



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO



FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

AUTOR: CARLOS FRANCISCO MÉNDEZ CRUZ

Bases de datos		Clave:	1365
Plan:	2005	Créditos:	8
Licenciatura:	Informática	Semestre:	3
Área: Informática (Desarrollo de sistemas)		Hrs. Asesoría:	4
Requisitos:	Ninguno	Hrs. Por semana:	4
Tipo de asignatura:	Obligatoria (x)	Optativa	()

Objetivo general de la asignatura

Al finalizar el curso, el alumno obtendrá los conocimientos necesarios sobre los diferentes modelos de bases de datos, así como la metodología para construir la base de datos de un sistema informático.

Temario oficial

1. Plataforma teórico conceptual (4 h.)
2. Modelo relacional (10 h.)
3. Modelo orientado a objetos (10 h.)
4. Diseño (12 h.)
5. Construcción (10 h.)
6. Administración (12 h.)
7. Nuevas Tecnologías (6 h.)



Introducción

Una de las principales actividades profesionales a las que podrás dedicarte como licenciado en informática es al desarrollo de sistemas de información para las organizaciones. Esta labor es compleja, pero apasionante y de gran creatividad. Uno de los aspectos de mayor reto en el desarrollo de sistemas de información es el que tiene que ver con la tecnología de bases de datos organizacionales.

Por tal razón, en esta materia revisaremos los temas más importantes relacionados a la teoría, diseño, construcción y administración de una base de datos. Conocerás en qué consiste esta tecnología, que hoy en día es fundamental y necesaria para las empresas, desde una perspectiva teórica y práctica.

Así, en el primer tema, **plataforma teórico conceptual**, trataremos los fundamentos de las bases de datos mediante una revisión de los conceptos básicos para entender esta tecnología. Entre los principales están el concepto de base de datos y el concepto de un sistema manejador de bases de datos.

El segundo tema te proporciona las bases teóricas del modelo de bases de datos más utilizado actualmente en las organizaciones: el **modelo relacional**. Echarás un vistazo a la propuesta de Edgar Codd, fundador de este modelo de base de datos.

Una vez revisado el modelo relacional, atenderemos a un novedoso modelo que está ganando terreno en la industria. Nos referimos al **modelo orientado a objetos**, que si bien no tiene tanto rigor teórico como el relacional, parece una excelente opción para mejorar el desarrollo de sistemas de información.

En el tema de **diseño** de bases de datos discutirás la propuesta de modelado de Peter Chen y manejarás los conceptos para el desarrollo de un modelo Entidad-Relación. Además, este tema incluye el uso de una representación gráfica llamada Diagrama Entidad-Relación.



Cuando en un proyecto de desarrollo de sistemas hemos terminado con el diseño de la base de datos, éste se implementa en un sistema manejador de bases de datos mediante programación. Esta etapa es cubierta en el tema de **construcción** de la base de datos en el cual enumeraremos los objetos programables de un sistema de base de datos relacional. Además, revisaremos cómo se almacenan datos en tablas y cómo son recuperados con consultas. Todo el tema está basado en el lenguaje de bases de datos relacionales SQL.

Pero la labor de un experto en bases de datos no queda sólo en su construcción, también es necesario el resguardo, mantenimiento y monitoreo del funcionamiento de la misma. Todas estas actividades que realiza el experto de bases de datos serán tocadas en la unidad dedicada a la **administración** de la base de datos.

Finalmente, dado el enorme crecimiento de la cantidad de información que guardan las bases de datos en las empresas, repasarás algunos aspectos introductorios a dos **nuevas tecnologías** del manejo de grandes bases de datos: el *Data Warehousing* y la Minería de Datos.



TEMA 1. PLATAFORMA TEÓRICO-CONCEPTUAL

Objetivo particular

Al terminar el tema, reconocerás cuál es el antecedente histórico de la tecnología de bases de datos y con ello las ventajas que ésta trajo al procesamiento automatizado de información. Además, identificarás los conceptos básicos de las bases de datos y sus sistemas manejadores.

Temario detallado

1.1. Historia

1.1.1 Manejadores de archivos (campo y registro)

1.2 Definición de bases de datos

1.3 Definición de sistema administrador de bases de datos

1.3.1 Elementos

1.3.2 Modelo

1.3.3 Objetivos

Introducción

Con el fin de conocer el concepto y la importancia de las bases de datos, en este tema estudiaremos su antecedente histórico: los manejadores de archivos. Esta tecnología de almacenamiento de información tuvo un uso muy extendido entre las empresas. Consistía básicamente en archivos de datos y lenguajes de programación que accedían a ellos. A pesar de que dichos lenguajes de programación se volvieron mejores en su labor, con el tiempo la tecnología de bases de datos vino a resolver fácilmente problemas que los manejadores de archivos resolvían de forma más compleja. Es importante mencionar que hoy en día la utilización de archivos de datos no ha quedado en desuso.



Después, destacaremos las definiciones de base de datos y de sistema manejador de bases de datos. Éstas son fundamentales para la formación de un informático y son retomadas en muchas de las materias de la carrera.¹ Profundizaremos en los elementos de un sistema administrador de bases de datos, el modelo que sirve de base para su constitución y los objetivos que persigue. Esto nos dará una base conceptual para entender la importancia y repercusión de las base de datos en la vida diaria de las empresas

1.1. Historia

Es bien sabido que desde la antigüedad el hombre ha tenido la necesidad de guardar información sobre su acontecer. Por ello, en un pasado remoto, los sucesos importantes eran preservados en pinturas, grabados, papiros y después en papel. Con el paso del tiempo, la sociedad se volvió más compleja y la manera de guardar la información que ésta producía también cambió.

El surgimiento de organizaciones bien establecidas con distintos fines: económicos o sociales, trajo consigo la utilización de libros de registros. El crecimiento de estas empresas produjo que dichos registros se volvieran difíciles de manejar. Afortunadamente, la llegada de las computadoras proporcionó medios de registro y procesamiento más simples y ágiles, naciendo una nueva tecnología de almacenamiento de datos. En seguida revisaremos la primera solución tecnológica al almacenamiento de datos.

1.1.1 Manejadores de archivos (campo y registro)

El surgimiento de las computadoras brindó la posibilidad del procesamiento de grandes cantidades de datos. Esta situación requirió de la invención de una

¹ En primer lugar estaría la materia Desarrollo de Aplicaciones con Manejadores de Bases de Datos Relacionales, además, las referentes al proceso de desarrollo de sistemas, ingeniería de software y construcción de aplicaciones.



manera de almacenar el conjunto de datos que serían posteriormente procesados. La primera solución y que resolvió los problemas tecnológicos de las empresas durante mucho tiempo fueron los archivos de datos.

Con estos archivos de datos surgió la primera tecnología de almacenamiento. En ella, los datos del mundo real se representaban como un conjunto de caracteres. Cuando un conjunto de caracteres se referían a un dato particular, por ejemplo el nombre de una persona, podíamos hablar de un **campo**. El conjunto de campos relacionados entre sí de acuerdo con una asociación del mundo real formaba un **registro**, por ejemplo el nombre, edad y dirección de una persona. Finalmente, el grupo de registros asociados a un concepto determinado, digamos una nómina o el catálogo de una biblioteca, formaba un **archivo**.

Hoy en día, podemos hacer un archivo de datos tan sólo con abrir un editor de textos y formar campos y registros. Por ejemplo, en la figura 1.1 puedes ver el fragmento de un archivo de personas. Cada campo: nombre, edad y rfc, está separado por una coma (,) y en él encontramos tres registros, uno por cada línea.²



Figura 1.1. Ejemplo de archivo de datos

Al principio, estos archivos eran procesados por lenguajes de programación de aplicación general, como Pascal o C. Después fueron manejados con lenguajes

² Este tipo de archivo es conocido como archivo separado por comas o archivo de valores separados por comas, calco del inglés *Comma Separated Values* (CSV). Este no es el único formato de archivos que ha sido utilizado en tecnologías de almacenamiento. Podemos encontrar también archivos separados por tabuladores o cualquier otro carácter. Algunas veces se prefieren archivos de ancho fijo, es decir, donde cada campo es del mismo tamaño.



específicos para procesar archivos de datos, como Cobol o Clipper, Finalmente, surgieron sistemas manejadores de archivos especializados como DBase, Informix y FoxPro, en sus primeras versiones. Estos últimos comenzaron a utilizar archivos en formato binario y no sólo formato de texto o ASCII.

Estos manejadores de archivos fueron utilizados mucho tiempo para dar respuesta a las necesidades de información de las empresas. Esta situación permitió encontrar los límites y debilidades de esta tecnología. Los principales problemas eran:

- ◆ Ya que los grandes sistemas requerían de muchos archivos, mantener relacionada la información entre unos y otros a veces resultaba en programas muy complejos. Relacionado con esto, la cantidad de archivos que el sistema operativo podía mantener abiertos era otro problema.
- ◆ Por ser simples archivo de texto o binarios, era posible utilizar distintos lenguajes o programas para modificarlos, brincando las rutinas que aseguraban la relación entre archivos o las rutinas de seguridad de los mismos.
- ◆ Era común que interrupciones de energía o problemas de memoria del sistema operativo dañara los archivos cuando estaban abiertos, provocando registros perdidos.
- ◆ La complejidad de los programas para procesar los archivos de datos hizo que las personas que los desarrollaban se volvieran indispensables. De igual manera, muchos de los lenguajes quedaron en desuso o las escuelas ya no los enseñaron.



Por estos y otros problemas, la tecnología de almacenamiento y procesamiento de grandes cantidades de datos evolucionó en lo que hoy conocemos como bases de datos.³

1.2. Definición de bases de datos

Para establecer una definición del concepto de base de datos vamos a separar los datos en sí mismos, de los programas de aplicación que los procesan y controlan. En este sentido, podemos definir una **base de datos** como una colección de datos relacionados, organizados, estructurados y almacenados de manera persistente. La **persistencia** es la característica de los datos que nos permite recuperarlos en el futuro, es decir que un dato es persistente si los podemos almacenar a través del tiempo.

También, la colección de datos debe estar organizada de acuerdo con un modelo que dictará la forma de las estructuras que almacenarán los datos. Estos modelos serán abordados en los temas siguientes, en los que hablaremos preferentemente del modelo relacional, ya que es el más utilizado en las empresas.

Una base de datos es finalmente un reflejo de la realidad. Esto quiere decir que a partir de observar un hecho del mundo, podemos modelarlo en términos de datos y crear una estructura que los almacene. En este sentido, y siendo estrictos, una base de datos no necesariamente debe estar computarizada, pero hoy en día no es fácil concebirlo así. Las organizaciones privadas y públicas de nuestra actualidad ya no pueden existir sin una base de datos computarizada que les brinde información veraz y oportuna para su toma de decisiones.

Para terminar este tema, debemos puntualizar que una base de datos requiere de programas que procesen, recuperen, compartan, aseguren y controlen sus datos.

³ Para leer algo más sobre la historia de los sistemas de bases de datos revisa esta bibliografía: (Silberschatz, 2006: 22-24).



El conjunto de programas que hacen esto conforman lo que llamaremos Sistema Administrador de Bases de Datos, y que estudiaremos en la siguiente sección.

1.3 Definición de sistema administrador de bases de datos

Una vez que contamos con una colección de datos, surge la necesidad de programas de aplicación que nos permitan almacenar, procesar, recuperar, compartir y asegurar esos datos, a este conjunto de programas lo llamaremos **Sistema Administrador de Bases de Datos**. Estos sistemas son conocidos también como *Sistemas gestores de bases de datos*, *Sistemas manejadores de bases de datos*, *Sistemas de bases de datos* o *DBMSs*, por las siglas del inglés *Database Management Systems*.

Los sistemas de base de datos permiten manejar grandes volúmenes de información. Son ellos los que brindan posibilidades de modificar y recuperar datos de forma ágil. Además, un sistema de base de datos debe tener mecanismos de seguridad que garanticen la integridad de la información y que impidan intentos de accesos no autorizados. Esta seguridad se vuelve aún más importante porque los datos están compartidos para muchos usuarios al mismo tiempo en una red de cómputo.

Con el fin de reafirmar el concepto de base de datos y de sistema administrador de base de datos, vamos a exponer algunas definiciones provistas por varios autores. Recopilo de diversas fuentes varias definiciones y te las presento en el siguiente cuadro (Cuadro 1.1).



Autor	Definición
C. J. Date	Una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada. (2001: 10)
James L. Johnson	<p>Una <i>base de datos</i> es un conjunto de elementos de datos que se describe a sí mismo, con relaciones entre esos elementos, que presenta una interfaz uniforme de servicio. Un <i>sistema de administración de bases de datos</i> (DBMS) es un producto de software que presta soporte al almacenamiento confiable de la base de datos, pone en marcha las estructuras para mantener relaciones y restricciones, y ofrece servicios de almacenamiento y recuperación a usuarios; más funciones se ocupan de otras tareas, como son el acceso simultáneo, seguridad, respaldo y recuperar (lectura) de datos. (1997: 8)</p> <p>Un sistema de administración de bases de datos (DBMS) proporciona el método de organización necesario para el almacenamiento y recuperación flexibles de grandes cantidades de datos. (1997: 3)</p>
Abraham Silberschatz	Un sistema gestor de bases de datos (SGBD) 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 SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto <i>práctica</i> como <i>eficiente</i> . (2006: 1)

Cuadro 1.1. Definiciones de bases de datos y sistemas administradores de bases de datos



Una de las principales ventajas que ofrece el uso de un sistema de administración de bases de datos es la división de niveles de abstracción de datos. En el cuadro de abajo (Cuadro 1.2) presento en dos columnas los tres niveles y su descripción.⁴

Nivel	Descripción
Nivel físico o interno	En este nivel se describe cómo están almacenados físicamente los datos.
Nivel conceptual o lógico	Describe la base de datos en término de estructuras de almacenamiento. Este conjunto de estructuras es también llamado esquema . Las estructuras están basadas en el modelo de datos que seleccionemos.
Nivel externo o de vistas	Es un conjunto de vistas a los datos que ocultan la base completa y están orientados a usuarios específicos.

Cuadro 1.2. Niveles de abstracción

Un sistema administrador de bases de datos debe incluir un conjunto de lenguajes que le permitan definir estructuras de almacenamiento, manipular y consultar datos, y controlar su acceso. En la práctica, estos lenguajes se encuentran unidos en uno solo, como el lenguaje SQL que revisaremos en el tema II. El cual detalla estos lenguajes.

Lenguaje

La división de lenguajes no es consistente entre los distintos autores, algunos consideran que son sólo dos: DML y DDL. Es común que se diga que el DML incluye al DQL y el DDL al DCL; así lo hace, por ejemplo, Silberschatz (2006: 6).

Para terminar esta sección, creemos pertinente mencionar que un DBMS cuenta con una arquitectura. Ésta muestra la interacción de los distintos programas

⁴ Si quieres profundizar en los niveles de abstracción de un DBMS, revisa el texto de Date (2001: 33-40) pues allí extiende la explicación de estos niveles en el capítulo 2.



involucrados en la operación del sistema, es decir, cómo son procesadas las peticiones del usuario y cómo son manipulados los datos. Presentamos a continuación la arquitectura propuesta por Date (2001: 45) a manera de ejemplo (Figura 1.2.). Así que confronta esta arquitectura con la de Johnson (1997: 17) y la de Silberschatz (2006: 20).

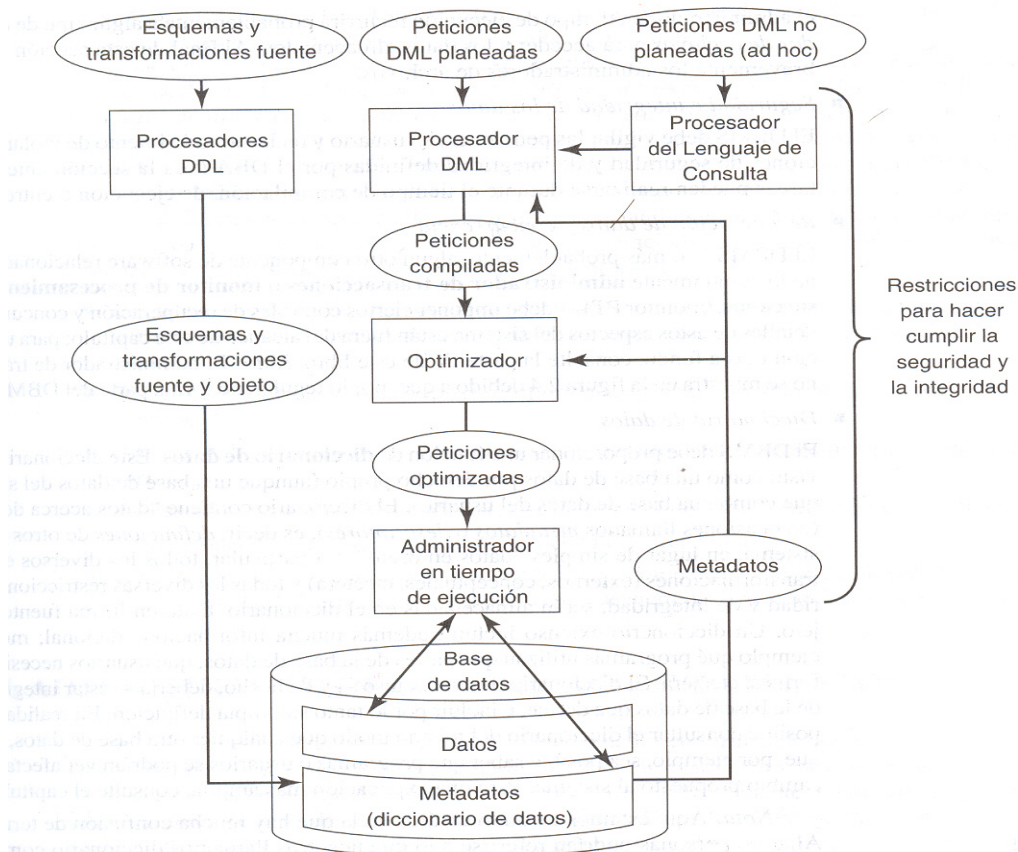


Figura 2.4 Funciones y componentes principales del DBMS.

Figura 1. 2. Arquitectura de un DBMS

1.3.1 Elementos

Para Date (2001: 5), un sistema de administración de base de datos comprende cuatro elementos: **datos**, **hardware**, **software** y **usuarios**.

Los datos deben estar disponibles para varios usuarios al mismo tiempo, esto significa que el DBMS proporciona **conurrencia de datos**. Además, deben estar





protegidos contra caídas del sistema e intentos de modificación por personas ajenas a la organización.

El software de un sistema administrador de bases de datos debe ser instalado en computadoras con características de hardware suficientes para brindar buen desempeño. Hoy en día, existen fabricantes especializados en sistemas de cómputo idóneos para bases de datos corporativas. Por lo general, basta con ponerse en contacto con ellos y exponerles las necesidades de información y las proyecciones de tamaño de nuestra base de datos.

Un DBMS comprende también un software encargado de hacer las gestiones con el sistema operativo y de dar los servicios de cómputo de la base de datos. Cuando este software está en funcionamiento, es frecuente llamarle **servidor de base de datos**. Este software incluye programas especializados para actualizar, recuperar, asegurar y compartir los datos de la base. Es habitual referirse al sistema administrador de bases de datos como un producto de software ofrecido por alguna compañía tecnológica. En el siguiente cuadro (Cuadro 1.3) listo algunos de los manejadores comerciales y de software libre más conocidos:

Compañía	Software	Tipo
Oracle	Oracle http://www.oracle.com	Comercial
Microsoft	SQL Server http://www.microsoft.com	Comercial
PostgreSQL Developer Group	PostgreSQL http://www.postgresql.org	Libre
	MySQL http://www.mysql.com	Libre
IBM	DB2 Universal Database	Comercial

Cuadro 1.3. Manejadores de bases de datos comerciales y libres



Los usuarios que entran en juego en un sistema de bases de datos son principalmente los programadores de aplicaciones, programadores de bases de datos, los usuarios finales y el administrador de bases de datos. Los primeros se encargan de programar las interfaces gráficas que usarán los usuarios finales para almacenar y recuperar datos de la base. Esta actividad la realizan con distintos entornos de desarrollo mediante varios lenguajes de programación (java, php, c++). Los segundos crean las estructuras de almacenamiento y los objetos de base de datos necesarios para procesar los datos. Estos objetos serán revisados en el tema V del temario.

Por otro lado, los usuarios finales son muy importantes ya que determinan las necesidades de información que deberá cubrir el sistema administrador de base de datos y finalmente serán los que alimentarán la base de datos. El administrador de la base de datos, llamado DBA por el inglés *Database Administrator*, es el encargado de llevar a cabo las tareas necesarias para un funcionamiento óptimo del DBMS, es común también que diseñe la base de datos y establezca las configuraciones necesarias al nivel de software y de seguridad. Las actividades del DBA se verán con mayor amplitud en el tema VI.

1.3.2 Modelo

Un modelo de datos es una “colección de herramientas conceptuales para describir los datos, sus relaciones, su semántica y las restricciones de consistencia” (Silberschatz 2006: 6). Existen dos modelos principales: el relacional y el orientado a objetos. Adoptamos un determinado modelo para crear la base de datos, de esta manera las estructuras de almacenamiento y sus relaciones estarían basadas en principios preestablecidos por el modelo. Por ejemplo, si nos decidimos por el modelo orientado a objetos tendremos a nuestra disposición para construir la base de datos los conceptos de herencia, polimorfismo y encapsulación. Repasaremos este modelo en el tema III.



Hoy en día, el modelo más extendido y utilizado es el relacional, que surgió a raíz de la propuesta de Edgar Codd en los años 70; sobre éste profundizaremos en el tema II.

1.3.3 Objetivos

Los objetivos principales de un sistema de base de datos son disminuir los siguientes aspectos:

Redundancia e inconsistencia en los datos. Es necesario evitar, en la medida de lo posible, la información repetida ya que aumenta el costo de almacenamiento y puede provocar problemas en el acceso a los datos. La inconsistencia en los datos se da cuando se pierde la relación lógica entre la información, por ejemplo, permitir que en la base de datos se registre un cargo sin su correspondiente abono.

Dificultad para tener acceso a los datos. Un DBMS debe cubrir las necesidades de información del usuario mediante un lenguaje de consultas sólido, esto implica prevenir cualquier petición o situación posible de ser solicitada.

Aislamiento de los datos. Antes del surgimiento de los sistemas administradores de bases de datos se utilizaban grupos de archivos por cada departamento de la empresa, los cuales muchas veces eran de distintos tipos, textuales o binarios, y eran tratados mediante diversos lenguajes de programación. Dicha situación causaba problemas para tener información centralizada. Los sistemas de bases de datos deben permitir la centralización de datos reduciendo su aislamiento.

Anomalías de acceso concurrente. Evitar inconsistencias por actualizaciones de usuarios que acceden al mismo tiempo a la base de datos. Era común que los administradores de archivos tuvieran problemas con la concurrencia.



Problemas de seguridad. La información que se guarda en una base de datos no debe ser vista con la misma profundidad por todos los usuarios de la misma. Por esta razón, el DBMS debe admitir niveles de usuarios y restricciones para consultar la información. También se requieren niveles de seguridad en contra de *haking* o *craking*.

Problemas de integridad. Los datos que ingresan a una base deben estar bien filtrados de manera que no se almacene información errónea o sin el formato adecuado. Para esto será necesario que el DBMS tenga mecanismos para implementar restricciones de integridad basadas en reglas de negocio.

Hemos expuesto arriba una cantidad considerable de conceptos asociados a la tecnología de bases de datos. Dos de ellos son los fundamentales: base de datos y sistema manejador de base de datos. Hoy en día, es prácticamente imposible imaginar una organización que no utilice bases de datos como parte de su labor cotidiana. Por ello es importante que seas capaz de reconocer los fundamentos expuestos en este tema.

Como te habrás dado cuenta, las bases de datos vinieron a mejorar la tecnología de almacenamiento de datos y se han vuelto indispensables gracias a los beneficios que ofrecen los DBMSs actuales. También notaste que conocer esta tecnología requiere de estudiar a los sistemas de bases de datos, sus elementos y modelos asociados. Por esto, en el siguiente tema abordaremos las especificaciones del modelo de datos más utilizado en la actualidad, el modelo relacional.



Bibliografía del tema 1

Date, C. J., (2001), *Sistemas de Bases de Datos*, 7ª ed., México, Pearson Education.

Elmasri, Ramez, (2002), *Fundamentos de sistemas de bases de datos*, México, Pearson Educación y Addison-Wesley.

Johnson, James L., (1997) *Bases de datos. Modelos, lenguajes, diseño*, México, Oxford University Press.

Silberschatz, A., et. al., (2006), *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw Hill.

Actividades de aprendizaje

A.1.1. Elabora un mapa conceptual que integre y relacione todos los conceptos presentados en el tema.

A.1.2. A partir del cuadro 1.1 construye tus propias definiciones de base de datos y de sistema administrador de base de datos.

A.1.3. Elabora un cuadro comparativo con el resultado de la confrontación de las arquitecturas de un DBMS propuestas por Date (2001: 45), Johnson (1997: 17) y Silberschatz (2006: 20).

A.1.4. Elabora un ensayo sobre el concepto de “modelo de datos” basándote en la bibliografía del tema.

Cuestionario de autoevaluación

1. ¿Qué son campo y registro?
2. ¿Qué es un archivo de datos?
3. ¿En qué consiste la tecnología de los manejadores de archivos?
4. ¿Cuáles son los problemas de la tecnología de los manejadores de archivos?
5. Define el concepto de base de datos.
6. Define un sistema administrador de bases de datos.
7. ¿Cuáles son los lenguajes de datos de un DBMS?
8. Describe cada uno de los elementos de un sistema de base de datos.



9. ¿Qué entiendes por un modelo de datos?
10. Explica tres objetivos de un DBMS.

Examen de autoevaluación

1. Un campo es un conjunto de registros.
a) Verdadero b) Falso
2. Un archivo de datos es un conjunto de campos relacionados entre sí.
a) Verdadero b) Falso
3. La persistencia es una característica de los datos.
a) Verdadero b) Falso
4. Un sistema administrador de bases de datos permite almacenar, recuperar y compartir datos.
a) Verdadero b) Falso
5. Un sistema de bases de datos brinda tres niveles de abstracción de datos.
a) Verdadero b) Falso
6. Todo sistema manejador de bases de datos incluye lenguajes de manipulación y definición de datos.
a) Verdadero b) Falso
7. Un sistema de bases de datos incluye cuatro elementos: datos, hardware, software y usuarios.
a) Verdadero b) Falso
8. La concurrencia de datos permite que sean recuperados en el futuro.
a) Verdadero b) Falso
9. El DBA es uno de los usuarios de un sistema administrador de bases de datos.
a) Verdadero b) Falso
10. Los dos modelos principales de bases de datos son el extendido y el redundante.
a) Verdadero b) Falso



TEMA 2. MODELO RELACIONAL

Objetivo particular

Al terminar el tema, reconocerás las características teóricas que conforman el modelo relacional de base de datos. Además, identificarás los elementos del mismo modelo, las reglas propuestas por Edgar Codd y el proceso de normalización de relaciones.

Temario detallado

2.1 Introducción

2.1.1 Modelos pre-relacionales

2.1.2 Modelos pos-relacionales

2.2 Definición de relación

2.2.1. Partes

2.3. Propiedades de una relación

2.4 Dominio y tipos de datos

2.5 Álgebra relacional y cálculo relacional

2.6 Normalización

2.6.1. Formas normales

2.6.2. Proceso de descomposición sin pérdida

2.7 Reglas de CODD

2.8 Estándar SQL



Introducción

El modelo relacional de base de datos surge a finales de los 60, no obstante, es hoy en día el modelo más utilizado en sistemas empresariales. Los principales manejadores de bases de datos comerciales o de software libre están basados en este modelo y brindan soluciones tecnológicas robustas para todo tipo de empresas. Es por estas razones que el licenciado en informática debe conocer el modelo relacional de base de datos.

En este tema, revisaremos algunas características de los modelos pre-relacionales y pos-relacionales para brindar parámetros de diferenciación con el modelo relacional. También, estudiaremos los fundamentos teóricos del modelo a partir de los conceptos de relación, dominio, álgebra y cálculo relacional.

En el proceso de desarrollo de una base de datos relacional, resulta importante evitar problemas de redundancia y de actualización de datos, por esto repasaremos el procedimiento conocido como normalización, el cual se basa en la descomposición sin pérdida de distintas relaciones para ajustarlas a formas normales.

Fue Edgar F. Codd quien puso las bases de este modelo y formuló lo que hoy se conoce como las “12 reglas de Codd”. En este tema dedicaremos un apartado a repasar estas reglas. Finalmente, discutidos los fundamentos del modelo, brindaremos aspectos generales del lenguaje estándar de desarrollo de bases de datos relacionales llamado SQL.

2.1. Introducción a los modelos

A finales de 1968, Edgar F. Codd, matemático investigador de IBM, propuso el uso de las matemáticas para dar cierto rigor al campo de las bases de datos. Codd puso sus ideas en un artículo que hoy es clásico: “A Relational Model of Data for



Large Shared Data Banks”⁵. En ese artículo y subsiguientes, Codd presenta los conceptos fundamentales del modelo y sus beneficios frente a la tecnología de ese tiempo.

Tal como lo mencionamos en el tema 1, la manera de manejar datos antes del advenimiento de las bases de datos era mediante archivos de datos. El uso de estos sistemas de archivos causaba distintos problemas que provocaban pérdidas de datos, datos duplicados innecesariamente, datos incompletos y erróneos.

Los investigadores en computación de aquellos tiempos comenzaron a proponer nuevas opciones de almacenamiento hasta que surgieron los primeros modelos de bases de datos. A continuación hablaremos de algunos de ellos.

2.1.1 Modelos pre-relacionales

Son básicamente dos modelos los antecesores del modelo relacional de base de datos. Ambos están basados en una estructura de nodos interconectados que almacenan la información.

El primero es el modelo jerárquico, que interconecta nodos en una jerarquía estricta de padre e hijos, donde no podía haber relación entre nodos de distintos niveles o entre los del mismo nivel. Este modelo fue útil hasta que se empleó para resolver problemas de almacenamiento más complejos. La interconexión de nodos y por consiguiente el uso de apuntadores comenzó a ser un inconveniente difícil de manejar por los sistemas de aplicación. El otro problema principal de este modelo es que no puede implementar relaciones de M:M entre instancias de entidades del mundo.

⁵ E. F. Codd. “A Relational Model of Data for Large Shared Data Banks” en *CACM* 13, no. 16 (junio, 1970).



El segundo modelo es el de red. Se trata de una interconexión de nodos mediante apuntadores, sin la restricción del jerárquico, es decir, pueden salir de cada nodo varios arcos apuntando a otros nodos. A diferencia del modelo jerárquico, en este modelo se permiten las conexiones entre nodos de cualquier tipo, por lo que resulta bueno para relaciones M:M. Su principal desventaja radica en los problemas de implementación para lograr un rendimiento óptimo.

2.1.2 Modelos pos-relacionales

Los modelos pos-relacionales más conocidos son el modelo orientado a objetos, el distribuido, el deductivo, el multidimensional y el modelo semiestructurado.

Tendremos la oportunidad de revisar el modelo orientado a objetos en el tema 3, así que no diremos nada sobre él en este apartado. El modelo distribuido está basado en el modelo relacional o en el orientado a objetos y su principal característica es que la base de datos está fragmentada en una red. Así, partes de las tablas de usuarios se encuentran en distintos lugares ya sea a nivel horizontal (renglones) o a nivel vertical (columnas). Uno de los retos principales de estos manejadores es que deben presentar la información al usuario como si las tablas estuvieran almacenadas localmente. Tanto la consulta como la actualización de datos deben estar garantizadas desde cualquier punto de la red.

El modelo deductivo se apoya en un conjunto de datos y reglas de inferencia. Es conocido también como modelo inferencial o modelo basado en la lógica. Las consultas de datos en este modelo son demostraciones de axiomas deductivos. El objetivo primordial de este tipo de bases de datos es inferir ciertos datos a partir de las reglas.

Las bases de datos multidimensionales son llamadas así porque los datos están almacenados en arreglos multidimensionales. Estos arreglos pueden ser interpretados en términos de variables dependientes e independientes. Las últimas



determinan las dimensiones sobre las que se guardan valores (variable dependiente). Este modelo forma parte de los sistemas conocidos como MOLAP (procesamiento analítico en línea multidimensional) que tiene por objeto la generación de informes analíticos de datos.

Por último, el modelo semiestructurado surge ante la necesidad de que cada elemento almacenado cuente con un conjunto de atributos propio, sin necesidad de que este conjunto sea obligatorio como sucede en el modelo relacional. Este tipo de bases de datos se crean con el lenguaje de marcado extensible XML.

2.2 Definición de relación

El modelo relacional de bases de datos es llamado así porque se basa en estructuras de almacenamiento de datos llamadas relaciones. Las dos características fundamentales del modelo relacional de bases de datos son:

1. Los datos son percibidos por el usuario como relaciones y nada más que relaciones.
2. Para consultar los datos, el usuario cuenta con operadores que generan nuevas relaciones a partir de otras (álgebra relacional).

El modelo relacional se fundamenta en una teoría abstracta de datos que toma ciertos aspectos de las matemáticas (teoría de conjuntos y lógica de predicados). De esta manera, podemos definir una relación como un conjunto de tuplas y atributos, y que se compone de dos partes: encabezado y cuerpo.

2.2.1 Partes

Como te mencionaba, una relación se forma de encabezado y cuerpo. El encabezado de una relación denota un cierto predicado o función valuada como verdadera. Por su parte, cada fila en el cuerpo denota una cierta proposición



verdadera obtenida del predicado por medio de la sustitución de ciertos valores, del tipo apropiado, en los indicadores de posición o parámetros de ese predicado. En otras palabras, podemos decir que el cuerpo es un conjunto de instancias o ejemplos del predicado. Veámoslo en un ejemplo:

Predicado:

El empleado NUMEMP se llama NOMBRE_EMP, trabaja en el departamento NUMDEPTO y gana un salario SALARIO_EMP

Proposiciones verdaderas obtenidas del predicado:

El empleado E1 se llama Azucena López, trabaja en el departamento D1 y gana el salario 20,000.

El empleado E2 se llama Alberto Juárez, trabaja en el departamento D1 y gana el salario 8,000.

Si representamos gráficamente lo anterior tendríamos una relación como la siguiente:

NUMEMP	NOMBRE_EMP	NUMDEPTO	SALARIO_EMP
E1	Azucena López	D1	20,000
E2	Alberto Juárez	D1	8,000

Ahora bien, desde la perspectiva de la teoría de conjuntos, el encabezado es un conjunto de n atributos necesariamente distintos de la forma NombredeAtributo:NombredeTipo. Mientras que el cuerpo es un conjunto de m tuplas en donde cada una es un conjunto de componentes de la forma NombredeAtributo:ValordeAtributo. El número de atributos de una relación se conoce como grado y el número de tuplas como cardinalidad.



2.3 Propiedades de una relación

Una relación debe contar con cuatro propiedades que son:

1. No existen tuplas duplicadas. La teoría de conjuntos descarta los elementos duplicados, además, no podemos afirmar dos veces el mismo hecho verdadero. La siguiente no es una relación:

NUMEMP	NOMBRES	APELLIDOS
1	Arturo	Jiménez
1	Arturo	Jiménez

2. Las tuplas están en desorden. La teoría de conjuntos nos dice que sus elementos están en desorden. Esto implica que no existen los conceptos de “tupla 1”, “siguiente tupla”, “cuarta tupla”; entonces nos debemos referir a una tupla por su proposición verdadera.
3. Los atributos están en desorden. Esto es porque el encabezado es un conjunto de atributos y los conjuntos no permiten elementos duplicados. Por tanto, tampoco hay conceptos de “siguiente atributo”, “primer atributo”, siempre se hace referencia a ellos por nombre, nunca por su posición
4. Los atributos deben contener valores atómicos. Esto significa que cada valor para cada tupla en un determinado atributo no debe ser semánticamente divisible y por tanto tiene que representar un valor único. La siguiente no es una relación, debido a que el atributo DEPTO no guarda valores atómicos:

NUMEMP	NOMBRES	APELLIDOS	DEPTO
1	Arturo	Jiménez	01 Finanzas
2	Arturo	Jiménez	02 Contabilidad

En una relación es posible encontrar tres tipos de claves:

Clave candidata

Es un atributo que identifica de manera única a cada tupla de una relación. Una relación puede tener varias claves candidatas y estas pueden ser simples,





formadas por un atributo, o compuestas cuando se forman por varios atributos. A las últimas se les llama superclaves.

Clave principal

Una clave principal es una clave candidata seleccionada para ser la clave principal de la relación. En este caso, a diferencia de las claves candidatas, sólo existe una por relación. También pueden ser superclaves.

Clave foránea

Una clave foránea es un atributo que hace referencia a una clave principal en otra relación. Esta referencia implica que no pueda existir un valor en la clave foránea que no exista primero en la clave principal.

2.4 Dominio y tipos de datos

Un dominio es el conjunto de valores válidos para un atributo. De esta manera, podríamos decir que cada atributo tendrá asociado siempre un dominio. Es común relacionar este concepto con el de tipo de dato, pero no son lo mismo. Un dominio es más restrictivo en cuanto a la semántica de lo que guarda, por ejemplo, el dominio CIUDAD es el conjunto de todas las ciudades posibles, misma que se expresan en caracteres alfanuméricos mediante un tipo de dato carácter (char, varchar) que podría aceptar cualquier combinación de estos.

Además, es posible utilizar ciertas restricciones para que los dominios acepten sólo determinados valores, pero siempre sobre la base de un tipo de dato simple como integer o char. Los tipos de datos están predefinidos en nuestro manejador de base de datos y contamos con la posibilidad de hacer conversiones entre distintos tipos.



2.5 Álgebra relacional y cálculo relacional

Álgebra relacional

El álgebra relacional es el conjunto de operadores que permiten manipular las relaciones (consultar datos). Estos operadores siempre generan nuevas relaciones a partir de otras. Las operaciones de álgebra relacional son:

Restricción

Regresa una relación que contiene todas las tuplas de una relación especificada, que satisfacen una condición. Un ejemplo sería:

R(ALUMNO)			R(RESULTADO)	
IDALUMNO	NOMBRE		IDALUMNO	NOMBRE
1	Jaime	Restricción IDALUMNO = 2	2	Raúl
2	Raúl			
3	Aurora			

Proyección

Regresa una relación que contiene un subconjunto de atributos de una relación especificada. Por ejemplo:

R(ALUMNO)			R(RESULTADO)
IDALUMNO	NOMBRE		NOMBRE
1	Jaime	Proyección NOMBRE	Jaime
2	Raúl		Raúl
3	Aurora		Aurora

Producto

Regresa una relación que contiene todas la tuplas posibles obtenidas por la combinación de todas las tuplas de la primera relación con todas las tuplas de la segunda relación. El siguiente es un ejemplo:



R(A)			R(B)		R(RESULTADO)	
IDA		PRODUCTO	IDB		IDA	IDB
1			10		1	10
2			20		1	20
					2	10
					2	20

Unión

Regresa una relación que contiene todas las tuplas que aparecen en las dos relaciones, omitiendo las tuplas duplicadas. Mira el siguiente ejemplo:

R(A)			R(B)		R(RESULTADO)	
IDA	CVE	UNIÓN	IDB	CVE	IDAB	CVE
1	A		2	B	1	A
2	B		3	C	2	B
			4	D	3	C
					4	D

Intersección

Regresa una relación que contiene todas las tuplas que aparecen en las dos relaciones especificadas. El siguiente es un ejemplo:

R(A)			R(B)		R(RESULTADO)	
IDA	CVE	INTERSECCIÓN	IDB	CVE	IDAB	CVE
1	A		2	B	2	B
2	B		3	C		
			4	D		



Diferencia

Regresa una relación que contiene todas las tuplas que aparecen en la primera pero no en la segunda de las dos relaciones específicas. Por ejemplo:

R(A)			R(B)			R(RESULTADO)	
IDA	CVE		IDB	CVE		IDAB	CVE
1	A	DIFERENCIA	2	B		1	A
2	B		3	C			
			4	D			

Junta (join)

Regresa una relación que contiene tuplas de dos relaciones a partir de la combinación de valores de uno o varios atributos en común. De no existir una columna en común, no sería posible la junta. Observa el siguiente ejemplo:

R(PROD)			R(VTA)			R(RESULTADO)	
IDP	NOMBRE		IDP	CANT		NOMBRE	CANT
1	Aretes	JUNTA IDP = IDP	2	100		Pulseras	100
2	Pulseras		1	30		Aretes	30
3	Collares		3	250		Collares	250

División

A partir de dos relaciones unarias $r(A)$ y $r(B)$ y una relación binaria $r(C)$, regresa una relación que contiene todas las tuplas de $r(A)$ siempre y cuando su combinación con todas las tuplas de $r(B)$ aparezca en $r(C)$. Observa bien el siguiente ejemplo y verifica por qué esas tuplas de $r(A)$ aparecen en la relación RESULTADO.



R(A)	R(B)	R(C)			R(RESULTADO)
IDA	IDB	IDA	IDB	DIVISIÓN	IDA
1	10	1	10		2
2	20	2	10		
		2	20		

El resultado es 2 ya que es la única tupla de R(A) que al combinarse con todas las de R(B) (2-10, 2-20) aparece en R(C).

Cálculo relacional

Es posible identificar dos tipos de cálculo relacional: el cálculo de tuplas y el cálculo de dominios. En ambos casos no se especifica cómo realizar la consulta, esto significa que se indica únicamente una proposición verdadera que deben cumplir las tuplas o los dominios; en el álgebra relacional se indica el procedimiento para resolver la consulta, en el cálculo no. El principal operador de este tipo de consultas es el operador “existe” (*exists* en SQL).

2.6. Normalización

La normalización es el proceso de ajuste de relaciones a ciertas reglas llamadas formas normales. Tiene por objetivo reducir los problemas de redundancia y actualización en las relaciones; su ventaja es que lo hace mediante un procedimiento bien formalizado.

A continuación te presentamos la definición de las formas normales. E. F. Codd propuso las tres primeras y con la aportación de posteriores investigadores en computación se establecieron las demás.



2.6.1 Formas normales

Como habíamos mencionado, las formas normales son reglas estrictas que deben cumplir las relaciones para disminuir problemas de redundancia y actualización. Varias de ellas están basadas en el concepto de Dependencia Funcional que explicaremos brevemente.

Dependencias funcionales

Una dependencia funcional (DF) es una relación muchos a uno que va de un conjunto de atributos a otro dentro de una determinada relación.

En otras palabras: sea R una relación y sean X y Y subconjuntos cualesquiera del conjunto de atributos de R , entonces podemos decir que Y es dependiente funcionalmente de X , en símbolos $X \rightarrow Y$ ⁶, si y sólo si en todo valor válido posible de R , cada valor X está asociado precisamente con un valor de Y ; siempre que dos tuplas coincidan en su valor X , también coincidirán en su valor Y .

La parte izquierda de una DF se denomina determinante y la derecha dependiente.

Dependencias triviales y no triviales

Una dependencia trivial se da si y solo si la parte derecha es un subconjunto de la parte izquierda. Las no triviales son la que no son triviales. Estas son las que realmente importan en el proceso de normalización mientras que las otras no son tomadas en cuenta.

Dependencias transitivas

Supongamos que tenemos una relación R con tres atributos: A , B y C , tales que las DFs $A \rightarrow B$ y $B \rightarrow C$ son válidas para R . Entonces es fácil ver que la DF $A \rightarrow C$ también es válida en dicha relación. Aquí la DF $A \rightarrow C$ es un ejemplo de DF transitiva y decimos que C depende de A transitivamente a través de B .

⁶ También se puede decir que “ X determina funcionalmente a Y ”.



Cierre de un conjunto de dependencias

Al conjunto de todas las DFs implicadas en una relación se le llama cierre de dependencias y se escribe S^+ . Para obtener este cierre se utilizan reglas de inferencia para obtener nuevas DFs a partir de las ya dadas. Los axiomas de Armstrong nos permiten realizar esta labor y son los siguientes:

Sean A, B y C subconjuntos cualesquiera del conjunto de atributos de la relación dada R, entonces pueden ser aplicados los siguientes axiomas para producir nuevas DFs:

Reflexividad: si B es subconjunto de A, entonces $A \rightarrow B$.

Aumento: si $A \rightarrow B$, entonces $AC \rightarrow BC$.

Transitividad: si $A \rightarrow B$ y $B \rightarrow C$, entonces $A \rightarrow C$.

Autodeterminación: $A \rightarrow A$.

Descomposición: si $A \rightarrow BC$, entonces $A \rightarrow B$ y $A \rightarrow C$.

Unión: si $A \rightarrow B$ y $A \rightarrow C$, entonces $A \rightarrow BC$.

Composición: si $A \rightarrow B$ y $C \rightarrow D$, entonces $AC \rightarrow BD$.

Te pongo a continuación un ejemplo del uso de estos axiomas. Dadas las siguientes DF:

1) $A \rightarrow BC$

2) $B \rightarrow E$

3) $CD \rightarrow EF$

Se pueden producir nuevas DFs a partir de la aplicación de los siguientes axiomas:

4) $A \rightarrow C$ por descomposición de 1.

5) $AD \rightarrow CD$ por aumento en 4.

7) $AD \rightarrow EF$ por transitividad entre 5 y 3.

8) $AD \rightarrow F$ por descomposición en 7.



Dependencias irreducibles

Una dependencia funcional se considera irreducible si la parte derecha involucra sólo un atributo y no es posible descartar ningún atributo del determinante sin cambiar el cierre. En otras palabras, en el caso de que la relación presente una clave principal compuesta, todo atributo a la izquierda de la DF debe depender de toda la clave y no sólo de una parte.

Primera forma normal (1FN)

Una relación está en 1FN si y sólo si toda tupla contiene exactamente un valor para cada atributo. Este concepto es el que ya habíamos explicado como la propiedad de una relación, por lo que muchos autores proponen que toda relación está en 1FN por definición.

Segunda forma normal

Una relación está en 2FN si y sólo si está en 1FN y todo atributo que no sea clave es dependiente irreduciblemente de la clave primaria. Podemos decir, en otras palabras que la 2FN exige que todas la DFs de una relación sean irreducibles.

Tercera forma normal

Una relación está en 3FN si y sólo si está en 2FN y todos los atributos que no son clave son dependientes en forma no transitiva de la clave primaria. La 3FN elimina la posibilidad de DFs transitivas.

Forma normal de Boyce/Codd

Una relación está en FNBC si y sólo si toda DF no trivial, irreducible a la izquierda, tiene una clave candidata como su determinante.

2.6.2. Proceso de descomposición sin pérdida

Pero qué sucede si una relación no cumple plenamente con una determinada forma normal. Como te habrás dado cuenta, las formas normales también son



acumulativas, es decir, no es posible estar en una forma normal más alta sin cumplir las formas normales inferiores.

Bueno, si una relación no cumple una forma normal, debemos descomponerla mediante un proceso conocido como descomposición sin pérdida. Éste consiste en una proyección de la relación para obtener nuevas relaciones que cumplan la forma normal exigida, y decimos que es sin pérdida si al juntar de nuevo las proyecciones regresamos a la relación original, preservando tanto el grado como la cardinalidad.

Revisemos ahora las llamadas formas normales superiores: 4FN y 5FN, para lo cual será necesario estudiar dos tipos más de dependencias.

Dependencias multivaluadas

Las dependencias multivaluadas se dan entre dos atributos. Uno de ellos caracteriza o determina a un conjunto bien definido de valores del otro atributo. Esta caracterización es independiente de otros atributos. Las dependencias multivaluadas se dan entre atributos multivaluados independientes entre sí donde se establece una combinatoria de todos contra todos. Veamos un ejemplo ilustrativo de De Miguel (2000: 178).

R(ASIGNATURAS)

NOM_ASIGNATURA	PROFESOR	TEXTO
Ficheros y BD	Sr. Sánchez Sra. Hidalgo	Concepción y diseño de BD Fundamentos de BD
BD avanzadas	Sra. Hidalgo Sr. Martín	Diseño de BD avanzadas

Podemos hacer las siguientes observaciones sobre el ejemplo:

Un profesor debe utilizar todos los textos. Por esto, un profesor no puede determinar un texto.

La clave primaria sería una superclave con los tres atributos.



Si normalizamos el modelo inicial obtendríamos una relación en FNBC, pero que tendría mucha redundancia.

R(ASIGNATURAS)

NOM_ASIGNATURA	PROFESOR	TEXTO
Ficheros y BD	Sr. Sánchez	Concepción y diseño de BD
Ficheros y BD	Sr. Sánchez	Fundamentos de BD
Ficheros y BD	Sra. Hidalgo	Concepción y diseño de BD
Ficheros y BD	Sra. Hidalgo	Fundamentos de BD
BD avanzadas	Sra. Hidalgo	Diseño de BD avanzadas
BD avanzadas	Sr. Martín	Diseño de BD avanzadas

Una dependencia multivaluada $X \twoheadrightarrow Y$ se da cuando para cada valor de X hay un conjunto de cero o más valores de Y, independientes de los valores de los otros atributos de la relación. La siguiente relación, tomada de De Miguel (2000: 180), no tiene una dependencia multivaluada ya que el texto “Modelo Relacional” no aparece en inglés.

R(CURSOS)

COD_CURSO	TEXTO	IDIOMA
A2783	Introducción a las BD	Español
A2783	Introducción a las BD	Inglés
A2783	Modelo relacional	Español
B2341	Concepción y diseño de BD	Francés
B2341	Modelo relacional	Español

Cuarta forma normal (4FN)

Una relación está en cuarta forma normal si, y sólo si, las dependencias multivaluadas tienen como determinante una clave. En este sentido, la relación



ASIGNATURA no está en 4FN, por lo que es necesario descomponerla en dos relaciones:

ASIGNATURA_PROFESOR (Nom_asignatura, Profesor)

ASIGNATURA_TEXTO (Nom_asignatura, Texto)

Dependencias de combinación

Este tipo de dependencia se da cuando existe interdependencia entre los atributos de una relación y la descomposición en dos relaciones causa pérdida de información. Por tanto, la descomposición tiene que ser en varias proyecciones. Esta dependencia es también llamada de junta y se expresa así:

$$DJ * (R1, \dots, Rj)$$

Y significa que $R = R1 \text{ join } R2 \text{ join} \dots \text{ join } Rj$. Veamos un ejemplo de De Miguel (2000: 190).

R(EDITA)

EDITORIAL	IDIOMA	TEMA
RA-MA	Inglés	BD
RA-MA	Español	CASE
Addison Wesley	Español	BD
RA-MA	Español	BD

Esta relación implica que si se publica un tema de bases de datos (BD) en un determinado idioma, por ejemplo español, entonces todas las editoriales deben publicar ese tema y en ese idioma. Por esto, si Addison Wesley publicara BD en francés, sería necesario que RA-MA también lo hiciera, provocando insertar una tupla adicional.

La descomposición de la relación en dos relaciones provocaría tuplas espurias. Por ejemplo:



R(EDITA1)

Editorial	Idioma
RA-MA	Inglés
RA-MA	Español
Addison Wesley	Español

R(EDITA2)

Idioma	Tema
Inglés	BD
Español	CASE
Español	BD

R(EDITA1) join R(EDITA2)

Editorial	Idioma	Tema
RA-MA	Inglés	BD
RA-MA	Español	CASE
RA-MA	Español	BD
Addison Wesley	Español	CASE
Addison Wesley	Español	BD

← Tupla espuria

Para que sea una descomposición sin pérdida sería necesario descomponer en tres relaciones, agregando la siguiente relación:

R(EDITA3)

Editorial	Tema
RA-MA	BD
RA-MA	CASE
Addison Wesley	BD

Quinta forma normal (5FN)

Una relación está en 5FN si, y sólo si, no existen dependencias de combinación. Si dicha relación no está en 5FN se deben hacer tantas proyecciones como descriptores involucrados en la dependencia. En el ejemplo eran tres atributos involucrados en la dependencia de junta y por eso se hicieron tres proyecciones.



2.7 Reglas de Codd

E. F. Codd propuso doce reglas que definen los requisitos de un manejador de base de datos relacionales. No obstante la mayoría de los manejadores comerciales no cumplen al 100 por ciento todas las reglas, buena parte de ellas han sido contempladas en los software de base de datos. Las reglas son:

1. Regla de la información

Toda la información de una base de datos relacional está representada explícitamente a nivel lógico mediante valores en tablas.

2. Regla del acceso garantizado

Todo dato (valor atómico) en una base de datos relacional es accesible de manera garantizada mediante la combinación del nombre de tabla, llave primaria y nombre de columna.

3. Regla del tratamiento sistemático de valores nulos

Los valores nulos (Null), que son diferentes a la cadena vacía, el carácter de espacio en blanco y al cero, son manejados por un sistema de bases de datos relacionales de manera sistemática con el objeto de representar la información desconocida o faltante; debe hacerlo de forma independiente del tipo de dato.

4. Regla del catálogo basado en el modelo relacional

La descripción de los datos dentro de una base de datos, es decir, el catálogo (nombres de tablas, nombres de columnas, tipos de datos de cada columna, nombres de restricciones, etc.) debe estar representada a nivel lógico de la misma manera que los datos normales de usuario, es decir, a través de tablas. Esto permitirá utilizar el mismo lenguaje relacional para recuperar datos del catálogo y datos normales de usuario.



5. Regla del sub-lenguaje de datos entendible

Un sistema relacional debe soportar varios tipos de lenguajes y varios modos de uso por parte del usuario. Sin embargo, debe existir al menos un sub-lenguaje relacional que permita expresar sentencias, mediante una sintaxis bien definida, con cadenas de caracteres. Este sub-lenguaje debe ser capaz de soportar las siguientes operaciones:

Definición de datos.

Consulta de datos.

Manipulación de datos.

Restricciones de integridad.

Manejo de autorizaciones para los datos.

6. Regla de la actualización de vistas

Todas las vistas que sean teóricamente actualizables deben ser actualizables por el sistema de bases de datos.

7. Regla de inserciones, actualizaciones y eliminaciones de alto nivel

La posibilidad de manejar una relación como un único operador aplica no sólo a la recuperación de datos sino también a la inserción, actualización y eliminación de datos. Debe ser posible realizar estas operaciones para un conjunto de renglones.

8. Independencia física de los datos

Los programas de aplicación no sufren modificaciones a pesar de los cambios en el nivel de físico de almacenamiento o en los métodos de acceso.

9. Independencia lógica de los datos

Los programas de aplicación no sufren modificaciones a pesar de los cambios hechos a las tablas.



10. Regla de independencia de integridad

Las restricciones de integridad especificadas para una relación deben ser definidas con el sub-lenguaje de datos relacional, y almacenadas en el catálogo y no en los programas de aplicación.

11. Independencia de distribución

Un sistema relacional puede estar distribuido en distintos equipos o sitios de una red y las tablas deben ser vistas como si estuvieran localmente.

12. Regla de la no subversión

Ningún lenguaje de bajo nivel puede ser usado para violar las restricciones de integridad expresadas en el lenguaje relacional de alto nivel.

2.8. Estándar SQL

En la sección anterior hemos visto que Codd propuso la utilización de un sub-lenguaje relacional que permitiera operaciones de definición, consulta, manipulación, restricción y control de acceso a datos. Hoy en día contamos con un lenguaje de programación para bases de datos relacionales que cumple en buena medida los requerimientos de Codd. Este lenguaje es conocido como SQL (Structured Query Language).

SQL89

La primera versión reconocida por la ANSI como estándar del SQL fue la de 1989, aunque el trabajo para desarrollar un estándar para bases relacionales había comenzado tiempo atrás en los laboratorios de investigación de IBM. Una versión inicial de la ANSI es de 1986, pero sufrió mejoras hasta llegar a la versión que hoy conocemos como SQL89.

Las principales características es que contaba con instrucciones DDL (create y drop), DML (select, insert, delete, update) y DCL (grant y revoke). Se había





establecido que toda sentencia SQL debería comenzar con un verbo y después una serie de cláusulas que comienzan por palabras claves. Otra característica importante fue que SQL podía ser integrado a otros lenguajes de aplicación o podría ser ejecutado por una terminal interactiva directamente sobre la base de datos relacional.

SQL2 (SQL92)

En 1992 fue revisado el estándar por la ANSI y resueltos varios problemas de la primera versión. Entre las mejoras estuvieron:

- Soporte a caracteres de longitud variable.
- Expresiones para cambios de tipo de datos.
- Operador *join*.
- Manejo de transacciones.
- Uso de subconsultas.
- Uso de operadores de unión, intersección y diferencia.

SQL3 (SQL99)

Este nuevo estándar introdujo conceptos de la orientación a objetos y el uso de tipos de datos complejos. Hoy en día los principales sistemas manejadores de bases de datos han adoptado este estándar y se han convertido en lo que podemos llamar manejadores objeto-relacionales de bases de datos.

No es claro si las empresas cambiarán su tecnología de bases de datos en un futuro cercano. Además, es innegable que el modelo relacional sigue siendo el más utilizado y el que ha logrado resolver los requerimientos de información más significativos de distintas organizaciones. Por esto es muy importante que como informático conozcas bien este modelo y seas capaz de utilizarlo para resolver problemas de información.



Bibliografía del tema 2

- Codd, E. F., (1970), "A relational model of data for large shared data banks", en *Communications of the ACM*, 13, 6.
- _____, "Is Your DBMS Really Relational?", en *ComputerWorld*, 14 de octubre. 1985.
- Date, C. J., (2001) *Sistemas de Bases de Datos*, 7ª ed., México, Pearson Education.
- De Miguel, Adoración, et al, (2001), *Diseño de bases de datos relacionales*, Madrid, Alfaomega-Rama.
- Elmasri, Ramez, (2002), *Fundamentos de sistemas de bases de datos*, México, Pearson Educación y Addison-Wesley.
- Silberschatz, A., H. Korth, et. al., 2006 *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw-Hill.

Actividades de aprendizaje

- A.2.1.** Elabora un mapa conceptual sobre el modelo relacional de bases de datos que incluya todos los conceptos presentados en el tema.
- A.2.2.** Define con tus propias palabras en qué consiste el modelo relacional de bases de datos.
- A.2.3.** Construye un cuadro comparativo entre los modelos pre-relacionales, el modelo relacional y los modelos pos-relacionales. Asegúrate de mostrar las principales características de cada uno.
- A.2.4.** Elabora un cuadro de dos columnas que resuma las formas normales y el tipo de dependencia funcional que eliminan.
- A.2.5.** Realiza un cuadro sinóptico que resuma los tipos de dependencias funcionales y su definición.
- A.2.6.** A partir de la siguiente relación EMBARQUE, determina si cumple con la primera, segunda y tercera forma normal y explica por qué.



R(EMBARQUE)

idembarque	iddestino	idproducto	fecha	destino	producto
1	100	V1	01/01/2006	España	Nombre: Vino, Precio: 12,000.
1	100	M1	01/01/2006	España	Nombre: Mueble, Precio: 10,000.
2	100	E1	01/02/2006	España	Nombre: Electrodoméstico, Precio: 13,000
2	100	V1	01/02/2006	España	Nombre: Vino, Precio: 12,000.
3	200	V1	01/03/2006	Argentina	Nombre: Vino, Precio: 12,000.
4	100	M1	01/04/2006	España	Nombre: Mueble, Precio: 10,000.
5	200	M1	01/04/2006	Argentina	Nombre: Mueble, Precio: 10,000.

Cuestionario de autoevaluación

1. ¿Cuáles son las características del modelo relacional?
2. ¿Qué es una relación?
3. Menciona las cuatro propiedades de una relación
4. ¿Qué es grado y qué es cardinalidad de una relación?
5. ¿Qué operación de álgebra relacional obtiene las tuplas que están en la primera relación y no en la segunda?
6. ¿En qué consiste la operación de álgebra relacional llamada junta?
7. Define la 1FN
8. Define la 2FN
9. Define la 3FN
10. ¿En qué consiste el proceso de descomposición sin pérdida?



Examen de autoevaluación

1. La 3FN exige que todas las dependencias sean irreducibles.
a) Verdadero b) Falso
2. La 4FN exige la eliminación de las dependencias de junta.
a) Verdadero b) Falso
3. La regla del acceso garantizado exige que el catálogo de una base de datos sea consultado usando el mismo sublenguaje de datos.
a) Verdadero b) Falso
4. La regla de la independencia física de los datos exige que si cambian las tablas o relaciones, los programas de aplicación no se modifiquen.
a) Verdadero b) Falso
5. La regla del tratamiento sistemático de valores nulos exige que el sistema de bases de datos cuente con un valor distinto al 0, la cadena vacía o el espacio.
a) Verdadero b) Falso
6. La restricción obtiene un subconjunto de atributos de una relación.
a) Verdadero b) Falso
7. El producto obtiene aquellos atributos que coinciden en un atributo en común entre las relaciones.
a) Verdadero b) Falso
8. La normalización tiene por objeto reducir los problemas de redundancia y actualización de datos.
a) Verdadero b) Falso
9. El cuerpo de una relación es el conjunto de atributos de la misma.
a) Verdadero b) Falso
10. Una relación tiene 12 propiedades.
a) Verdadero b) Falso



TEMA 3. MODELO ORIENTADO A OBJETOS

Objetivo particular

Después de estudiar este tema, serás capaz de distinguir un manejador de bases de datos orientado a objetos de uno relacional. Podrás enumerar las razones por las que surgieron estos manejadores. Además, identificarás las generaciones de manejadores orientados a objetos y sus principales características.

Temario detallado

3.1 Introducción

Antecedentes

- 3.1.1 Retos actuales de los sistemas manejadores de bases de datos
- 3.1.2 Tendencias actuales en la tecnología de bases de datos
- 3.1.3 Orientación a objetos
- 3.1.4 Persistencia

3.2 Sistemas de administración de bases de datos orientadas a objetos

- 3.2.1 Historia
- 3.2.2 Primera generación
- 3.2.3 Segunda generación
- 3.2.4 Tercera generación
- 3.2.5 Definición
- 3.2.6 Características

3.3 Estándar ODMG



3.1.Introducción

La tecnología de bases de datos vive un momento de lenta transición del modelo relacional a otros novedosos modelos con aplicaciones interesantes. Entre estos modelos están el multidimensional para sistemas OLAP, el semiestructurado para bases de datos XML de intercambio electrónico de información y el modelo dimensional para creación de *data warehouses*. Un modelo que llama mucho la atención por sus ventajas tecnológicas es el orientado a objetos y será producto de la exposición en este tema.

La orientación a objetos llegó para quedarse y desde los años 80 se convierte día a día en la tecnología de desarrollo de aplicaciones por excelencia. Este hecho bastaría para entender el surgimiento de los sistemas manejadores de bases de datos orientados a objetos (OODBMS), pero en las siguientes páginas revisaremos algunas motivaciones adicionales para el origen de esta tecnología.

Algo importante que debemos resaltar es que a pesar de las ventajas para el desarrollo de aplicaciones empresariales que tienen los OODBMS, hoy en día las empresas siguen utilizando los manejadores de bases de datos relacionales y no se ve aún para cuándo serán suplantadas por completo.

En los apartados siguientes revisaremos las situaciones tecnológicas que dieron origen a los manejadores de bases de datos orientados a objetos. Repasaremos los fundamentos de la orientación a objetos. Detallaremos la evolución de este tipo de sistemas de bases de datos. Finalmente te proporcionaremos una definición y expondremos sus principales características.

Antecedentes

Los DBMS surgieron para responder a las necesidades de información de las organizaciones. Se tratan de un conjunto de datos persistentes y de programas para acceder a ellos y actualizarlos.



La tecnología de bases de datos tiene más de treinta años. Primero surgieron sistemas administradores de archivos de tipo ISAM (*Indexed Secuencial Access Mode*), que trabajaban con archivos separados. Después vinieron sistemas que centralizaban los archivos en una colección llamada base de datos; los primeros de éstos utilizaron el modelo jerárquico (sistemas IMS y 2000). A continuación surgieron los desarrollados por la CODASYL, como IDS, TOTAL, ADABAS e IDMC. La siguiente generación fue la de bases de datos relacionales. Éstas utilizan lenguajes más accesibles y poderosos en la manipulación de datos como el SQL, QUEL y QBE.

Los DBMS deben contar con un modelo de datos, es decir, estructuras lógicas para describir los datos, y operaciones para manipularlos (recuperación y actualización). Las operaciones sobre los datos se hacen por medio de tres lenguajes: un DDL para definir el esquema y la integridad, un DML para la actualización de los datos y un DCL para el manejo de las autorizaciones en la base de datos. Adicionalmente un DBMS incluye mecanismos de seguridad, acceso a los datos, recuperación, control de concurrencia y optimización de consultas.

Los DBMS evolucionan con el afán de satisfacer nuevos requerimientos tecnológicos y de información. En seguida describiremos los que han motivado el surgimiento de sistemas orientados a objetos.

3.1.1 Retos actuales de los sistemas manejadores de bases de datos

Aunque los DBMS relacionales (RDBMS) son actualmente líderes del mercado y brindan las soluciones necesarias a las empresas comerciales, existen aplicaciones que necesitan funciones con las que no cuentan. Ejemplos de ellas son las CAD/CAM y CASE. Adicionalmente, los sistemas multimedia, como los geográficos y de medio ambiente, sistemas de gestión de imágenes y



documentos, y los sistemas de apoyo a las decisiones necesitan de modelos de datos complejos difíciles de representar como tuplas de una tabla.

En general, estas aplicaciones necesitan manipular objetos y los modelos de datos deben permitirles expresar su comportamiento y las relaciones entre ellos. Los nuevos DBMS deben tomar en cuenta las siguientes operaciones:

- Ser capaces de definir sus propios tipos de datos.
- Manejar versiones de objetos y estados de evolución.
- El tamaño de los datos puede ser muy grande.
- La duración de las transacciones puede ser muy larga.
- Recuperar rápidamente objetos complejos.
- Ofrecer comunicación efectiva a los clientes del sistema, principalmente en desarrollos grupales.
- Permitir cambios en el esquema de la base.
- Manejar objetos completos y sus componentes.
- Lenguajes de consulta de objetos y lenguajes computacionalmente complejos.
- Mecanismos de seguridad basados en la noción de objeto.
- Funciones para definir reglas deductivas y de integridad.
- Tener la capacidad para comunicarse con las aplicaciones ya existentes y manipular sus datos.

3.1.2 Tendencias actuales en la tecnología de bases de datos

Con miras a superar los retos antes mencionados, las bases de datos están tomando varias tendencias. En general se están auxiliando de los lenguajes de programación orientados a objetos, los lenguajes lógicos y la inteligencia artificial. En este sentido, podemos determinar cuatro tendencias actuales:

- Sistemas relacionales extendidos. Incluyen manejo de objetos y *triggers*.



- Sistemas de bases de datos orientadas a objetos. Integran el paradigma de la orientación a objetos a la tecnología de bases de datos.
- Sistemas de bases de datos deductivas. Unen las bases de datos con la programación lógica. Cuentan con mecanismos de inferencia, basados en reglas, para generar información adicional a partir de los datos almacenados en la base.
- Sistemas de bases de datos inteligentes. Incorporan técnicas desarrolladas en el campo de la inteligencia artificial.

3.1.3 Orientación a objetos

La orientación a objetos representa el mundo real y resuelve problemas a través de objetos, ya sean tangibles o digitales. Este paradigma tecnológico considera un sistema como una entidad dinámica formada de componentes. Un sistema sólo se define por sus componentes y la manera en que éstos interactúan. Las principales características de la orientación a objetos son:

- Es una tecnología para producir modelos que reflejen un dominio de negocio y utiliza la terminología propia de tal dominio.
- Cuenta con cinco conceptos subyacentes: objeto, mensajes, clases, herencia y polimorfismo.
- Un objeto tiene un estado, un comportamiento y una identidad.
- Los mensajes brindan comunicación entre los objetos.
- Las clases son un tipo de plantilla usada para definir objetos, los cuales son instancias del mundo real.
- Cada objeto tiene un nombre, atributos y operaciones.

3.1.4 Persistencia

La persistencia es una característica necesaria de los datos en un sistema de bases de datos. Recordemos que consiste en la posibilidad de recuperar datos en el futuro. Esto implica que los datos se almacenan a pesar del término del



programa de aplicación. En resumen, todo manejador de base de datos brinda persistencia a sus datos.

En el caso de los OODBMS, la persistencia implica almacenar los valores de atributos de un objeto con la transparencia necesaria para que el desarrollador de aplicaciones no tenga que implementar ningún mecanismo distinto al mismo lenguaje de programación orientado a objetos.

Lo anterior traería como ventaja que no sería necesario el uso de dos lenguajes de programación para construir una aplicación. Es decir, actualmente, el desarrollo de aplicaciones se hace con lenguajes de programación orientada a objetos almacenando datos en bases relacionales, por lo que el desarrollador debe utilizar un lenguaje para la aplicación (Java, PHP, C++) y otro para la base de datos (SQL).

El objetivo básico de un OODBMS es entonces darle persistencia a los objetos. Por lo anterior algunos autores ven estos sistemas sólo como lenguajes de orientación a objetos con persistencia y no como manejadores completos. Para ver una discusión acerca de si los OODBMS son en realidad DBMS puedes leer a C. J. Date (2001: 845-847).

3.2 Sistemas de administración de bases de datos orientadas a objetos

Los sistemas de bases de datos orientadas a objetos parecen ser la tecnología más prometedora para los próximos años, aunque carecen de un modelo de datos común y de fundamentos formales, además de que sus comportamientos en seguridad y manejo de transacciones no están a la altura de los productos actuales.



Hay organismos en pro de la estandarización de este tipo de sistemas manejadores de bases de datos como el OMG (Object Management Group), la CAD Framework Initiative y el grupo de trabajo de ANSI.

Algo que apoya esta tendencia es que a pesar de que la ingeniería de software orientada a objetos requiere mucho tiempo de análisis, la mayoría de los proyectos de desarrollo son más cortos y requieren menos personas, además de que la cantidad de código es menor.

A pesar de lo dicho anteriormente, sería difícil para las empresas dejar de un día para otro los sistemas actuales, debido principalmente a la falta de personal calificado, al efecto sobre la continuidad de sus operaciones y a la ausencia de garantías en la reutilización de los datos.

3.2.1 Antecedentes

La evolución de los sistemas de bases de datos orientados a objetos está muy ligada al mercado de manejadores y a las compañías que han apostado por este tipo de bases de datos. Como mencionamos, este tipo de DBMS no tiene estrictos fundamentos teóricos como el modelo relacional y por tanto no se puede establecer una historia de su concepción.

A continuación te describimos brevemente las tres generaciones de OODBMS según Bertino y Martino (1995).

3.2.2 Primera generación

Comienza en 1986 cuando el sistema G-Base fue lanzado por la compañía francesa Grápale. En 1987 Servio Corp introduce GemStone y en 1988 Ontologic promueve su Vbase, seguido de Statics por la empresa Symbolics. Estos sistemas estaban basados en lenguajes propios y plataformas independientes del mercado.



Estos sistemas fueron considerados lenguajes orientados a objetos con persistencia.

3.2.3 Segunda generación

Se da con la salida al mercado de Ontos en 1989. Siguieron los productos Object Design, Objectivity y Versant Object Technology. Todos utilizaron una arquitectura cliente/servidor y una plataforma en C++, X Windows y UNIX.

3.2.4 Tercera generación

La generación comienza con Itasca, lanzado en agosto de 1990 por Microelectronics and Computer Corporation. Le siguieron O2 producido por la compañía francesa Altair, y después Zeitgeist por Texas Instruments. Estos ya son sistemas administradores de bases de datos con características avanzadas, un DDL y DML orientados a objetos.

3.2.5 Definición

No obstante, en la actualidad hay mucha atención hacia los OODBs, tanto en el terreno de desarrollo como en el teórico, no hay una definición estándar de lo que estos sistemas significan.

Existen tres problemas principales que impiden una definición generalizada:

- La falta de un modelo de datos común entre los diferentes sistemas. Los sistemas de bases de datos relacionales cuentan con especificaciones claras dadas por Codd, pero los orientados a objetos no tienen algo así. Se pueden encontrar muchos textos que describen diferentes modelos, pero no hay uno como estándar.
- La carencia de fundamentos formales. El fundamento teórico de la programación orientada a objetos es escaso en comparación con otras



áreas como la programación lógica. Además se carece de definiciones de diversos conceptos.

- Una actividad experimental muy fuerte. Existe mucho trabajo experimental, la mayoría de los desarrollos son sistemas prototipo no comerciales, no hay trabajo de conceptualización y definición de estándares. El diseño de estos sistemas está orientado por las aplicaciones que los requieren y no por un modelo común.

El problema de estos sistemas es similar al de las bases de datos relacionales a mitad de los setenta. La gente se dedicaba a desarrollar implementaciones en lugar de definir las especificaciones para luego hacer la tecnología que permitiera implementarlas.

Se espera que de los prototipos y desarrollos actuales de los OODBS surja un modelo. Aunque también se corre el riesgo de que alguno de estos se convierta en el estándar por su demanda en el mercado.

A manera de definición podemos decir que un OODBS debe satisfacer dos criterios:

1. Debe ser un DBMS.
2. Debe ser un sistema orientado a objetos (consistente con los lenguajes de programación orientada a objetos).

El primer criterio incluye características de cualquier DBMS, que podemos listar como: persistencia, administración de almacenamiento secundario, concurrencia, recuperación y facilidad de consultas personalizadas.

El segundo criterio corresponde a características que se comparten con la programación orientada a objetos: objetos complejos, identidad de objetos, encapsulación, herencia, sobreescritura y sobrecarga, y completa capacidad computacional (*computational completeness*).



3.2.6 Características

A continuación exponemos las características de un OODBMS.

Objetos complejos

Los objetos complejos son creados a partir de objetos (tipos de datos) simples. Estos objetos simples son: enteros, caracteres, cadenas de bytes, booleans y números de punto flotante. Los objetos complejos pueden ser por ejemplo: tuplas, conjuntos (sets), listas y arreglos. Un OODBMS debe tener como mínimo conjuntos (set), listas y tuplas.

Los conjuntos (sets) son la manera natural de representar colecciones del mundo real. Las tuplas permiten representar de manera natural las propiedades de una entidad y son importantes por la aceptación ganada con el modelo relacional. Finalmente, las listas o arreglos resultan importantes porque capturan orden, cosa que ocurre en el mundo real; además de que ayudan a representar matrices y series de datos en el tiempo.

Identidad de objetos

La identidad de objetos ha existido desde hace mucho tiempo en los lenguajes de programación, pero en las bases de datos es más reciente. El objetivo es contar con objetos que tengan una existencia independiente de sus valores. Así, dos objetos pueden ser idénticos si son el mismo objeto o pueden ser iguales si tienen los mismos valores.

La identidad de objetos cobra relevancia cuando un objeto se comparte con otros y cuando se actualiza. En un modelo basado en identidad, dos objetos pueden compartir un objeto hijo. Por ejemplo, pensemos en dos personas, Arturo y Valeria, y cada una tiene un hijo llamado Jaime. En la vida real hay dos posibles situaciones:

- Arturo y Valeria son padres de Jaime, por lo que existe identidad del objeto Jaime, es decir, se trata en realidad del mismo objeto.



- Hay dos niños del mismo nombre. Esto implica igualdad de dos objetos Jaime, porque tienen el mismo nombre, pero no son el mismo objeto.

Asumiendo que Arturo y Valeria son en realidad padres de un mismo niño Jaime, todas las actualizaciones al hijo de Arturo, también son aplicadas al hijo de Valeria, ya que se trata del mismo objeto, Jaime. Si el sistema no tuviera identidad, sino que se basara en igualdad de objetos, serían necesarias dos actualizaciones a los dos objetos Jaime.

Soportar identidad de objetos implica que el OODBMS ofrece operaciones como asignación de objetos, copiado de objetos y comprobación de la identidad o igualdad de objetos.

La principal manera de implementar la identidad de objetos es mediante un OID (*Object Identifier*) independiente de los valores de los atributos del objeto. Estos son implementados por el sistema y muchas veces a bajo nivel, lo que mejora el rendimiento.

Según Date (2001: 847), los OID son innecesarios e indeseables en el nivel del modelo. En comparación con las claves primarias, los OID están ocultos al usuario mientras que las claves no. El uso de estos *object identifiers* no elimina el uso de claves primarias ya que son necesarias para unir al sistema con la realidad en la que se inserta; pensemos, por ejemplo, en los folios de facturación.

Encapsulación

La idea de encapsulación es tomada de los lenguajes de programación en los que para todo objeto existe una parte visible que permite especificar el conjunto de operaciones que pueden ser realizadas sobre el objeto. Otra parte es no visible y contienen los datos que almacena ese objeto.



Traducido a bases de datos, un objeto encapsula programas y datos. Por ejemplo, en un sistema relacional, un empleado es representado por una tupla. Para ser consultado desde una aplicación es necesario usar un lenguaje de programación para programar procedimientos que serán almacenados fuera de la base de datos; es más, ese lenguaje puede ser en realidad la combinación de un lenguaje de alto nivel con el lenguaje estándar relacional SQL.

En un sistema de bases de datos orientado a objetos, definimos al empleado como un objeto que tiene una parte de datos (probablemente muy similar al registro que definiríamos en el sistema relacional) y una parte de operaciones, la cual consiste en operaciones de `incrementarSalario()` y `eliminarDefinitivo()`, por ejemplo, que accederían a los datos del empleado. Cuando se almacene un nuevo empleado, datos y operaciones serían almacenados en la base de datos y no fuera de ella.

Herencia

La herencia tiene dos ventajas: es una herramienta poderosa de modelado, ya que brinda una descripción precisa del mundo, y ayuda a simplificar la implementación de las aplicaciones. Para entender el manejo de la herencia en los sistemas de bases de datos orientados a objetos vamos a establecer un ejemplo.

Asumamos que tenemos empleados y estudiantes. Cada empleado tiene un nombre, edad y salario, además podemos aumentar su sueldo. Por su parte, cada estudiante tiene edad, nombre y un conjunto de calificaciones con las cuales podemos obtener su promedio.

En un sistema relacional, el diseñador de bases de datos definiría una relación empleado y una estudiante, también escribiría el código para la operación de aumentar sueldo. Para la relación empleado tendría que escribir el código para la operación de obtener promedio.



En un sistema orientado a objetos, usando adecuadamente la herencia, nos daríamos cuenta de que empleado y estudiante son personas y comparten los atributos: nombre y edad. Entonces, declararíamos una clase empleado como un tipo especial de la clase persona, el cual incluiría una operación especial de aumentarSueldo() y un atributo de salario. De forma similar se declararía el estudiante como un tipo especial de la clase persona con el atributo adicional de conjunto de Grados y la operación especial obtenerPromedio()).

El modelo es más cercano a la realidad y nos permite ahorrar código de programación. Por esto se dice que la herencia ayuda a reutilizar código ya que cada programa está disponible para ser compartido.

Sobreescritura y sobrecarga

En la programación orientada a objetos tenemos la ventaja de poder reescribir métodos con el mismo nombre. Esto significa contar con varios métodos que se llamen igual, pero que realicen distintas operaciones. Para poder programar estos métodos llamados sobrecargados, es necesario que cambie algo en sus parámetros, como el número, orden o tipo de dato. Esto mismo es posible de una base de datos orientada a objetos.

Completa capacidad computacional (*Computational completeness*)

Los manejadores de bases de datos relacionales cuentan con un lenguaje para realizar procesos computacionales sobre los datos: el SQL. Además, adicionan un lenguaje *procedural* (“de procedimiento”) que permite la definición de variables, manejo de excepciones, ciclos y estructuras condicionales. Estos lenguajes son pl/sql para Oracle, pl/pgsql para PostgreSQL y Transact-SQL para SQL Server de Microsoft, por dar unos ejemplos.

Los manejadores de bases de datos orientadas a objetos, también deben contar con un lenguaje que puede realizar cualquier procesamiento. En este sentido, lo más común es que los OODBMS integren lenguajes computacionalmente



completos dentro de la base de datos. Estos pueden ser los que ya existen en el mercado y que se usan como lenguajes aplicación general (Java, C++, etc.).

3.3 Estándar ODMG

El *Object Database Management Group* (ODMG) surge en 1991 formado por un grupo de proveedores de OODBMS. Generaron un primer estándar en 1993 en donde exponían las características que consideraban necesarias en un sistema de bases de datos de este tipo. En 2001 aparece la nueva versión del estándar que se acomoda a las especificaciones de la tecnología de Java. Los principales componentes de un OODBMS según el ODMG son:

- Lenguaje de definición de objetos (ODL).
- Lenguaje de consulta de objetos (OQL).
- Conexión con los lenguajes C++, Smalltalk y Java.

Bibliografía del tema 3

Atkinson, Malcom., (et. al.), *The Object-Oriented Database System Manifesto*, documento digital disponible en:

<http://www.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/index.html>. 1989. Consultada el 13 de noviembre de 2008.

Bertino, Elisa y Lorenzo Martino, (1995), *Sistemas de bases de datos orientadas a objetos. Conceptos y arquitectura*, Massachussets, Addison-Wesley.

Date, C. J., (2001), *Sistemas de Bases de Datos*, 7ª ed., México, Pearson.

Hughes, J. G., (1991), *Object-Oriented databases*, New york, Prentice Hall.

Johnson, James L., (1997), *Bases de datos. Modelos, lenguajes, diseño*, México, Oxford.

Silberschatz, A., KORTH H., et. al., (2006), *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw-Hill.



Actividades de aprendizaje

- A.3.1.** Desarrolla un mapa conceptual sobre las bases de datos orientadas a objetos que incluya su evolución, definición y características.
- A.3.2.** Elabora un breve ensayo a partir de la lectura de las páginas 845 a 847 del libro de C. J. Date donde discute acerca de si los OODBMS son en realidad un DBMS. Confronta esta lectura con el contenido de tu material.
- A.3.3.** Realiza un cuadro sinóptico que resuma las razones por las que surgieron o fueron necesarios los OODBMS.

Cuestionario de autoevaluación

1. ¿Qué es la orientación a objetos?
2. ¿Cuáles fueron las necesidades tecnológicas que dieron pie al surgimiento de los sistemas de bases de datos orientados a objetos?
3. ¿Cuáles son las tendencias actuales de la tecnología de bases de datos?
4. Describe la evolución de los OODBMS.
5. ¿En qué consiste la persistencia en los sistemas de bases de datos orientados a objetos?
6. ¿Qué es un OODBMS?
7. Describe la característica de identidad de objetos en un sistema de base de datos de este tipo.
8. Detalla cómo se aplica la herencia en una base de datos orientada a objetos.
9. ¿Cuál es la diferencia entre sobreescritura y sobrecarga?
10. ¿Qué características propone el ODMG para un OODBMS?

Examen de autoevaluación

1. Los OODBMS surgen por la necesidad de tipos de datos de longitud variable.
a) Verdadero b) Falso
2. Los sistemas de bases de datos inteligentes son una tendencia de la tecnología actual de bases de datos.
a) Verdadero b) Falso



3. La orientación a objetos evita utilizar la terminología del dominio del negocio.
a) Verdadero b) Falso
4. Los objetos tienen estado y comportamiento.
a) Verdadero b) Falso
5. El modelo de base de datos orientado a objetos tiene un sustento fuertemente teórico y formal.
a) Verdadero b) Falso
6. Un OODBMS debe contar con herencia.
a) Verdadero b) Falso
7. Las tuplas, conjuntos y listas pueden considerarse objetos complejos.
a) Verdadero b) Falso
8. Dos objetos con distinto OID, pero valores de atributos iguales son idénticos.
a) Verdadero b) Falso
9. La sobrecarga y la sobreescritura son conceptos equivalentes.
a) Verdadero b) Falso
10. La orientación a objetos se caracteriza por la poca reutilización de código.
a) Verdadero b) Falso



TEMA 4. DISEÑO

Objetivo particular

Al terminar el tema podrás identificar los elementos del modelo entidad-relación para el diseño de bases de datos. También reconocerás los pasos fundamentales para transformar este modelo a un modelo relacional de tablas.

Temario detallado

4.1 Introducción al diseño

4.2 Modelo semántico

4.3 Modelo lógico

4.3.1 E/R

- Entidad
- Atributo
 - Atributos compuestos
 - Atributos clave
 - Atributos multivaluados
 - Atributos derivados
- Interrelación
 - Grado
 - Tipo
 - Papel (rol)
 - Cardinalidad

4.3.2 E/R extendido

4.4 Modelo físico

4.4.1 Implementación de un E/R al modelo relacional

4.5 Modelo de clases (UML)



Introducción

El tema que presentamos a continuación tiene gran relevancia para tu formación como informático, principalmente en el aspecto del desarrollo de sistemas de información. Así, una fase del proceso de desarrollo de sistemas es el diseño de la base de datos.

Con la información obtenida en la etapa de análisis se desarrolla una solución de almacenamiento de datos mediante un modelo (relacional, orientado a objetos, etc.). Por lo tanto, revisaremos en este tema la herramienta de modelado o diseño de datos más utilizada en la vida real: el modelo entidad-relación.

En este tema estudiarás los fundamentos teóricos de este modelo y los procedimientos para llevarlo a la práctica. Su principal ventaja es que resulta independiente del modelo de base de datos en el que será implementado.

El diseño es fundamental para la buena construcción de nuestra base de datos. Este proceso nos permitirá definir las restricciones de integridad del mundo real y llevarlas a la base de datos, además nos brindará la posibilidad de determinar cómo serán las estructuras de almacenamiento de datos y sus relaciones.

El éxito de la base de datos y del sistema en general depende (entre otras cosas), de un buen diseño. Planearlo bien nos garantiza eliminar problemas de redundancia y actualización de datos, además de ahorrar de recursos de cómputo y facilidad de consultas de datos.

4.1 Introducción al diseño

El diseño de bases de datos consiste en traducir un conjunto de datos inmersos en una realidad a un modelo manejable en una base de datos. Esta traducción debe decirnos la estructura lógica de las estructuras para almacenar los datos y las restricciones sobre estos.



Diseñar es más un arte que una ciencia, o al menos no es posible encuadrarlo en rigurosos principios científicos.⁷ En las siguientes secciones conocerás una metodología de diseño. Esta metodología estará basada en el desarrollo de un modelado semántico a través de un modelo entidad-relación y su posterior transformación en un modelo relacional.

El proceso de diseño de base de datos puede verse como una interacción de tres etapas generales:

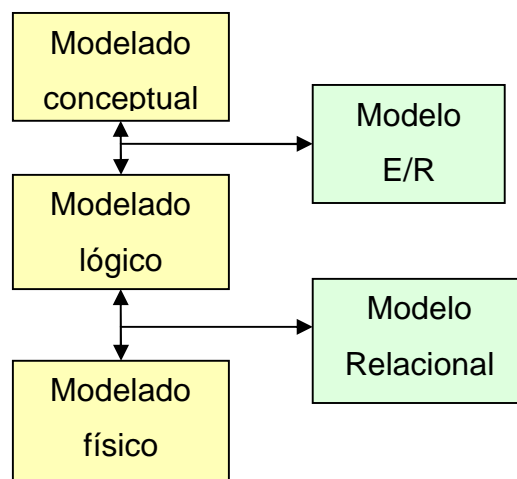


Figura 1. Proceso del diseño de base de datos

4.2 Modelo semántico

El modelo relacional ha demostrado ser un modelo muy útil para el desarrollo de sistemas de información organizacionales y en algunas otras áreas de la actividad humana. Sin embargo, desde sus inicios ha sido criticado porque no representa mucha de la semántica de la realidad. El hecho de estar basado en relaciones y nada más que relaciones, le impide captar el significado de las interacciones entre éstas, su jerarquía y las restricciones asociadas a estas interacciones.

⁷ Una manera de diseño de bases de datos relacionales con cierto rigor científico es el proceso de normalización que revisamos en el tema 2.



Por esta razón, y entre otras, fue propuesta una novedosa manera de modelar la realidad. Su principal característica era que intentaba representar en buena medida la semántica de la realidad. La propuesta se conoce como “Modelado Semántico”. En palabras de C. J. Date (2001: 419), lo que se trató de resolver es que “por lo regular los sistemas de bases de datos sólo tienen una comprensión muy limitada de lo que significa la información de la base de datos”.

El modelado semántico tuvo su principal desarrollo en un modelo que veremos adelante: el modelo Entidad-Relación (E/R). Este modelo materializó el objetivo del modelado semántico y tuvo las ventajas necesarias para convertirse en el más utilizado en nuestro tiempo. Claro que resulta importante aclarar que tampoco capta todos los sentidos de la realidad que representa, pero sí la gran mayoría.

4.3 Modelo lógico

Mediante un modelado semántico de la realidad es posible obtener un modelo de datos en el nivel conceptual que nos permitirá representar la realidad en una forma dirigida al almacenamiento de datos. Como ya lo habíamos mencionado, es el modelo Entidad-Relación la manera más utilizada para hacerlo. Los elementos y características de este modelo se verán en la siguiente sección.

A partir del modelo conceptual se deriva un modelo lógico específico para un tipo de base de datos: orientado a objetos, relacional, jerárquico, etc. En nuestro caso, el modelo lógico será basado en el modelo relacional.

4.3.1 E/R

El modelo Entidad-Relación (E/R) ayuda a realizar un diseño de bases de datos sin atender a un modelo en especial (jerárquico, relacional, orientado a objetos).

Fue propuesto por Peter Chen en el artículo “The entity-relationship model - Toward a unified view of data” (1976). En éste, Chen propone utilizar un enfoque más natural del mundo real basado en entidades e interrelaciones. Hoy en día,



podríamos decir que más bien existe una familia de modelos, ya que muchos autores han realizado propuestas que han enriquecido al modelo E/R.

El modelo E/R, permite visualizar la base de datos desde un alto nivel de abstracción. Los elementos interesantes de la realidad que queremos modelar son las entidades, además modelamos sus atributos y las interacciones entre ellas.

Una ventaja del modelo E/R es que utiliza una representación gráfica conocida como Diagrama Entidad-Relación (DER). Es importante mencionar que existen distintas representaciones de un DER en las que cambian los aspectos gráficos, pero se modelan los mismos elementos. A continuación detallo los elementos del modelo E/R e incluyo su representación gráfica en el DER.

- **Entidad**

Una entidad es cualquier “objeto (real o abstracto) que existe en la realidad y acerca del cual queremos almacenar información en la base de datos” (De Miguel 2001: 49). Las entidades se agrupan en tipos de entidades, de los cuales podemos identificar ejemplares, por ejemplo, el Sr. Ruiz sería un ejemplar del tipo de entidad PERSONA. Los tipos de entidades se representan con un rectángulo con el nombre del tipo de entidad en su interior. Existen entidades, llamadas débiles, cuya existencia depende de que exista otra entidad, denominada fuerte o regular; las entidades débiles se representan con doble rectángulo.

Algunos ejemplos son los siguientes. El primer ejemplo corresponde a la entidad ESTUDIANTE. El segundo, se refiere a una interrelación (ESCRIBIR) entre una entidad fuerte, TESISTA, y una entidad débil, TESIS. Ésta última es débil porque su existencia depende de la entidad TESISTA.

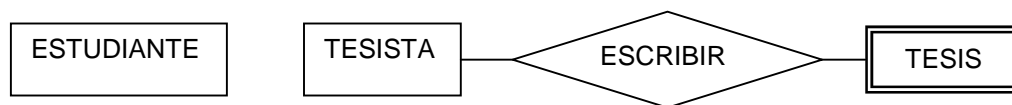


Figura 2. Entidades



- **Atributo**

Un atributo es una característica o propiedad de un tipo de entidad o interrelación, que puede tomar distintos valores. Al conjunto de valores se le distingue como dominio. Los atributos se representan con elipses que incluyen el nombre en su interior.

- **Atributos compuestos**

Existen atributos compuestos de varios dominios, aunque también podemos decir que se forman por varios subatributos. Para representarlos debemos conectarlos mediante líneas con el atributo que componen.

- **Atributos clave**

Todo ejemplar de una entidad debe ser identificado de manera única por un atributo llamado *clave*. Para representarlo gráficamente es necesario subrayar el nombre del atributo.

- **Atributos multivaluados**

Otro tipo de atributo es el multivaluado, caracterizado porque cada ejemplar de entidad puede tomar varios valores del dominio, por ejemplo, un curso puede impartirse en dos idiomas. Su representación es con elipse doble.

- **Atributos derivados**

Otro tipo de atributo es el derivado de una operación entre otros atributos. Éste lo podemos ubicar mediante una elipse punteada.

Es importante comentar que un atributo puede presentar cardinalidad, la que consistiría en el número mínimo y máximo de valores que puede tomar ese atributo en cada ejemplar del tipo de entidad al cual pertenece.

Tomemos como ejemplo nuevamente a la entidad ESTUDIANTE, y coloquémosle algunos de los atributos que ya hemos estudiado.

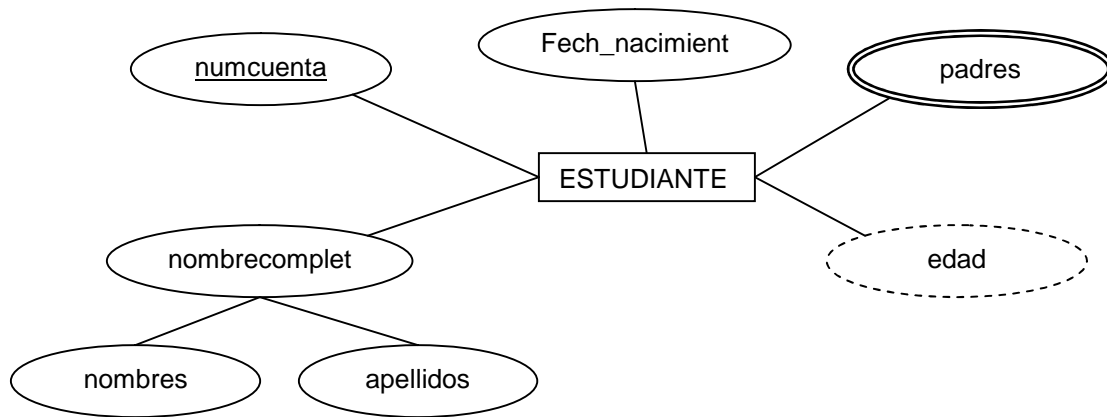


Figura 3. Tipos de atributos en una entidad

Como podemos observar, número de cuenta (**numcuenta**) corresponde a un atributo clave para cada estudiante, ya que lo identifica de manera única, por lo que aparece subrayado. Nombre completo (**nombrecompleto**) es un atributo compuesto de los atributos **nombres** y **apellidos**, por lo que aparecen unidos a través de líneas. El atributo **padres** se encuentra con una elipse doble, refiriéndose a un atributo multivaluado, ya que el estudiante puede tener hasta 2 padres. Finalmente, **edad** es un atributo derivado de una operación entre la fecha de nacimiento (**fech_nacimiento**) y la fecha actual.

▪ Interrelación

Una interrelación es “una asociación, vinculación o correspondencia entre entidades” (De Miguel 2001: 51). Igual que las entidades, las interrelaciones se agrupan en diferentes tipos. Por ejemplo, el tipo de interrelación IMPARTE es la vinculación entre las entidades PROFESOR y CURSO. Un ejemplar de esta interrelación sería la vinculación entre el profesor Sr. Ruiz y el curso Bases de datos. Las interrelaciones se representan gráficamente con un rombo que incluye el nombre del tipo de interrelación en su interior.

En algunos casos de diseño, será posible ver una interrelación como un tipo especial de entidad débil. Para decidir si lo modelaremos como entidad o como



interrelación dependerá de la conveniencia en el modelo y de la realidad que analicemos.

Las interrelaciones cuentan con varias características:

- **Grado**

“Es el número de tipos de entidad que participan en un tipo de interrelación” (De Miguel 2001: 61). Una interrelación de grado dos se refiere a la vinculación de dos tipos de entidades. Un tipo especial de interrelación de grado dos es la reflexiva, que asocia un tipo de entidad consigo misma.

- **Tipo**

“Es el número máximo de ejemplares de un tipo de entidad que pueden estar asociados” (De Miguel 2001: 62). Los tipos son uno a uno (1:1), uno a muchos (1:M) y muchos a muchos (M:M).

- **Papel (rol)**

“Es la función que cada uno de los tipos de entidad realiza en el tipo de interrelación” (De Miguel 2001: 63). Cuando la función de una entidad en la interrelación es ambigua o no se puede inducir de manera clara, se recomienda colocar el papel (rol) en la línea que conecta a la entidad con la interrelación.

- **Cardinalidad**

“Se define como el número máximo y mínimo de ejemplares de un tipo de entidad que pueden estar interrelacionados con un ejemplar del otro tipo” (De Miguel 2001: 63). Los valores que podrían tomar son (0,1), (1,1), (0,N) o (1,N), donde N significa muchos ejemplares.

Para comprender mejor las interrelaciones, veamos el siguiente ejemplo. En éste podemos observar una interrelación de tipo, muchos a muchos (M:M), entre TESISTA y TESIS. Esto significa que muchos tesistas escriben una tesis, pero



también que muchas tesis son escritas por un tesista. Pero, si miramos bien, tenemos una restricción establecida por la cardinalidad en TESISTA, ésta nos indica que pueden participar en la relación al menos 1 tesista y como máximo 4; en otras palabras, una tesis es escrita por mínimo 1 y máximo 4 tesistas.

En el caso de la cardinalidad de TESIS, ésta nos indica que un estudiante escribe 1 o muchas tesis, sin restricción de cuántas. También se indica el papel o rol del tesista en la interrelación, consistiendo su rol en “escritor”.

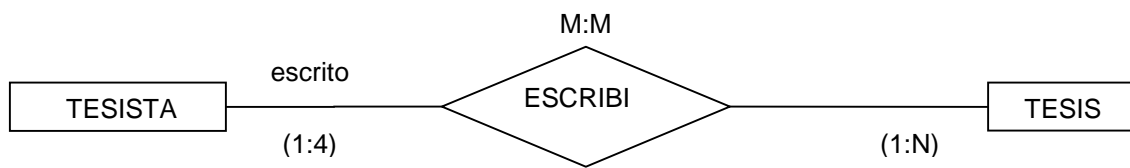


Figura 4. Ejemplo 1. Interrelaciones

En otro ejemplo, las situaciones cambian. De acuerdo con este tipo de relación, una habitación sólo puede estar ocupada por un huésped y un huésped sólo puede ocupar una habitación (1:1). La cardinalidad nos indica que un huésped sólo puede tomar una habitación como máximo (1:1), pero una habitación puede estar desocupada o puede estar ocupada por máximo un huésped, por lo que la cardinalidad de su interrelación es 0:1.

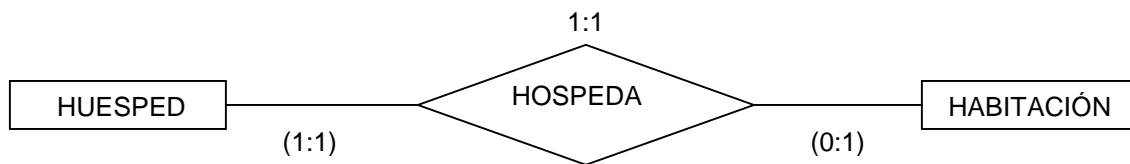


Figura 5. Ejemplo 2. Interrelaciones

4.3.2 E/R extendido

El modelo E/R puede captar la mayoría de las características semánticas para la base de datos, pero es posible extenderlo para captar otros aspectos. Esta



extensión es conocida como modelo Entidad-Relación extendido. Las características de modelo extendido son:

- Especialización.
- Generalización.
- Agregación.
- Herencia de atributos.⁸

4.4 Modelo físico

El modelo Entidad-Relación nos permite representar la realidad sobre la cual vamos a almacenar información. Mediante el Diagrama Entidad-Relación (DER), captamos el significado de todo aquello que queremos almacenar en la base de datos. Gracias a esto, podemos pasar de un modelo conceptual a un modelo lógico y finalmente a uno físico.

Para realizar lo anterior, es necesario determinar el modelo de datos con el que construiremos nuestra base. En nuestro caso, utilizaremos el modelo relacional, que como recordarás, está basado en relaciones o tablas. En el modelo físico se determinan aspectos físicos de almacenamiento como: registros, punteros, direccionamiento y asignación de espacio para memorias intermedias.

La realidad es que hoy en día son los manejadores de bases de datos los que se encargan de esto y el diseñador de bases de datos no juega un papel decisivo en estos aspectos físicos del almacenamiento. La parte donde sí se involucra el experto en bases de datos es en la modificación de parámetros de rendimiento del manejador.

Dado que los RDBMS (Sistema Administrador de Bases de Datos Relacionales) son los más utilizados hoy en día, es común que el proceso de diseño de una

⁸ Si quieres revisar aspectos de estas características puedes leer Silberschatz, et. al. (2006: 190-196).



base de datos se realice inicialmente con el modelo E/R a través de un DER y después se realice un mapeo o transformación a relaciones o tablas de un modelo relacional. A continuación veremos cómo se hace esto.

4.4.1 Implementación de un modelo E/R al modelo relacional

Una vez realizado el modelado para obtener el Diagrama Entidad-Relación, es necesario seguir un conjunto de pasos para convertirlo en un modelo relacional.

Las tres reglas generales para realizar esto son:

- 1) Los tipos de entidades se convierten en relaciones.
- 2) Las interrelaciones N:M se transforman en relaciones.
- 3) Para las interrelaciones 1:N se realiza una propagación de clave a partir del lado de 1 hacia el lado de N.

A continuación, explicaremos a detalle los pasos necesarios para esta derivación. Es necesario recordar que, de forma ideal, se debe mantener la semántica del modelo conceptual en el modelo lógico. Desafortunadamente, no siempre es posible hacerlo. Las reglas específicas para derivar un modelo Entidad-Relación son:

1. Derivación de dominios. Debe crearse los dominios de todos los atributos. Para esto utilizamos la sentencia CREATE DOMAIN, por ejemplo:

```
CREATE DOMAIN edo_civil AS CHAR(1)
    CHECK (VALUE IN ('S', 'C'));
```

2. Derivación de entidades. Cada tipo de entidad se transforma en una relación o tabla. Para ello, utilizamos la sentencia CREATE TABLE.

3. Derivación de atributos. Todo atributo de una entidad se transforma en una columna de su relación correspondiente.

- 3.1 Atributos identificadores (claves o principales). Se transforman en claves primarias. Para ello se usa la restricción de integridad PRIMARY KEY. Por ejemplo:



```
cod_alumno INTEGER CONSTRAINT pk_cod_alumno PRIMARY KEY
```

3.2 Atributos identificadores alternativos (claves candidatas). Se les aplica la restricción de integridad UNIQUE. Por ejemplo:

```
rfc CHAR(13) CONSTRAINT un_rfc UNIQUE
```

3.3 Atributos no identificadores (no clave, no principales). Se les aplica la restricción de integridad de NOT NULL, sólo si es necesario. Por ejemplo:

```
nombre VARCHAR(30) NOT NULL
```

4. Derivación de interrelaciones

4.1 Interrelaciones N:M. La interrelación se transforma en una relación que tendrá como superclave los atributos identificadores (claves o principales) de las entidades que relaciona. Los atributos de la relación resultante (relación o tabla de interrelación) se convierten en llaves foráneas (claves ajenas) mediante la restricción FOREIGN KEY. Para esta restricción es necesario indicar la tabla padre y la llave primaria en esa tabla. Además, podemos indicar las restricciones de borrado y actualización en cascada. Por ejemplo:

```
CREATE TABLE imparte
(
cod_curso INTEGER,
cod_profesor INTEGER,
PRIMARY KEY (cod_curso, cod_profesor), /*Primaria
compuesta*/
CONSTRAINT fk_cod_curso FOREIGN KEY (cod_curso)
REFERENCES curso(cod_curso)
ON DELETE CASCADE
ON UPDATE CASCADE, /* Foránea con borrado y
actualización en cascada*/
CONSTRAINT fk_cod_profesor FOREIGN KEY (cod_profesor)
REFERENCES profesor(cod_profesor)
ON DELETE CASCADE
```




```
        ON UPDATE CASCADE /* Foránea con borrado y
actualización en cascada*/
    )
```

Otro aspecto importante a derivar, es la cardinalidad de las entidades participantes. Como sabemos, existen casos en los que hay restricciones de negocio que impiden que un determinado número de ejemplares de una entidad interactúe con ejemplares de otra entidad. En este caso es necesario agregar una restricción de aserción. Desafortunadamente no todos los manejadores de bases de datos incluyen esta posibilidad. En ese caso, debemos utilizar disparadores (*triggers*) de integridad. Por ejemplo, supongamos que un curso sólo puede ser impartido por cuatro profesores, es decir, cardinalidad (0, 4). En ese caso tendríamos que crear la siguiente aserción:

```
CREATE ASSERTION profesor_curso
CHECK NOT EXIST (SELECT COUNT(*)
                FROM imparte
                GROUP BY cod_curso
                HAVING COUNT(*) >= 4);
```

4.2 Interrelaciones 1:N. La manera general de transformar esta interrelación consiste en propagar (pasar) el atributo identificador (clave o principal) de la entidad con cardinalidad 1 hacia la entidad con cardinalidad N. Como resultado la interrelación se pierde, es decir, no se convierte en entidad como en el caso de N:M.

El atributo propagado se convierte en llave foránea, para el cual debemos indicar si existe borrado en cascada. La cardinalidad de la interrelación también debe ser implementada como ya establecimos, esto es, utilizando restricciones de aserción o *triggers*. Adicionalmente, si la cardinalidad indica que el número de ejemplares puede ser 0, la llave foránea aceptará valores nulos. Si, por el contrario, indica que al menos un ejemplar debe estar



relacionado con uno o más ejemplares (cardinalidad 1), la llave foránea debe tener restricción de NOT NULL.

4.3 Interrelaciones 1:1. No hay regla fija para la transformación de esta interrelación. Lo más común es aplicar la regla 4.2, aunque puede ser posible la aplicación de la regla 4.1. Es importante notar que al aplicar la regla 4.2 necesitamos decidir en qué dirección se propaga la llave primaria. Para decidirlo podemos tomar en cuenta al menos dos aspectos:

- a) Pasar la llave primaria de una entidad fuerte a una débil es generalmente más conveniente.
- b) Pasar la llave primaria de la entidad con cardinalidad (1, 1) a la entidad con cardinalidad (0, 1) permite aplicar NOT NULL a la llave foránea.

5. Atributos de interrelaciones. Si la interrelación se transforma en relación, los atributos de la interrelación se transforman en columnas de la relación resultante. En caso de que se aplicase una propagación de identificador (clave), los atributos de la interrelación pasan en la misma dirección que dicha clave.
6. Otras restricciones. Podemos encontrar restricciones de valores posibles para un atributo. Para transformarlas al modelo relacional utilizamos la restricción CHECK. Además, si el manejador de bases de datos no acepta creación de dominios, con el CHECK es posible restringir los valores de un dominio.
7. Transformación de dependencias de existencia e identificación. En este caso se realiza una propagación de identificador (clave) asegurándonos de poner en la llave foránea, si es necesaria, la restricción de borrado y actualización en cascada. Cuando se trate de una dependencia de identificación, la llave primaria propagada se combina con algún atributo de la entidad débil para formar una superclave.
8. Derivación de tipos y subtipos. No hay una manera de conservar la semántica de los tipos y subtipos en el modelo relacional. Lo más recomendable es crear



relaciones para cada supertipo y para cada subtipo. Después, propagar la clave principal del supertipo en cada relación de los subtipos. Esta clave propagada será, además de llave foránea, la llave primaria de cada subtipo. Finalmente, si queremos asegurarnos de que un tipo no pueda aparecer en varios subtipos, tendremos que usar disparadores (*triggers*) que revisen que la clave del tipo no exista ya en algún subtipo.

4.5. Modelo de clases (UML)

El modelo de clases es una herramienta para modelado de sistemas orientado a objetos. Muestra la estructura estática del sistema mediante clases. Estas clases representan objetos involucrados en el sistema; pueden mantener relaciones entre ellas, ser especializaciones de otras clases o tener dependencias.

Algunas de estas clases pueden terminar almacenadas en la base de datos, pero no necesariamente. El diagrama de clases es producto del modelado de sistemas orientado a objetos, de tal modo que si la base de datos es orientada a objetos, sería de esperarse que las clases del sistema sean persistentes de forma transparente.

El problema radica en si la implementación se realizará en una base de datos relacional. En este caso parece necesario, además de este diagrama, elaborar un Diagrama Entidad-Relación y luego pasarlo a tablas. El diagrama de clases ayuda bastante a modelar las entidades que intervienen en el sistema junto con sus atributos, pero recordemos que no está orientado al modelado de datos.

Existen estándares de representación de este tipo de modelos de clases. Uno muy conocido y utilizado es el Lenguaje de Modelado Unificado (UML). Se trata de una norma de modelado mediante aspectos gráficos auspiciada por el Grupo de Administración de Objetos (*Object Management Group*, OMG) dedicado al desarrollo de especificaciones y estándares para crear componentes de software.



Hemos visto un acercamiento al diseño de base de datos llamado modelado semántico. Éste se realiza mediante un modelo Entidad-Relación que se representa en un Diagrama Entidad Relación. Una vez hecho, hemos obtenido un modelo conceptual de la realidad que queremos almacenar en la base de datos.

Después, siguiendo las reglas que te propusimos en la sección correspondiente, transformamos el modelo E/R en un modelo específico de base de datos. En nuestro caso, se transformó en un modelo relacional, el cual representa el modelo lógico de la base de datos que se implementa en un manejador de base de datos específico mediante instrucciones de SQL. Es precisamente éste último paso, el objetivo del siguiente tema.

Bibliografía del tema 4

- CHEN, Peter P. (1976), "The entity-relationship model - toward a unified view of data", en *ACM Transactions on Database Systems (TODS)*. 1, 1.
- CHURCHER, Clare. (2007), *Beginning Database Design. From Novice to Professional*, Berkeley, CA, Apress.
- DATE, C. J., (2001), *Sistemas de Bases de Datos*, 7ª ed., México, Pearson Education.
- DE MIGUEL, Adoración, et al., (2001), *Diseño de bases de datos relacionales*, Madrid, Alfaomega-Rama.
- JOHNSON, James L., (1997), *Bases de datos. Modelos, lenguajes, diseño*, México, Oxford University Press.
- SILBERSCHATZ, A., et. al., (2006), *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw-Hill.



Actividades de aprendizaje

- A.4.1.** Realiza la lectura del “Proceso de desarrollo de una base de datos para un banco”, que propone Silberschatz, (2006: 197:207) y elabora un resumen. Pon especial atención en el proceso de modelado E/R y en la transformación al modelo relacional (“reducción a esquemas relacionales”).
- A.4.2.** Realiza el modelado E/R del siguiente caso de diseño, y represéntalo mediante un DER.

Se necesita un registro de los becarios y los proyectos en los que participan. Los becarios pueden participar en varios proyectos y en cada proyecto siempre trabajan varios becarios. Los becarios tienen los siguientes atributos: número de cuenta, nombre y tareas (grupo de tareas que simultáneamente realiza el becario en el proyecto). De los proyectos necesitamos conocer la fecha de inicio, nombre, número de proyecto y si cuenta con patrocinio de PAPIIT O CONACYT. Existen tres tipos de becarios de acuerdo a sus estudios: licenciatura, maestría y doctorado. Los de licenciatura, además de sus datos generales, cuentan con créditos; los de maestría, nombre de tesis; y los de doctorado, comité doctoral, compuesto por tres o cuatro profesores.

- A.4.3.** Observa el siguiente DER y, empleando los pasos proporcionados en la lectura, obtén un modelo relacional de tablas.

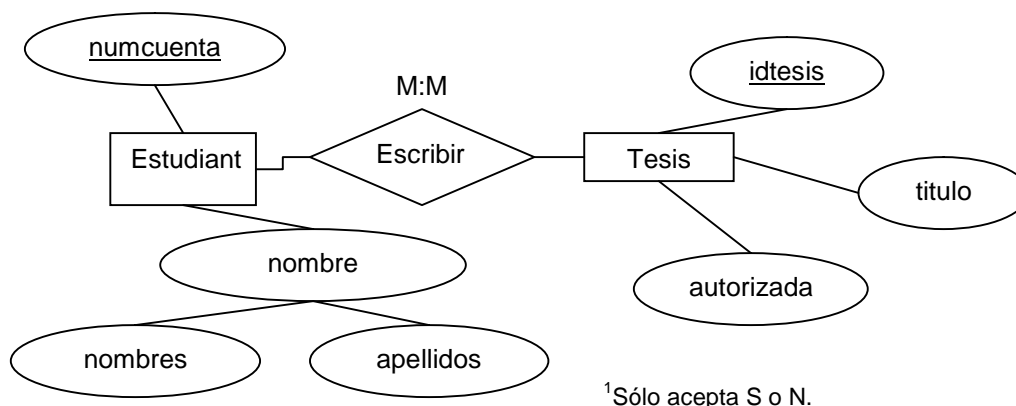


Figura 6. Diagrama Entidad-Relación



A.4.4. Lee y elabora un breve ensayo de las páginas 845 a la 847 del libro de C. J. Date, donde aborda el tema de los OODBMS (Sistemas de gestión de bases de datos orientada a objetos). Confronta esta lectura con el contenido de tu material.

A.4.5. Desarrolla el ejercicio de modelado A.1 propuesto en el libro De Miguel (2001: 435-436).

Cuestionario de autoevaluación

1. ¿En qué consiste el modelado semántico de base de datos?
2. ¿Cuáles son las carencias del modelo relacional que dieron paso al surgimiento del modelo semántico?
3. ¿Cómo se llama el modelo semántico más utilizado para diseñar bases de datos?
4. ¿Qué es una entidad?
5. ¿Qué es una interrelación?
6. Explica cada uno de los tipos de atributos que existen en el modelo E/R.
7. ¿Cuáles son los tipos de entidades?
8. Describe cada una de las características de una interrelación.
9. ¿Cuáles son los pasos generales a seguir para la implementación de un modelo E/R en un modelo relacional?
10. ¿Cuál es el proceso que se sigue para implementar una interrelación de M:M al modelo relacional?

Examen de autoevaluación

1. El modelo semántico es una representación a nivel de modelo lógico de la base de datos.
a) Verdadero b) Falso
2. Edgar Codd propuso el modelo entidad-relación.
a) Verdadero b) Falso



3. Un atributo multivaluado es aquel que se compone de múltiples subatributos.
a) Verdadero b) Falso
4. Un atributo derivado es aquel que se obtiene de una operación entre otros atributos.
a) Verdadero b) Falso
5. La cardinalidad de un atributo consistiría en el número mínimo y máximo de valores que puede tomar ese atributo en cada ejemplar del tipo de entidad al cual pertenece.
a) Verdadero b) Falso
6. Una interrelación tiene tres grados: 1 a 1, 1 a muchos y muchos a muchos.
a) Verdadero b) Falso
7. Una interrelación M:M se implementa al modelo relacional como una nueva relación con los atributos clave de las relaciones involucradas.
a) Verdadero b) Falso
8. Para la implementación del DER en el modelo relacional todos los atributos se convierten en columnas.
a) Verdadero b) Falso
9. La relación 1 a muchos se implementa en el modelo relacional mediante la propagación de clave del lado de 1 al lado de muchos.
a) Verdadero b) Falso
10. En la implementación del modelo E/R al modelo relacional todas las entidades se convierten en relaciones.
a) Verdadero b) Falso



TEMA 5. CONSTRUCCIÓN

Objetivo particular

Al terminar el tema identificarás las principales actividades que realiza un implementador (programador) de bases de datos. Además, lograrás enumerar los objetos que son creados en una base de datos relacional, así como diferenciarlos mediante sus principales características.

Temario detallado

- 5.1 Roles del implementador
- 5.2 Tablas
- 5.3 Integridad
- 5.4 Índices
- 5.5 Vistas
- 5.6 Triggers
- 5.7 Stored Procedures
- 5.8 Manejo de Transacciones
- 5.9 Recuperación



Introducción

Una vez que hemos realizado el diseño de la base de datos y obtenido el modelo relacional (ver tema 4), ha llegado el momento de implementarlo (programarlo) en un manejador de base de datos específico. Para realizar esto será necesario determinar cuál nos conviene. Hoy en día existe una gran variedad de ellos, desde los que cuentan con licencia de uso comercial hasta los basados en la postura del software libre.

Algunos de los aspectos a considerar para la selección son: el volumen de información que pueden almacenar, generalmente medido en *megabytes* o *terabytes*; el número de usuarios que pueden acceder al mismo tiempo; sus ventajas en el manejo de transacciones; su velocidad de respuesta a un número considerable de transacciones, y sus características para imponer seguridad a los datos.

Como recordarás, la programación en una base de datos relacional se realiza con el lenguaje SQL, y a lo largo de este tema conocerás de manera general del uso de este lenguaje. El objetivo de la materia de bases de datos no consiste en que aprendas a programar con SQL, por lo que sólo haremos mención de los comandos más importantes y necesarios. Te explicaremos para qué sirve cada uno de los objetos de una base de datos, cuál es su objetivo y sus principales características.

Afortunadamente el lenguaje SQL es un estándar y, si bien hay diferencias de programación entre los distintos manejadores de bases de datos, lo que revisemos en este tema aplicará para cualquiera de ellos. En otras palabras, desarrollaremos el tema sin pensar en un software específico. En caso de que sea necesario, haremos alguna anotación adicional a este tema. Si te interesa conocer al detalle aspectos de los principales manejadores de bases de datos puedes revisar los capítulos 26 al 29 del libro de Silberschatz (2006: 807-921), en los que presentan cuatro manejadores: PostgreSQL, Oracle, DB2 UDB y SQL Server.



Revisemos entonces los objetos de base de datos que utilizamos para implementar una base de datos útil en un sistema de información.

5.1 Roles del implementador

El diseñador de bases de datos entrega el modelo lógico de tablas al implementador o programador de bases de datos para que construya la base de datos en el sistema manejador. Sus principales roles o actividades son:

- Programar las estructuras de almacenamiento (tablas).
- Implementar las reglas de integridad mediante restricciones o triggers.
- Agilizar las consultas mediante la creación de índices.
- Encapsular consultas en vistas.
- Programar procedimientos almacenados para implementar la lógica de procesamiento de datos dentro de la base de datos.

Para que conozcas a fondo estas actividades, en los apartados siguientes revisaremos cada uno de los objetos programables en una base de datos.

5.2 Tablas

Las tablas son un conjunto de filas y columnas que nos permiten almacenar los datos bajo el enfoque de un modelo relacional. Como sabes, el término tabla se conoce de manera formal como relación, al renglón como tupla y a la columna como atributo. Son en éstas donde almacenamos los datos mediante instrucciones DML.

Para crear una tabla utilizamos el comando CREATE TABLE, con la siguiente sintaxis general⁹:

⁹ La sintaxis que presentamos aquí está simplificada en comparación con la que puedes encontrar en los manuales de programación de SQL de los sistemas de base de datos. Ya que nuestro objetivo no es presentar la manera de programar sino el entendimiento de los objetos



```
CREATE TABLE nombre_tabla
(
  nombre_columna tipo_dato
  [DEFAULT valor_default]
  [CONSTRAINT nombre_constraint TIPO (condición)],
  nombre_columna tipo_dato...,
  nombre_columna tipo_dato...
)
```

Para ejemplificar la manera de crear una tabla veamos un ejemplo. Pensemos que deseamos crear la siguiente tabla:

Autor		
idautor	Nombres	apellidos
1	Ignacio Manuel	Altamirano
2	Manuel	Payno
3	Jorge Luis	Borges
4	Sor Juana Inés	De la Cruz
5	Julio	Cortázar

Siguiendo la plantilla de sintaxis expuesta arriba, podemos crear la tabla de la siguiente manera:

```
CREATE TABLE autor
(
  idautor integer CONSTRAINT pkautor PRIMARY KEY,
  nombres varchar(40) NOT NULL,
  apellidos varchar(40) NOT NULL
);
```

programables de una base de datos, creemos que es necesario ser más específicos. Para entenderla mejor, recuerda que los [] encierran elementos que pueden o no incluirse en la programación y que el uso del carácter | es para proponer distintas opciones agrupadas entre llaves {}.





Creamos una tabla de nombre **auto** con tres columnas. Cada una de ellas se declara por separado y termina su declaración con una coma. Observa que la última columna no tiene una coma al final ya que no le sigue ninguna columna más. Cada definición de columna se compone de nombre, por ejemplo **idautor**, su tipo de dato, en este caso entero **integer**, y la declaración de una restricción de integridad o **constraint**.

El *constraint* se forma por la palabra reservada **CONSTRAINT**, seguida de un nombre para esa restricción, **pkauto**, y el tipo de restricción, en este caso de clave primaria, **PRIMARY KEY**. Las otras columnas se definen de la misma forma y sólo cambian de tipo de dato y de restricción por una de **NOT NULL**.

Otras operaciones que podemos realizar con una tabla son:

- Renombrarla.
- Renombrar una columna.
- Agregar o eliminar columnas.
- Cambiar el tipo de dato de una columna.
- Agregar o eliminar una restricción (*constraint*).
- Agregar o eliminar un DEFAULT.

Estas operaciones se realizan con el comando ALTER TABLE.

Un DEFAULT es un valor predefinido para una columna que se inserta automáticamente si no definimos un valor específico. Otra operación es la de eliminar una tabla con todo y sus datos que ejecutamos con el comando DROP TABLE.

5.3 Integridad

La integridad de una base de datos se establece con restricciones, en inglés *constraints*. Ponemos restricciones en las columnas de una tabla para que acepten o rechacen ciertos valores. Con ello evitamos que los datos ingresados a la base sean incorrectos o inapropiados. A continuación te ofrezco una lista con





posibles restricciones de una columna y la descripción de los valores que acepta o rechaza.

Restricción	Valores que acepta	Valores que rechaza
NOT NULL	Distintos a Null	Valor Null
CHECK	Los que cumplen con la condición establecida por la restricción	Los que no cumplen con la condición establecida por la restricción.
UNIQUE	Valores únicos en la columna	Valores repetidos en la columna.
PRIMARY KEY	Valores únicos distintos a Null	Valores repetidos y valor Null.
FOREIGN KEY	Valores que existan previamente en la clave primaria	Valores que no existan previamente en la clave primaria.

Cuadro 1. Tabla de restricciones de una columna

Las restricciones se programan dentro de la definición de cada columna en un CREATE TABLE. En la siguiente tabla puedes ver ejemplos de la programación de cada tipo de *constraint*.

Restricción	Ejemplo
NOT NULL	<code>titulo text NOT NULL</code>
CHECK	<code>sexo char(1)</code> <code>CONSTRAINT chksexo CHECK (sexo IN ('F', 'M'))</code>
UNIQUE	<code>rfc char(13) CONSTRAINT unirfc UNIQUE</code>
PRIMARY KEY	<code>idlibro integer CONSTRAINT pklibro PRIMARY KEY</code>
FOREIGN KEY	<code>REFERENCES autor (idautor)</code> <code>ON DELETE CASCADE</code> <code>ON UPDATE CASCADE</code>

Cuadro 2. Ejemplos de programación de restricciones

El *constraint* de FOREIGN KEY es particularmente más elaborado. La parte de REFERENCES hace referencia a la tabla padre de la relación, es decir, aquella



donde está la clave primaria a la que hace referencia la clave foránea que se está declarando. La restricción incluye dos cláusulas:

- ON DELETE action
- ON UPDATE action

Éstas indican la acción a ejecutar con los valores de la clave foránea, en caso de que los valores de la clave primaria sean borrados o actualizados. El *action* se refiere a las siguientes posibles acciones:

- NO ACTION: en caso de borrado o actualización, no hacer nada.
- RESTRICT: en caso de borrado o actualización, producir error.
- CASCADE: en caso de borrado o actualización, los valores de la clave foránea son borrados o actualizados al mismo valor de la clave primaria.
- SET NULL: en caso de borrado o actualización, asignar nulos a los valores de la clave foránea.
- SET DEFAULT: en caso de borrado o actualización, asigna los valores por *default* a la columna que es clave foránea.

En seguida puedes ver el código completo para crear una tabla con todos los tipos de *constraints*.

```
CREATE TABLE empleado
(
  idempleado integer CONSTRAINT pkempleado PRIMARY KEY,
  nombres varchar(40) NOT NULL,
  apellidos varchar(40) NOT NULL,
  rfc char(13) CONSTRAINT unirfc UNIQUE,
  idarea integer REFERENCES area(idarea)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```



5.4 Índices

Los índices permiten incrementar el rendimiento de la base de datos haciendo más rápida la ejecución de consultas. En otras palabras, una cláusula SELECT con WHERE encontraría más rápido los registros que cumplen la condición.

Se hacen sobre una o más columnas de una tabla. Es importante resaltar que debemos poner índices sólo en columnas usadas frecuentemente en nuestras expresiones de comparación con la cláusula WHERE.

Para crear un índice utilizamos la cláusula CREATE INDEX y para borrarlo se hace con el comando DROP INDEX, por ejemplo:

```
CREATE INDEX idx_nombres ON empleado(nombres);  
DROP INDEX idx_nombres;
```

5.5 Vistas

Una vista es un objeto de la base de datos que almacena una consulta. Funciona como una tabla, pero no existe físicamente en la base de datos, se genera de forma dinámica. Una vista nos permite encapsular consultas que utilizamos de forma recurrente, nos evita escribirlas de nuevo. También nos ayuda a manipular consultas muy complejas de una forma más sencilla.

La instrucción SQL para consultar datos es SELECT. La sintaxis básica general de esta instrucción es:

```
SELECT nombre_columna, nombre_columna,...  
FROM nombre_tabla  
WHERE condición
```

A partir de este tipo de consultas se crean las vistas con el comando CREATE VIEW, de la siguiente forma:

```
CREATE VIEW nombre_vista AS  
SELECT ...
```



Veamos algunos ejemplos de vistas creadas a partir de diversas consultas:

Seleccionar todos los renglones y todas las columnas:

```
CREATE VIEW vempleados AS
  SELECT *
  FROM empleado;
```

Seleccionar algunas columnas:

```
CREATE VIEW vemp AS
  SELECT idempleado, nombres, apellidos
  FROM empleado;
```

Seleccionar renglones a partir de una condición expresada en la cláusula WHERE:

```
CREATE VIEW vempleado2 AS
  SELECT nombres, apellidos
  FROM empleado
  WHERE idempleado = 2;
```

Recuperar registros de una sola tabla es muy inusual. En la realidad siempre obtenemos datos de diferentes tablas, a veces de muchas. Es importante entonces conocer la manera de “unirlas” para poder obtener valores almacenados en columnas de unas y de otras. Esto lo realizamos con la operación relacional llamada junta o *join*. Existen varios tipos de *join*, entre ellos podemos mencionar:

Cross join

El resultado es un producto cartesiano, es decir, una combinación de todos los valores de una tabla contra todos los valores de otra tabla.

Inner join

El resultado es un conjunto de registros que resultan de la combinación de dos o más tablas, siempre y cuando existan columnas en común y los valores de dichas



columnas coincidan. Este *join* necesita de una cláusula ON que iguale las columnas en común.

Outer join

El resultado es un *inner join* que además incluye aquellos valores donde no hay coincidencia en el origen del lado izquierdo (LEFT OUTER JOIN) o del lado derecho (RIGHT OUTER JOIN) o aquellos que no coinciden en ningún lado (FULL OUTER JOIN).

A continuación un ejemplo de consulta de dos tablas.

```
CREATE VIEW vlibroautor AS
  SELECT libro.titulo, autor.nombres, autor.apellidos
  FROM libro INNER JOIN autor
  ON (libro.idautor = autor.idautor);
```

El resultado de este *join* no incluiría los libros sin autor en caso de que no haya coincidencia entre las claves idautor de las dos tablas. Esto sucede porque el *inner join* rescata sólo aquellos que cumplen la condición de igualdad entre columnas en común. Si quisiéramos rescatar los libros que sí tienen autor y los que no tienen, deberíamos utilizar un *outer join*.

```
CREATE VIEW vlibroautor AS
  SELECT libro.titulo, autor.nombres, autor.apellidos
  FROM libro LEFT OUTER JOIN autor
  ON (libro.idautor = autor.idautor);
```

Se trata de un *left outer join* porque la tabla libro está a la izquierda del *join*. Si lo que quisiéramos fueran todos los autores con y sin libro asociado, entonces tendríamos que programar un *right outer join*.

```
CREATE VIEW vlibroautor AS
  SELECT libro.titulo, autor.nombres, autor.apellidos
  FROM libro RIGHT OUTER JOIN autor
```



```
ON (libro.idautor = autor.idautor);
```

5.6 Triggers

Un *trigger* ejecuta un determinado procedimiento almacenado en la base de datos cuando se realiza cualquier modificación en una tabla. Un *trigger* es una función que se “dispara” antes o después de la instrucción que actualiza la tabla a la que está asociado. Son usados para implementar reglas de integridad de datos, auditoría de tablas importantes y actividades de mantenimiento de datos.

Para crear un *trigger* debemos usar la siguiente sintaxis general:

```
CREATE TRIGGER nombre_trigger
{ BEFORE | AFTER } { DELETE OR UPDATE OR INSERT }
ON nombre_tabla FOR EACH { ROW | STATEMENT }
EXECUTE PROCEDURE nombre_procedimiento;
```

Las opciones de BEFORE o AFTER permiten definir si el trigger se ejecutará antes o después de un evento INSERT, DELETE o UPDATE. La cláusula FOR EACH ROW indica que el procedimiento que dispara el *trigger* será ejecutado por cada renglón actualizado por el evento INSERT, DELETE o UPDATE. Si por el contrario se indica FOR EACH STATEMENT, el procedimiento será disparado una sola vez.

Para borrar un *trigger* contamos con la sentencia:

```
DROP TRIGGER nombre_trigger ON nombre_tabla;
```

Siendo redundantes, un *trigger* dispara un procedimiento almacenado, el cuál puede realizar cualquier acción en la base de datos. De esta manera, podemos crear un *trigger* para los siguientes casos:

- Después de borrar o actualizar en la tabla ventas, actualizar en la tabla venta_total el total de venta.



- Después de modificar, eliminar o actualizar la tabla venta, registrar en la tabla venta_bitacora el usuario que modificó la venta y la hora de modificación.
- Antes de actualizar o eliminar en la tabla cliente, revisar si el cliente no tiene registros en la tabla clientes_morosos, en caso de tener registros detener la actualización.
- Después de eliminar un registro de la tabla préstamo, insertar en la tabla prestamo_historico el registro eliminado.

5.7 Stored Procedures

Un procedimiento almacenado (*stored procedure*) es un conjunto de instrucciones en un lenguaje de programación que se almacenan como un objeto de la base de datos y puede ser ejecutado en cualquier momento. Por lo general estos procedimientos almacenados están escritos en un lenguaje peculiar que combina un lenguaje *procedural* (o de procedimiento) con el SQL.

Este lenguaje se vuelve bastante poderoso en tanto que una declaración de variables, manejo de excepciones, expresión de ciclos y condicionales con las capacidades del SQL. Todos los manejadores de bases de datos incluyen un lenguaje de este tipo, por ejemplo: Oracle tiene el pl/sql, PostgreSQL el pl/pgsql, y SQL Server cuenta con el Transact-sql.

Todo procedimiento almacenado sigue una estructura basada en tres secciones:

1. Sección declarativa. Permite declarar variables y cursores.
2. Sección ejecutiva. Es donde se expresan las instrucciones que procesan los datos.
3. Sección de excepciones. Aquí podemos manejar excepciones producidas por el procedimiento.



Hay tres tipos de procedimientos almacenados. Los procedimientos como tales, que no regresan valor; las funciones, que sí regresan algún valor; y los *triggers*, que como ya vimos, se ejecutan automáticamente cuando se actualiza la tabla a la que están vinculados. Tanto las funciones como los procedimientos reciben parámetros que utilizan dentro de la sección ejecutiva. Dependiendo del lenguaje que utilicemos podremos contar con parámetros de entrada, salida o entrada-salida.

Es justo decir que la programación de procedimientos almacenados varía de manejador en manejador. Además, es muy importante, ya que nos permite encapsular la lógica del procesamiento de datos al interior de la base de datos. Este hecho es una ventaja en comparación con programar el procesamiento de datos en los programas de aplicación, ya que los *stored procedures* se ejecutan más rápido y están precompilados y probados desde antes.

Es posible programar dentro de la base de datos, desde cálculos completos de impuestos y procesamientos estadísticos hasta modificaciones complejas de datos. Desafortunadamente, por el desconocimiento de estos objetos de bases de datos, en muchos equipos de desarrollo se continúa programando fuera de la base de datos.

5.8 Manejo de Transacciones

Una transacción de base de datos es un conjunto de dos o más instrucciones que modifican la información de la base (UPDATE, DELETE o INSERT), las cuales son tratadas como una unidad, de tal forma que se realizan todas o no se realiza ninguna. Una transacción puede terminar en una actualización de los datos (*commit*) o puede terminar sin actualización regresando la base de datos al estado consistente con el cuál empezó (*rollback*).



Una transacción comienza con la palabra `BEGIN TRANSACTION`, todas las instrucciones siguientes a ella forman parte de la transacción. Para que los cambios sean permanentes debe terminar con el comando `COMMIT TRANSACTION`, en caso de querer deshacer los cambios utilizaríamos `ROLLBACK TRANSACTION`.

Las transacciones son utilizadas en los casos en los que las empresas requieren que las operaciones se completen como una unidad. La falta de actualización completa implicaría una falta de consistencia en la información. El ejemplo más claro es el de un cargo y su respectivo abono, sabemos que de no completarse las dos operaciones sufriríamos de un problema en los resultados financieros. Toda base de datos es susceptible de sufrir un error, originado por diversas fuentes, que impida que las operaciones se completen en conjunto.

El peligro de realizar sólo parte de la transacción es que algunos registros queden actualizados y otros no, dando como resultado información errónea. Para evitar estos problemas de consistencia, los manejadores de bases de datos implementan automáticamente un mecanismo de recuperación que revisaremos en la siguiente sección.

5.9 Recuperación

Como decíamos, la recuperación es un mecanismo automático de un sistema de bases de datos que entra en acción cuando ocurre un error en medio de una transacción y ésta no se ha completado. El principio fundamental de la recuperación es que se hacen todas las operaciones o ninguna; si ya se habían registrado algunas operaciones, éstas deben deshacerse. Es mejor que los datos no se actualicen a que se actualicen de manera incompleta y, por tanto, errónea. En seguida verás dos esquemas que muestran los dos comportamientos de una transacción, uno termina con un *commit* y el otro con *rollback*. En el segundo caso, los datos no se actualizan ya que ocurrió un error antes de terminar la transacción.



Estado consistente inicial			Transacción	Estado consistente final																		
<table border="1"><thead><tr><th>Id</th><th>Nombre</th><th>Saldo</th></tr></thead><tbody><tr><td>1</td><td>Alberto</td><td>2000</td></tr><tr><td>2</td><td>Alma</td><td>5000</td></tr></tbody></table>	Id	Nombre	Saldo	1	Alberto	2000	2	Alma	5000		1) Update saldo = 3000 Where id =1; 2) Update saldo = 2500 Where id =2;		<table border="1"><thead><tr><th>Id</th><th>Nombre</th><th>Saldo</th></tr></thead><tbody><tr><td>1</td><td>Alberto</td><td>3000</td></tr><tr><td>2</td><td>Alma</td><td>2500</td></tr></tbody></table>	Id	Nombre	Saldo	1	Alberto	3000	2	Alma	2500
Id	Nombre	Saldo																				
1	Alberto	2000																				
2	Alma	5000																				
Id	Nombre	Saldo																				
1	Alberto	3000																				
2	Alma	2500																				

Cuadro 3. Transacción sin error que termina en *commit*

Estado consistente inicial			Transacción	Estado consistente final																		
<table border="1"><thead><tr><th>Id</th><th>Nombre</th><th>Saldo</th></tr></thead><tbody><tr><td>1</td><td>Alberto</td><td>2000</td></tr><tr><td>2</td><td>Alma</td><td>5000</td></tr></tbody></table>	Id	Nombre	Saldo	1	Alberto	2000	2	Alma	5000		1) Update saldo = 3000 Where id =1; ERROR 2) Update saldo = 2500 Where id =2;		<table border="1"><thead><tr><th>Id</th><th>Nombre</th><th>Saldo</th></tr></thead><tbody><tr><td>1</td><td>Alberto</td><td>2000</td></tr><tr><td>2</td><td>Alma</td><td>5000</td></tr></tbody></table>	Id	Nombre	Saldo	1	Alberto	2000	2	Alma	5000
Id	Nombre	Saldo																				
1	Alberto	2000																				
2	Alma	5000																				
Id	Nombre	Saldo																				
1	Alberto	2000																				
2	Alma	5000																				

Cuadro 4. Transacción con error que termina en *rollback*

En el primer esquema, dado que no existió error en la transacción, ésta finaliza con un *commit* y los datos se actualizan. Pero en el segundo caso, observa que ningún registro es actualizado a pesar de que la operación (1) sí se realizó; al presentarse el error ésta es desecha y la tabla queda como en el estado inicial.

Hemos descrito una serie de objetos de bases de datos que son programables y que permiten implementar el sistema relacional de almacenamiento de datos junto con sus restricciones. Con estos, el programador construye una base de datos funcional para un sistema de información. En el siguiente tema revisaremos cómo administrar la base de datos una vez que está construida.



Bibliografía del tema 5

- DATE, C. J., (2001), *Sistemas de Bases de Datos*, 7ª ed., México, Pearson Education.
- GESCHWINDE, Ewald y SCHÖNIG, Hans-Jürgen. (2002), *PHP and PostgreSQL. Advanced Web Programming*: Sams, Indianapolis, Ind.
- JOHNSON, James L., (1997), *Bases de datos. Modelos, lenguajes, diseño*, México, Oxford University Press.
- SILBERSCHATZ, A., KORTH H., et. al., (2006), *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw-Hill.
- WORSLEY, J., y DRAKE, D., (2002), *Practical PostgreSQL*, O'Reilly, USA.

Actividades de aprendizaje

- A.5.1.** Elabora un mapa conceptual que abarque todos los objetos de base de datos expuestos a lo largo del tema.
- A.5.2.** Realiza un cuadro de tres columnas que resuma los comandos de SQL presentados en el material, colocando en la primera columna el comando, en la segunda el objetivo y en la tercera un ejemplo.
- A.5.3.** Escribe tres operaciones que podrían implementarse con un trigger.
- A.5.4.** Construye un cuadro sinóptico con los tipos de restricciones que existen.
- A.5.5.** Lee el capítulo 17 de Silberschatz (2006: 567-598) y 14 de Date (2001: 454-472), y realiza un ensayo acerca de la recuperación de transacciones.

Cuestionario de autoevaluación

1. ¿Cuáles son las actividades que realiza un implementador de bases de datos?
2. ¿Qué es una tabla?
3. ¿Qué es una vista?
4. ¿Qué es un trigger?
5. ¿Qué es un procedimiento almacenado?
6. ¿Qué es una restricción?



7. Describe los tipos de restricciones que existen.
8. Explica las dos maneras de término de una transacción.
9. ¿Qué es un índice?
10. ¿Cuál es la importancia de la recuperación de transacciones?

Examen de autoevaluación

1. La restricción de NOT NULL rechaza valores que no cumplen una condición.
a) Verdadero b) Falso
2. La restricción de FOREIGN KEY rechaza valores que no aparecen en la clave primaria.
a) Verdadero b) Falso
3. Los índices se crean en columnas que aparecen en la cláusula WHERE.
a) Verdadero b) Falso
4. Un trigger se dispara cuando ocurre un error de actualización en una tabla.
a) Verdadero b) Falso
5. La sección ejecutiva de un procedimiento almacenado permite ejecutar excepciones.
a) Verdadero b) Falso
6. Una vista encapsula una consulta de base de datos.
a) Verdadero b) Falso
7. Los triggers se ejecutan antes o después de una instrucción DML.
a) Verdadero b) Falso
8. Si se ejecuta un trigger FOR EACH ROW, el procedimiento almacenado ligado al trigger se ejecuta sólo una vez.
a) Verdadero b) Falso
9. La restricción de PRIMARY KEY acepta valores null.
a) Verdadero b) Falso
10. La instrucción ALTER TABLE permite renombrar una columna de una tabla.
a) Verdadero b) Falso



TEMA 6. ADMINISTRACIÓN

Objetivo particular

Al terminar el tema reconocerás los roles que cumple un administrador de bases de datos y listarás las actividades fundamentales de esta actividad.

Temario detallado

- 6.1 Roles del administrador
- 6.2 Seguridad
- 6.3 Respaldo
- 6.4 Otras actividades

Introducción

En el tema 1 se mencionó que uno de los usuarios más importantes de un sistema de bases de datos es el administrador o DBA (Database Administrator). Un DBA es el experto responsable de asegurar la continua funcionalidad y operación eficiente de las bases de datos de una organización y de las aplicaciones que acceden a ellas.

La actividad del DBA es de suma importancia, ya que en estos días las bases de datos son vitales para las organizaciones. Este almacenaje de datos tiene la finalidad de ofrecer una ventaja competitiva a las organizaciones mediante el acceso a información oportuna y veraz. Veamos en seguida las actividades de este personaje.

6.1 Roles del administrador

Las principales tareas que realiza el DBA son: administración del software (servidor) de bases de datos (instalación, configuración, monitoreo y actualización



del sistema manejador de bases de datos), implementación de medidas de seguridad, operaciones de respaldo, recuperación, importación y exportación de datos y ajustes de rendimiento.

6.2 Seguridad

Actualmente el tema de la seguridad es un aspecto fundamental que no debe omitirse en todo sistema de cómputo y, en especial, en un sistema de bases de datos. Esta actividad consiste en garantizar que los datos sean accesibles únicamente al personal autorizado. Asimismo debe tener especial cuidado en reducir los riesgos y vulnerabilidades del sistema de bases de datos, a fin de impedir cualquier intrusión de usuarios no autorizados que puedan extraer o modificar los datos. En esta época, en la que la información se ha vuelto un activo crítico en las organizaciones, la seguridad de bases de datos es una gran responsabilidad para el DBA.

El DBA implementa un esquema de seguridad basado en los siguientes elementos principales: un *login* para cada usuario, un password, un nombre de usuario asociado al *login* y grupos de usuarios. Para determinar el esquema se toman en cuenta los siguientes aspectos:

- Determinar las tareas que los usuarios van a realizar en la base de datos. Se establecen las funciones de cada usuario y las acciones que le estarán permitidas realizar: actualizar, capturar o consultar los datos.
- Agrupar de manera lógica a los usuarios con tareas comunes. Estos grupos se basan en lo que harán los usuarios con la base.

Una vez abordados estos aspectos, la implementación del esquema de seguridad se lleva a cabo de la siguiente manera:

- Crear grupos por cada base de datos usando nombres acordes a la organización de la empresa.
- Crear un *login* por cada usuario.



- Asignar una base de datos por *default* a cada *login*.
- Asociar a cada *login* un nombre de usuario.
- Asignar cada nombre de usuario a uno de los grupos determinados.

Otro aspecto de seguridad de bases de datos a considerar son los privilegios que cada usuario o grupo tienen sobre los datos. Estos privilegios son:

Privilegio	Acción que permite
SELECT	Seleccionar datos de una tabla, vista o columna.
INSERT	Insertar nuevos datos a una tabla o vista.
UPDATE	Actualizar datos existentes en una tabla, vista o columna.
DELETE	Borrar datos de una tabla o vista.
EJECUTAR	Ejecutar procedimientos almacenados.

En el siguiente cuadro se describe las actividades de administración de usuarios y privilegios con su correspondiente comando SQL.

Actividad	Comando SQL
Crear usuarios	CREATE USER
Modificar usuarios	ALTER USER
Eliminar usuarios	DROP USER
Crear grupos	CREATE GROUP
Borrar grupos	DROP GROUP
Asignar privilegios	GRANT
Quitar privilegios	REVOKE

6.3 Respaldo

Un respaldo de bases de datos consiste en hacer copias de las tablas de sistema, los objetos creados por el programador (tablas, vistas, procedimientos almacenados, restricciones, etc.) y los datos del usuario. Es el DBA el encargado



de realizar esta labor mediante un esquema de respaldo (*backup*). El respaldo es lo que permite al administrador recuperar los datos en caso de una contingencia, como puede ser:

- Fallas de disco.
- Fallas de software.
- Borrado o actualización incidental.
- Virus.
- Desastres naturales.
- Robo.

Para establecer su esquema, el DBA toma en cuenta los siguientes aspectos:

- ¿Con qué periodicidad debe realizarse el respaldo?
- ¿Qué se debe respaldar?
- ¿Qué medio electrónico se debe usar para el respaldo?
- ¿Debe efectuarse en línea o fuera de línea?
- ¿Existe un mecanismo para asegurarnos que el respaldo se hizo correctamente?
- ¿Dónde se almacenarán los respaldos?
- ¿Cuánto tiempo deben conservarse los respaldos?
- ¿Deben ser hechos de forma manual o automática?
- Si son automáticos ¿cómo se verifican?
- Cuando ocurre una falla ¿cuánto tiempo toma restaurar las bases de datos?

Algunas de las respuestas las encontramos al conocer el volumen de transacciones que se realizan sobre las bases de datos. Pensemos que un respaldo toma tiempo y distrae al procesador de su actividad normal, por lo que el DBMS (Sistema de gestión de base de datos) deja de atender con la misma velocidad las transacciones de los usuarios. De aquí parte que muchos respaldos se hagan por la noche o los fines de semana y de manera automática. El principio



básico consiste en hacer los respaldos en horas en las que se efectúe el menor número de transacciones.

Por otro lado, es primordial verificar que los respaldos se hayan copiado correctamente, ya que puede suceder que al momento de necesitarse no funcionen. Para probar su efectividad, se puede hacer un simulacro de falla en el DBMS, de esta manera se prueban los respaldos y se determina el tiempo que tarda el DBA en restaurar el servicio de la base de datos.

6.4 Otras actividades

Otras de las actividades que realiza el DBA son:

- Determinación de los requerimientos de espacio lógico para la base de datos.
- Monitoreo del espacio disponible para la base de datos.
- Importar y exportar datos.
- Mantener un sistema de tareas automáticas y alertas en caso de problemas con el DBMS.
- Ajustes de configuración de rendimiento
- Monitoreo constante del sistema de base de datos.

Bibliografía del tema 6

DATE, C. J. (2001) *Sistemas de Bases de Datos*, 7ª ed., México, Pearson.

MULLINS, Craig, (2002), *Database administration: the complete guide to practices and procedures*. Boston, Addison-Wesley.

JOHNSON, James L. (1997) *Bases de datos. Modelos, lenguajes, diseño*, Oxford, México.

SILBERSCHATZ, A., KORTH H., et. al., (2006) *Fundamentos de bases de datos*, 5ª ed., México, McGraw-Hill.



Actividades de aprendizaje

- A.6.1.** A partir de la revisión del tema 6, elabora un mapa conceptual para tener un panorama general del mismo.
- A.6.2.** Define con tus propias palabras en qué consiste la administración de base de datos.
- A.6.3.** Realiza un listado de por qué es importante la figura de un administrador de bases de datos en una organización.
- A.6.4.** Realiza la lectura del Capítulo I “What Is a DBA?” del libro de Craig Mullins (2002), y elabora un resumen.
- A.6.5.** Investiga sobre los factores que impactan en el rendimiento de un sistema de bases de datos y elabora un reporte.

Cuestionario de autoevaluación

1. ¿Qué significan las siglas DBA?
2. ¿Por qué es importante un DBA en una organización?
3. ¿Cuáles son las principales actividades de un DBA?
4. ¿Cuál es el objetivo primordial de la administración de bases de datos?
5. ¿En qué consiste un esquema de seguridad de una base de datos?
6. Explica la administración de privilegios de una base de datos.
7. ¿Qué es un grupo en un esquema de seguridad?
8. ¿Con qué comandos de SQL se otorgan y quitan privilegios?
9. ¿Qué aspectos se toman en cuenta para definir un esquema de respaldos para una base de datos?
10. ¿Qué es un respaldo de bases de datos?



Examen de autoevaluación

1. El DBA se encarga de la administración del sistema operativo de un servidor.
a) Verdadero b) Falso
2. Los días y horas de respaldo se definen a partir de la carga transaccional en la base de datos.
a) Verdadero b) Falso
3. Para establecer un esquema de seguridad es necesario definir algunos elementos como: login, password, nombre de usuario y grupo.
a) Verdadero b) Falso
4. El privilegio de EJECUTAR permite a los usuarios consultar tablas y vistas.
a) Verdadero b) Falso
5. El comando SQL para crear un grupo es GRANT GROUP.
a) Verdadero b) Falso
6. El comando SQL para eliminar privilegios es REVOKE.
a) Verdadero b) Falso
7. En un respaldo se involucran tablas de sistema, datos de usuario y objetos de la base de datos.
a) Verdadero b) Falso
8. Algunas causas por las que son necesarios los respaldos son: fallas de hardware, desastres naturales, virus.
a) Verdadero b) Falso
9. Un simulacro de falla en la base de datos nos permite verificar si un respaldo fue bien realizado.
a) Verdadero b) Falso
10. Los respaldos se realizan en horas de alta carga de transacciones.
a) Verdadero b) Falso



TEMA 7. NUEVAS TECNOLOGÍAS

Objetivo particular

Al finalizar el tema, identificarás los objetivos, estrategias, técnicas, enfoques y componentes de las nuevas tecnologías en bases de datos: minería de datos y data warehousing.

Temario detallado

7.1 Minería de datos

- Definición
- Pasos generales de la minería de datos
- Proceso de minería de datos
- Estrategias de minería de datos
- Técnicas de minería de datos
- Aplicaciones de la minería de datos

7.2 *Data Warehousing*

- Conceptos básicos
- Características de un *data warehouse*
- Componentes de un *data warehouse*
- Áreas que usan un *data warehouse*



Introducción

Las necesidades de información del ser humano y de las organizaciones evolucionan con el tiempo. Con el surgimiento de las bases de datos se pudo resolver el problema de registrar información útil para su futura recuperación; fueron posibles cálculos más rápidos y se le dio confiabilidad al procesamiento de grandes cantidades de transacciones. Hoy en día, podemos decir que estas necesidades están prácticamente cubiertas.

De esta forma, las necesidades de información se han vuelto más complejas. Las cantidades de información que guardan las bases de datos empresariales son enormes y su análisis se ha vuelto complicado. La primera necesidad ya no consiste en el almacenamiento y procesamiento, sino en el análisis. Por tanto, las nuevas tecnologías de bases de datos se están orientando al análisis automático de información para brindar soporte a las decisiones y generar conocimiento.

Antes, las necesidades de información eran del estilo: ¿cuánto fue vendido en Nuevo León en septiembre del año pasado?, cuya solución era aplicada con una consulta simple de SQL (*query*). Hoy en día, éstas son: ¿cuántas unidades se vendieron en Nuevo León con respecto a Guadalajara el año pasado, con respecto a este año? La solución ya no es un solo *query* de SQL.; O por ejemplo, con base en las ventas de los últimos cinco años, ¿cuántas podrían ser las unidades vendidas este año? Esta situación obliga a usar métodos que permitan predecir el comportamiento de los datos.

Por otro lado, no es raro encontrar organizaciones que utilicen las bases de datos como centros de almacenaje del acontecer diario de la organización, ya que por ejemplo, esto les permite facturar y llevar control de stocks, pero pocas veces las usan para producir un análisis.

Entonces, ¿qué necesitan las organizaciones para utilizar este análisis automático de información y predecir el comportamiento? Primero, un almacén de datos



construido con ese fin, un *Data Warehouse*, y después, un conjunto de estrategias y métodos de análisis, es decir, la minería de datos. Este tema presenta las características generales de ambas tecnologías.

7.1. Minería de datos

El volumen de datos que guardan actualmente las bases de datos se ha convertido en un recurso importante que debe analizarse. Este análisis permitiría describir y entender los procesos económicos y financieros de las organizaciones. Además, el ser humano cuenta hoy en día con enormes bases de datos científicas como las relacionadas al genoma o la astronomía, que encierran conocimiento que debe ser descubierto.

Una manera de obtener este análisis es mediante el descubrimiento de patrones en los datos. Un patrón es una serie de características o de eventos que presenta alguna regularidad, suceden cada cierto tiempo, en las mismas circunstancias o con los mismos efectos. Si vemos una base de datos, será difícil percibir a simple vista si existe un patrón, de aquí que necesitemos utilizar técnicas especializadas llamadas en conjunto: minería de datos.

▪ Definición

La minería de datos (MD) es también conocida como Descubrimiento de Conocimiento en Bases de Datos (*Knowledge Discovery in Databases*, KDD). Podemos definirla como la aplicación de técnicas estadísticas y de aprendizaje automático para encontrar patrones no triviales en bases de datos, que resulten de interés para el experto de un dominio determinado.

La MD tiene dos enfoques. El primero es descriptivo y consiste en encontrar patrones que nos permitan describir la situación actual de la organización. El segundo es predictivo y trata de obtener modelos que pronostiquen, a partir de los patrones, algún comportamiento interesante en el futuro.



- **Pasos generales de la minería de datos**

La MD se lleva a cabo siempre bajo la idea de cooperación entre el experto del dominio (finanzas, mercadotecnia, ventas) y el personal de informática. A continuación los pasos generales de la MD:

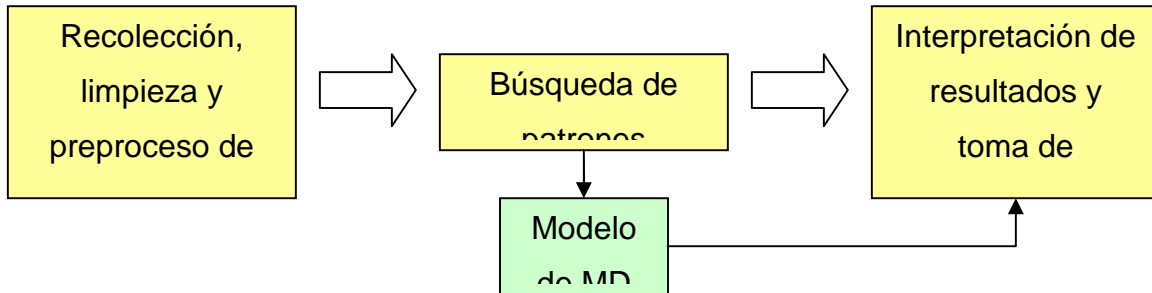


Figura 1. Pasos generales de la Minería de Datos

El proceso de MD produce un modelo de MD, que puede ser descriptivo o predictivo, según los patrones identificados en los datos. Con él se analizan posibles cursos de acción para tomar decisiones. Este modelo puede ser desde una gráfica hasta una red neuronal. La recolección de datos se realiza a partir de las bases de datos transaccionales o de un *data warehouse*. Finalmente, debes saber que el modelo de MD puede ser estadístico o de aprendizaje automático.

- **Proceso de minería de datos**

Primer paso

El proceso de MD comienza con la definición del objetivo de la minería. Esto se hace entre el experto del dominio y el experto en minería de datos. El objetivo puede ser explorar los datos de forma general (conocer las características de los clientes), obtener un modelo descriptivo o clasificador (obtener un modelo que clasifique a los clientes en sujetos de crédito o no), o demostrar una hipótesis (¿es cierto que los cursos más rentables se dan en Guadalajara a personas mayores de 40 años con puestos gerenciales y con ingresos superiores a \$20,000?).



Segundo paso

Una vez identificado el objetivo, será necesario definir las fuentes de datos y los datos relevantes para el análisis. Estos datos pueden estar resumidos o acumulados.

Tercer paso

En seguida, se puede hacer un acercamiento preliminar a los datos. Por lo general se tratan de aplicar medidas estadísticas y obtener gráficas representativas de los datos, por ejemplo: promedios, máximos, mínimos, histogramas, diagramas de barras, pareto o análisis de correlación.

Cuarto paso

En esta etapa se deciden los métodos estadísticos o de aprendizaje automático que serán aplicados a los datos. Estos se verán en los apartados de estrategias y técnicas de minería.

Quinto paso

Este paso consiste en obtener los datos y adaptarlos al formato de entrada de nuestro método. Es común que se llegue a una sola tabla con datos no normalizados como insumo de la minería. Esta tabla suele llamarse “vista minable”.

Sexto paso

Se corren los procesos de aprendizaje automático para obtener un modelo a partir de los datos. Por lo general se utiliza un 70% de los datos originales, como datos de entrenamiento, y el resto se emplea para evaluar si el modelo obtenido es bueno.

Séptimo paso

Evaluación del modelo obtenido mediante medidas especiales (*precision* y *recall*). Esto nos permitirá saber si el modelo obtenido es bueno.



Octavo paso

Evaluar los resultados del proceso de MD con el experto del dominio para saber si son útiles y no triviales.

▪ Estrategias de minería de datos

Las estrategias de minería de datos pueden clasificarse en: supervisadas (clasificación y estimación), no supervisadas (agrupamiento o *clustering*) y análisis de canasta.

Supervisadas

Clasificación

Construye modelos que puedan asignar o clasificar nuevos ejemplos a un conjunto de clases, definidas previamente. Los modelos de clasificación pueden servirnos para:

- Clasificar personas con riesgo de sufrir paro cardíaco.
- Clasificar aspirantes con bajo o alto riesgo crediticio.

Estimación

Consiste en determinar el valor de un atributo numérico. Ejemplos:

- Estimar las ventas de próximo mes.
- Estimar la probabilidad de que una tarjeta de crédito sea usada de manera fraudulenta.

No supervisadas

Agrupamiento

En este caso, no existen categorías predefinidas para clasificar ejemplos o instancias. Estas clases (*clusters*) se proponen como resultado del proceso de minería de datos. A este modelo de agrupamiento le acompañan medidas de cercanía entre los ejemplos o instancias agrupadas. Algunos ejemplos son:

- *Clustering* de clientes de acuerdo con su historial de ventas.
- Agrupamiento de nuevas especies de flores.



Análisis de canasta

Este análisis nos permite encontrar relaciones entre productos de acuerdo con sus ventas. Para este análisis suelen utilizarse algoritmos que producen reglas de asociación. Una regla sería la siguiente: “Cuando un cliente compra pan marca ‘Bready’ también compra leche ‘Milky’”.

▪ **Técnicas de minería de datos**

Las técnicas de MD se usan para aplicar una estrategia determinada a un conjunto de datos. Éstas cuentan generalmente con un algoritmo y una estructura de conocimiento. Las principales técnicas de minería son:

- Reglas de Producción.
- Regresión Lineal.
- Árboles de Decisión.
- *Clustering*.
- Reglas de asociación (análisis de canasta).

Reglas de producción

Son reglas con la forma:

IF antecedente
THEN consecuencia

Regresión lineal

Permite crear ecuaciones matemáticas con más de una variable independiente.

Árboles de decisión

Es un conjunto de nodos que representan preguntas con las que se clasifica un ejemplo o instancia en una categoría predefinida.



Clustering

Como lo habíamos mencionado, consiste en encontrar *clusters*, también llamados cúmulos, nubes, agrupamientos o categorías en conjuntos de datos. Lo empleamos cuando no estamos seguros de las categorías que existen en ellos.

Reglas de asociación

Representan asociaciones entre atributos contenidos en las bases de datos.

▪ Aplicaciones de la minería de datos

Algunas de las actividades en las que resulta útil el uso de la minería de datos:

- Identificar patrones de fraudes en el uso de tarjetas de crédito.
- Identificar buenos y malos clientes.
- Predecir los clientes que podrían cambiar de banco.
- Encontrar relaciones entre indicadores financieros.
- Desarrollar modelos para diagnóstico o identificación de enfermedades.
- Predecir el perfil de los clientes que podrían adquirir nuevos productos.

7.2. Data Warehousing

Los avances en la tecnología de bases de datos y el desarrollo de un conjunto variado de sistemas manejadores han brindado invaluables beneficios al procesamiento automatizado de información. Así, las organizaciones se han convertido en continuos desarrolladores de bases de datos transaccionales para apoyar sus actividades económico-administrativas.

Junto a la observación anterior es necesario rescatar dos aspectos que no han sido favorables para la misma organización.

Primero, se volvió práctica común que distintos departamentos de una misma organización decidieran trabajar con distintos sistemas manejadores de bases de datos, incluso no resulta raro encontrar en una misma área distinto software de





base de datos; situación que ha generado problemas para el análisis y procesamiento consolidado de datos.

Segundo, las organizaciones se preocuparon mucho - y con justa razón - por crear sistemas robustos para captar las operaciones diarias de la organización, y no contemplaron la labor del análisis de los datos. De esta manera, es normal encontrar en muchas organizaciones enormes bases de datos transaccionales y procesos de análisis de datos manuales basados en reportes automatizados, eso si hay buena suerte.

Con el afán de responder a estas problemáticas, se propusieron nuevos modelos de bases de datos que permiten contar con un gran almacén consolidado de datos listo para aplicarle procesos de análisis automático. A esta nueva tecnología se le llamó *data warehousing*.

▪ **Conceptos básicos**

Para Chaudhuri y Dayal, el *data warehousing* consiste en una colección de tecnologías que permiten mejores y más rápidas decisiones:

Data warehousing is a collection of decision support technologies, aimed at enabling the knowledge worker (executive, manager, analyst) to make better and faster decisions.¹⁰

El objetivo del *data warehousing* es brindar la tecnología necesaria para obtener resúmenes complejos de información y conocimiento. Asimismo, empleando las respectivas tecnologías, herramientas y metodologías, se podría crear, usar y mantener un *data warehouse*.

Por su parte, el concepto de *data warehouse* (DW) se puede entender como un depósito centralizado de datos que ayuda al análisis del negocio. Es un almacén que unifica las bases de datos empresariales o departamentales sin importar el

¹⁰ CHAUDHURI, S., DAYAL, U., *An Overview of Data Warehousing and OLAP Technology*, ACM SIGMOD Record, no. 26-1. 1997.



sistema manejador en el que se encuentren. Uno de los precursores de esta tecnología indica: “A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management’s decisions”.¹¹

Al respecto, también se suele escuchar el concepto de *Data Mart*. Vamos a entender un *data mart* como un *data warehouse* departamental, con las mismas características y componentes, pero a nivel de un departamento de la organización. Es más, podríamos decir que un *data warehouse* se forma de todos los *data mart* de la organización.

- **Características de un *data warehouse***

A partir de la definición del llamado padre del *data warehouse*, William H. Inmon, podemos identificar sus características:

Orientados a temas

Los datos de un DW deben estar recolectados para proporcionar información sobre los temas importantes de la empresa y no sobre sus operaciones diarias.

Integrado

Los datos están integrados a partir de una variedad de bases de datos transaccionales provenientes de los sistemas OLTP de la organización. Los datos deben ofrecer una imagen corporativa general de ella.

No volátil

Los datos de un DW no son eliminados ni modificados, ya que se quiere representar la historia de la organización.

Variante en el tiempo

Los datos están asociados a períodos de tiempo, específicos y bien identificados.

¹¹ William H. Inmon, *Building the data warehouse*, 3a ed., Nueva York, John Wiley & Sons. 2002.



▪ Componentes de un *data warehouse*

Los principales componentes de un DW incluyen el *data warehouse* como tal y algunos elementos adicionales como las fuentes y destinos del mismo.

Bases de datos transaccionales y otros recursos externos

Son las fuentes de origen de los datos las que forman el DW. De ellas se extraen los datos y se cargan en las estructuras del DW. Es común que los datos se tengan que limpiar y transformar. Este proceso es conocido como ETL (extract, transform and load).

Metadatos

Son datos que describen el contenido del *data warehouse*. Entre estos se incluyen el origen de los datos, el responsable y las descripciones sobre los tipos de resúmenes de datos.

La base de datos del DW

Es la base de datos que almacena los datos del *data warehouse*. Es normal observar que el almacenamiento se haya realizado bajo un modelo dimensional y no relacional. Este modelo consiste en un grupo de tablas de dimensiones que guardan una relación de uno a muchos con una tabla principal o tabla de hechos.



Measured facts in the retail sales schema.

Figura 2. Ejemplo de un Modelo dimensional de almacenamiento¹²

¹² Kimball (2002: 36).



Herramientas de consulta

La construcción de un *data warehouse* conlleva la creación de herramientas de procesamiento y análisis de los datos contenidos en él. Estas herramientas pueden ser parte de lo que se conoce como un sistema OLAP (*On line analytical processing*) o ser herramientas de minería de datos.

Los usuarios

Son el último componente de un DW y son los que explotan sus beneficios, ya sea mediante el resultado de la minería de datos o de sistemas de consulta para soporte de las decisiones.

▪ Áreas que usan un *data warehouse*

A continuación se listan las diversas áreas que en la actualidad han adoptado la tecnología del *data warehousing* para mejorar sus negocios:

- Comercio minorista
- Servicios financieros
- Análisis de rentabilidad
- Gestión de riesgo y prevención de fraude
- Análisis de marketing
- Análisis y planificación de distribución
- Telecomunicaciones
- Gobierno
- Salud
- Seguros

El *data warehouse* se está convirtiendo en la base de los procesos analíticos de negocios y es importante que tú, como estudiante de informática, conozcas sus aspectos fundamentales, que hemos expuesto a lo largo de este tema.



Bibliografía del tema 7

- CHAUDHURI, S., U. DAYAL, (1997), *An Overview of Data Warehousing and OLAP Technology*, ACM SIGMOD Record, no. 26-1.
- GUTIÉRREZ HERNÁNDEZ, Guadalupe Vanessa Carolina, y Verónica barranco serrano, 2008, *Minería de datos dentro del proceso de KDD aplicado a la base de datos de circulación bibliográfica de la Biblioteca Central*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.
- IBÁÑEZ CARRANZA, Alma Rebeca, 2008, *Data Warehouse. Guía para el diseño de un Data Warehouse*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.
- INMON, William H., (2002) *Building the data warehouse*, 3a ed., Nueva York, John Wiley & Sons.
- KIMBALL, Ralph, (2002), *The data warehouse toolkit: the complete guide to dimensional modeling*, 2a ed., Nueva York, John Wiley & Sons.
- MALLACH Efrem, (2000), *Decision support and data warehouse systems*, Nueva York, McGraw-Hill.
- REYES GARCÍA, Carlos Tomás. (2007), *La Minería de Datos como Herramienta para la Toma de Decisiones en el Proceso de Calendarización de Cursos de Cómputo*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.

Actividades de aprendizaje

- A.7.1.** Elabora un mapa conceptual del tema 7.1 Minería de datos, que abarque todos los aspectos expuestos en la lectura.
- A.7.2.** Realiza un cuadro sinóptico de las estrategias de minería de datos.
- A.7.3.** Construye un cuadro sinóptico que describa las técnicas de minería de datos.
- A.7.4.** Diseña un esquema que asocie las estrategias con las técnicas de minería de datos.



A.7.5. Elabora un mapa conceptual del tema 7.2. Data Warehousing, que abarque todo lo expuesto en la lectura.

A.7.6. Plasma, en un cuadro sinóptico, los componentes de un data warehouse.

A.7.7. Realiza un cuadro comparativo de las características del data warehousing y un data warehouse.

Cuestionario de autoevaluación

1. ¿En qué consiste la minería de datos?
2. ¿Quiénes están involucrados en el proceso de minería de datos?
3. Describe los dos enfoques que puede tomar la minería de datos.
4. Explica el proceso de minería de datos.
5. ¿Cuáles son las estrategias de minería de datos?
6. Menciona algunas de las técnicas de minería de datos.
7. Define el data warehousing.
8. ¿Qué es un data warehouse?
9. ¿Cuáles son las características de un data warehouse?
10. Describe los componentes de un data warehouse.

Examen de autoevaluación

1. Los dos enfoques de la minería de datos son: clasificador y estimador.
a) Verdadero b) Falso
2. La minería de datos busca descubrir patrones no triviales en los datos.
a) Verdadero b) Falso
3. La minería de datos se realiza únicamente por un experto en cómputo.
a) Verdadero b) Falso
4. La estrategia de clasificación permite obtener grupos o clusters de datos.
a) Verdadero b) Falso
5. El análisis de canasta permite obtener valores numéricos estimados.
a) Verdadero b) Falso



6. Las reglas de producción consisten de un grupo de nodos que clasifican un ejemplo.

a) Verdadero b) Falso

7. Un data warehouse refleja la actividad diaria de la empresa, es actualizable y sin referencia a un tiempo específico.

a) Verdadero b) Falso

8. Un data mart se forma de varios data warehouse.

a) Verdadero b) Falso

9. Los metadatos y las bases de datos transaccionales son componentes de un data warehouse.

a) Verdadero b) Falso

10. El modelo dimensional se basa en tablas de muchos a muchos entre diversos catálogos dimensionales.

a) Verdadero b) Falso



Bibliografía básica

- ATKINSON, Malcom., (et. al.), The Object-Oriented Database System Manifesto, documento digital en <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/clamen/OODBMS/Manifesto/index.html>. 1989. Visitada el 13 de noviembre de 2008.
- BERTINO, Elisa y MARTINO, (1995), Lorenzo. *Sistemas de bases de datos orientadas a objetos. Conceptos y arquitectura*, Massachussets, Addison-Wesley.
- CHAUDHURI, S., DAYAL, U., (1997), *An Overview of Data Warehousing and OLAP Technology*, ACM SIGMOD Record, no. 26-1.
- CHEN, Peter P., (1976), "The entity-relationship model - toward a unified view of data", en *ACM Transactions on Database Systems (TODS)*. 1, 1.
- CHURCHER, Clare, (2007), *Beginning Database Design. From Novice to Professional*, Berkeley, CA, Apress.
- CODD, E. F., (1970), "A relational model of data for large shared data banks", en *Communications of the ACM*, 13, 6.
- CODD, E.F., (1985), "Is Your DBMS Really Relational?", en *ComputerWorld*, 14 de octubre.
- DATE, C. J., (2001), *Sistemas de Bases de Datos*, 7ª ed., México, Pearson.
- DE MIGUEL, Adoración, et al., (2001), *Diseño de bases de datos relacionales*, Madrid, Alfaomega-Rama.
- ELMASRI, Ramez, (2002), *Fundamentos de sistemas de bases de datos*, México, Pearson Educación y Addison-Wesley.
- GESCHWINDE, Ewald y SCHÖNIG, Hans-Jürgen, (2002), *PHP and PostgreSQL. Advanced Web Programming*, Indianapolis, Ind., Sams.
- GUTIÉRREZ HERNÁNDEZ, Guadalupe Vanessa Carolina, BARRANCO SERRANO, Verónica, (2008), *Minería de datos dentro del proceso de KDD aplicado a la base de datos de circulación bibliográfica de la Biblioteca Central*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.



- HUGHES, J. G., (1991), *Object-Oriented databases*, Nueva York, Prentice Hall.
- IBÁÑEZ CARRANZA, Alma Rebeca, (2008), *Data Warehouse. Guía para el diseño de un Data Warehouse*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.
- INMON, William H., (2002), *Building the data warehouse*, tercera edición, John Wiley & Sons, Nueva York.
- JOHNSON, James L., (1997), *Bases de datos. Modelos, lenguajes, diseño*, México, Oxford.
- KIMBALL, Ralph, (2002), *The data warehouse toolkit: the complete guide to dimensional modeling*, segunda edición, John Wiley & Sons, Nueva York.
- MALLACH Efrem, (2000), *Decision support and data warehouse systems*, McGraw-Hill, Nueva York.
- MULLINS, Craig, (2002), *Database administration: the complete guide to practices and procedures*. Boston, Addison-Wesley.
- REYES GARCÍA, Carlos Tomás, (2007), *La Minería de Datos como Herramienta para la Toma de Decisiones en el Proceso de Calendarización de Cursos de Cómputo*, Tesis de licenciatura (Licenciado en Informática), Facultad de Contaduría y Administración, UNAM.
- SILBERSCHATZ, A., et. al., (2006), *Fundamentos de bases de datos*, 5ª ed., Madrid, McGraw-Hill.
- WORSLEY, J. y DRAKE, D., (2002), *Practical PostgreSQL*, O'Reilly, EE.UU.



RESPUESTAS A LOS EXÁMENES DE AUTOEVALUACIÓN

Bases de Datos

	Tema 1	Tema 2	Tema 3	Tema 4	Tema 5	Tema 6	Tema 7
1.	F	F	F	F	F	F	F
2.	F	F	V	F	V	V	V
3.	V	F	F	F	V	V	F
4.	V	F	V	V	F	F	F
5.	V	V	F	V	F	F	F
6.	V	F	V	F	V	V	F
7.	V	F	V	V	V	V	F
8.	F	V	F	V	F	V	F
9.	V	F	F	V	F	V	V
10.	F	F	F	V	V	F	F