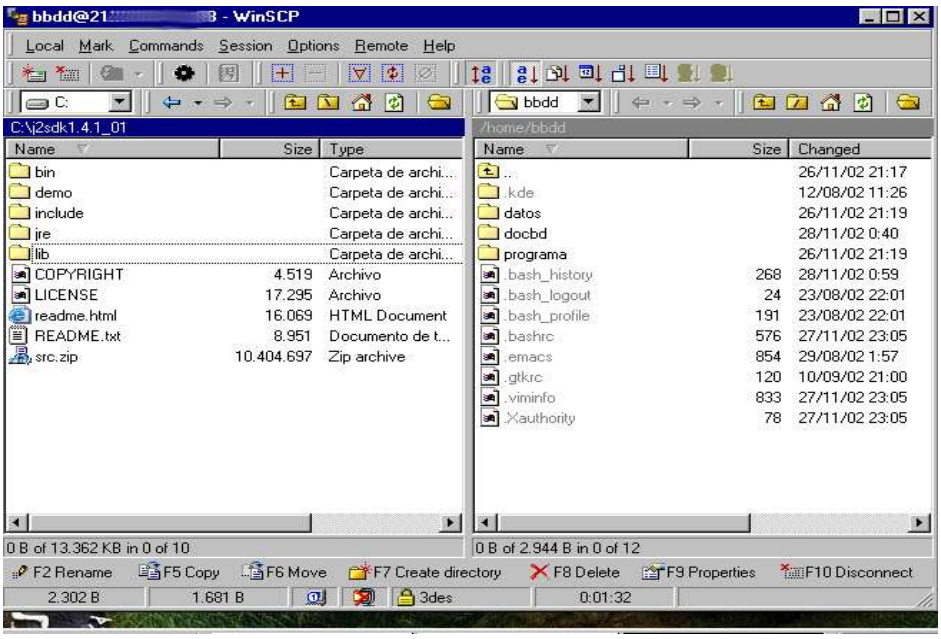
- En Windows se realiza de un modo similar. Lo primero será bajar PuTTY, http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html, un cliente para conexiones ssh en windows. PuTTY tiene una interfaz amigable, en la que sólo tendremos que especificar la IP a la que conectarnos en indicar que pretendemos hacer una conexión ssh:



El aspecto de PuTTY funcionando es similar al de una consola de Unix. Tras un rato trabajando con PuTTY se nos olvidará que estamos utilizando Windows.

En ocasiones será necesario mover archivos de nuestra máquina local a la remota o viceversa. En Windows existe una aplicación que proporciona una interfaz de usuario a **scp** (secure copy). Esta aplicación de llama WinSCP

http://winscp.vse.cz/eng/download.php y es también muy sencilla de usar, con una configuración muy intuitiva:



8.4 Actividades Complementarias.

1. Realizar el montaje del manejador de bases de datos PostgreSql y diseñe una base de datos con el enunciado del capitulo 5, capitulo 6 ejercicio numero 4.

2. Investigue y explique como funcionamiento el manejador de base de datos PostgreSql bajo windows.

3. Investigue y realice una comparación entre los manejadores de bases de datos PostgreSQl y MySql software libre.

9. Glosario de Terminos

Archivo: Grupo de registros relacionados

Atributos: Es una unidad básica e indivisible de información acerca de una entidad o una relación. Por ejemplo la entidad proveedor tendrá los atributos nombre, domicilio, población.

Base de datos: Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en má3quina accesible en tiempo real y compatible con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

Campo: Grupo de caracteres

Claves: Campo o grupo de campos en un registro, que se usa para identificar a este ultimo.

Clave Primaria: Campo o grupo de campos que identifican exclusivamente un registró.

Diagrama Entidad- Relación: Modelo de datos que usa símbolos gráficos para mostrar la organización y las relaciones entre los datos.

Dominios: Es el conjunto de valores que puede tomar cada atributo. Por ejemplo el dominio del atributo población, será la relación de todas las poblaciones del ámbito de actuación de nuestra empresa.

Entidades: Son objetos concretos o abstractos que presentan interés para el sistema y sobre los que se recoge información que será representada en un sistema de bases de datos. Por ejemplo, clientes, proveedores y facturas serían entidades en el entorno de una empresa.

Integridad de datos: Grado hasta el cual son exactos los datos en cualquier archivo individual.

Lenguaje de Consulta estructurado(SQL): Lenguaje de manipulación de datos estandarizado.

Metodos: Una operación que realiza acceso a los datos. Podemos definir método como un programa procedimental o procedural escrito en cualquier lenguaje, que está asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por éste o por sus descendientes.

Métodos heredados: Están definidos en un objeto diferente, antepasado de éste (padre, "abuelo", etc.). A veces estos métodos se llaman métodos miembro porque el objeto los posee por el mero hecho de ser miembro de una clase.

Modelo de datos: Mapa o diagrama de entidades y sus relaciones.

Modelo de datos en red: Es una variación del modelo de datos jerárquico con relación de propietario-miembro, en la que un miembro quizá tenga muchos propietarios

Modelo Entidad – Relación : Se trata de una técnica de diseño de base de datos gráfica, que nos muestra información relativa a los datos y la relación existente entre ellos.

Modelo jerárquico de datos: Una clase de modelo lógico de bases de datos que tiene una estructura arborescente. Un registro subdivide en segmentos que se interconectan en relaciones padre e hijo y muchos más. Los primeros sistemas administradores de bases de datos eran jerárquicos. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos

Modelo relacional de datos: Es el más reciente de estos modelos, supera algunas de las limitaciones de los otros dos anteriores. El modelo relacional de datos representa todos los datos en la base de datos como sencillas tablas de dos dimensiones llamadas relaciones. Las tablas son semejantes a los archivos planos, pero la información en más de un archivo puede ser fácilmente extraída y combinada..

Registro: es el concepto básico en el almacenamiento de datos. El registro agrupa la información asociada a un elemento de un conjunto y está compuesto por campos.

Redundancia de datos: Duplicación de datos en archivos independientes.

Relación: Es la asociación que se efectúa entre entidades. Por ejemplo la relación entre las entidades facturas emitidas y clientes.

Relaciones: Las relaciones entre objetos son, precisamente, los enlaces que permiten a un objeto relacionarse con aquellos que forman parte de la misma organización.

Selección: Manipulación de datos que elimina filas de acuerdo con ciertos criterios.

10. Bibliografía

- ADORACIÓN DE MIGUEL CASTAÑO, MARIO G.PIATTINI VELTHUIS, (1.998) Fundamentos y Modelos de Bases de datos. Ed. Alfaomega S.A.
- ADORACIÓN DE MIGUEL CASTAÑO, MARIO G.PIATTINI VELTHUIS, (1.998) Diseño de Bases de datos Relaciónales, Editorial Alfaomega S.A.
- C.J. DATE, (2.001) Introducción a los Sistemas de Bases de datos, edición, Addison Wesley Iberoamericana.
- DEITEL Y DEITEL. Análisis y diseño de bases de datos.
- GIO WIEDERNOLD, Diseño de Bases de Datos
- HENRY F.KORTH, ABRAHAM SILBERCHATZ , Fundamentos de Bases de Datos., Editorial Mc Graw Hill, 1998.
- MIGUEL A. RODRIGUEZ ALMEIDA, (1.992) Bases de Datos, Editorial Mc Graw Hill.
- Miguel A. Rodríguez Almeida, Bases de Datos, Editorial Mac Graw Hill,1992
- Luis Hernando Rojas, Guía Procesamiento de Datos, Editorial UNAD, 1.992

SITIOS WEB

- www.postgresql.org
- www.programacion.com
- www.lawebdelprogrmador.com
- http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x332.html
- http://www.tramullas.com/nautica/documatica/2-7.html
- http://atenea.udistrital.edu.co/profesores/jdimate/basedatos1/tema3_1.htm
- http://dis.um.es/~barzana/Informatica/IAGP/IAGP_BD_Relacional.html