

ISSN: 0718 – 1701



UTEM

Serie Bibliotecología y Gestión de Información N° 42, Noviembre - Diciembre 2008

**Introducción al modelamiento de bases de datos y SLQ
básico para Bibliotecarios.**

**Ana Chamorro Malagueño
Claudio Escobar Arriagada**



D · G · I

Departamento
de Gestión de
Información
Escuela de
Bibliotecología

Serie Bibliotecología y Gestión de Información es publicada desde Octubre de 2005 por el Departamento de Gestión de Información de la Universidad Tecnológica Metropolitana. Dr. Hernán Alessandri, 722, 6º piso, Providencia, Santiago, Chile, www.utem.cl

Sus artículos están disponibles en versión electrónica en E-prints in Library and Information Science: <http://eprints.rclis.org> y están indizados e integrados en la base de datos "Fuente Académica" de EBSCO Information Services.

Está registrada en el Sistema Regional de Información en Línea para Revistas Científicas de América Latina, El Caribe, España y Portugal (LATINDEX)

Sitio Web: <http://www.bibliotecarios.cl/servicios/serie-bibliotecologia-y-gestion-de-informacion/>

Dirección Editorial

- Héctor Gómez Fuentes, Director (s) Departamento de Gestión de la Información
- Carmen Pérez Ormeño, Directora Escuela de Bibliotecología

Editor Jefe

Héctor Gómez Fuentes

Consejo Editorial

Académicos del Departamento de Gestión de Información

- Mariela Ferrada Cubillos
- Cecilia Jaña Monsalve
- Guillermo Toro Araneda
- Alicia Ramírez González

Presidenta del Colegio de Bibliotecarios de Chile A. G.

Paola Roncatti Galdames

Representante Legal

Haydée Gutiérrez Vilches, Rectora (s)

Decano Facultad de Administración y Economía

Enrique Maturana Lizardi

Secretaria del Departamento de Gestión de Información

Carolina Osorio Silva

Autorizada su reproducción con mención de la fuente.

LAS IDEAS Y OPINIONES CONTENIDAS EN LOS TRABAJOS Y ARTÍCULOS SON DE RESPONSABILIDAD EXCLUSIVA DE LOS AUTORES Y NO EXPRESAN NECESARIAMENTE EL PUNTO DE VISTA DE LA UNIVERSIDAD TECNOLÓGICA METROPOLITANA.

TABLA DE CONTENIDO

Introducción

1. Bases de datos	8
1.1. Historia	8
1.2. Tipos de bases de datos	10
1.3. Sistemas de gestión de bases de datos	11
1.4. Uso de bases de datos en bibliotecas	13
2. Diseño de modelos de datos	14
2.1 Modelo entidad – relación (E – R)	15
2.2 Modelo relacional	15
3. SQL	20
3.1 Historia	20
3.2 Características generales	22
3.3 Lenguaje de definición de datos	22
3.3.1 Create	23
3.3.2 Alter	29
3.3.3 Drop	31
3.4 Lenguaje de manipulación de datos	32
3.4.1 Insert	33
3.4.2 Update	35
3.4.3 Delete	37
3.4.4 Select	38
Conclusiones	41
Bibliografía	42
Anexos	
Anexo A. Formas normales	43
Anexo B. Cómo instalar MySQL en Windows	45

Índices de figuras

Fig. Nº 1.	Tipos de bases de datos	10
Fig. Nº 2.	Modelo entidad-relación	16
Fig. Nº 3.	Ejemplo de tupla	18
Fig. Nº 4.	Ejemplo de atributo	19
Fig. Nº 5.	Modificaciones de SQL	21
Fig. Nº 6.	Tipos de contenido de campo	25
Fig. Nº 7.	Creación de base de datos y de tabla en MySQL	28
Fig. Nº 8.	Uso del commando ALTER	31
Fig. Nº 9.	Uso del comando DROP	32
Fig. Nº 10.	Uso del comando INSERT	35
Fig. Nº 11.	Uso del comando UPDATE	37
Fig. Nº 12.	Uso del comando DELETE	38
Fig. Nº 13.	Uso del comando SELECT	40

Introducción al modelamiento de bases de datos y SQL básico para Bibliotecarios

Claudio Escobar Arriagada

Bibliotecario Documentalista

Licenciado en Bibliotecología y Gestión de Información

Analista Programador Computacional (c)

Sistema de Bibliotecas Pontificia Universidad Católica de Chile

cescobai@uc.cl

Ana Carolina Chamorro Malagueño

Bibliotecaria Documentalista

Postítulo en Gestión Informática

Sistema de Bibliotecas Pontificia Universidad Católica de Chile

achamorr@uc.cl

Resumen

Aproximación al uso de las bases de datos en bibliotecas y centros de información. Se incluye una breve definición y evolución de las bases de datos, y el modelamiento de éstas, explicando los dos modelos mas empleados en la actualidad: modelo entidad – relación y modelo relacional. Se abordan las funciones básicas del lenguaje SQL, el cual permite la creación e interacción con las bases de datos. Para ejemplificar los conceptos tratados, se usa un caso cotidiano, que se aplica a cualquier biblioteca.

Palabras Claves:

Bases de datos, SQL, Bases de datos relacionales, Modelamiento de bases de datos.

Abstract:

The current article is about the use of databases in libraries and information centers. It begins with a short definition and evolution of the databases, and their modeling; by explaining the two most – frequently used models nowadays: entity – relation model and relational model. Also the basic functions of the SQL language will be explained; the language which allows the creation and interaction with the database. For explaining the topics, we will use an ordinary case that applies to any library.

Keywords:

Databases, SQL, Relational database, Database model.

INTRODUCCIÓN

La motivación para realizar el presente artículo nace de la inquietud por posicionar al bibliotecario en un rol más activo en cuanto al desarrollo y uso de las bases de datos, y a la importancia que tienen ellas en las diversas unidades o centros de información. Como bibliotecarios gestionamos información, que es el elemento más importante dentro de las organizaciones. Dicha información es almacenada en bases de datos, por lo que un correcto modelamiento, diseño e implementación, son vitales para el óptimo funcionamiento de los servicios y procesos bibliotecarios.

Encontramos distintas tecnologías que apoyan nuestro quehacer bibliotecario: acceso en línea a catálogos, referencia virtual, reserva en línea, entre otros. Sin embargo, estos avances tecnológicos tarde o temprano quedarán obsoletos o evolucionarán en otras tecnologías, pero la información siempre permanecerá.

Al comenzar a tratar, es importante rescatar una breve definición de las bases de datos, las cuales son entendidas como un conjunto de datos que pertenecen a un mismo contexto, y que son almacenados sistemáticamente para su posterior uso. Existe una serie de conceptos asociados al desarrollo y el uso de las bases de datos, como los sistemas de gestión de bases de datos (SGBD), las formas de representar los problemas y los lenguajes que nos permiten interactuar con los datos que almacenan dichas bases. El lenguaje por excelencia empleado por los SGBD es Structured Query Language (SQL), el cual abordaremos en el presente artículo en forma teórico-práctica y orientada al mundo de las bibliotecas.

1. Bases de datos

1.1. Historia

Antes de que aparecieran las bases de datos, existían los archivos, que si bien facilitaban el manejo y organización de la información, presentaban problemas como la separación y la duplicación de datos, además de problemas de dependencia que se generaba entre ellos. Cada departamento de una empresa desarrollaba sus propios archivos, por tanto muchas veces, los formatos entre éstos eran incompatibles. La estructura que tenían obligaba a hacer consultas fijas y ante los diferentes requerimientos, se desarrollaban diversas aplicaciones que requerían constantemente de un programador.

A diferencia de los archivos, las bases de datos permiten separar datos y programas, mantienen la consistencia e integridad de los datos, y permiten el acceso concurrente a la información.

La historia de las bases de datos se remonta a los años 60¹. En aquellos tiempos la Nasa trabajaba en el proyecto Apolo. Para manejar toda la información que genera este proyecto, la North American Aviation, desarrolla el software GUAM (*General Update Access Method*). Este se basa en una estructura jerárquica como una especie de árbol, en que cada dato se desprende de otro, que a su vez dependen de uno mayor, como un tronco y las ramas de un árbol. Al trabajo luego se suma IBM, dando como fruto el Information Management System (*IMS*). A mediados de los años 60, General Electric desarrolla IDS (*Integrated Data Store*), en éste Charles Bachmann, diseña un sistema de red, que permite graficar mejor la interrelación entre los datos. Estos primeros Sistemas de Gestión de Bases de Datos no tenían un fundamento teórico que los respaldara y

¹ Márquez, M. (2001). *Historia de los sistemas de bases de datos*. Recuperado el 30 de octubre de 2008, de <http://www3.uji.es/~mmarques/f47/apun/node6.html>

necesitaban de complicados programas de aplicación para interactuar con los datos.

La llamada segunda generación de los Sistemas de Gestión de Bases de Datos, se inicia con las investigaciones de Codd en 1970, quien propone el modelo relacional, dando inicio a una serie de sistemas como System R, DB2 y ORACLE, entre otros. Hacia 1976, Chen diseña el modelo entidad-relación, buscando aumentar la capacidad de los sistemas para modelar los datos. En los años 80, se desarrolla el que es conocido como el lenguaje estándar de los modelos relacionales, el SQL (*Structured Query Language*)

Una tercera generación, es la que desarrolla sistemas relacionales extendidos y aquellos orientados a objetos, en el que se manejan los objetos, sus propiedades y las asociaciones entre ellos. Aparecen conceptos como la minería de datos y OLAP (*On-Line Analytical Processing*), que son definidas como soluciones que permiten una mayor y más ágil manipulación de los datos. Hoy en día, se habla de arquitecturas de cliente delgado y modelos de datos semánticos, en que se busca simplificar las consultas del usuario, y aparecen conceptos de uso de XML (*Extensible Markup Language*) y Web semántica.

1.2. Tipos de bases de datos

Las bases de datos se pueden clasificar de acuerdo a dos grandes criterios:

Según el contenido de ellas según variabilidad de datos almacenados

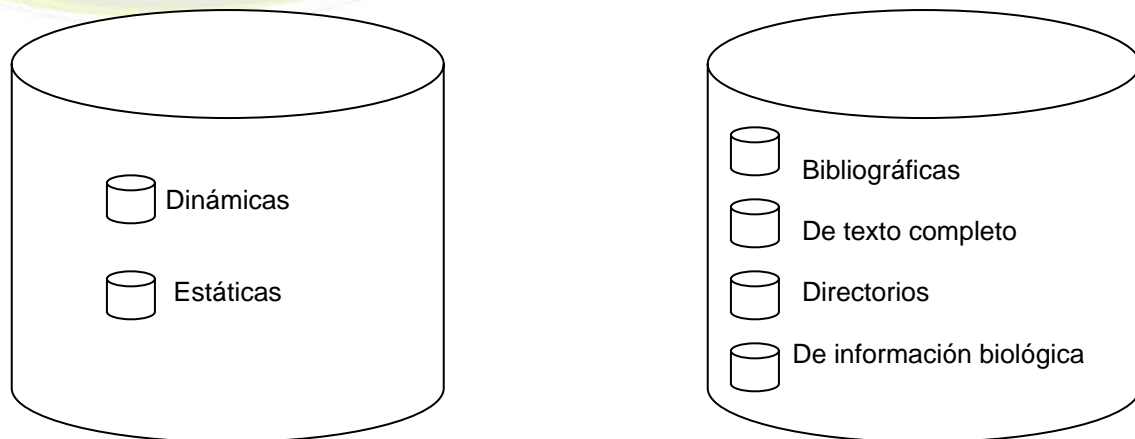


Figura Nº 1
Tipos de bases de datos

En un primer grupo, clasificadas según su contenido, encontramos las bases de datos dinámicas, las que almacenan datos que cambian en el tiempo. Ello porque no sólo permiten consultas, sino también agregar, eliminar o actualizar datos contenidos en ellas, como las usadas en bibliotecas o centros de documentación. Mientras que las bases estáticas, son aquellas que almacenan datos que permanecen en el tiempo sin sufrir cambios, llamados datos de lectura. Son útiles para realizar estudios de comportamiento, proyecciones y tomar decisiones.

Por otro lado, según variabilidad de datos, aparecen las bases bibliográficas. Estas son típicamente las que se usan en bibliotecas, pues direccionan hacia un registro que representa una fuente primaria, como lo es la descripción bibliográfica de un libro, a diferencia de las de texto completo que si llevan a la fuente primaria, como por ejemplo servicios de libros en línea. Mientras que los directorios, contienen una guía alfabética

de datos, típicamente identificados con las guías telefónicas. Finalmente, las llamadas bibliotecas de información biológica, contienen información relacionada a la medicina y la biología, como las bases de datos clínicas.

1.3 Sistemas de gestión de bases de datos (SGBD)

Los SGBD o DBMS (*Database Management System*) son programas de aplicación que permiten almacenar y posteriormente recuperar datos, rápida y estructuradamente, desde las bases. Hacen posible:

- Definir la base en relación a los datos, a las interrelaciones y las restricciones entre ellos,
- Manipular la base, es decir, insertar, modificar, borrar y consultar datos,
- Mantener la integridad y seguridad de los datos, sus relaciones y sus valores².

Los SGBD están constituidos por ciertos lenguajes que permiten la definición de datos, la definición de almacenamiento, y la manipulación de los mismos, además de un diccionario de datos, un gestor de la base de datos, un administrador y los usuarios.

Los lenguajes se dividen en tres:

Lenguaje de definición de datos DDL (*Data Definition Language*) es un lenguaje artificial que permite representar lógicamente los datos.

Lenguaje de definición de almacenamiento SDL (*Data Storage Definition Language*) lenguaje que explica las relaciones, propiedades y restricciones de los datos.

² *Bases de datos* (2002). México, D.F.: Alfaomega.

Lenguaje de manipulación de datos DML (*Data Manipulation Language*) es el que permite recuperar datos además de actualizar el contenido de la base de datos.

El diccionario de datos contiene todas las definiciones necesarias para explicar el problema que manejará el SGBD, es la descripción de la estructura de la base de datos y de las relaciones entre, datos y programas. Almacena el esquema lógico, el físico y los subesquemas de la base, además del mapa de reglas que vinculan los datos.

El gestor de la base, es un componente de software encargado de proporcionar una interfaz entre los datos almacenados y las aplicaciones. Debe garantizar la privacidad, integridad y seguridad de los datos. Además de facilitar el acceso de varios usuarios a la vez sobre la base.

El administrador es la persona encargada de administrar la base y el SGBD. Debe definir el esquema lógico de datos, el físico y los subesquemas, es decir representa el problema real mediante códigos, precisa las estructuras de almacenamiento y métodos de acceso a la información, y define las vistas externas de la base de datos. Además se encarga de entregar privilegios de acceso a los usuarios, ello para controlar la privacidad de los datos. Por otro lado, también debe especificar los procedimientos requeridos para mantener la seguridad de la base.

Los usuarios se dividen: en usuarios terminales, técnicos, especializados y críticos. **Los usuarios terminales** son aquellos no especializados, que interactúan con los datos mediante programas de aplicación.

Los técnicos son los informáticos que desarrollan los programas de aplicación para interactuar con la base de datos.

Los especializados, necesitan una buena gestión de la información, pues utilizan los SGBD como herramientas para el desarrollo de otros sistemas.

Finalmente, **los usuarios críticos** son aquellos que por posiciones gerenciales o directorios, exigen características detalladas de un cierto formato de información y bajo criterios específicos, que no han sido contempladas en la definición inicial de la base.

Existen algunos SGBD que se encuentran disponibles gratuitamente en Internet, como por ejemplo MySQL para Windows o DreamCoder para Oracle, una lista de ellos se encuentra en <http://www.portalprogramas.com/gratis/bases-datos>

1.4 Uso de base de datos en bibliotecas

En el mundo de la bibliotecología trabajamos en la gestión de la información, la que es necesario clasificar dando un determinado orden para su posterior recuperación. Si bien es cierto se puede tener un sistema manual de procesamiento y recuperación de la información, las innovaciones tecnológicas permiten ir automatizando distintos procesos. En ese sentido, la gestión de las bases de datos es un aspecto primordial, pues permite hacer un manejo más efectivo y eficaz de los procesos. La modelación de datos permite que se integren las distintas situaciones que se presentan en cualquier biblioteca o centro de documentación, optimizando recursos tanto tecnológicos como humanos, pues el personal no necesita gastar tiempo en tareas que se pueden definir en la gestión de la base. También es interesante rescatar el aspecto de la independencia entre datos y programas que ofrecen las bases de datos, pues se puede mantener la consistencia e integridad de los datos que se manejan y desarrollar programas o aplicaciones según las necesidades que se manifiesten, favoreciendo la integración de nuevas soluciones a nuevos problemas.

Por otro parte, si bien es cierto, los lenguajes de definición y almacenamiento de datos, presentan un nivel de complejidad que requiere conocimiento del área informática, la tendencia a simplificar los lenguajes de consulta a las bases de datos, favorecen a los usuarios de éstas, pues los acercan más naturalmente a los datos. Bajo este mismo prisma, los usuarios de sistemas de información se ven favorecidos al acceder a la información, cada vez más, en un lenguaje natural y rápidamente, gracias a los procedimientos definidos en la gestión de las bases de datos.

2. Diseño de modelo de datos

Ante un problema dado, es necesario representar los elementos involucrados, las relaciones entre ellos y el comportamiento que caracteriza la situación. Para simplificar la explicación de los problemas del mundo real, aparece el concepto de la abstracción³, la que permite agrupar los objetos en clases que compartan algunas propiedades. Por ejemplo, en una biblioteca universitaria tenemos usuarios que pueden ser alumnos, profesores, administrativos o investigadores, los cuales comparten características que los identifican y que permite agruparlos en una clase general llamada usuarios.

Para lograr una representación lo más completa posible, es necesario entender la naturaleza del sistema a estudiar. Se requiere definir el problema, la función del sistema. Además se requiere definir la arquitectura del problema, que busca precisar los límites del sistema y su relación con otros, diferenciando el problema de otros que se relacionen al principal. Luego, corresponde describir los elementos que forman el sistema, las relaciones y restricciones presentes entre los objetos que interactúan, esto es la definición de la estructura del problema. Desde un

³ *Bases de datos* (2002). México, D.F.: Alfaomega

ámbito dinámico es necesario describir la evolución del sistema, las acciones que tienen o pueden tener los objetos, y analizar el comportamiento del modelo propuesto, facilitando la detección de posibles diferencias entre el comportamiento real y el esperado. Con todo este análisis se logra tener un conjunto de reglas que grafican el problema, un pseudolenguaje que describe las propiedades estáticas y dinámicas del problema, y un conjunto de restricciones que limitan el ámbito del mismo.

Otro aspecto a considerar en el modelo de datos, son las diferentes formas de describir la información que almacena la base. Para ello existen distintas visiones de los datos, por ello se habla de visión conceptual, externa y física. La visión conceptual es la que representa el problema real, define la estructura de toda la base; determina objetos, características de ellos y las relaciones entre éstos. La visión externa en tanto, es la vista que tiene el usuario final de los datos, contiene información parcial que generan las aplicaciones o programas. Finalmente, la visión física es la representación del almacenamiento de los datos, la estructura física de los datos.

2.1 Modelo entidad-relación (E-R)

Este modelo desarrollado por Chen, es uno de los más utilizados, por su capacidad de representar conceptualmente los problemas y la visión integral que refleja. Para representar los problemas reales se utilizan distintos elementos. Estos son: entidades, atributos, vínculos o relaciones y cardinalidades. Las entidades representan el objeto del mundo real. Estos objetos tienen atributos que son las características que tienen los objetos. Las relaciones que se establecen entre las entidades son conocidas como vínculos. Y finalmente, las cardinalidades representan el número de entidades que se relacionan con otra.

En el mundo de la bibliotecología, un caso típico para representar es la situación en que un usuario se dirige a una biblioteca a solicitar un libro determinado. La siguiente figura ejemplifica el caso de préstamo de material bibliográfico en el modelo entidad-relación.

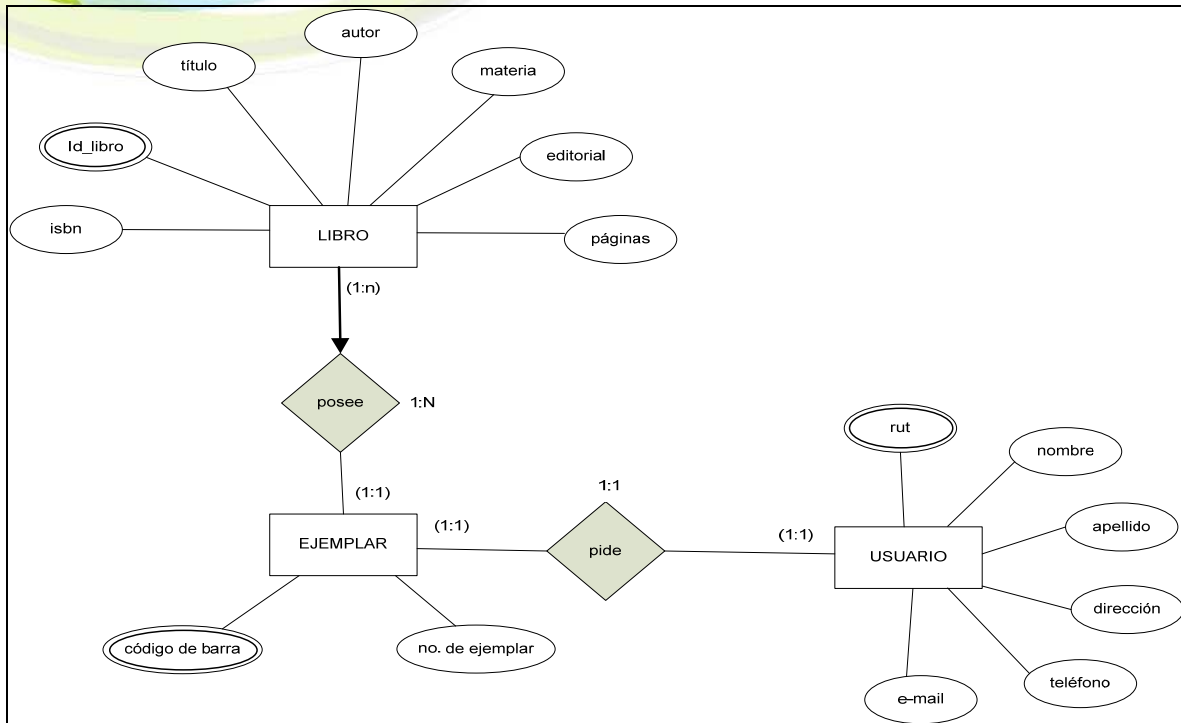



Figura Nº 2
Modelo entidad-relación

En el que cada entidad está representada por un que tiene ciertos atributos graficados por una . Tal como se aprecia en la figura anterior, el objeto denominado usuario, tiene características que permiten identificarlo. Tiene un nombre, un apellido, una dirección, un teléfono, un correo electrónico y un RUT. Pero el rasgo que lo identifica unívocamente es el RUT. Ello porque puedo tener un usuario con el mismo nombre pero no pueden tener el mismo RUT. Estos atributos que permiten la identificación única del objeto, se grafican como una doble elipse.

Por otro lado, los vínculos tienen su representación gráfica en un  en él se grafica la interacción o tipo de asociación que tiene una entidad con otra. En la figura anterior un usuario “pide” un ejemplar X.

Finalmente, los números muestran la cardinalidad entre las entidades. En ella se expresa la forma que tiene una entidad de relacionarse con otra. En el caso planteado, entre la entidad usuario y la de ejemplar, se aprecia una cardinalidad 1:1, que significa que *un* usuario pide *un* ejemplar determinado. El tipo de cardinalidad puede ser:

- | | |
|-----------|---|
| 1:1 | En que una instancia de la entidad se relaciona sólo con una instancia de otra en particular. |
| 1: N | En que una instancia de la entidad se relaciona con una o muchas de otra. |
| N:N o N:M | En que una o muchas instancias de la entidad se relacionan con una o muchas de otra |

A su vez existen tipos de formas de relacionarse con un mínimo y un máximo para que ocurra la relación. En el caso graficado se tiene (1:n) entre la entidad libro y la de ejemplar. Ellos significa que un libro puede tener como mínimo *un* ejemplar y como máximo *muchos* ejemplares. De esta misma forma, se puede tener:

- | | |
|---------------|--|
| (1:1) | un mínimo de uno y máximo de uno |
| (1:n) | un mínimo de uno y máximo de muchos |
| (n:n) o (n:m) | un mínimo de muchos y máximo de muchos |

2.2 Modelo relacional

Es un modelo que permite representar la información relativa a un problema dado, graficando las relaciones entre los objetos. De tal forma, que pueda ser entendida por usuarios sin una preparación en el área. Por otro lado, busca mantener la independencia entre los datos que contienen una base y las aplicaciones que actúan en ella, porque a través de este modelo se busca eliminar la redundancia y pérdida de información, y mejorar la integridad de los datos. También favorece la flexibilidad para realizar consultas sobre la base de datos.

En este modelo el problema se grafica en tablas, con filas y columnas, que representan las relaciones entre datos. Las filas, denominadas tuplas, contienen un registro de información elemental de datos, por ejemplo la información de un usuario:

RUT	Nombre	Apellido	Dirección	Teléfono	E-mail
20.883.045-M	Gonzalo	Mendoza	Ankara 015	5550324	gmix@yahoo.com

Figura Nº 3
Ejemplo de tupla

Cada fila contiene un valor definido que la identifica unívocamente, éste es conocido como clave primaria, en el ejemplo anterior la clave primaria es el RUT. Por esta razón, no es posible que esta clave se repita en otra fila o tupla de la misma tabla. Esta misma clave primaria puede ser usada en otra tabla tomando la forma de clave foránea, lo permite crear una referencia de integridad de los datos.

Las columnas, denominadas campos o atributos, almacenan determinada información relacionada, por ejemplo en una tabla usuarios se tiene una columna donde se ingresa el nombre de ellos.

nombre
Gonzalo
Enrique
Catalina

Figura Nº 4
Ejemplo de atributo

En una misma tabla no pueden existir dos columnas con el mismo nombre, pero ese mismo nombre sí se puede usar en otra tabla relacionada. Cuando una columna no almacena datos, recibe el valor NULL. Los valores permitidos en un campo o atributo, se denominan dominios, y ellos determinan el tipo de dato a ingresar. Por ejemplo si contienen una fecha los valores son del tipo *DATE*, si contienen sólo números del tipo *INTEGER* si sólo son letras del tipo *CHARACTER*, y si combina números y letras del tipo *STRING*.

Para manipular la información se utiliza un lenguaje relacional, álgebra y cálculo relacional, lo que expresa como realizar una consulta y como se devolverá dicha consulta, respectivamente. Pero el lenguaje más usado para realizar consultas es el lenguaje estándar SQL (*Structured Query Language*) el cual se explica más adelante.

Para garantizar la integridad y la duplicidad de registros, el modelo relacional considera un proceso de normalización de la base. Este consiste en el uso de una serie de reglas o formas normales, que optimizan el uso de la base, evitando la redundancia. Por ejemplo, en una base de datos en Primera Forma Normal (*1FN*) no hay campos múltiples.

Si desea más información sobre las formas normales remítase al anexo A del artículo.

3. SQL

Es un 'Lenguaje de consulta estructurado' (Structured Query Language), en el cual se basan la mayoría de los Sistemas de Gestión de Bases de Datos, tanto para crear la estructura de una base de datos, como para gestionar los datos.

3.1 Historia

Sus orígenes los podemos encontrar a comienzo de los años 70. Edgard Frank Codd, investigador de IBM, propone el modelo entidad relación para representar conceptualmente las relaciones entre los elementos que intervendrán en la creación de una base de datos. Asociado a este modelo, Codd plantea la necesidad de un sublenguaje que permita el acceso a los datos. Basándose en estas ideas, un equipo de investigación de IBM desarrolla el lenguaje SEQUEL (Structured English Query Language), el cual fue empleado por la misma compañía en el desarrollo de 'System R' (uno de los primeros SGBD). De esta forma SEQUEL se convirtió en el predecesor de SQL. En el año 1979, ORACLE emplea este sublenguaje en su Sistema de Gestión de Base de Datos.

En los años posteriores SEQUEL sufrió varias modificaciones, dando paso a la primera versión de SQL, convirtiéndose en el lenguaje por excelencia de los SGBD relacionales.

En el año 1986 fue estandarizado por ANSI, dando a lugar la primera versión estandarizada de este lenguaje, el "SQL-86" o "SQL 1". Al año siguiente esta versión de SQL es aceptada por la ISO.

El ANSI SQL ha sufrido varias modificaciones, siendo las principales las que se muestran en la siguiente tabla:

Año	Nombre	Alias	Comentarios
1986	SQL-86	SQL-87	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	SQL-89		Revisión menor.
1992	SQL-92	SQL2	Revisión mayor.
1999	SQL:1999	SQL2000	Se agregaron expresiones regulares, consultas recursivas (para relaciones jerárquicas) y algunas características orientadas a objetos.
2003	SQL:2003		Introduce algunas características de XML, cambios en las funciones y de las columnas auto numéricas.
2006	SQL:2006		ISO/IEC 9075-14:2006 Define las maneras en las cuales el SQL se puede utilizar conjuntamente con XML. Define maneras importar y guardar datos XML en una base de datos SQL, manipulándolos dentro de la base de datos y publicando el XML y los datos SQL convencionales en forma XML. Además, proporciona facilidades que permiten a las aplicaciones integrar dentro de su código SQL el uso de XQuery, lenguaje de consulta XML publicado por el W3C (World Wide Web Consortium) para acceso concurrente a datos ordinarios SQL y documentos XML.

Figura Nº 5
Modificaciones de SQL

3.2 Características generales

SQL está conformado por 3 sublenguajes, cumpliendo cada uno de ellos una función específica.

1. **Lenguaje de definición de datos (LDD):** Es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma, llevar a cabo las tareas de definición de las estructuras que almacenarán los datos, así como de los procedimientos o funciones que permitan consultarlos.
2. **Lenguaje de manipulación de datos (LMD):** Es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma, llevar a cabo las tareas de consulta o manipulación de los datos.
3. **Lenguaje de Control de Datos (DCL):** Es un lenguaje proporcionado por el sistema de gestión de base de datos que incluye una serie de comandos SQL, que permiten al administrador controlar el acceso a los datos contenidos en la base de datos.

Para fines del presente artículo sólo se tratarán los dos primeros sublenguajes, con los cuales podremos crear una base de datos, y realizar los procedimientos básicos, tales como insertar, modificar, eliminar y seleccionar datos.

3.3 Lenguaje de definición de datos

Como ya se ha hablado en el apartado anterior, este sublenguaje es el encargado de la 'estructura' de la base de datos. Un Sistema de Gestión de Bases de Datos debe poder crear, eliminar y modificar tablas, que cómo se ha mencionado anteriormente, son el contenedor de los datos.

Es de suma importancia señalar que los SGBD, en su mayoría, poseen una interfaz gráfica que permite realizar el manejo de la estructura de las bases de datos a través de botones. De todos modos es necesario conocer los códigos y la sintaxis correcta para poder solucionar cualquier inconveniente que se pueda encontrar.

A continuación veremos los códigos principales, y para desarrollar algunos ejemplos usaremos el modelo relacional visto en el capítulo anterior.

3.3.1 Create

Este código permite crear una base de datos y sus respectivas tablas.

Para crear una base de datos basta con escribir la siguiente sintaxis:

```
create database nombre_base_de_datos;
```

*nombre_base_de_datos corresponde al nombre que le queremos dar a la Base de Datos.

Con esa línea de código ya tenemos creada nuestra base de datos, ahora procederemos a crear las tablas que contendrán los datos, para ello también haremos uso del comando 'create'.

```
create table nombre_tabla  
(  
campo1 tipo_campo not null auto_increment,  
campo2 tipo_campo not null,  
campo3 tipo_campo,  
primary key(campo1)  
);
```

Este código, que es bastante más extenso que el anterior necesita de una explicación un poco más profunda. Comenzaremos por explicar el concepto de 'tipo de campo'.

Toda tabla contiene información relacionada con algo en específico, por ejemplo, la tabla Libro posee varios campos con información relacionada sobre libros, tales como autor, ISBN, materia, entre otros. Si ponemos especial cuidado en los campos, nos podremos dar cuenta de que existen campos con diferente tipo de contenidos, tales como texto, números, números y texto (alfanuméricos), imágenes, fechas, entre otros. A estos diversos tipos de contenido se les denomina 'tipo de campo' o 'tipo de columna'. Al decir que el campo 1 es de tipo NUMERIC estamos diciendo que ese campo sólo puede almacenar números. A continuación se brinda un listado con los principales tipo de campo.

Nombre del tipo	Descripción
NUMERIC	Almacena números
CHAR	Cadena de caracteres de longitud fija
VARCHAR	Cadena de caracteres de longitud variable, es necesario especificar la cantidad de dígitos que almacenará, por ejemplo: un campo varchar(500) podrá almacenar hasta 500 caracteres.
BLOB	Binari Large Object. Es capaz de almacenar imágenes.
YEAR	Valor de año en formato AAAA, por ejemplo: 2008
DATE	Valor de fecha en formato AAAA-MM-DD, ejemplo: 2008-12-20
TIME	Valor de hora en formato de hh:mm:ss, ejemplo: 23:58:09
DATETIME	Valor de fecha y hora en formato AAAA-MM-DD hh:mm:ss, ejemplo: 2008-12-20 23:58:09

Figura Nº 6
Tipos de contenido de campo

Notemos también que algunos de los campos en el código de ejemplo contienen la instrucción NOT NULL. La función de esta instrucción es indicarle al SGBD que ese campo siempre debe contener información, es decir, que al ingresar un registro o tupla a la tabla todos los campos que hayan sido definidos como *not null* deben poseer datos, de lo contrario no se permitirá ingresar el registro. Si deseamos que algún campo tenga 'habilitada' la opción de no contener datos no debemos escribir la sentencia not null.

Otro punto importante a tratar es la instrucción AUTO_INCREMENT. Esta instrucción sólo se puede aplicar a campos de tipo numérico (NUMERIC), ya que su función consiste en incrementar en uno el contenido del campo anterior, es decir, si el campo autonumérico del último registro de la tabla

posee el número 99, al insertar un nuevo registro, el SGBD otorgará de forma automática al campo el número 100. Esta instrucción es muy usada en campos de identificación, por ejemplo en bibliotecas para diferenciar usuarios.

Para finalizar la explicación del código anterior explicaremos el concepto de *clave primaria* (primary key). Como se explicó en el primer capítulo, es necesario que una base de datos no posea registros duplicados (que sean exactamente iguales), es por ello que al menos un campo de cada tabla debe ser declarado como único, así, al insertarse un nuevo registro, el SGBD buscará en el o los campos declarados como primary key, si ya existe el dato que se desea insertar. En caso de existir, se arrojará un mensaje diciendo que no se puede insertar el nuevo registro porque se produciría una 'violación de restricción'.

Después de esta breve explicación del uso del comando create es el turno de llevarlo a la práctica. Para ello emplearemos MySQL como SGBD (si no sabe como instalar MySQL puede remitirse al Anexo B del presente trabajo).

El primer paso consiste en crear la base de datos, para ello escribimos el siguiente código:

```
CREATE DATABASE SISTEMA_BIBLIOTECA;
```

**El código puede ser escrito en mayúscula o minúscula.*

Lo cual se puede leer como: "Crear una base de datos llamada Sistema_Biblioteca"

Si todo ha salido bien nuestra base de datos ha sido creada, y deberá aparecer el siguiente mensaje:

"Query OK, 1 row affected"

En MySQL se debe seleccionar la base de datos en la cual se desea trabajar, ya que de otro modo no se sabrá en cual de ellas buscar, eliminar, modificar o actualizar las tablas y los datos. Para ello se debe utilizar el comando 'USE'.

En este caso debemos escribir lo siguiente:

```
USE SISTEMA_BIBLIOTECA
```

En estos momentos MySQL sabe que queremos trabajar en esa base de datos y no en otra. Deberá aparecer el siguiente mensaje en pantalla:

"Database changed"

Con la base de datos creada y seleccionada para trabajar en ella estamos en condiciones de poder crear, actualizar y eliminar tablas. Lo primero que veremos será la creación de tablas, para ello nos basaremos en la tabla 'libro' del modelo en cuestión.


```
CREATE TABLE LIBRO
(
id_libro int(9) not null auto_increment,
titulo varchar(75) not null,
autor varchar(100) not null,
ISBN int(13) not null,
materia varchar(50),
editorial varchar(75),
paginas int(4),
primary key(id_libro)
);
```

Este código se puede interpretar como: “Crear una tabla llamada libro, la cual contiene 7 campos: id_libro (tipo numérico de hasta nueve dígitos, autoincrementable y siempre debe contener datos), titulo (campo de texto de hasta 75 caracteres y siempre debe contener datos), autor (campo de texto de hasta 100 caracteres y siempre debe contener datos), ISBN (tipo numérico de hasta 13 dígitos y siempre debe contener datos), materia (campo de texto de hasta 50 caracteres y permite registros sin datos), editorial (campo de texto de hasta 75 caracteres y permite registros sin datos), paginas (tipo numérico de hasta 4 dígitos y permite registros sin datos). La clave primaria será id_libro”

MySQL internamente interpretará el código y creará la tabla. En pantalla aparecerá el siguiente mensaje:

“Query OK, 0 rows affected”

En la figura 7 se puede ver la creación de una tabla en MySQL.



```
C:\Program Files\MySQL\MySQL Server 6.0\bin>mysql.exe
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 6.0.3-alpha-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database sistema_biblioteca;
Query OK, 1 row affected (0.01 sec)

mysql> use sistema_biblioteca
Database changed
mysql> CREATE TABLE LIBRO
-> <
-> id_libro int(9) not null auto_increment,
-> titulo varchar(75) not null,
-> autor varchar(100) not null,
-> ISBN int(13) not null,
-> materia varchar(50),
-> editorial varchar(75),
-> paginas int(4),
-> primary key(id_libro)
-> ;
Query OK, 0 rows affected (0.07 sec)

mysql> _
```

Figura N°7

Creación base de datos y tabla en MySQL

3.3.2 Alter

Hay ocasiones, una vez que se ha creado la tabla en la base de datos, en que se desea cambiar la estructura de la tabla. Los casos más comunes son los siguientes:

- Agregar un campo (columna)
- Eliminar un campo.
- Cambiar el nombre de un campo.
- Cambiar el tipo de datos para un campo.
- Agregar o modificar la clave primaria.

Para realizar cualquiera de los puntos listados se emplea la instrucción ALTER.

```
alter table "nombre_tabla"  
[código de modificación]
```

[código modificación] depende del tipo de modificación que deseamos realizar. Para los usos mencionados anteriormente, las instrucciones [código modificación] son:

- Agregar un campo: ADD "campo 1" "tipo de datos campo 1"
- Eliminar un campo: DROP "campo 1"
- Cambiar el nombre de un campo: CHANGE "nombre antiguo del campo" "nuevo nombre del campo" "tipo de datos para el nuevo campo"
- Cambiar el tipo de datos para un campo: MODIFY "campo 1" "nuevo tipo de datos"

Ahora algunos ejemplos del uso de ALTER TABLE empleando la tabla *libro*.

```
ALTER TABLE LIBRO  
ADD notas varchar(200);
```

Lo cual se puede leer como: “Modificar la tabla libro agregando el campo llamado notas de tipo texto con un máximo de 200 caracteres”

```
ALTER TABLE LIBRO  
CHANGE notas notitas varchar(100);
```

Lo cual se puede leer como: “Modificar la tabla libro cambiando el nombre del campo notas por el nombre notitas de tipo numérico de hasta 100 caracteres”

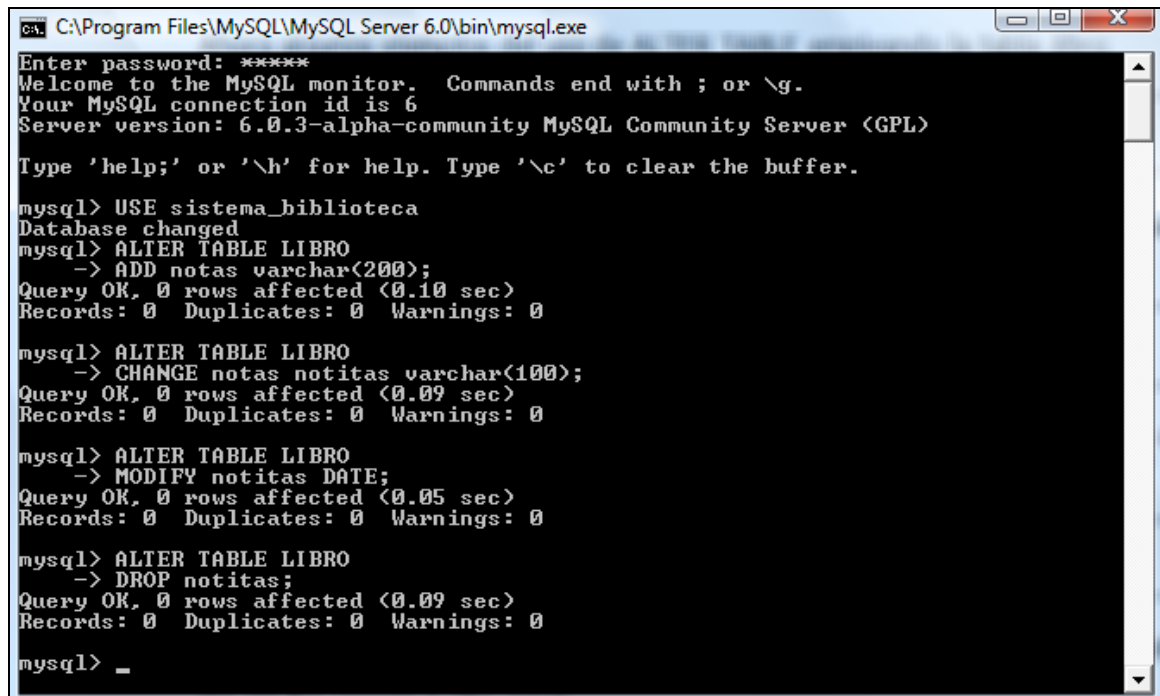
```
ALTER TABLE LIBRO  
MODIFY notitas DATE;
```

Lo cual se puede leer como: “Modificar la tabla libro cambiando el tipo de datos del campo notitas a tipo fecha”

```
ALTER TABLE LIBRO  
DROP notitas;
```

Lo cual se puede leer como: “Modificar la tabla libro eliminando el campo notitas”

Es importante señalar que cuando queremos modificar el tipo de datos de algún campo en específico la tabla debe estar vacía, ya que de lo contrario el SGBD no nos dejará realizar la operación.



```
C:\Program Files\MySQL\MySQL Server 6.0\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 6.0.3-alpha-community MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE sistema_biblioteca
Database changed
mysql> ALTER TABLE LIBRO
-> ADD notas varchar(200);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE LIBRO
-> CHANGE notas notitas varchar(100);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE LIBRO
-> MODIFY notitas DATE;
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE LIBRO
-> DROP notitas;
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> _
```

Figura Nº 8
Uso del comando ALTER

3.3.3 Drop

A veces es necesario eliminar una tabla, ya sea porque nos equivocamos al crearla, o porque la información que contiene ya no es útil. Independiente de la razón, los SGBD ofrecen la opción de eliminar datos a través de una simple instrucción SQL:

```
drop table nombre_tabla;
```

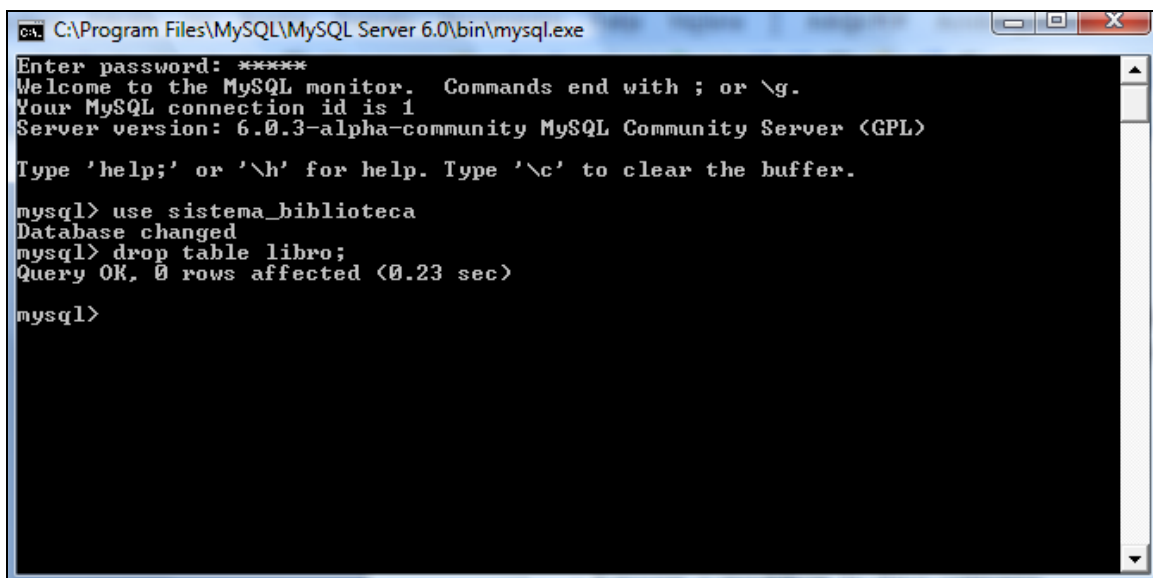
Es importante señalar que hay que estar muy seguro de querer eliminar una tabla, ya que no sólo se elimina la estructura de la tabla, sino que también se eliminan los datos que ella contiene. Se debe tener claro el

impacto que puede provocar en la base de datos, ya que es muy probable que esa tabla esté relacionada con otra u otras tablas.

Siguiendo con los ejercicios anteriores, para eliminar la tabla libro deberíamos escribir la siguiente sentencia:

```
DROP TABLE LIBRO;
```

Lo cual se puede leer como: “Eliminar la tabla libro”



```
C:\Program Files\MySQL\MySQL Server 6.0\bin>mysql.exe
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 6.0.3-alpha-community MySQL Community Server <GPL>
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use sistema_biblioteca
Database changed
mysql> drop table libro;
Query OK, 0 rows affected (0.23 sec)

mysql>
```

Figura Nº 9
Uso del comando DROP

3.4 Lenguaje de Manipulación de Datos

El segundo sublenguaje de SQL es el Lenguaje de Manipulación de Datos, el cual es el encargado de que los SGBD puedan realizar las cuatro operaciones básicas, las cuales son insertar, actualizar (modificar), eliminar y buscar (consultar) registros. Siguiendo la metodología del punto anterior, veremos las instrucciones básicas y un ejemplo de ellas utilizando la tabla libro y el SGBD MySQL.

3.4.1 Insert

Esta instrucción se utiliza para insertar registros en la base de datos. Se utiliza una única sentencia que puede poseer algunas modificaciones dependiendo de los datos que queramos insertar. La sintaxis básica es:

```
insert into libro  
(campo1, campo2, campo3)  
values  
(dato1, dato2, dato3);
```

Es de suma importancia que el orden en que se inserten los datos sea el mismo que el orden en que se escribieron los nombres de los campos, es decir, si el campo autor lo escribimos en tercer lugar, el dato correspondiente a autor lo escribimos en la tercera posición.

En algunos manuales de SQL es posible encontrar la sentencia anterior de forma más abreviada (sin escribir el nombre de los campos), dejándola de la siguiente forma:

```
insert into libro values  
(dato1, dato2, dato3);
```

Esta forma de codificación es útil siempre y cuando se inserten datos en todos los campos que conforman la tabla, y lo que es más importante, se deben insertar en el mismo orden en que aparecen en la tabla, ya que de lo contrario el SGBD no sabrá a que campo insertar el dato. Es por esta razón que se recomienda utilizar la instrucción insert de la forma más completa.

Si deseamos insertar sólo algunos de los campos que conforman la tabla deberemos necesariamente escribir los nombres de los campos a insertar, y los datos deben ser insertados en el mismo orden en que se escribieron los campos.

Cundo una tabla tiene un campo como autoincrementable, éste no se debe escribir en la sentencia del insert, ya que el SGBD lo genera automáticamente.

A modo de ejemplo insertaremos algunos datos en la tabla libro.

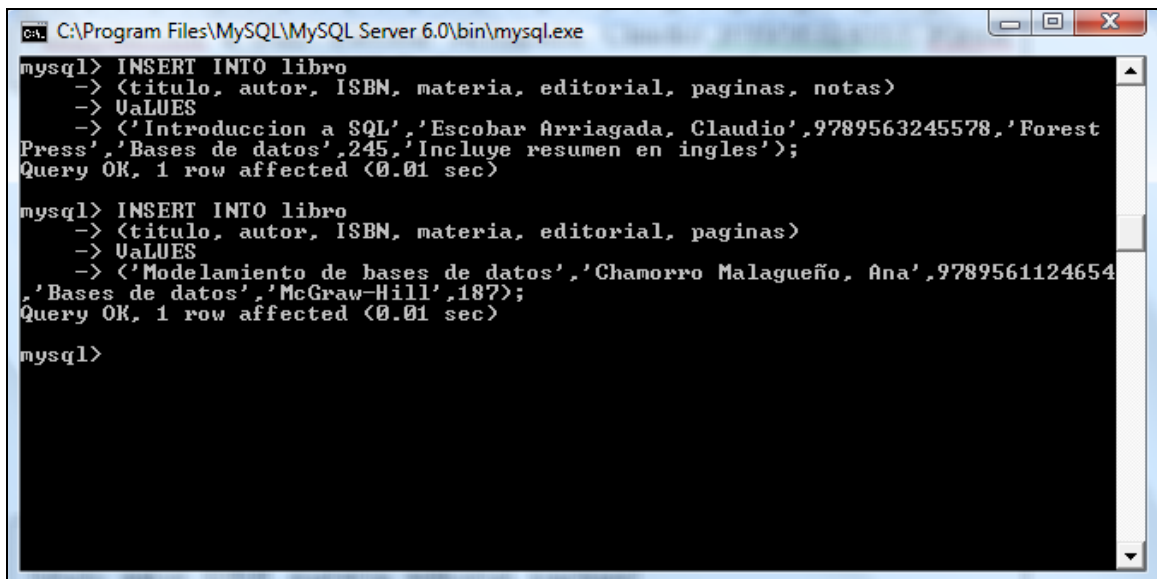
```
INSERT INTO libro
(titulo, autor, ISBN, materia, editorial, paginas, notas)
VALUES
('Introduccion      a      SQL','Escobar      Arriagada,
Claudio','978956324557','Forest Press','Bases de datos',245,'Incluye
resumen en ingles');
```

Se puede leer como: “Insertar en la tabla libro los campos titulo, autor, ISBN, materia, paginas y notas con los contenidos Introducción a SQL, Escobar Arriagada, Claudio, 9789563245578, Forest Press, Bases de datos, 245, Incluye resumen en inglés”.

Note que los campos que fueron declarados como varchar (campos de texto) se insertan entre comilla simples ("). Antes de insertar fechas es necesario leer la documentación del SGBD a utilizar, ya que suele variar entre uno y otro, por ejemplo en MySQL se inserta empleando el formato AAAA-MM-DD (año – mes - día), por ejemplo: date(2008-12-24). Mientras que en ORACLE se inserta de la siguiente manera: to_date('dd/mm/yyyy','24/12/2008').

```
INSERT INTO libro
(titulo, autor, ISBN, materia, editorial, paginas)
VaLUES
('Modelamiento de bases de datos','Chamorro Malagueño,
Ana',9789561124654,'Bases de datos','McGraw-Hill',187);
```

Muy similar al ejemplo anterior, esta vez no se ingresa el campo nota, el cual no fue declarado como not null.



```
C:\Program Files\MySQL\MySQL Server 6.0\bin>mysql.exe
mysql> INSERT INTO libro
-> <titulo, autor, ISBN, materia, editorial, paginas, notas>
-> VaLUES
-> ('Introduccion a SQL','Escobar Arriagada, Claudio',9789563245578,'Forest
Press','Bases de datos',245,'Incluye resumen en ingles');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO libro
-> <titulo, autor, ISBN, materia, editorial, paginas>
-> VaLUES
-> ('Modelamiento de bases de datos','Chamorro Malagueño, Ana',9789561124654
,'Bases de datos','McGraw-Hill',187);
Query OK, 1 row affected (0.01 sec)

mysql>
```

Figura Nº10
Uso del comando INSERT

3.4.2 Update

La mayoría de las bases de datos son dinámicas, es decir, actualizan sus datos. Tal es el ejemplo de las bases de datos de un banco en la que el saldo de los clientes cambia prácticamente todos los días. En las bibliotecas se pueden actualizar las fechas de devolución de los préstamos. Para realizar la actualización de datos recurrimos al comando UPDATE.

La sintaxis es bastante simple:

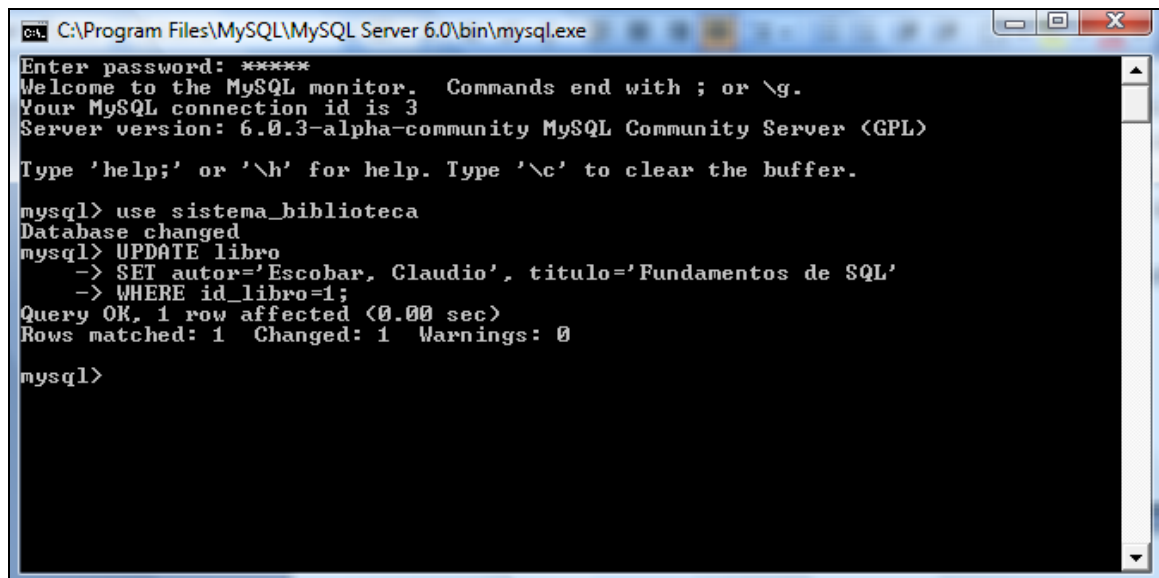
```
UPDATE nombre_tabla  
SET campo2='dato2', campo3='dato3'  
WHERE campo1='dato1';
```

En una sola sentencia se pueden actualizar uno o más campos. La sentencia UPDATE va seguida del nombre de la tabla en la que se actualizará el o los campos. Luego se escribe el nombre del o los campos con los nuevos valores que contendrán. Finalmente se escribe la sentencia where, la cual delimitará los registros a actualizar, si no escribimos ninguna sentencia where se actualizarán todos los registros. El uso de la sentencia where es muy importante, ya que delimita los campos a actualizar, por lo general deseamos actualizar un registro en específico, por lo que se utiliza la clave primaria como referencia. La sentencia where puede estar conformada por más de un campo.

Como ejemplo actualizaremos uno de los registros insertados anteriormente.

```
UPDATE libro  
SET autor='Escobar, Claudio', titulo='Fundamentos de SQL'  
WHERE id_libro=1;
```

Lo cual se puede leer como: “Actualice la tabla libro modificando los valores de autor y titulo con los siguientes valores ‘Escobar, Claudio’ y ‘Fundamentos de SQL’ donde el campo id_libro sea igual a 1”



```
CA: C:\Program Files\MySQL\MySQL Server 6.0\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 6.0.3-alpha-community MySQL Community Server <GPL>

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use sistema_biblioteca
Database changed
mysql> UPDATE libro
-> SET autor='Escobar, Claudio', titulo='Fundamentos de SQL'
-> WHERE id_libro=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

Figura Nº11
Uso del comando UPDATE

3.4.3 Delete

Para eliminar uno o más registros se emplea la instrucción DELETE. A diferencia de drop, que elimina la estructura de la tabla y los datos que contiene, esta sentencia sólo elimina los datos. La sentencia es muy simple:

```
delete from nombre_tabla
where campo1=dato1;
```

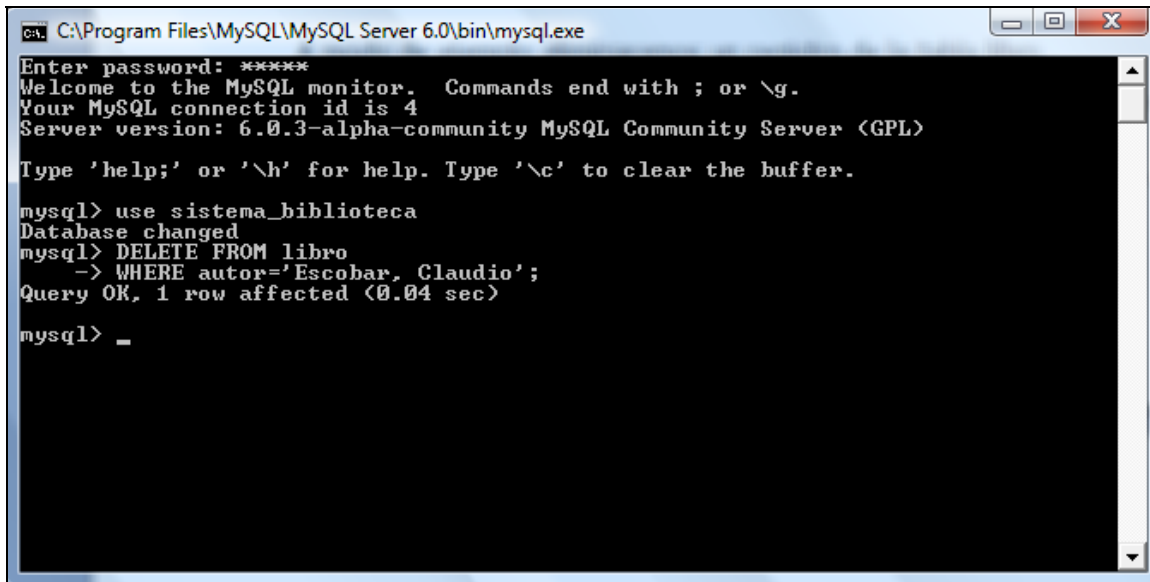
Se utiliza la sentencia DELETE seguido de otra sentencia SQL, la sentencia FROM y el nombre de la tabla, indicándole al SGBD de qué tabla tiene que eliminar el registro. La sentencia where tiene la misma importancia y función que en la sentencia UPDATE.

A modo de ejemplo eliminaremos un registro de la tabla libro.

```
DELETE FROM libro
WHERE autor='Escobar, Claudio';
```

Lo cual se puede leer como: "Eliminar el o los registros de la tabla libro donde autor sea igual a Escobar, Claudio".

Note que esta sentencia borrará todos los registros donde el campo autor sea igual a Escobar, Claudio. Si lo que usted desea es eliminar un registro en específico debe ser más especial en el where.



```
ca: C:\Program Files\MySQL\MySQL Server 6.0\bin\mysql.exe
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 6.0.3-alpha-community MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use sistema_biblioteca
Database changed
mysql> DELETE FROM libro
      -> WHERE autor='Escobar, Claudio';
Query OK, 1 row affected (0.04 sec)

mysql> _
```

Figura Nº 12
Uso del comando DELETE

3.4.4 Select

Esta instrucción es para muchos una de las más utilizadas en SQL, ya que nos permite realizar búsquedas de todo tipo. Como bibliotecarios estamos acostumbrados a realizar búsquedas de información, por lo que a diario empleamos el comando SELECT, aunque muchas veces no estamos al tanto de ello.

Para realizar una búsqueda se emplea por obligación los comandos SELECT y FROM, siendo alternativo el uso de WHERE y otros comandos menos utilizados.

La sentencia básica es:

```
select nombre_campos_a_buscar  
from nombre_tablas;
```

Un select puede hacer búsquedas en una o más tablas de forma paralela, para ello solo basta escribir el nombre de las tablas separadas por coma (,). Si empleamos la sentencia básica nos arrojará los campos escritos en la sentencia de todos los registros que contiene la o las tablas a consultar. Si deseamos refinar la búsqueda debemos utilizar el comando where.

```
select nombre_campos_a_buscar  
from nombre_tablas  
where campo1=dato1;
```

Un ejemplo utilizando la tabla libro

```
SELECT titulo, autor, materia  
FROM libro  
WHERE autor='Chamorro Malagueño, Ana';
```

Lo cual se puede leer como: “Seleccionar los campos titulo, autor y materia de la tabla libro donde el campo autor sea igual a Chamorro Malagueño, Ana”

El resultado de la consulta arrojará todos los registros que cumplan con la condición. Si deseamos que la consulta arroje un único registro se debe usar el where con la clave primaria.

```

C:\Program Files\MySQL\MySQL Server 6.0\bin\mysql.exe
Server version: 6.0.3-alpha-community MySQL Community Server <GPL>
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> use sistema_biblioteca
Database changed
mysql> SELECT titulo, autor, materia
  -> FROM libro
  -> WHERE autor='Chamorro Malagueño, Ana';
+-----+-----+-----+
| titulo                | autor                | materia                |
+-----+-----+-----+
| Modelamiento de bases de datos | Chamorro Malagueño, Ana | Bases de dato          |
| Modelo entidad relacion y relacional | Chamorro Malagueño, Ana | Bases de dato          |
| Introduccion a PHP      | Chamorro Malagueño, Ana | Programacion           |
+-----+-----+-----+
3 rows in set (0.00 sec)
mysql>
    
```

Figura Nº 13
Uso del comando SELECT

CONCLUSIONES

Para finalizar el presente artículo podemos concluir que:

- ✓ Es muy importante estar atentos frente a las nuevas tecnologías que pueden aportar y optimizar el servicio que podemos ofrecer a los usuarios de nuestras bibliotecas.
- ✓ Existen SGBD gratuitos, como MySQL, que nos pueden ser de ayuda para optimizar o crear nuevos servicios.
- ✓ El estar al tanto de las situaciones cotidianas que suceden en una biblioteca, nos permite tener una visión más amplia, que se acerca de mejor forma a estas realidades. Por tanto, nuestro aporte en el uso de sistemas de gestión de bases de datos resulta muy valioso. Por un lado es necesario trabajar en todo lo que es modelamiento de datos de la base, graficando objetos, relaciones y acciones, para luego abocarse a la programación, para la cual podemos trabajar en conjunto con profesionales del área informática.
- ✓ Los sistemas de gestión de bases de datos nos ayudan a automatizar procesos y aprovechar eficientemente nuestros recursos.
- ✓ El uso de lenguajes estandarizados facilita la interacción con las bases de datos y permite definir procesos.
- ✓ La utilización de SGBD favorece un rápido y eficiente acceso a la información, mejorando los procesos empleados en las bibliotecas.

BIBLIOGRAFÍA

DUBOIS, Paul. MySQL. Madrid, Prentice Hall, 2001. 789 p.

GROFF, James R. y WEINBERG, Paul. Aplique SQL. Madrid: McGraw-Hill, 1991. 619 p.

LUQUE Ruiz, Irene. Bases de datos: desde Chen hasta Codd con Oracle. México, Alfaomega, 2002. 422 p.

MÁRQUEZ, M. (2001). Historia de los sistemas de bases de datos [en línea]. [fecha de consulta: 30 octubre 2008]. Disponible en: < <http://www3.uji.es/~mmarques/f47/apun/node6.html> >

ROZIC, Sergio Ezequiel. Bases de datos y su aplicación con SQL. Buenos Aires : MP Eds., 2004. 249 p.

Tutorial de SQL. [en línea]. [fecha de consulta: 15 noviembre 2008]. Disponible en: < <http://sql.1keydata.com/es> >

ANEXO A

Formas normales

La normalización es un proceso que caracteriza y clasifica relaciones, objetos, formas de relación y demás elementos en grupos. Sobre los cuales se aplican ciertas reglas. Las formas normales son 5.

1. Primera forma normal

Una tabla está en Primera Forma Normal sólo si cumple los siguientes requisitos:

Todos los atributos son atómicos, ello quiere decir que no se pueden subdividir, son una unidad mínima.

La tabla no contiene atributos nulos

2. Segunda forma normal

Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave primaria. Es decir que no existen dependencias parciales hay una dependencia completamente funcional de la clave primaria.

3. Tercera forma normal

La tabla se encuentra en 3FN si es 2FN y cada atributo que no forma parte de ninguna clave, depende directamente y no transitivamente, de la clave primaria, es decir que dependa directamente de la clave primaria.

4. Cuarta forma normal

Para poner una relación o sub-relación en 4FN debe estar en 3FN y deben existir una o más multidependencias. Mejora las relaciones,

eliminando la redundancia, de forma que los valores no se repitan en distintos registros.

5. Quinta forma normal

Una tabla se encuentra en 5FN si la tabla esta en 4FN, y si no existen relaciones de dependencias no triviales que no siguen los criterios de las claves. Una tabla que se encuentra en la 4FN se dice que esta en la 5FN si, y sólo si, cada relación de dependencia se encuentra definida por las claves candidatas. Esto quiere decir que con el fin de evitar la redundancia se usan las claves primarias para unir o relacionar dos o más tablas.

ANEXO B

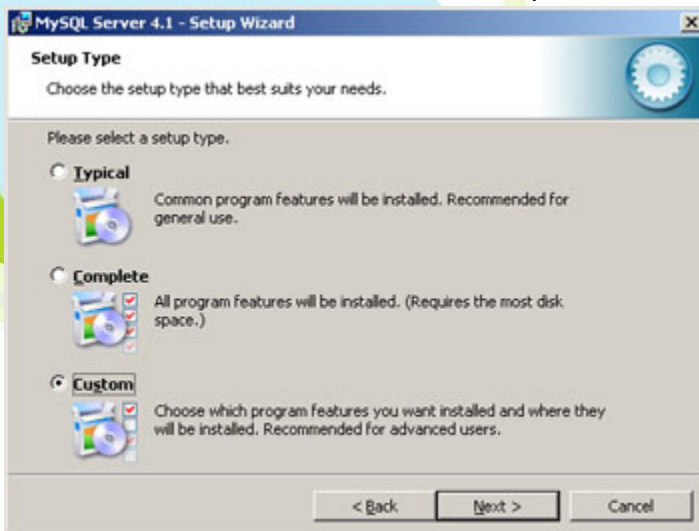
Cómo instalar MySQL Server en Windows

Este artículo muestra paso a paso cómo instalar MySQL Server (Base de Datos SQL gratuita y muy difundida por Internet):

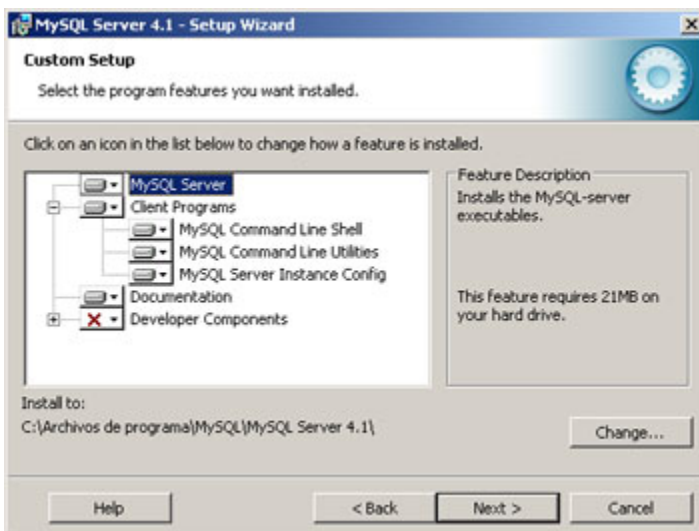
1. En primer lugar necesitaremos disponer del programa de instalación. Se puede descargar gratuitamente de "<http://dev.mysql.com/downloads>". Una vez descargado el programa de instalación de MySQL (versión que queramos, mientras escribíamos este artículo ha aparecido la versión 5.0, en nuestro caso instalaremos la "<http://dev.mysql.com/get/Downloads/MySQL-4.1/mysql-4.1.12-win32.zip/from/pick>" lo ejecutaremos y seguiremos las instrucciones que nos muestra el asistente de instalación:



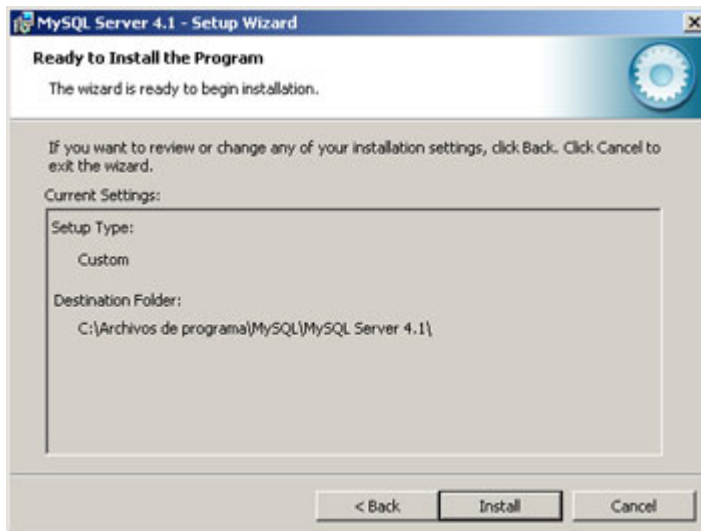
Pulsaremos en "Next" y marcaremos "Custom":



Seleccionamos las utilidades a instalar, por defecto se instalará todo salvo las herramientas para desarrolladores (sólo necesarias para desarrollos en Perl, C++ y MySQL Embedded Server):



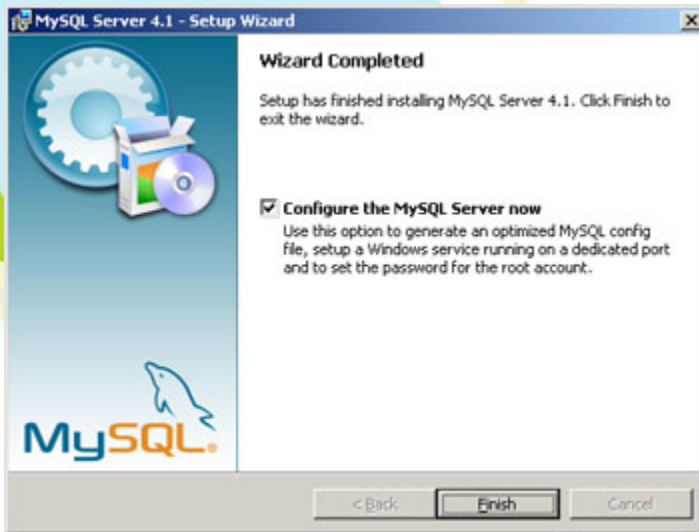
Pulsamos en "Next" y a continuación en "Install":



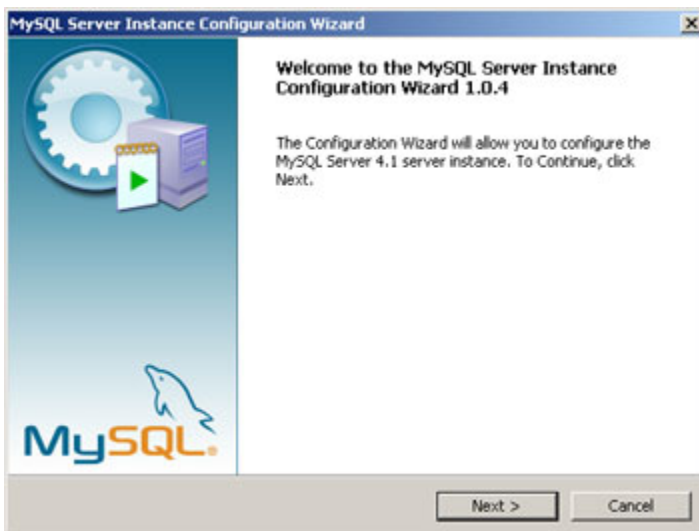
En la siguiente ventana podemos registrarnos en MySQL.com o si ya estamos registrados introducir email de registro y contraseña. También podemos cancelar el registro. En nuestro caso, puesto que ya estamos registrados marcaremos "Login to MySQL.com e introduciremos email y contraseña". Para registrarse desde aquí marcaremos en "Create a new free MySQL.com account" y iremos rellenando los datos que nos pide:



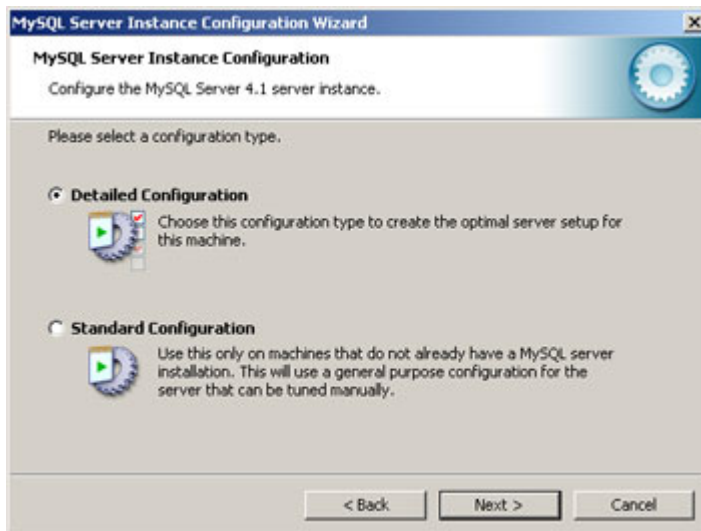
Si queremos configurar MySQL en este momento dejaremos marcada la opción "Configure the MySQL Server now" y pulsaremos en "Finish":



Ahora nos aparecerá un asistente para la configuración "MySQL Server Instance Configuration Wizard" y pulsaremos en "Next":



Marcaremos la opción "Detailed Configuration" y pulsaremos en "Next", de esta forma podremos configurar más opciones de MySQL utilizando el asistente. Si marcásemos "Standard Configuration" el asistente nos pediría menos información pero habría que configurar algunas opciones manualmente:



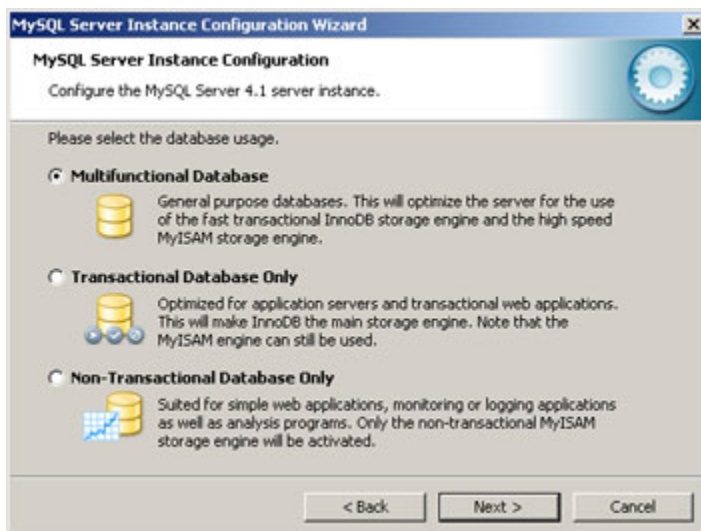
Dependiendo del uso que queramos dar al equipo en el que se instala marcaremos una de las tres opciones:

- **Developer Machine:** marcaremos esta opción si en el equipo donde hemos instalado MySQL Server se utiliza también para otras aplicaciones. MySQL Server utilizará la memoria mínima necesaria.
- **Server Machine:** marcaremos esta opción si vamos a utilizar el equipo para algunas aplicaciones (no demasiadas). Con esta opción MySQL Server utilizará un nivel medio de memoria.
- **Dedicated MySQL Server Machine:** marcaremos esta opción sólo si queremos utilizar el equipo como un servidor dedicado exclusivamente a MySQL. Con esta opción MySQL Server utilizará el máximo de memoria disponible. Se obtendrá un rendimiento elevado pero el equipo sólo servirá para MySQL.

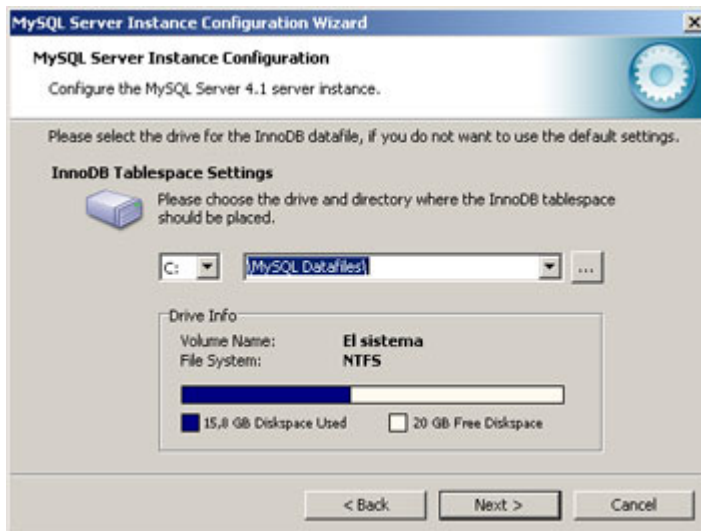
En nuestro caso marcaremos "Developer Machine" (consume el mínimo de memoria necesaria para su funcionamiento), este tipo de configuración de la instancia de MySQL no es recomendable si la base de datos va a soportar múltiples conexiones concurrentes con un volumen importante de información.



Dependiendo del uso que queramos dar a la Base de Datos marcaremos una de las tres opciones siguientes, normalmente se marcará "Multifunctional Database" salvo que queramos utilizar MySQL como base de datos para transacciones de otra Base de Datos MySQL:



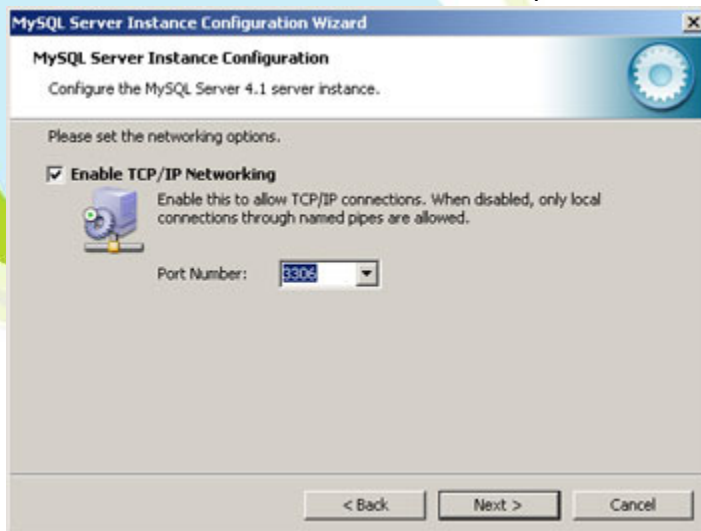
Seleccionaremos la unidad y la carpeta donde queramos guardar los ficheros de datos (Tablespace) de la Base de Datos. A partir de la versión 4.0, MySQL incorpora soporte para el control de la integridad referencial. A este nuevo tipo de tablas lo llama InnoDB:



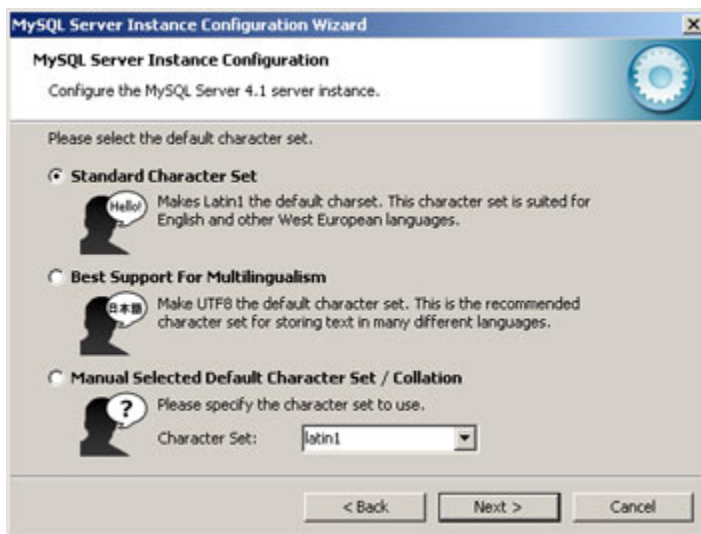
Seleccionaremos ahora el número aproximado de conexiones concurrentes (varios clientes conectados a la vez) que tendrá nuestro servidor de MySQL. La primera opción asume unas 20, la segunda unas 500 y la tercera permite especificarlas manualmente. Este parámetro es aproximado no tiene por qué ser exacto:



Dejaremos marcada la opción "Enable TCP/IP Networking" si queremos que los clientes se puedan conectar mediante TCP/IP al equipo servidor de MySQL. Podremos cambiar el puerto por el que lo harán, por defecto se suele dejar 3306 (si tenemos instalado algún cortafuegos deberemos abrir dicho puerto):



Seleccionaremos el juego de caracteres que queremos utilizar, por defecto está marcado "Latin1" válido para Inglaterra y Europa:



El siguiente paso es importante pues nos pide que especifiquemos el tipo de arranque de MySQL Server. Si seleccionamos la primera opción ("Install As Windows Service") el programa de instalación nos creará un Servicio que será el encargado de ejecutar MySQL Server, también nos permite especificar el nombre del servicio y si queremos que arranque automáticamente al iniciar el sistema ("Launch the MySQL Server automatically"). La segunda opción "Include Bin Directory in Windows

PATH" añadirá las variables de entorno necesarias para la ejecución de los ficheros necesarios para iniciar MySQL .

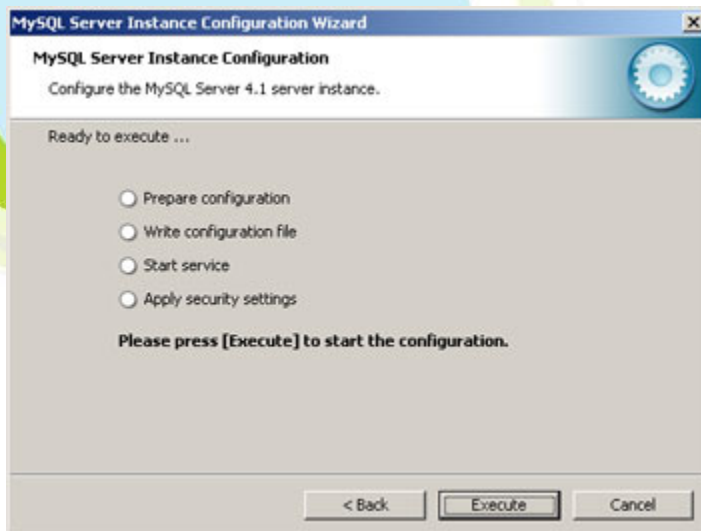
La opción recomendada es "Install As Windows Service":



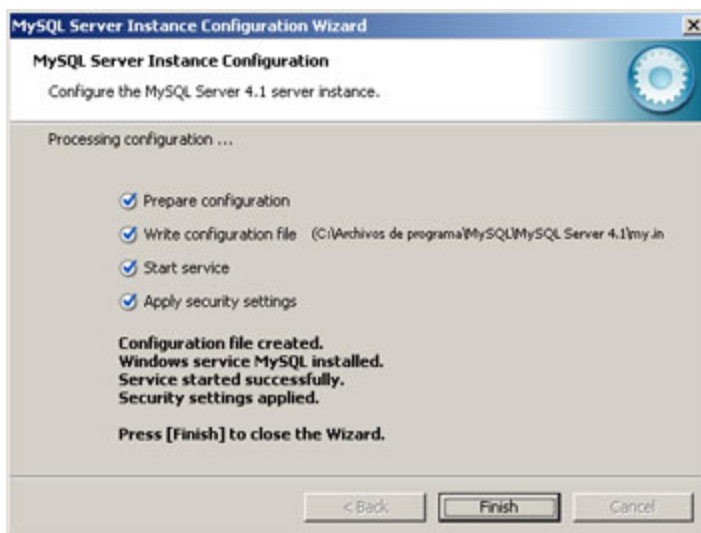
Introduciremos la contraseña para el usuario administrador (root) y marcaremos la opción "Enable root access from remote machines" si queremos que se pueda acceder como administrador desde otros equipos:



Por último pulsaremos en "Execute" para finalizar la configuración de MySQL:



Si no hay problemas mostrará esta ventana indicando que el proceso de instalación y configuración de MySQL Server ha terminado y se ha instalado e iniciado el Servicio que ejecutará MySQL:



Tras la instalación podemos comprobar (si hemos seleccionado la opción de iniciar MySQL como servicio) que el servicio se está ejecutando. Esto se puede ver en el administrador de tareas:



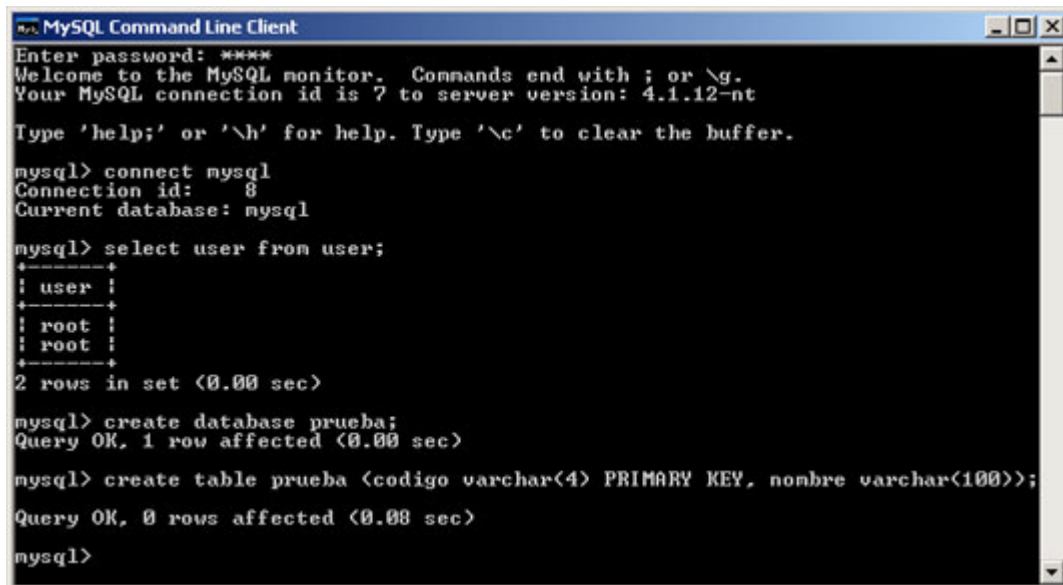
Allí nos aparecerá un servicio con el nombre "mysqld-nt.exe" que, como se puede observar, usa unas 12 MB de memoria RAM (sin conexiones de clientes).

Configuración alternativa de MySQL

Si lo deseamos podemos volver a configurar la instancia de MySQL desde "Inicio" - "Programas" - "MySQL" - "MySQL Server 4.1" - "MySQL Server Instance Config Wizard". El asistente que aparecerá será similar al explicado en el programa de instalación.

También podremos configurar mediante la línea de comandos MySQL, para ello iremos a "Inicio" - "Programas" - "MySQL" - "MySQL Server 4.1" - "MySQL Command Line Client". Nos pedirá una contraseña (la que hayamos introducido en la instalación):

Verificación de la correcta instalación de MySQL



Como ejemplo para comprobar que la instalación ha sido correcta nos hemos conectado a la base de datos que MySQL crea automáticamente llamada "mysql", la cual contiene los usuarios y configuración de MySQL, mediante:

```
connect mysql
```

Hemos ejecutado una consulta sobre la tabla "user" para mostrar el nombre de los usuarios de la BD:

```
select user from user;
```

Hemos creado una nueva base de datos llamada "prueba":

```
create database prueba
```

Nos hemos conectado a la BD "prueba":

```
connect prueba
```

Hemos creado una tabla en dicha BD llamada "prueba" con dos campos:

- código: de tipo texto, tamaño 4 y que será clave primaria de la tabla.
- nombre: de tipo texto, tamaño 100.

```
create table prueba (codigo varchar(4) PRIMARY KEY, nombre varchar(100));
```

Hemos ejecutado un Select sobre dicha tabla (aunque no tiene registros):

```
select * from prueba;
```

Por supuesto existen herramientas gráficas para administrar MySQL gratuitas, también disponibles en "<http://dev.mysql.com/downloads>", como por ejemplo: MySQL Administrator.

Serie Bibliotecología y Gestión de Información.

Títulos publicados 2008

Nº 34 Partidos políticos en Chile: Una mirada desde la Gestión de la Información. Georgina Caniffrú.

Nº 35 Leer y escribir en la web social: uso de blogs, wikis y multimedia compartida en educación. Cristian Cabezas Mardones.

Nº 36 Lecturas sobre la profesión de bibliotecario, las bibliotecas, los libros y la lectura. Mariela Ferrada.

Nº 37 Web, bibliotecas y fomento de la lectura. Enrique Ramos Curd.

Nº 38 Pautas para un trabajo de investigación en Bibliotecología. Héctor Gómez Fuentes y Beatriz Mercado Martinic.

Nº 39 Usuarios de bibliotecas con discapacidad psiquiátrica. Mariela Ferrada Cubillos.

Nº 40 La producción bibliográfica chilena de impacto mundial y regional: un análisis de las revistas nacionales en Web o Science. Leonardo Reyes R.

Nº 41 Guía para la preparación de resúmenes. María Texia Iglesias Maturana.

Disponible en : <http://eprints.rclis.org>

NORMAS DE PUBLICACION

• Objetivos

La **Serie Bibliotecología y Gestión de Información** tiene por objetivo difundir la productividad, académica, las investigaciones y las experiencias de profesionales del área de la de Bibliotecología y Ciencia de la Información y del sector afin al mundo del libro y la lectura.

• Alcance y política editorial

Los trabajos a ser considerados en la Serie Bibliotecología y Gestión de Información, deben ser inéditos, no publicados en otras revistas o libros. Excepcionalmente el Comité Editorial podrá aceptar artículos que no cumplan con este requisito.

- **Arbitraje:** Los artículos recibidos serán sometidos a evaluación, a recomendación del Director de la Serie, donde el Comité Editorial enviará los trabajos a árbitros independientes para su aceptación o rechazo. En este último caso, se emitirá un informe al autor/a donde se señalen las razones de la decisión. El Comité Editorial podrá solicitar trabajos a autores de reconocido prestigio, quienes no serán sometidos al proceso de evaluación por árbitros.

• Forma y preparación de manuscritos

- **Extensión:** El artículo deberá tener una extensión entre 12 y 100 páginas, tamaño carta, espacio 1,5, cuerpo 12, incluidos gráficos, cuadros, diagramas, notas y referencias bibliográficas.

- **Idiomas:** Se aceptan trabajos en castellano, portugués e inglés, los cuales serán publicados en su idioma original.

- **Resumen y palabras claves:** El trabajo deberá tener un resumen en español e inglés en la primera página, de no más de 200 palabras, que sintetice sus propósitos y conclusiones más relevantes. De igual modo, deben incluirse tres palabras claves, que en lo posible no se encuentren en el título del trabajo, para efectos de indización bibliográfica.

- **Nota biográfica:** En la primera página, en nota al pie de página, deben consignarse una breve reseña curricular de los/as autores/as, considerando nacionalidad, título y/o grados académicos, desempeño y/o afiliación profesional actual y sus direcciones de correo electrónico, para posibles comunicaciones de los/las lectores/as con los autores/as.

- **Referencia bibliográfica:** Utilizar para las referencias bibliográficas la modalidad de (Autor, año) en el texto, evitando su utilización a pie de página. Ejemplo: (González, 2006). Agregar al final del texto, la bibliografía completa. Sólo con los/las autores/as y obras citadas, numeradas y ordenadas alfabéticamente. Para el formato de la bibliografía, utilizar la "Guía para la presentación de referencias bibliográficas de publicaciones impresas y electrónicas" disponible en formato electrónico en : <http://eprints.rclis.org/archive/00005163/01/ReferenciasBibliograficas.pdf>

- **Derechos:** Los derechos sobre los trabajos publicados, serán cedidos por los/as autores/as a la **Serie**.

- **Investigadores jóvenes:** El Comité Editorial considerará positivamente el envío de trabajos por parte de profesionales y/o investigadores/as jóvenes, como una forma de incentivo y apoyo a quienes comienzan su carrera en investigación.

- **Ejemplares de cortesía:** Los/as autores/as recibirán un ejemplar de cortesía del trabajo publicado.

• Envío de manuscritos

Todas las colaboraciones deberán ser enviadas impresas en duplicado. Los autores/as podrán remitir sus artículos en CD, o al correo electrónico: hector.gomez@utem.cl , en programa Word (office).