UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

“ANÁLISIS Y DISEÑO DE UNA BASE DE DATOS DE PROCESAMIENTO, CONTROL Y ADMINISTRACIÓN DE INFORMACIÓN ESCOLAR”

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

L I Z E T H B E R E N I C E
G A R C Í A P É R E Z

ASESOR: MAT. LUIS RAMÍREZ FLORES

MÉXICO, 2011.

Dedicatorias y Agradecimientos

"A mis padres
Jorge García Benitez y
Teresa Pérez Narváez,

Por todo su apoyo, comprensión y cariño, pero sobre todo
por los valores inculcados que me han llevado a realizar esta meta,
en especial a ti madre, que has sido como un roble, eres una madre excepcional,
sin ti no hubiera podido realizar este sueño,
mil gracias, los amo mucho, que Dios los bendiga siempre.

"A mi esposo
Horacio Rivera García,

Por ser mí ejemplo a seguir, por apoyarme y darme ánimo,
pero sobre todo por ser un hombre maravilloso
y cuidar de mi más grande tesoro… Muchas gracias amor, te amo.

"A mis hermanos
Nallely Jazmín García Pérez
Jorge Martín García Pérez,

Por ser incondicionales conmigo cuando más los necesite,
y estar dispuestos a brindarme su ayuda sin reparo,
muchas gracias por todo,
sin ustedes no hubiese llevado a cabo este proyecto,
los quiero mucho.

"A mi hija
Ameyalli Rivera García,

Muy especialmente, por ser la mejor hija y un angelito caído del cielo,
por tu tolerancia y paciencia a lo largo de la realización de este proyecto,
y del cual eres responsable, por ser mi motivación e impulso de seguir
adelante y ser mejor persona, mil gracias hija por estar a mi lado, te amo.

"A mi asesor de tesis
Maestro Luis Ramírez Flores

Por todo el apoyo incondicional que me brindo,
y la confianza que tuvo en mi para llevar a cabo esta meta,
infinitamente, muchas gracias por brindarse como lo hace.

I N D I C E

CAPITULO II

CAPITULO III

1. INTRODUCCIÓN

1.1 Antecedentes

Actualmente las bases de datos se han convertido en una herramienta importante e indispensable para el mundo, gracias a su funcionamiento se realizan actividades que facilitan la vida, por ejemplo, retirar o ingresar dinero en un cajero automático, reservar un vuelo aéreo, suscribirse a algún servicio, consultar un catálogo de libros en una biblioteca, o simplemente buscar información en Internet, todo esto se lleva a cabo gracias a que pueden procesar, administrar y controlar grandes cantidades de información.

La idea de llevar a cabo la planeación y creación de una base de datos surge de la necesidad de satisfacer y dar solución a un determinado problema o situación detectada, en un lugar determinado llámese empresa, banco, escuela o línea aérea. Una vez identificada esta necesidad se lleva a cabo el análisis y el diseño de la misma, para posteriormente llegar a la implementación y cubrir satisfactoriamente el objetivo de su creación. Por lo tanto es de suma importancia diseñarla de manera adecuada, lo que permitirá a corto y largo plazo un rendimiento óptimo.

Las etapas que comprenden este diseño son de gran importancia y no se deben pasar por alto, ya que de ellas dependerá su buen funcionamiento. Cada una de ellas lleva consigo tareas que están relacionadas con la organización para la cual se creará la base de datos, con esto garantizamos que la base cubra todas las necesidades y objetivos de la misma. Todas estas actividades deben estar calendarizadas para iniciarlas y finalizarlas en el tiempo estimado y una vez identificada la necesidad se lleva a cabo el proceso de análisis, donde se realiza la investigación preliminar de la situación actual recopilando toda la información existente que será la base para el desarrollo del futuro sistema.

Todos estos elementos son necesarios para realizar el diseño lógico y físico de la base de datos, que en conjunto serán el cuerpo de la misma. La finalidad es tener los datos disponibles para los usuarios en el menor tiempo posible, eliminando o minimizando la redundancia, todo esto podrá llevarse a cabo si se cuenta con los elementos necesarios como son: el hardware, el software (SMBD), los datos a manejar y el personal encargado de manejar el sistema.

El diseño conceptual describe el contenido de la información, sin tomar en cuenta las estructuras de almacenamiento que se utilicen para manejarla. El diseño lógico parte de lo anterior, para generar las estructuras de datos que puede procesar un determinado SMBD (Sistema Manejador de Bases de Datos). El diseño físico describe las estructuras de almacenamiento y los métodos utilizados para tener acceso a los datos, por medio de un lenguaje de definición de datos (DDL). Todas estas partes del diseño darán como resultado una base de datos.

Una vez terminado el diseño se realizan pruebas para verificar que se satisfacen las necesidades de los usuarios finales, cargando datos ficticios en un primer momento, creando las consultas más solicitadas que se utilizarán en la base de datos y finalmente implementándola para ser usada.

1.2 Planteamiento del problema

El Colegio Particular "Los Ángeles", ubicado en la Colonia Centro del municipio de Huehuetoca Estado de México, ha crecido paulatinamente, aumentando su matrícula año con año en todos sus niveles educativos (Preescolar, Primaria, Secundaria, Preparatoria y Licenciatura), esto resulta favorecedor para el prestigio de la escuela y el Director General, pero también aumenta la carga de trabajo para el personal encargado del área de Control Escolar.

Actualmente el Colegio no cuenta con ningún sistema basado en computadora que puede administrar la información académica y personal de los alumnos de todos sus niveles, todo se almacena y manipula en archivos de Excel, manual e individualmente, es decir, cada nivel educativo la administra y procesa de la manera que mejor le conviene, dando como resultado la duplicidad de actividades y datos, ya que la mayoría de los alumnos egresa de un nivel para ingresar a otro haciendo un histórico en la escuela.

1.3 Objetivos

Objetivo general

Analizar y diseñar una base de datos que administre y procese la información personal y académica de los alumnos de cada nivel educativo que existe en el Colegio, a través de un Sistema Manejador de Bases de Datos que proporcione el almacenamiento, recuperación y manipulación de los datos a los usuarios del sistema.

Objetivos particulares:

- Obtener información personal y académica de cualquier alumno de la escuela.
- Realizar solo una captura de datos evitando repetir tareas.
- Realizar altas, bajas y cambios en los datos de los alumnos.
- Evitar la redundancia de datos.
- Editar plantillas, reportes o formatos que necesite cada uno de los distintos niveles.
- Tener organizada toda la información escolar.
- Tener almacenadas todas las calificaciones de los alumnos así como sus inasistencias.
- Obtener estadísticas de los datos personales de los alumnos como (edad, talla, peso, grado que cursa, etc.)
- Estimar promedios parciales, generales o anuales de los alumnos.
- Obtener estadísticas cuantitativas de cada una de las materias cursadas.

1.4 Justificación

La cantidad de información que generan cada uno de los alumnos es basta por lo que se propone la creación de una base de datos que mejore la administración y procesamiento de datos dentro de la institución, con la finalidad de reducir las tareas administrativas de las áreas de control escolar de cada nivel educativo, teniendo almacenada la información en un solo lugar.

1.5 Hipótesis

El análisis, diseño y creación de una base de datos que pueda administrar y procesar la información personal y académica de los alumnos de una Institución mejora y reduce en gran medida las tareas administrativas de las áreas encargadas y el tiempo de respuesta para los usuarios finales.

C A P I T U L O I

1. BASES DE DATOS

1.1 Definición

Una base de datos es el conjunto de varios archivos interrelacionados creados en un lenguaje de consulta estructurado (SQL), bajo un Sistema Manejador de Bases de Datos (SMBD) en el cual se formarán numerosas rutinas de software, que realizarán una tarea en específico (figura 1).

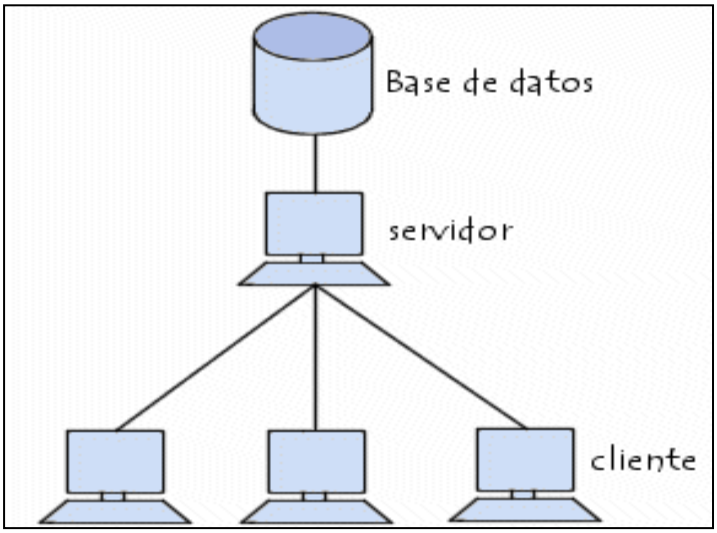


Fig. 1 Estructura de una Base de Datos

Una base de datos está compuesta de tres elementos importantes que son: el hardware, el software DBMS y los datos a manejar, así como el personal encargado del manejo del sistema (DBA). (M. Kroenke, 1996).

Todos estos elementos son importantes para el desarrollo del sistema, cada uno tiene una función determinante para su buen funcionamiento. El hardware la parte física que soporta la instalación y proporciona los recursos materiales necesarios para poder visualizar la base de datos. El software SMBD (Sistema Manejador de bases de datos) en él se crea la estructura interna y las rutinas de software que permiten extraer, almacenar y manipular la información, es decir genera las peticiones de acceso, además de ser la interface entre los usuarios y la base de datos. Los datos la parte más delicada e importante, sin ella no tendría finalidad su creación, y por último el administrador de la base de datos, que es o son las personas profesionales con experiencia en los conocimientos requeridos para su control y manejo.

El objetivo principal de un sistema de base de datos es brindarle a los usuarios una visión genérica de los datos, permitiendo que solo visualicen lo que ellos necesitan y no como están almacenados. Otro de sus principales motivos de creación es eliminar o minimizar en lo posible la redundancia e inconsistencia, ya que si hay duplicidad de datos se tiene un costo más alto de almacenamiento y acceso a ellos, el tenerlos en diferentes archivos provoca que al actualizarlos no se haga de manera correcta teniendo información en un lado y en otro. Se debe crear un entorno que permita la flexibilidad de los datos, en caso de surgir un imprevisto se pueda solucionar el problema, toda esta información debe ser supervisada y controlada para mantener un grado de protección e integridad que hagan confiable el sistema. (L. Gillenson, 2006).

1.2 Tipos de Bases de Datos

i. Base de datos jerárquica: son bases de datos que almacenan su información de forma jerárquica, es decir en forma de un árbol (pero al revés), en donde existe un *nodo raíz* del cual se desprenden los *nodos padres* que pueden o no tener hijos, en el caso de no tenerlos se les conoce como hojas. Debido a que son útiles para manejar grandes volúmenes de información y comparten datos no se puede eliminar totalmente la redundancia de datos.

ii. Base de datos de red: esta base de datos tiene una diferencia notable en cuanto a la jerárquica, sus nodos pueden tener varios padres, esto evita el problema de la redundancia de datos, pero su administración resulta difícil por lo que regularmente solo es manejado por programadores y no por usuarios finales.

iii. Bases de datos transaccionales: el objetivo de estas bases de datos es el envío y recepción de datos a una velocidad muy rápida, no son muy comunes, se utilizan por lo general para el análisis de calidad en el ámbito industrial, aquí la redundancia y duplicidad de datos no es problema, para su mejor aprovechamiento se permite que tengan conectividad con las bases relacionales.

iv. Bases de datos relacionales: modelo de bases de datos utilizado en la actualidad para aplicaciones reales y gestionar información, su base es el uso de relaciones, las cuales se transforman a tablas compuestas por registros (filas de la tabla) que representan tuplas, y campos (columnas de la tabla). La forma y acomodo de la información no tiene importancia, ya que se puede recuperar o almacenar mediante consultas que proporcionan una gran flexibilidad y administración de la información. El lenguaje utilizado por estas bases de datos es el SQL *Structured Query Language* o *Lenguaje Estructurado de Consultas*.

v. Bases de datos multidimensionales: esta base de datos tiene una mínima diferencia con las bases relacionales, la diferencia radica en el nivel conceptual, ya que los campos o atributos de la tabla pueden tomar dos valores, o bien representan dimensiones de la tabla o métricas para estudiar.

vi. Bases de datos orientadas a objetos: modelo de bases de datos más reciente el cual almacena los objetos completos (estado y comportamiento). Esta base contiene todos los conceptos del modelo orientado a objetos:

- Encapsulación: Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En las bases de datos orientadas a objetos, los usuarios definen operaciones sobre los datos. La operación (función) definida consta de dos partes, la interfaz que es el nombre de la operación y los tipos de datos que se van a manejar, y por último la implementación que puede realizarse de forma separada y sin afectar la interfaz. De este modo los programas de aplicación manipulan los datos solicitando que se ejecuten dichas operaciones sin importar como están implementadas.

vii. Bases de datos documentales: permiten la búsqueda de textos completos o líneas específicas teniendo una indexación más potente.

viii. Bases de datos deductivas: este sistema de base de datos permite realizar deducciones a través de conocimientos o datos previos que son almacenados, son también llamadas bases de datos lógicas siendo su base la lógica matemática. (Castaño & Piattini Velthuis, Fundamentos y modelos de bases de datos, 1999).

1.3 Aplicaciones de los sistemas de Bases de Datos

El uso de las bases de datos es amplio y se pueden aplicar a diferentes ámbitos de la sociedad, a continuación daremos algunos ejemplos de sus aplicaciones:

- Bancos: para administrar la información de los clientes, cuentas, préstamos, transacciones y en los cajeros para retiro de dinero.
- Universidades: para administrar la información de los estudiantes, las asignaturas, sus evaluaciones y los cursos que se imparten.
- Telecomunicaciones: para el registro de las llamadas telefónicas realizadas, así como la facturación de las mismas y para almacenar la información de las redes de comunicación.
- Finanzas: para almacenar la información de empresas grandes, como por ejemplo compras y ventas de documentos formales financieros (bolsa y bonos).
- Ventas: para el almacenamiento de los datos de clientes, productos y compras.
- Producción: para gestionar la información relacionada al proceso de producción y para el seguimiento del mismo. Inventarios de almacenes y de pedidos de elementos.
- Recursos Humanos: para almacenar la información de los empleados, como salarios, impuestos y remuneraciones que sirven para generar las nóminas. (Silberschatz, Korth, & Sudarshan, 2002)

Un sin número de aplicaciones se pueden enumerar, lo importante es destacar que a través de los años se ha interactuado con las bases de datos, tal vez sin darse cuenta, pero en todos lados están disponibles las bases de datos. El avance de las tecnologías exige nuevos SMBD que puedan resolver los futuros problemas, algunas de las áreas en donde se podrían aplicar las bases de datos, son:

- En la Ingeniería de Soporte Lógico Asistido por Computadora (en inglés: CASE *Computer Aided Software Engineering*): administrando toda la información relacionada con el desarrollo de sistemas informáticos.
- La Fabricación Integrada por Ordenador (en inglés: CIM *Computer Integrated Manufacturing*): administrando todas las etapas relacionadas en la operación de una planta de producción.
- Imágenes: para el reconocimiento de patrones, almacenamiento de millones de bits para el tratamiento de imágenes, etc.
- Datos espaciales: almacenando todo tipo de datos relacionados a la superficie terrestre, a su interior o el de la atmósfera y codificación de mapas, que son utilizados para investigaciones militares y ambientales. (Castaño & Piattini Velthuis, Concepción y Diseño de Bases de Datos: Del modelo E/R al modelo relacional, 1993).

2. ANALISIS DE UNA BASE DE DATOS

2.1 Ciclo de vida del desarrollo de una base de datos

El conjunto de pasos a seguir desde que se concibe una idea o proyecto hasta que finaliza o se cambia a otro, forman el ciclo de vida de una base de datos, en general las fases que comprenden este ciclo son:

a. Estudio previo y plan de trabajo
b. Recolección y análisis de requisitos
c. Diseño conceptual de la base de datos
d. Elección del SMBD y diseño lógico de la base de datos
e. Diseño físico e implementación del sistema de base de datos

Por tanto el ciclo de vida del desarrollo de una base de datos se puede ver en el siguiente diagrama (figura 2).

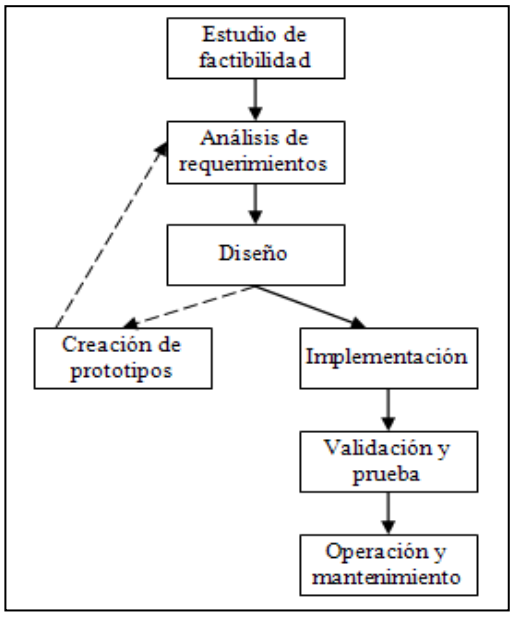


Figura 2. Ciclo de vida del desarrollo de una base de datos.

En la primera etapa se determina el propósito de la base de datos, es aquí donde se lleva a cabo el análisis de los datos que se almacenarán, la necesidad que se tiene que satisfacer, los objetivos de su creación, que tan factible de informatizar puede ser, como se puede satisfacer (recursos materiales), que problemas o riesgos se pueden encontrar en el camino, que beneficios se obtendrán de ella, así como el tiempo que requiere cada una las tareas a realizar a lo largo del desarrollo del sistema.

El diseño conceptual es una descripción del contenido de la información que será utilizada para el diseño de la base de datos. A partir de esto se pueden determinar las entidades, relaciones y atributos, la información que debe ser almacenada, las restricciones de integridad, etc., para posteriormente representarlo gráficamente en un diagrama Entidad Relación independiente del SMBD. En este diseño solo se describirá la información relacionada con la base de datos, no las estructuras de almacenamiento que se necesitan para manejarla.

Una vez realizado el diseño conceptual de la base de datos se hace la elección del SMBD en el cual se tienen que analizar factores técnicos y económicos de acuerdo a las capacidades de la organización que la ha solicitado, así mismo el diseño lógico dependerá del modelo de datos que soporte el SMBD, dando como resultado un conjunto de sentencias escritas en un Lenguaje de Definición de Datos (LDD) soportado por el mismo manejador.

El diseño lógico comprende la descripción de la estructura de la base de datos, bajo los términos que cada tipo de SMBD puede procesar. Este diseño depende del tipo de manejador que se utilice y no depende del producto concreto.

El diseño físico define las estructuras internas de almacenamiento y los medios para tener acceso a los datos, este diseño si depende del SMBD y se expresará mediante su propio lenguaje de definición de datos. Por último se realiza la implementación, en donde se adquieren e integran los componentes y recursos necesarios para que el sistema funcione.

2.2 Diagrama de Gantt

El diagrama de Gantt es una herramienta que permite calendarizar las tareas que se realizan para el desarrollo de un proyecto. A través de ella se puede hacer una representación gráfica del avance del proyecto, haciendo fácil su interpretación para los involucrados en el desarrollo del mismo, y ver si es factible su creación. Una herramienta que se utiliza para realizarlo es Microsoft Project, además de otras que son libres y gratuitas.

Para su creación es necesario definir las tareas o actividades a realizar, el tiempo que cada una requiere para el desarrollo del proyecto, determinando así su inicio y su final. Las tareas se pueden realizar secuencial o simultáneamente, esto no importa ya que se pueden graficar de igual manera. Una vez que se determinan estos elementos, empiezan a graficarse todas las actividades, formando una secuencia de barras horizontales que definen el periodo de tiempo para cada una de ellas y para el desarrollo total del proyecto (figura 3).

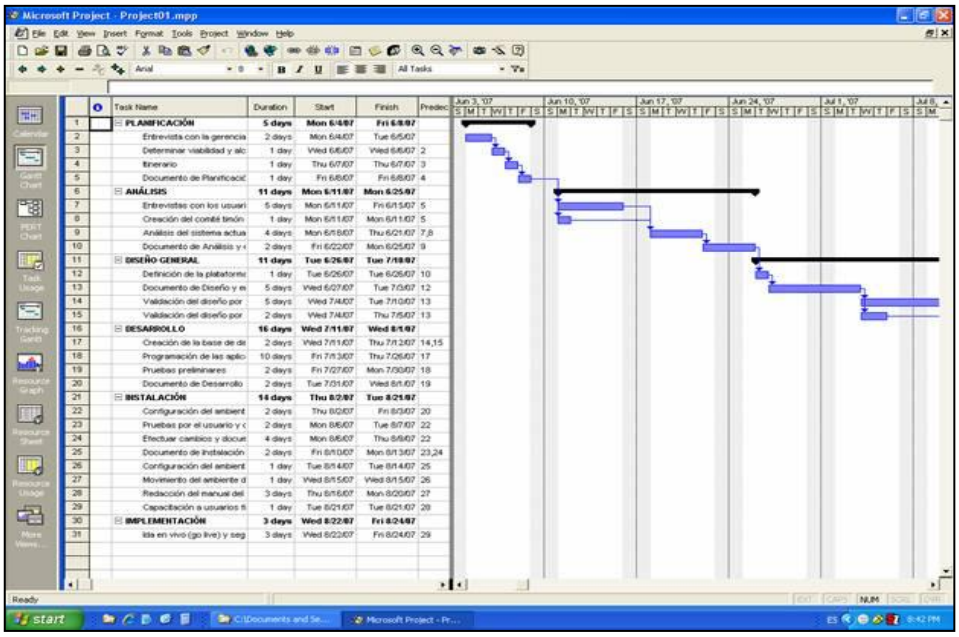


Fig. 3 Ejemplo de Diagrama de Gantt de un proyecto (tomada de Robles, abril 2007).

3. DISEÑO DE UNA BASE DE DATOS

El diseño de una base de datos abarca diferentes etapas, cada etapa se pueden trabajar de mejor manera si se descompone en otra menor, haciendo su resolución más fácil y rápida, utilizando técnicas específicas. De esta manera el diseño de una base de datos se compone de tres fases diseño conceptual, diseño lógico y diseño físico (Marqués Andrés, 2001).

3.1 Diseño Conceptual

El diseño conceptual nos brinda una idea más clara y real del problema, previo a la formulación de esta idea se hace una recopilación de todos los requerimientos que necesita la base de datos, para iniciar su creación. Su descripción mostrará la semántica de los datos la cual es independiente del SMBD que se vaya a utilizar. El objetivo es entender lo que los usuarios perciben de los datos, la naturaleza de los mismos y su uso a través de la aplicación. Esta descripción se puede hacer mediante un esquema conceptual que utilice la notación más común, estamos hablando del modelo entidad-relación (MER). Este esquema será la fuente de información para el diseño lógico de la base de datos (Marqués Andrés, 2001).

3.1.1 Modelo Entidad-Relación (MER)

El modelo entidad-relación establece una percepción del mundo real, mediante objetos llamados entidades y relaciones. Su objetivo es facilitar el diseño de bases de datos permitiendo realizar un esquema de la empresa que representa la estructura lógica, es decir nos permite comparar los significados e interacciones del mundo real con un esquema conceptual (Silberschatz, Korth, & Sudarshan, 2002).

Los elementos importantes del modelo entidad-relación son: Entidades, Atributos y Relaciones.

Entidad: es aquel objeto (real o abstracto) del cual se quiere obtener información, este objeto puede ser una persona, lugar, cosa, objeto o situación real o abstracta, que sea de interés para la organización. Pueden ser fuertes o débiles, todo depende de las claves primarias que se puedan formar entre sus atributos, de esta manera si entre sus atributos se selecciona uno para ser llave primaria ésta entidad es fuerte, y si no cuenta con ninguna llave primaria entre sus atributos es una entidad débil. La representación grafica de este objeto es un rectángulo con el nombre (sustantivo común en singular) para entidades fuertes y un rectángulo doble para entidades débiles, como se muestra en la figura 4 y 5.

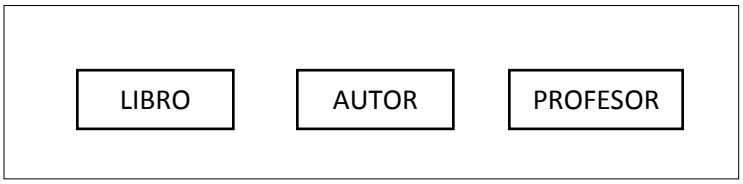


Figura 4. Representación de entidades fuertes.

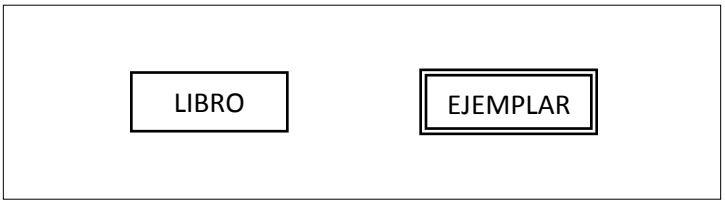


Figura 5. Representación de una entidad débil.

En la figura 5 se muestra una entidad débil, ya que EJEMPLAR es dependiente de LIBRO, así que si eliminamos LIBRO no habrá ningún vínculo por el cual exista EJEMPLAR.

Relación: es la asociación o correspondencia entre las entidades, este objeto se representa mediante un rombo etiquetado con el nombre de la relación (verbo en singular), esta relación puede tener una cardinalidad o numero de ocurrencias entre entidades como pueden ser: 1:1, 1:N, N:M, 0:M ó 0:1. Un grado dependiendo del número de entidades involucradas en las interrelaciones (grado 1, grado 2, grado 3, grado n).

Por tanto la cardinalidad de relaciones entre entidades puede ser:

- uno a uno (1:1): una entidad en A se asocia única y exclusivamente con una entidad en B.
- uno a varios (1:N): una entidad en A se asocia con cualquier número de entidades en B.
- varios a uno (N:1): varias entidades en A se asocian con una entidad en B.
- varios a varios (N:N): varias entidades en A se asocian con varias entidades en B.

En la figura 6, se muestran los tipos de cardinalidad que pueden tener las relaciones entre entidades.

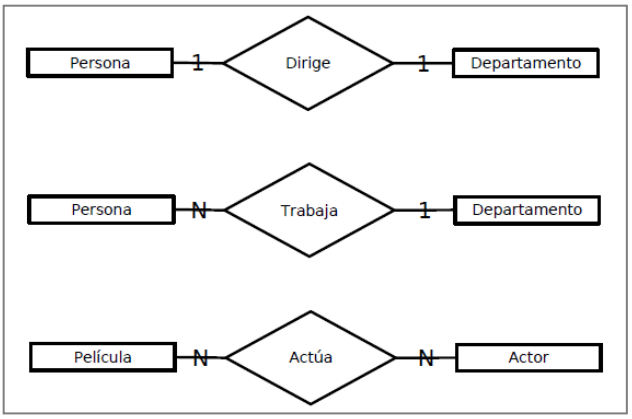


Figura 6. Representación de la cardinalidad entre entidades.

Clave: es un conjunto de atributos que identifican de manera única y mínima cada tupla de la relación. Casi siempre hay una clave candidata en una relación, ya que no puede haber tuplas iguales. Entre los atributos debe haber más de una clave candidata, entre las que podemos distinguir:

i. *Clave primaria*: es aquella que identifica las tuplas de una relación.
ii. *Clave alternativas*: son aquellas que no han sido escogidas como claves primarias.
iii. *Clave ajena*: es aquella clave que formará parte de una nueva relación pero que es clave primaria en otra.

Las restricciones semánticas se representan de la siguiente manera:

i. Restricción de clave primaria (PRIMARY KEY): define el atributo o atributos que serán los identificadores inequívocos para cada tupla de la relación, esta clave primaria no puede admitir valores nulos. Gráficamente aparece subrayado el nombre del atributo seleccionado.

ii. Restricción de clave única (UNIQUE): define el atributo o atributos que deben ser únicos e irrepetibles en las tuplas de una relación. Gráficamente aparece subrayado de forma discontinua el nombre del atributo seleccionado.

iii. Restricción de obligatoriedad (NOT NULL): esta restricción define que atributos deben tomar siempre un valor, es decir no pueden ser nulos. Gráficamente dentro del diagrama E/R estos atributos son opcionales y aparecen con un asterisco (Cuadra, y otros, 2008).

Todas estas restricciones las podemos visualizar en la figura 7.

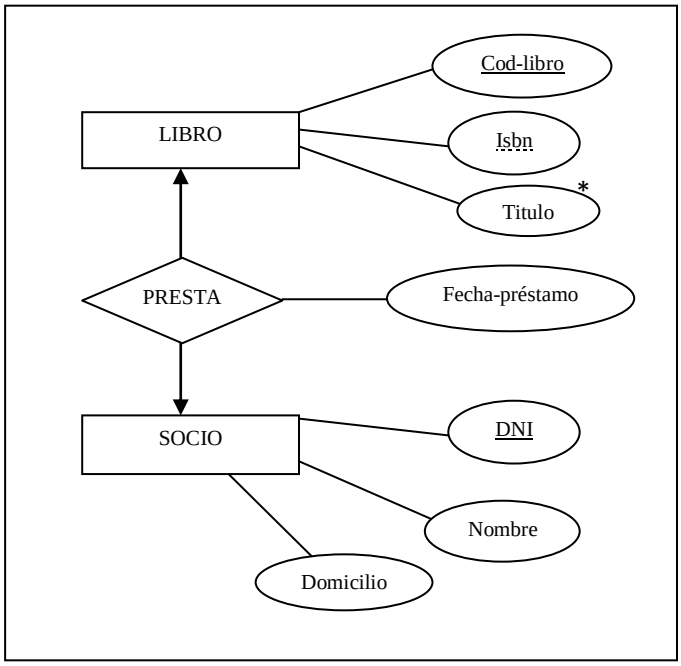


Figura 7. Representación de restricciones semánticas.

Atributo: son todas aquellas propiedades o características de un tipo de entidad o relación, se representan mediante un círculo u ovalo con el nombre que califica a este atributo. Los atributos pueden ser simples o compuestos, es decir que no se divida en más atributos o que se pueda dividir en más atributos, como por ejemplo: el atributo *nombre_empleado* puede dividirse en *nombre*, *apellido_paterno*, *apellido_materno*, en cambio el *CURP* no, ya que es una clave única de registro, figura 8 inciso *a)* y *b)*. Entre todos los atributos puede existir una llave primaria que será el atributo identificador y único para esa entidad, el cual la distinguirá de las demás entidades, y se representa subrayando el nombre del atributo, figura 8 inciso *c)*, un atributo candidato y el resto pueden ser atributos alternativos. Un atributo puede ser multivaluado cuando toma más de un valor (por ejemplo: varias direcciones de proveedores, teléfonos de una persona, nombres de hijos, etc.) ó derivables cuando son el resultado de alguna operación entre atributos (por ejemplo la edad, resultado de la operación del atributo fecha_nacimiento y fecha_actual), figura 8 inciso *d)* (Cuadra, y otros, 2008).

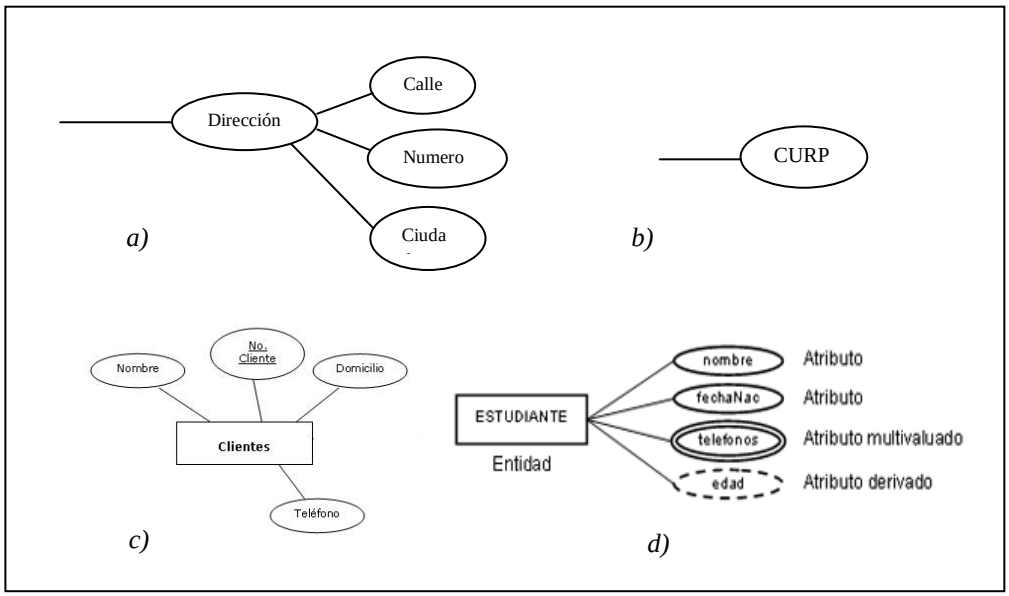


Figura 8. Representación de atributos compuestos *a)*, simples *b)*, identificadores primarios *c)* y multivaluados o derivados *d)*.

3.1.2 Diagrama Entidad-Relación (DER)

Los diagramas E-R sirven para expresar la estructura lógica de una base de datos, son expresiones gráficas que nos permiten describir conceptos, mediante simples dibujos o grafos que hacen más fácil su entendimiento. Los componentes principales del diagrama E-R son:

- Rectángulos, representan entidades.
- Elipses, representan atributos.
- Rombos, representan relaciones.
- Líneas, unen atributos a entidades, y entidades a relaciones.
- Los atributos que son claves primarias aparecen subrayados
- La cardinalidad aparece con las razones siguientes: uno a uno (1:1), uno a varios (1:N), varios a uno (N:1) y varios a varios (N:M).

En el siguiente diagrama de la figura 9 se muestra el diagrama E/R del departamento de embarques de un Centro de distribución, describiendo las entidades CAMIONEROS, PAQUETE, CAMIONES Y CIUDAD, cada una con sus respectivos atributos y colocando su identificador primario (llave primaria) subrayado, así como sus interrelaciones entre las entidades descritas.

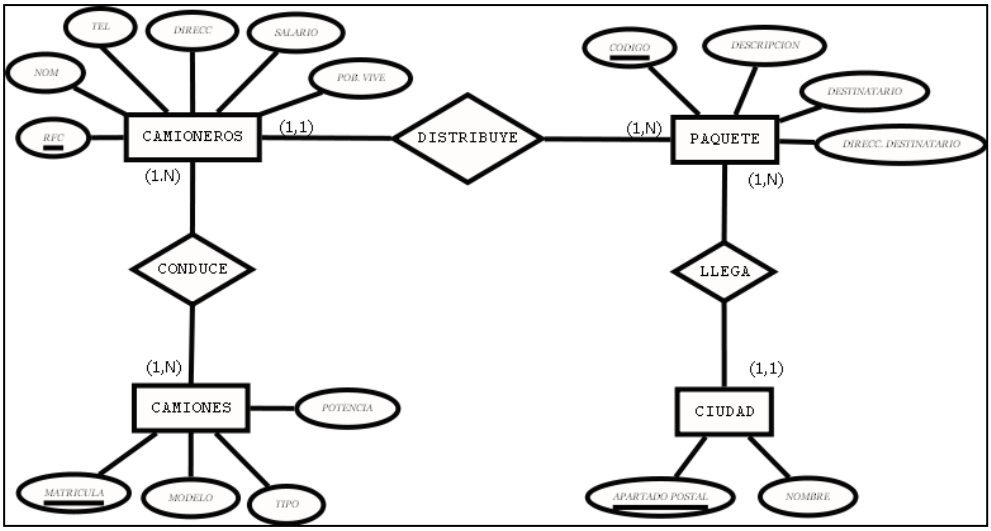


Figura 9. Diagrama Entidad-Relación del departamento de embarques de un Centro de Distribución.

3.2 Modelado de Datos

3.2.1 Modelo de Datos

Es una serie de herramientas conceptuales que se encargan de hacer una descripción de los datos, incluyendo las relaciones, la semántica y las restricciones de consistencia. La mayoría de los modelos de datos poseen un conjunto de operaciones básicas para especificar consultas y actualizaciones de la base de datos (Silberschatz, Korth, & Sudarshan, 2002).

Los modelos de datos se pueden clasificar en:

- Modelo Relacional
- Modelo orientado a objetos
- Modelo relacional-objeto
- Modelo jerárquico
- Modelo de red

En particular se describirá el Modelo Relacional siendo este el utilizado para el desarrollo del presente trabajo.

3.2.2 Modelo Relacional

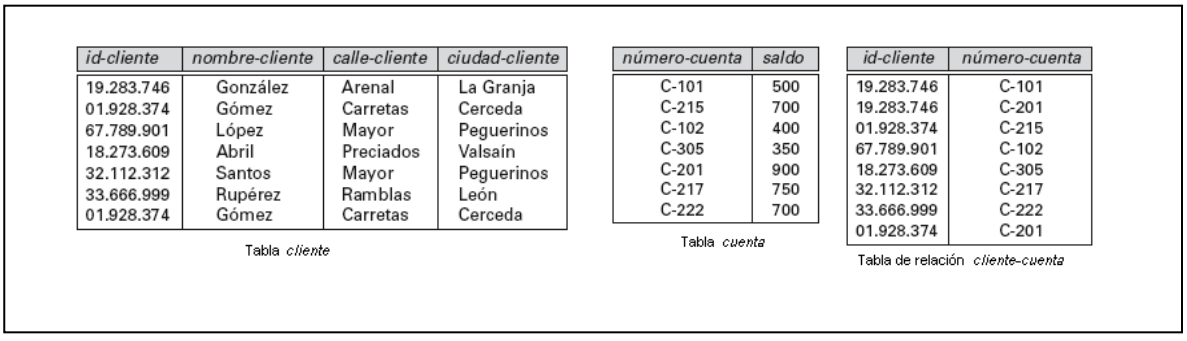
A finales de los años sesenta, CODD presenta un nuevo modelo de datos basado en relaciones, este modelo es uno de los más utilizados en el diseño de las bases de datos, utiliza tablas bidimensionales para representar los datos y sus relaciones. Es el más usado por casi todos los sistemas de bases de datos y comúnmente se realiza primero el diseño de la base de datos en el modelo E-R, y posteriormente se traduce al modelo relacional (Castaño & Piattini Velthuis, Fundamentos y modelos de bases de datos, 1999).

Sus principales características son:

- El modo de almacenamiento físico de los datos no influye en la manipulación lógica de los mismos, así que el usuario no tiene que hacer cambios en sus programas (independencia física).
- La administración de la base de datos no debe perjudicar en los programas y/o usuarios que acceden a las vistas del sistema (independencia lógica).
- Los usuarios no necesitan saber donde se encuentran los datos físicamente, solo se les debe ofrecer lo que necesitan a través de las aplicaciones (flexibilidad).
- La representación en forma de tablas de las estructuras lógicas de la base de datos hace más fácil su entendimiento y su uso para los usuarios (Uniformidad y Sencillez).

El elemento principal de este modelo es la relación que se representa mediante una tabla la cual está basada en registros, cada tabla tiene registros de un tipo particular, que a su vez tienen un número fijo de campos (atributos), cada columna corresponde a los atributos del tipo de registro (Lucas Gómez, Romera García, Fraile Dotes, Argente del Castillo, & Alfaro Pesa, 1993).

En la figura 10 se presenta un ejemplo del modelo relacional el cual contiene 3 tablas, la tabla *cliente* que muestra los clientes de un banco, la tabla *cuenta* con el monto de cada una de esas cuentas y la tabla *cliente-cuenta* con las cuentas que pertenecen a cada cliente.



id-cliente	*nombre-cliente*	*calle-cliente*	*ciudad-cliente*
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
18.273.609	Abril	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Ramblas	León
01.928.374	Gómez	Carretas	Cerceda

Tabla *cliente*

número-cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

Tabla *cuenta*

id-cliente	*número-cuenta*
19.283.746	C-101
19.283.746	C-201
01.928.374	C-215
67.789.901	C-102
18.273.609	C-305
32.112.312	C-217
33.666.999	C-222
01.928.374	C-201

Tabla de relación *cliente-cuenta*

Figura 10. Representación del Modelo Relacional.

3.3 Sistemas Manejadores de Bases de Datos (SMBD)

Un Sistema Manejador de Bases de Datos (SMBD), (en inglés: *Database Management System*, abreviado DBMS) es un conjunto de programas especializados, diseñados para describir, introducir, proteger, almacenar y recuperar datos, es decir para administrar la base de datos. Hay varios tipos de manejadores de bases de datos como: relacional, jerárquico, de red, etc., entre otros, pero el más utilizado para PC's es el modelo relacional que almacena los datos en forma de tabla, descrito anteriormente en el apartado 3.2.2.

Algunas de sus funciones en las bases de datos son:

- Definir los datos empleados y especificar sus relaciones.
- Proporcionar un método para dar de alta, baja y modificar los datos.
- Permitir que múltiples usuarios compartan la base de datos.
- Permitir la recuperación de datos mediante un lenguaje entendible (lenguaje estructurado de consulta SQL).

Y sus beneficios son:

- Mejorar la integración de los datos.
- Aumentar la accesibilidad de los datos.
- Mejorar el control de los datos.
- Facilidad en el desarrollo y administración de las aplicaciones.
- Mejorar la seguridad de los datos.

Un SMBD tiene dos componentes:

- Sistema de control de la base de datos, en el que el software del SCBD interactúa con los programas de aplicación de los usuarios para recuperar los datos de la base de datos.
- Sistema de almacenamiento de la base de datos, en el que el software del SSBD manipula los archivos de datos necesarios para almacenar los datos dentro de la base de datos.

En la figura 11 se presenta el esquema general de un SMBD y los componentes que intervienen en el proceso de un sistema de base de datos.

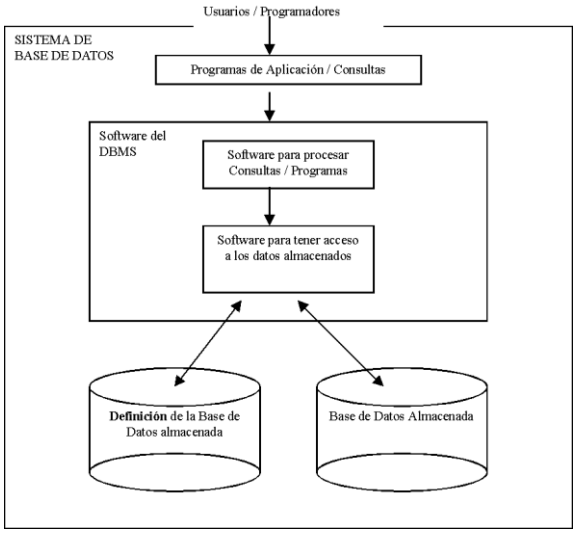


Figura 11. Esquema General de un SMBD.

Todas las funciones que realiza el SMBD se pueden agrupar en vistas o niveles. Cada uno de estos niveles tiene una función en específico dentro de la base de datos.

- Nivel externo o de vistas: el cual está compuesto por lenguajes de manipulación de datos, que son empleados para extraer datos de la base de datos, un ejemplo de ellos es el SQL.

- Nivel conceptual: en este nivel se hace la definición de los datos empleados para modelar la base de datos, se puede utilizar cualquiera de los tres modelos de datos: modelo jerárquico, modelo de red y modelo relacional.

- Nivel interno: en este nivel se hace la definición de los datos así como sus relaciones. Se hace uso del diagrama entidad-relación.

- Nivel organización física de los datos: aquí se define donde y como serán almacenados los datos. Tomando en cuenta los requerimientos del procesamiento de datos, los procesos que se realicen, y las relaciones de los datos mismos.

Para entender mejor un SMBD en la tabla 1 se muestran los niveles o vistas que lo componen.

Programas que emplean este nivel.	Nivel de datos	Descripción
Usuarios finales y programador de aplicaciones	Nivel externo	Vista de la BD, orientada a los usuarios del sistema.
Programador de aplicaciones y Administrador de la BD	Nivel conceptual	Vista de toda la BD.
Administrador de la BD y programador de sistemas.	Nivel interno	Vista de relaciones entre entidades.
Programador de sistemas	Nivel de organización física	Vista de distribución física de los datos en un dispositivo de almacenamiento secundario.

Tabla 1. Modelo de referencia de los niveles o vistas de un DBMS

3.3.1 Sistemas Manejadores de Bases de Datos disponibles en el mercado (libres y no libres)

Para poder satisfacer las necesidades de los servicios informáticos se han desarrollado diversos Sistemas Manejadores de Bases de Datos, que cubran la demanda del mercado, permitiendo organizar y administrar la información de manera sencilla siendo más fácil su manejo. Ofreciendo a la comunidad distintas formas de licenciamiento como son la libre y la comercial.

El licenciamiento es un convenio en donde están estipuladas todas las normas y clausulas bajo las que se va a trabajar con un programa (alcances, limitaciones de uso, instalación, distribución y reproducción), por tanto al realizar la descarga el usuario estará aceptando todas las condiciones especificadas en el licenciamiento de cada software.

Para poder diferenciar entre una y otra hay que tomar en cuenta algunas características importantes de cada una de estas formas de licenciamiento.

Características del software libre:

- Puede ser utilizado por cualquier usuario sin restricción alguna.
- Puede ser estudiado y adaptado a las necesidades propias, permitiendo el acceso a su código fuente.
- Se pueden distribuir copias sin que por ello se cometa algún delito.
- Puede ser modificado y publicar esas modificaciones o mejoras en beneficio de todos.
- Debe estar disponible para uso, desarrollo y distribución comercial.

Características del Software No Libre:

- El software es propiedad de la empresa que lo crea o lo adquiere.
- Su distribución, implementación y uso queda especificado bajo una licencia entre el cliente y el comprador.
- Tienen un costo elevado.
- Solo se entrega una copia del ejecutable del producto al cliente.
- No se puede realizar ninguna modificación en el código fuente del producto, por tanto el cliente depende del proveedor, siendo este el único que puede realizar cambios en él.
- Cuentan con soporte técnico especializado.

Cada uno de estos Sistemas Manejadores de Bases de Datos tiene sus características generales, ventajas y desventajas, y para el caso específico de este trabajo se describirá más a detalle en el Capítulo III el SMBD *MySQL*, siendo este el manejador seleccionado para el diseño de la Base de Datos propuesta.

A continuación se presentan en la tabla 2 los Sistemas Manejadores de Bases de Datos disponibles en el mercado.

Libres	No libres	No libres y gratuitos
MySQL	Microsoft SQL Server	Microsoft SQL Server Edición Básica
FireBird	PervasiveSQL	Sybase ASE (Linux)
SQLite	Oracle	Oracle Express Edition 10g
PostgreSQL	dBase	
BDB	FileMaker	
DB2 Express-C	Fox Pro	
Apache Derby	IBM DB2 Universal Database (DB2 UDB)	
	IBM Informix	
	Magic	
	Open Access	
	Paradox	
	Progress	
	Sybase ASE, Sybase ASA, Sybase IQ	
	Windows Base	

Tabla 2. SMBD libres y comerciales disponibles en el mercado

3.3.2 Usuarios de los SMBD

Una base de datos tiene como objetivo principal almacenar y recuperar información, para eso se debe tener personal especializado y no especializado que tiene la función de administrar, manipular y modificar el sistema. Podemos clasificar a los usuarios de la base de datos tomando en cuenta la manera en que interactúan con el sistema.

- Usuario final: tiene acceso a la base de datos desde su computadora mediante el lenguaje de consulta (LMD) o por medio de un programa de aplicación previamente implantado. Estos usuarios no necesitan ser especialistas en bases de datos, ya que pueden manejar la información eficientemente a través de los programas de aplicación.

- Administrador de la base de datos (ó en inglés, DBA Database Administrator): es el encargado de controlar la base de datos, está especializado en el manejo y manipulación de la misma. Será la persona responsable de la seguridad e integridad de la base, deberá tener amplio conocimiento del SMBD en cuanto a la creación y administración de tablas, índices, consultas, formularios, reportes, macros, etc. En resumen será el intermediario entre el programador y el usuario final.

- Programador de aplicaciones: será el encargado de escribir y diseñar los programas de aplicación que son utilizados por las bases de datos, en lenguajes de alto nivel (Cobol, Clipper, VisualBasic, 4GL), con los cuales se puede acceder y actualizar los datos. Deberán de ser capaces de realizar soluciones a la medida. Debe tener un conocimiento aun más profundo del SMBD. (Cuadra, y otros, 2008).

3.3.3 Lenguajes del Sistema Manejador de Bases de Datos

Un sistema manejador de bases de datos está estructurado por dos lenguajes, uno especifica el esquema de la base de datos y el otro expresa las consultas y las modificaciones a la base de datos. Aunque tienen funciones diferentes no trabajan por separado ambos forman parte del lenguaje SQL ampliamente usado.

Estos lenguajes son:

- Lenguaje de Definición de Datos (LDD), (en inglés: *Data Definition Language*, abreviado DDL): por medio de este lenguaje el SMBD identifica las descripciones de los elementos de los esquemas y actualiza un conjunto especial de tablas denominado *diccionario de datos* o *directorio de datos.*

  El diccionario de datos contiene los metadatos, es decir datos referentes a los datos del sistema, por eso cada vez que se quiere realizar alguna lectura o modificación de los datos reales la base de datos consulta primero en él.

  En el siguiente ejemplo la instrucción en lenguaje SQL define la creación de una tabla:

  Create table cuenta

  (numero_cuenta *char* (10),

  saldo *integer*)

- Lenguaje de Manipulación de Datos (LMD), (en inglés: *Data Manipulation Language*, abreviado DML): Permite la manipulación de las operaciones de recuperación, inserción, eliminación y modificación de datos en la base de datos.

Hay dos tipos de LMD's:

- De alto Nivel o No procedimental: aquí el usuario especifica que datos quiere sin especificar como los obtendrá. SQL es un ejemplo de este lenguaje.
- De bajo Nivel o por procedimientos: en este caso el usuario especifica que datos quiere y como los obtendrá.

Para la recuperación de información se utiliza el lenguaje de consulta que es lo mismo que el lenguaje de manipulación de datos, mediante la consulta que no es más que una instrucción para recuperar información, se hace la petición a la base de datos para que muestre lo que se le pide.

En el siguiente ejemplo se hace la petición a la base de datos para traer información de un cliente determinado por el identificador '2014':

select cliente.nombre_cliente
from cliente
where cliente.id_cliente='2014'

En conjunto estos dos lenguajes forman parte del lenguaje SQL el más utilizado en las bases de datos, el cual se describirá más ampliamente en el Capítulo II.

3.3.4 Elección del Sistema Manejador de Bases de Datos

Las bases de datos siempre están presentes en casi todas las organizaciones, la decisión de cambiar una aplicación basada en un sistema de ficheros a base de datos tiene que ver con ciertos factores, enunciaremos algunos de ellos:

- Complejidad de los datos.
- Compartimiento entre aplicaciones.
- Cambios constantes en los datos.
- Solicitud de datos de acuerdo a las necesidades del usuario.
- Cantidades grandes de datos requieren un control eficiente.

Una vez analizados todos estos factores y elegido el Modelo de datos para diseñar la base de datos, se debe hacer la elección de un determinado SMBD, tomando en cuenta los puntos antes mencionados y de acuerdo a las necesidades y capacidades del sistema o en un determinado momento de la misma organización.

Estos elementos son:

1. Factores técnicos

    - Tipo de SMBD que se utilizará (relacional, objeto-relacional, orientado a objetos o de otra clase).
    - Los datos deben organizarse independientemente de las aplicaciones a utilizar y de los archivos donde se almacenan.
    - El SMBD debe ser portable, es decir que las versiones del mismo se puedan ejecutar en muchas configuraciones de hardware/software (diferentes plataformas).
    - Los datos y aplicaciones deben ser accesibles a los usuarios en un lenguaje de consulta amigable (SQL, Query-by-example, etc).
    - Los datos no deben estar duplicados.
    - Debe proporcionar protección frente a fallos.
    - Los datos deben ser asegurados de manera que no cualquier persona tenga acceso a ellos.
    - Los datos deben ser portables, replicables y distribuidos.

2. Factores “no técnicos”

    - Se debe tomar en cuenta el costo de la compra de software, hardware y mantenimiento.
    - Costo de la creación y conversión de la base de datos.
    - Costo de adquisición de personal nuevo para la administración de la base de datos.
    - Costo de la capacitación al personal.
    - La disponibilidad de servicios del proveedor.

Todos estos aspectos son importantes y decisivos para elegir el SMBD, por tanto no se deben pasar por alto ya que de ello dependerá el buen diseño de la base de datos (Navathe & Elmasri, 2002).

3.4. Diseño Lógico

El diseño lógico de las bases de datos es la fase donde se decide la colocación de los atributos de las entidades de una organización (llámese empresa, banco, escuela, línea aérea, etc.) a estructuras de bases de datos, como lo son las tablas de las bases de datos relacionales. El objetivo del diseño lógico es la creación de tablas con una estructura bien definida y que representen de manera correcta el entorno de una organización, estas estructuras deben tener la capacidad de almacenar los datos de la organización de manera no redundante, colocando identificadores que permitan soportar las relaciones entre entidades (L. Gillenson, 2006).

El esquema lógico se obtendrá de acuerdo al modelo de datos del sistema manejador de bases de datos elegido (jerárquico, red o relacional). En este sentido se tendrá que transformar el esquema conceptual a un determinado modelo, esto no quiere decir que no se puedan usar características de otro modelo buscando con ello hacer la mejor representación de la información conservando la semántica, la redundancia, su fácil comprensión, su adaptabilidad, la protección de datos, entre otras (Castaño & Piattini Velthuis, Concepción y Diseño de Bases de Datos: Del modelo E/R al modelo relacional, 1993).

A continuación se transformará el esquema E/R al modelo de base de datos relacional.

3.4.1 Reglas de transformación de un esquema E/R a un esquema relacional

El paso de un esquema en el modelo E/R al relacional está basado en lo siguiente:

- La entidad se convierte en relación.
- La interrelación N:M se transforma en una relación.
- La interrelación 1:N se traduce a una propagación de clave o se crea una nueva relación.
- No puede haber tuplas semejantes en una relación (restricción de llave primaria).
- La ordenación de atributos y tuplas no es significativo.
- El atributo designado como llave primaria no puede tomar valores nulos.

En estas transformaciones se pierde un poco la semántica pero eso no importa mucho, no se pierde la integridad de los datos, ya que se definen restricciones de integridad referencial para evitarlo. Esta transformación da como resultado un conjunto de sentencias LDD escritas en el lenguaje del SMBD que se ha seleccionado, especificando los esquemas en el nivel conceptual y externo del sistema de base de datos.

Tomando en cuenta lo anterior hay reglas que seguir para la transformación de cada uno de los elementos del Diagrama E/R, y son las siguientes:

a. Transformación de entidades: el modelo lógico permite representar una entidad mediante una relación o tabla, la cual tendrá el mismo nombre de la entidad.

b. Transformación de atributos de entidades: cada atributo de la entidad se convertirá en una columna de la tabla, identificando aquellos que son principales o claves primarias, alternativos o claves únicas, y los que no son principales.

c. Transformación de interrelaciones (N:M): En las transformaciones N:M se creará una relación donde las claves primarias de las entidades que enlazan la interrelación serán los atributos de la nueva relación y se convertirán en claves ajenas con respecto a las entidades donde son llaves primarias. Si la relación contiene algún otro atributo este formará parte de la nueva relación creada.

d. Transformaciones de interrelaciones (1:N): en esta transformación se realiza la propagación de clave, es decir a la entidad que se encuentra del lado de N se le incluirá la clave primaria de la entidad que está del lado 1 y que forma parte de esta interrelación, la cual se convertirá en clave ajena en la entidad incluida y puede tomar otro nombre, sin afectar por ello la transformación. Si la interrelación tiene atributos estos también se propagan (Cuadra, y otros, 2008).

Con el siguiente modelo E/R de la figura 11, se demuestra lo anterior.

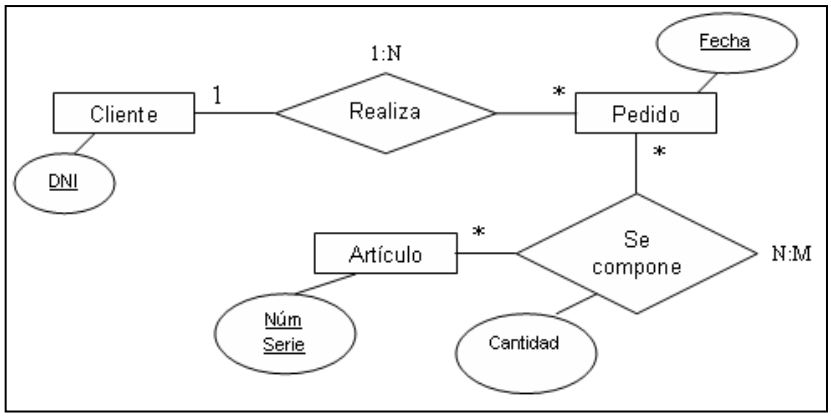


Figura 11. Diagrama Entidad Relación del departamento de Ventas.

1: Transformación de las entidades Cliente, Pedido y Artículo a relaciones:

Pedido (Fecha, DNI) ----------transformación de la interrelación 1:N

Cliente (DNI)

Pedido_Articulo (Núm. Serie, Fecha, Cantidad) ------- transformación de la interrelación N:M

Artículo (Núm. Serie)

2: Transformación de las entidades y sus atributos a tablas:

Pedido

Fecha	DNI
12/04/2010	0010
15/05/2010	0011
25/01/2011	0012

Clave primaria

Clave ajena

Pedido_Articulo

Num.Serie	Fecha	Cantidad
0000145	12/04/2010	451
0000248	15/05/2010	521
0000478	25/01/2011	569

Claves ajenas

Cliente

DNI
0010
0011
0012

Articulo

Num.Serie
0000145
0000248
0000478

Claves primarias

3: Representación en lenguaje lógico estándar (SQL):

```
CREATE TABLE Pedido
        (Fecha  fechas de pedidos,
         DNI identificador de clientes,
         PRIMARY KEY (Fecha),
         FOREIGN KEY (DNI));

CREATE TABLE Cliente
        (DNI identificador de clientes,
         PRIMARY KEY (DNI));


CREATE TABLE Pedido_Articulo
        (Num.Serie  num_artículo,
         Fecha fechas de pedidos,
         Cantidad num_piezas,
         PRIMARY KEY (Num.Serie, Fecha),
         FOREIGN KEY (Num.Serie) REFERENCES  Articulo
                                 ON DELETE CASCADE
                                 ON UPDATE CASCADE,
        FOREIGN KEY (Fecha) REFERENCES Pedido
                                 ON UPDATE CASCADE);
```

```
CREATE TABLE Artículo
        (Num.Serie  num_articulo,
         PRIMARY KEY (Num.Serie));
```

3.4.2 Normalización de datos en tablas

Durante el diseño de una base de datos relacional puede suceder que no se defina de manera adecuada el esquema conceptual y por consecuencia se realice una inadecuada transformación al modelo relacional, para que no suceda esto se puede aplicar una técnica de verificación llamada normalización de datos.

La normalización de datos es una técnica que sirve para organizar los atributos en tablas excluyendo la redundancia entre aquellos que no son claves, es decir lo que se busca es que cada tabla resultante describa un tipo de entidad único o una relación única muchos-a-muchos, tomando en cuenta que las llaves externas solo aparecerán donde sean necesarias, obteniendo con ello una base de datos bien estructurada. Este proceso se basa en la descomposición de los atributos en subgrupos y para llegar al objetivo final se pasará por varias formas normales.

Antes de describir cada una de estas formas es importante conocer un concepto que será utilizado dentro de la normalización de datos, este concepto es la dependencia funcional, que no es más que la forma de expresar que un valor de un atributo particular se asocia con un valor único y especifico de otro atributo. Por ejemplo para un identificador de empleado denominado Id_empleado = 001 solo puede haber un nombre_de_empleado = Jorge, es decir un valor es dependiente de otro.

Pasos del proceso de normalización:

a. Primera forma normal: en esta primera forma se deben eliminar los atributos repetidos de las tablas, crear tablas independientes para cada conjunto de datos relacionados, y por ultimo identificar cada conjunto de datos con una llave primaria, lo que logramos con esto es que cada valor de atributo sea atómico, y ningún atributo contenga valores múltiples.

b. Segunda forma normal: en esta fase se crearán tablas independientes para los conjuntos de valores que se apliquen a varios registros y se relacionarán mediante una clave ajena, es decir no se permitirán dependencias funcionales parciales, ya que un atributo que no es llave no puede depender solo de una parte de la llave primaria.

c. Tercera forma normal: en esta fase eliminaremos los campos que no dependen de la llave primaria de esta manera los valores de un registro que no formen parte de la llave primaria no deben estar contemplados en la tabla (L. Gillenson, 2006).

Ejemplo en el que se aplica el proceso de normalización para una tabla de alumnos:

1) Tabla sin normalizar

Matricula-alumno	Tutor	Despacho-Tutor	Clase1	Clase2	Clase3
1022	García	412	101-07	143-01	159-02
4123	Díaz	216	201-01	211-02	214-01

2) Primera forma normal: no se repiten grupos, pero los campos Clase1, Clase2 y Clase3 deben estar en una tabla independiente, para corregir este error se puede crear una nueva tabla donde se elimine el grupo repetido (Num-Clase).

Matricula-alumno	Tutor	Despacho-Tutor	Num-clase
1022	García	412	101-07
1022	García	412	143-01
1022	García	412	159-02
4123	Díaz	216	201-01
4123	Díaz	216	211-02
4123	Díaz	216	214-01

3) Segunda forma normal: como se puede ver hay datos redundantes por tanto no se cumple la segunda forma normal, en este caso el Num-clase no depende funcionalmente de Matricula-alumno que es la llave primaria, por tanto se pueden dividir en dos tablas.

Alumnos

Matricula-alumno	Tutor	Despacho-Tutor
1022	García	412
4123	Díaz	216

Registros

Matricula-alumno	Num-clase
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

4) Tercera forma normal: en esta fase eliminaremos los datos que no dependen de la clave primaria. Examinando los datos el Despacho-tutor no depende de la clave primaria Matricula-alumno, por tanto la podemos pasar a otra tabla de Personal-tutor.

Alumnos

Matricula-alumno	Tutor
1022	García
4123	Díaz

Personal-tutor

Nombre	Despacho-tutor	Depto
García	412	42
Díaz	216	42

De esta manera quedan normalizados los datos, obteniendo valores atómicos en cada uno de los registros, eliminando dependencias funcionales así como datos que no son dependientes de la clave primaria. Por tanto quedan tres tablas en lugar de una: Alumnos, Personal-tutor y Registros.

3.5 Diseño Físico

El Diseño Físico de una Base de Datos es el proceso mediante el cual se determinan y modifican sus estructuras, mejorando con ello el desempeño del ambiente durante el tiempo de consulta. Las tablas que se sometieron al proceso de normalización se van a modificar para que las aplicaciones que las utilizan tengan un mejor tiempo de respuesta y cumplan los siguientes objetivos:

- Disminución de tiempos de respuesta
- Minimizar el espacio disponible para almacenar
- Impedir que se reorganice la base de datos
- Obtener la máxima seguridad
- Optimizar el consumo de los recursos

Cabe mencionar que esta parte del Diseño de Bases de Datos es dependiente del SMBD que se va a utilizar, por tanto se debe conocer su funcionalidad, así como el sistema informático sobre el que se trabajará.

Para poder alcanzar una organización física que minimice el tiempo de respuesta debemos conocer la arquitectura física de la base de datos y sus mecanismos de acceso (estructuras de memoria, procesos, etc.). En la figura 12 se muestra la Arquitectura Física de la base de datos.

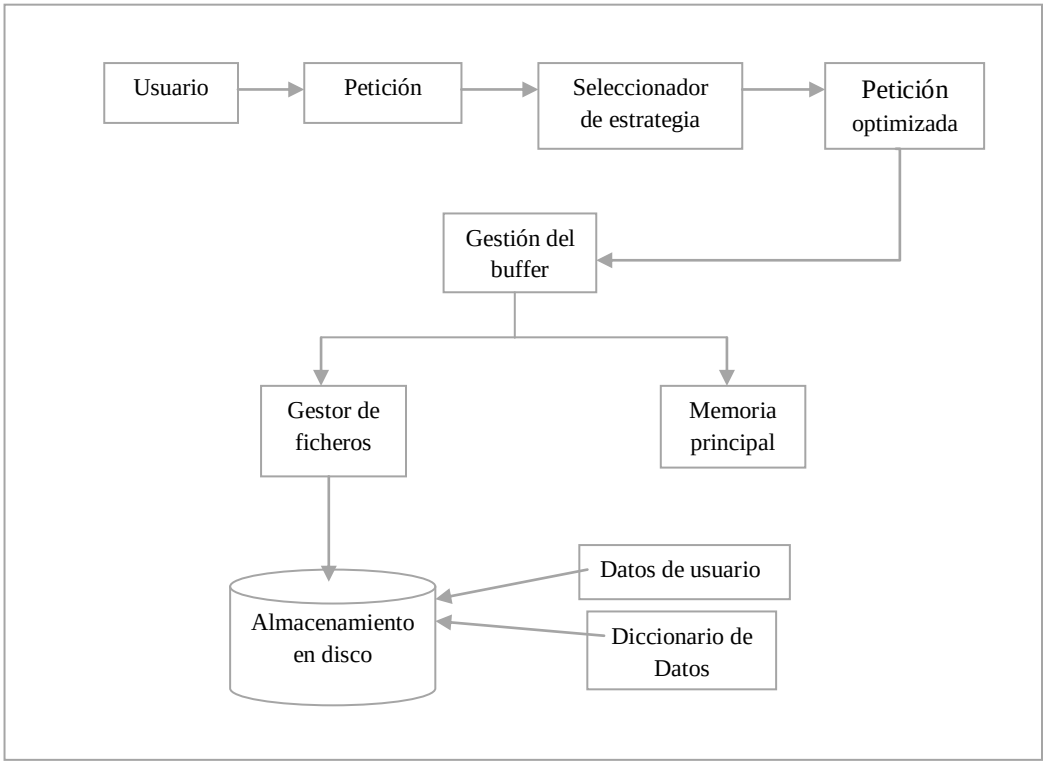


Figura 12. Arquitectura física de la base de datos.

En esta arquitectura se puede ver como el usuario interactúa con el SMBD mediante una petición, la cual es convertida en una estrategia más eficiente de ejecución, entonces esta petición optimizada activa la gestión del buffer, encargado de controlar el movimiento entre la memoria principal y los datos almacenados en memoria secundaria. El gestor de buffer esta soportado por un gestor de ficheros el cual controla y gestiona la asignación de espacio en disco junto con sus estructuras de datos asociadas. El disco contiene los datos de los usuarios y el diccionario de datos en el cual está definida la estructura de los datos de los usuarios y como pueden ser usados (Pastor López & Blesa Pons, 2000).

El punto de partido de esta fase serán las tablas normalizadas, a partir de ellas se deben tomar en cuenta ciertos factores como:

- Requerimientos del negocio (tiempos de respuesta y rendimiento).
- Estimación del volumen de datos y volatilidad de los mismos.
- Requisitos y prioridades de las aplicaciones.
- Requerimientos operativos (seguridad, respaldo y recuperación de los datos).
- Características de Hardware y Software (equipo y SMBD).

Por tanto podemos dividir el Diseño físico en cuatro etapas que tendrán ciertos pasos a seguir:

1. Realizar la traducción del esquema lógico a un SMBD específico (en este caso con el que se va a trabajar).
2. Diseño de la forma física. (Cómo y dónde serán almacenados los datos).
3. Diseño del esquema de seguridad (vistas y accesos de usuarios).
4. Monitoreo del sistema (analizar el funcionamiento del sistema).

3.5.1 Traducción del esquema lógico a un SMBD específico

En esta primera etapa se transformará el esquema lógico (tablas) a un esquema del SMBD elegido para llevar a cabo esto debemos tomar en cuenta algunas consideraciones que tienen que ver con el SMBD como son:

- El soporte de claves primarias, ajenas y alternativas.
- El soporte de definir atributos no nulos (NOT NULL).
- El soporte de dominios.
- El soporte de restricciones que sean definidos por las reglas del negocio.
- Cómo crear las relaciones base.

Una vez analizados estos parámetros se procede a crear las relaciones base mediante el Lenguaje de Definición de Datos del SMBD, definiendo nombre de la relación, sus atributos entre paréntesis, llaves primarias y ajenas si tiene, así como las reglas de integridad de las claves ajenas. Por otro lado en el diccionario de datos se describen los atributos definiendo el tipo de dato, longitud y restricciones de dominio, su valor por defecto, el cual puede ser opcional, si va admitir nulos, si es derivado y en caso de serlo como calcularlo.

De la misma manera se describirán las restricciones de acuerdo a las reglas del negocio (normatividades de la empresa) que se tengan que aplicar sobre los datos.

En el siguiente ejemplo se muestra como se realiza esta primera etapa:

a) Creación de relaciones base

```
CREATE TABLE status_escolar
(id_status int NOT NULL auto_increment,
status varchar(15) NOT NULL,
PRIMARY KEY (id_status));
```

Creación de la tabla status_escolar en lenguaje SQL

b) Diccionario de datos

table_schema	table_name	column_name	data_type	column_type	column_key	extra
escuela	status_escolar	id_status	int	int(11)	PRI	auto_increment
escuela	status_escolar	status	varchar	varchar(15)		

c) Restricciones de las reglas de negocio

Por ejemplo, si no se quiere que un empleado tenga más de diez inmuebles asignados, se puede definir una restricción en la sentencia CREATE TABLE de la relación INMUEBLE:

```
CONSTRAINT inmuebles_por_empleado
  CHECK (NOT EXISTS (SELECT enum
          FROM   inmueble
          GROUP BY enum
          HAVING COUNT(*)>10))
```

Todas las restricciones que se definan deben estar documentadas. Si hay varias opciones posibles para implementarlas, hay que explicar porqué se ha escogido la opción implementada.

3.5.2 Diseño de la representación física

El principal objetivo del Diseño físico es almacenar los datos de manera eficaz, para lograrlo se deben tomar en cuenta tres aspectos importantes:

- El tiempo de respuesta requerido.
- La capacidad total o rendimiento requerido.
- El espacio en disco requerido.

Una vez tomado en cuenta lo anterior, se tendrán que ir ajustando algunos factores para conseguir el equilibrio entre ellos, ya que en muchas ocasiones por satisfacer uno se ponen en riesgo los demás. Para ello se deben conocer las estructuras de almacenamiento disponibles por el SMBD y así definir una para inicializar la base de datos y posteriormente cambiarla para poder realizar operaciones en ella, minimizando los tiempos de respuesta.

Además de estas estructuras propias del SMBD también debemos tener presentes las del hardware como:

- *Memoria principal:* Los accesos a memoria principal son mucho más rápidos que los accesos a memoria secundaria (decenas o centenas de miles de veces más rápidos). Generalmente, cuanta más memoria principal se tenga, más rápidas serán las aplicaciones. Sin embargo, es aconsejable tener al menos un 5% de la memoria disponible, pero no más de un 10%. Si no hay bastante memoria disponible para todos los procesos, el sistema operativo debe transferir páginas a disco para liberar memoria (*paging*). Cuando estas páginas se vuelven a necesitar, hay que volver a traerlas desde el disco (*faltas de página*). A veces, es necesario llevar procesos enteros a disco (*swapping*) para liberar memoria. El hacer estas transferencias con demasiada frecuencia empeora las prestaciones.

- *CPU:* controla los recursos del sistema y ejecuta los procesos de usuario. El principal objetivo con este dispositivo es lograr que no haya bloqueos de procesos, si el sistema operativo, o los procesos de los usuarios, hacen muchas demandas de CPU, éste se convierte en un cuello de botella. Esto suele ocurrir cuando hay muchas faltas de página o se realiza mucho *swapping*.

- *Entrada/salida a disco:* Los discos tienen una velocidad de entrada/salida. Cuando se requieren datos a una velocidad mayor que ésta, el disco se convierte en un cuello de botella. Dependiendo de cómo se organicen los datos en el disco, se conseguirá reducir la probabilidad de empeorar las prestaciones. Los principios básicos que se deberían seguir para repartir los datos en los discos son los siguientes:

  ⇒ Los ficheros del sistema operativo deben estar separados de los ficheros de la base de datos.
  ⇒ Los ficheros de datos deben estar separados de los ficheros de índices
  ⇒ Los ficheros con los diarios de operaciones deben estar separados del resto de los ficheros de la base de datos.

- *Red:* La red se convierte en un cuello de botella cuando tiene mucho tráfico y cuando hay muchas colisiones.

Resulta casi seguro que si se mejora en alguno de estos aspectos los demás serán beneficiados enormemente (L. Gillenson, 2006).

3.5.2.1 Análisis de consultas y transacciones

Para realizar un buen diseño físico es necesario conocer las consultas y las transacciones que se van a ejecutar sobre la base de datos y tomar en cuenta algunas características propias, como por ejemplo:

- La frecuencia con la que se van a ejecutar.
- Que tablas, clase u objetos se actualizarán.
- A que tablas, clases u objetos tendrá acceso la consulta.
- El tiempo de respuesta que requiere el usuario para presentar los datos en pantalla.
- Los atributos cuyos valores obtendrá la consulta.
- Los tipos de acceso que tienen los datos (consulta, inserción modificación o eliminación) tomando en cuenta que los atributos que se modifican no son buenos candidatos para construir estructuras de acceso.
- Los atributos que se utilizan en la clausula WHERE de las sentencias SQL ya que pueden ser candidatos para construir estructuras de acceso, etc.
- Los atributos sobre los que se especificarán las condiciones de reunión o condiciones para enlazar múltiples tablas u objetos en la consulta especificada ya que pueden ser candidatos para definir estructuras de acceso.

Además de estas características debemos tomar en cuenta la frecuencia esperada de invocación para las consultas y transacciones, en general cuando el volumen de procesamiento es alto se aplica la regla informal “80-20”, la cual establece que apenas el 20 por ciento de las consultas y transacciones representa aproximadamente el 80 por ciento del procesamiento, por tanto no es necesario recabar datos de todas las consultas y transacciones que se realizan en la base de datos, sino tomar aquellas que se consideren importantes y que cubran este 20 por ciento mencionado. (DuBois, 2001).

3.5.2.2 Organización de ficheros

En este paso se determinará en qué tipo de ficheros se almacenará la base de datos, El objetivo de este paso es escoger la organización de ficheros más óptima para cada relación. Por ejemplo, un fichero desordenado es una buena estructura cuando se va a cargar gran cantidad de datos en una relación al inicializarla, cuando la relación tiene pocas tuplas, también cuando en cada acceso se deben obtener todas las tuplas de la relación, o cuando la relación tiene una estructura de acceso adicional, como puede ser un índice.

Por otra parte, los ficheros dispersos (hashing) son apropiados cuando se accede a las tuplas a través de los valores exactos de alguno de sus campos (condición de igualdad en el WHERE). Si la condición de búsqueda es distinta de la igualdad (búsqueda por rango, por patrón, etc.), la dispersión no es una buena opción.

Hay otras organizaciones, como la ISAM o los árboles B+, en general los SMBD relacionales utilizan arboles B+ para la indexación, ya que soportan tanto las consultas de igualdad como las de rango sobre el atributo utilizado como clave de búsqueda. Las organizaciones de ficheros elegidas deben quedar especificadas, justificando en cada caso la opción escogida.

3.5.2.3 Crear índices secundarios

La creación de índices secundarios permite especificar caminos de acceso adicionales para las relaciones base. Por ejemplo, la relación ALUMNOS se puede haber almacenado en un fichero disperso a través del atributo *id_alumno*. Si se accede a menudo a esta relación a través del atributo *nom_alumno*, se puede plantear la creación de un índice sobre dicho atributo para favorecer estos accesos. Pero hay que tener en cuenta que estos índices conllevan un coste de mantenimiento que hay que sopesar frente a la ganancia en prestaciones. A la hora de seleccionar los índices, se pueden seguir las siguientes indicaciones:

- Construir un índice por tabla, puede ser sobre la clave primaria (llamado índice primario) o algún otro atributo de cada relación base (llamado índice de agrupación).
- No crear índices sobre relaciones pequeñas.
- Añadir un índice sobre los atributos que se utilizan para acceder con mucha frecuencia.
- Añadir un índice sobre las claves ajenas que se utilicen con frecuencia para hacer joins.
- Evitar los índices sobre atributos que se modifican a menudo.
- Evitar los índices sobre atributos poco selectivos (aquellos en los que la consulta selecciona una porción significativa de la relación).
- Evitar los índices sobre atributos formados por tiras de caracteres largas.

Los índices creados se deben documentar, explicando las razones de su elección.

3.5.2.4 Redundancia controlada

En ocasiones puede haber dilemas serios cuando el desempeño de las bases de datos no es satisfactorio, a pesar de que ya se ha hecho todo lo posible en cuestión de diseño físico, por tanto es conveniente introducir redundancia de forma controlada, con objeto de mejorar el sistema. En la etapa del diseño lógico se recomienda llegar al menos, hasta la tercera forma normal para obtener un esquema con una estructura consistente y sin redundancias y es a partir de ellas que se puede tomar la decisión de eliminar las uniones durante la corrida, recombinando tablas, haciendo que el sistema sea utilizable, esta técnica se conoce como desnormalización.

Pero al realizarla, hay que tener en cuenta los siguientes factores:

- La desnormalización hace que la implementación sea más compleja.
- La desnormalización hace que se sacrifique la flexibilidad.
- La desnormalización puede hacer que los accesos a datos sean más rápidos, pero hace más lentas las actualizaciones.

No se pueden establecer una serie de reglas que determinen cuándo desnormalizar relaciones, pero hay algunas situaciones muy comunes en donde puede considerarse esta posibilidad:

- *Combinar relaciones de uno a uno:* Cuando hay relaciones (tablas) involucradas en relaciones de uno a uno, se accede a ellas de manera conjunta con frecuencia y casi no se les accede separadamente, se pueden combinar en una sola relación (tabla).

- *Duplicar atributos no clave en relaciones de uno a muchos para reducir los joins.* Para evitar operaciones de join, se pueden incluir atributos de la relación (tabla) padre en la relación (tabla) hijo de las relaciones de uno a muchos.

- *Tablas de referencia.* Las tablas de referencia (*lookup*) son listas de valores, cada uno de los cuales tiene un código. Por ejemplo puede haber una tabla de referencia para los tipos de inmueble, con las descripciones de estos tipos y un código asociado. Este tipo de tablas son un caso de relación de uno a muchos. Si las tablas de referencia se utilizan a menudo en consultas críticas, se puede considerar la introducción de la descripción junto con el código en la relación (tabla) hijo, manteniendo la tabla de referencia para validación de datos.

- *Duplicar claves ajenas en relaciones de uno a muchos para reducir los joins.* Para evitar operaciones de join, se pueden incluir claves ajenas de una relación (tabla) en otra relación (tabla) con la que se relaciona (habrá que tener en cuenta ciertas restricciones).

- *Duplicar atributos en relaciones de muchos a muchos para reducir los joins.* Durante el diseño lógico se eliminan las relaciones de muchos a muchos introduciendo dos relaciones de uno a muchos. Esto hace que aparezca una nueva relación (tabla) intermedia, de modo que si se quiere obtener la información de la relación de muchos a muchos, se tiene que realizar el join de tres relaciones (tablas). Para evitar algunos de estos joins se pueden incluir algunos de los atributos de las relaciones (tablas) originales en la relación (tabla) intermedia.

- *Introducir grupos repetitivos.* Los grupos repetitivos se eliminan en el primer paso de la normalización para conseguir la primera forma normal. Estos grupos se eliminan introduciendo una nueva relación (tabla), generando una relación de uno a muchos. A veces, puede ser conveniente reintroducir los grupos repetitivos para mejorar las prestaciones.

Todas las redundancias que se introduzcan en este paso se deben documentar y razonar. El esquema lógico se debe actualizar para reflejar los cambios introducidos.

3.5.2.5 Espacio en disco

Existen algunas características del hardware que son importantes tomar en cuenta para el buen funcionamiento de la base de datos, como la velocidad del procesador y las tasas de transferencia de los datos en el disco, ya que están asociadas con el diseño físico, y en pocas palabras si el hardware es más veloz será capaz de soportar mejor todos aquellos cambios que se realicen en el diseño físico.

Si se tiene que adquirir nuevo hardware, se debe hacer una estimación del espacio necesario en disco para soportar la base de datos, esto depende básicamente del SMBD que se esté utilizando, además de considerar el número de tuplas de cada relación y su tamaño, así como el factor de crecimiento de cada relación (L. Gillenson, 2006).

3.5.2.6 Elementos para controlar la seguridad

Uno de los componentes más importantes y el activo de una empresa, organización, u otro, son los datos, y mantenerlos seguros es de vital importancia para ellos, en la etapa anterior de diseño lógico se definieron las especificaciones en cuanto a seguridad ahora lo que corresponde es implementarlos, esto dependerá de las capacidades y posibilidades que ofrezca el SMBD que se esté utilizando.

Algunos elementos que se pueden implementar para la seguridad de los datos pueden ser:

- *Crear vistas de los usuarios:* el objetivo de esta implementación es preservar la seguridad, mejorando la independencia de los datos y reduciendo la complejidad permitiendo que los usuarios vean solamente los datos que requieren.

- *Crear reglas de acceso:* el administrador de la base de datos asignará a cada usuario un password (palabra secreta) el cual le permitirá tener acceso y permiso para realizar acciones sobre los objetos de la base de datos como consultar, actualizar, eliminar o insertar. De esta manera se especificará quién o quienes tienen permisos a los objetos de la base de datos y que privilegios tienen sobre ellos.

3.5.2.7 Monitoreo del sistema

Una vez que queda implementado el diseño físico es el momento de hacer la carga de datos y poner en funcionamiento la base de datos para observar como es su rendimiento y si realiza lo que se planeo. Si no cubre las necesidades requeridas hay que cambiar el esquema para poder satisfacerlas, el sistema no permanecerá estático cambiará conforme lo requiera la empresa o los usuarios, por tanto el SMBD debe proporcionar las herramientas necesaria para monitorearlo mientras se encuentra en funcionamiento.

C A P I T U L O II

1. Sistema Manejador de Base de Datos Relacional *MySQL*

1.1 Introducción

Un sistema manejador de base de datos relacional (en inglés, RDBMS, *Relational Database Management System*) es una herramienta necesaria en casi todos los entornos, desde los negocios hasta las búsquedas en internet, almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén, esto añade velocidad y flexibilidad, pero no todos tienen los recursos financieros para acceder a ellas. Históricamente este software ha sido caro, debido a su mantenimiento y asesoría especializada, elevándose su costo si estos motores requieren de un hardware especializado para su buen funcionamiento.

Pero a través de los años el software de base de datos se ha vuelto más accesible y han llegado a ser disponibles de forma gratuita o a un bajo costo. Una de las entradas más recientes en el terreno de las bases de datos de coste mínimo es MySQL, un sistema gestor de bases de datos relacionales cliente-servidor SQL originario de Escandinavia. MySQL incluye un servidor SQL (Structured Query Language) que es el lenguaje estandarizado más común para acceder a bases de datos y está definido por el estándar ANSI/ISO SQL, muy rápido, multi-threaded, multiusuario y robusto, programas cliente con los cuales acceder al servidor, herramientas administrativas y una interfaz de programación para escribir programas propios. Funciona en muchas plataformas y está disponible en forma binaria como en código fuente. (DuBois, 2001).

MySQL es una marca registrada de MySQL AB, tiene una doble licencia, por lo tanto los usuarios pueden elegir entre usar el software MySQL como un producto Open Source bajo los términos de la licencia GNU General Public License o pueden adquirir una licencia comercial estándar de MySQL AB. Open Source significa que es posible para cualquiera usar y modificar el software. Cualquiera puede bajar el software MySQL desde internet y usarlo sin pagar nada, puede estudiarse el código fuente y cambiarlo para adaptarlo a las necesidades de cada usuario. Se ejecuta en computadoras personales, pero es gracias a su portabilidad que puede ejecutarse también en sistemas operativos comerciales como Solaris, Irix y Windows y en hardware de hasta servidores empresariales, gestionando grandes bases de datos con millones de registros. (ORACLE, 2011)

El servidor MySQL está diseñado para entornos de producción críticos, con alta carga de trabajo así como para integrarse en software para ser distribuido.

1.2 Características de MySQL

Algunas características importantes de porque MySQL es una excelente opción como el manejador para nuestra base de datos son:

- Ser la mejor y más usada base de datos en el mundo.
- Estar disponible y ser comprable por cualquiera.
- Fácil de usar.
- Se puede mejorar continuamente, entre tanto es rápido y seguro.
- Ser divertido de usar y mejorar.
- Libre de errores.
- Trabaja en entornos cliente/servidor o incrustados.
- Es portable, funciona en diferentes plataformas y compiladores.
- Está escrito en C y C++.
- Puede usarse fácilmente en múltiples CPU´s si están disponibles.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Uso de tablas en disco B-tree (MyISAM) rápidas con compresión de índice.
- Joins muy rápidos usando multi-join de un paso optimizado.
- Uso de tablas hash en memoria, usadas como tablas temporales.
- El servidor está disponible como un programa separado para usarse en un entorno de red cliente/servidor.
- Incluye un extenso Manual de Referencia (que sigue creciendo).
- Existe una lista de correo donde se encontrarán muchos participantes que pueden ayudar en las dudas o necesidades de los usuarios.

Existen muchas otras, todas importantes y decisivas para elegirlo como el gestor de una base de datos. Otro de los aspectos que debemos tener en cuenta es la terminología de la base de datos, del lenguaje de consulta y de la arquitectura de MySQL. (ORACLE, 2011).

1.3 Terminología elemental de la base de datos

1.3.1 Terminología estructural

Muchos de estos términos se han mencionado a lo largo del capitulado de este trabajo, por tanto solo recordaremos algunos conceptos básicos e importantes de las bases de datos:

- Tablas: conjunto de filas y columnas.
- Registro: cada fila de una tabla.
- Atributos: columnas de una tabla que contienen diversas partes de información.

El sistema de administración es el software que permite usar los datos para poder insertar, recuperar, modificar o borrar registros. En palabras sencillas podemos definir el concepto relacional como una simple operación de correspondencia donde se relaciona una tabla con otra haciendo coincidir los valores de las filas de una con los de otra.

1.3.2 Terminología de lenguaje de consulta

La comunicación entre MySQL y el usuario será el lenguaje de consulta SQL (en inglés: *Structured Query Language*, Lenguaje de Consulta Estructurado). SQL incluye muchas sentencias, todas diseñadas para hacer posible la interacción con su base de datos de modo interesante y útil. Por ejemplo para crear una tabla se indica algo como esto:

```
CREATE TABLE paises_direc
(
idpa_dir INT(2) NOT NULL auto_increment,
nompais VARCHAR(20) not null,
PRIMARY KEY (idpa_dir)
);
```

Esta en solo una de tantas estructuras que se pueden crear en MySQL, más adelante se explicará cada una de ellas.

1.3.3 Terminología de la arquitectura de MySQL

Cuando se usa MySQL, en realidad se están utilizando dos programas, porque MySQL opera bajo la arquitectura cliente-servidor.

- Servidor: es el programa situado en la maquina en el que se almacenan los datos, se encarga de escuchar las peticiones del cliente cuando entran en la red y accede a la base de datos para extraer la información que solicitan.

- Cliente: son los programas que se conectan con el servidor de la base de datos y emiten consultas para indicar la información que quieren.

El que más se usa es *mysql*, un cliente interactivo que permite emitir consultas y ver resultados. Existen otros que incluyen *mysqldump* y *mysqlimport*, que vuelcan los contenidos de la tabla en un archivo y viceversa, y *mysqladmin*, que permite verificar el estado del servidor y realiza tareas administrativas como el cierre del servidor.

1.4 Requisitos preliminares

1.4.1 Instalación de MySQL

Para poder empezar a trabajar en MySQL debemos instalarlo en la maquina donde se vaya a trabajar con él, y tener acceso a los clientes y a algunos de sus servidores. Antes de instalar MySQL, se debe hacer lo siguiente:

1. Determinar si la plataforma donde se desea hacer la instalación está soportada.
2. Elegir la distribución que se instalará.
3. Descargar la distribución que se desea instalar y verificar su integridad.

En relación al punto 1, se muestran una lista de las plataformas en las cuales esta soportado MySQL:

- AIX 4.x, 5.x con subprocesos nativos.
- Amiga.
- BSDI 2.x with con el paquete MIT-pthreads.
- BSDI 3.0, 3.1 y 4.x con subprocesos nativos.
- Digital Unix 4.x con subprocesos nativos.
- FreeBSD 2.x con el paquete MIT-pthreads.
- FreeBSD 3.x and 4.x con subprocesos nativos.
- FreeBSD 4.x con LinuxThreads.
- HP-UX 10.20 con el paquete DCE threads o MIT-pthreads.
- HP-UX 11.x con subprocesos nativos.
- Linux 2.0+ con LinuxThreads 0.7.1+ o glibc 2.0.7+ para varias arquitecturas de CPU.
- Mac OS X.
- NetBSD 1.3/1.4 Intel y NetBSD 1.3 Alpha (requiere GNU make).
- Novell NetWare 6.0.
- OpenBSD > 2.5 con subprocesos nativos. OpenBSD < 2.5 con el paquete MIT-pthreads.
- OS/2 Warp 3, FixPack 29 y OS/2 Warp 4, FixPack 4.
- SCO OpenServer 5.0.X con una versión del paquete FSU Pthreads recientemente portada.
- SCO UnixWare 7.1.x.
- SCO Openserver 6.0.x.
- SGI Irix 6.x con subprocesos nativos.
- Solaris 2.5 y posteriores con subprocesos nativos en SPARC y x86.
- SunOS 4.x con el paquete MIT-pthreads package.
- Tru64 Unix.
- Windows 9x, Me, NT, 2000, XP, y 2003.

En base a algunas investigaciones realizadas, las mejores plataformas para ejecutar MySQL en este momento son x86 con SuSE Linux (kernel versión 2.4 o 2.6), y ReiserFS (o cualquier distribución de Linux similar) y SPARC con Solaris (2.7-9). También las otras plataformas donde MySQL se compila y ejecuta en la actualidad podrían ser incluidas en la categoría principal, pero no con el mismo nivel de estabilidad y rendimiento. (López Quijado, 2001).

Con respecto al punto 2, lo primero que debemos considerar para tomar una decisión es si desea emplear una entrega "en producción" (estable) o una entrega de desarrollo. En el proceso de desarrollo de MySQL coexisten múltiples entregas, cada una con un diferente estado de madurez:

- MySQL 5.1 es la próxima serie de entregas de desarrollo, y en ella se implementarán las nuevas características. En breve se pondrán a disposición de los usuarios interesados en hacer pruebas integrales las entregas Alfa.

- MySQL 5.0 es la serie de entregas estables (para producción). Solamente se liberan nuevas entregas para corrección de errores, no se añaden nuevas características que pudieran afectar a la estabilidad.

- MySQL 4.1 es la anterior serie de entregas estables (para producción). Se liberarán nuevas entregas para solucionar problemas de seguridad o errores críticos. En esta serie no se agregarán nuevas características de importancia.

- MySQL 4.0 y 3.23 son las antiguas series de entregas estables (para producción). Estas versiones están discontinuadas, de modo que solamente se liberarán nuevas entregas para solucionar errores de seguridad extremadamente críticos. (ORACLE, 2011).

Una vez elegida la versión hay que pasar al punto 3, instalar MySQL en la plataforma correspondiente, para poder crear las estructuras de la base de datos y todos los elementos necesarios para administrarla. Tomando en cuenta características propias de la organización para la cual se realizo el proyecto, la instalación se realizara en Windows 2007. Se debe confirmar la instalación de MySQL para estar seguros de que se crearon los directorios por el instalador binario y por las distribuciones de código fuente.

Para la confirmación se abre una consola y se teclea el nombre de usuario con todos los privilegios (root) determinado por default en el momento de la instalación de MySQL, introduciendo la contraseña establecida en la instalación y el sistema tiene que mostrar una pantalla como la de la figura 1.

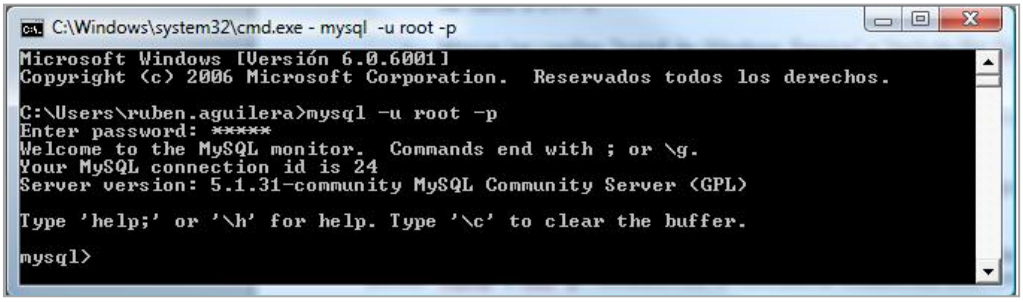


Figura 1. Instalación de MySQL en Windows.

1.4.2 Cuentas de usuarios MySQL

Cuando se instala MySQL, las tablas de permisos se inicializan con un conjunto inicial de cuentas. Las cuales tienen nombres y privilegios de acceso descritos. Las tablas de permisos definen las cuentas de usuario MySQL iniciales y sus permisos de acceso. Las cuentas que se crean son las siguientes:

- Dos cuentas de usuario root: Son cuentas de superusuario, es decir tienen el privilegio de hacer cualquier tarea, pero no tienen contraseñas, por lo que en el momento de la instalación se solicita una contraseña para poder garantizar la seguridad de la base de datos, de otra forma cualquier persona puede conectarse al servidor MySQL como root y obtener todos los privilegios.

- En Windows con la cuenta root se puede realizar la conexión desde el ordenador local (localhost) y la otra permite conectarse desde cualquier ordenador.

- Dos cuentas de usuario anónimo: las cuales no tienen nombre de usuario. Y no tienen contraseña, de modo que cualquier persona puede usarlos para conectarse al servidor MySQL.

- En Windows, una cuenta anónima es para conexiones desde el ordenador local. Tiene todos los privilegios, exactamente como la cuenta root. La otra sirve para conectarse desde cualquier ordenador y tiene todos los permisos sobre la base de datos test u otras cuyo nombre comience con test.

Cabe mencionar que se deben proteger ambas cuentas de usuario con contraseñas. En ambos casos, hay que asegurarse de cifrar el password utilizando la función PASSWORD(). O en su defecto eliminar las cuentas anónimas para no tener accesos inesperados.

Por tanto para cambiar las cuentas anónimas y root se puede hacer lo siguiente:

Mysql> SET PASSWORD FOR “@’localhost’=PASSWORD (‘nuevopassword’); → Cuenta anónima

Mysql> SET PASSWORD FOR “root@’localhost’=PASSWORD (‘nuevopassword’); → Cuenta root

La otra forma es mediante la sentencia UPDATE para modificar directamente la table user. Para realizar esta acción debe acceder al servidor como root y emitir la siguiente sentencia:

Shell> mysql –u root

Mysql> UPDATE mysql.user SET PASSWORD = PASSWORD (‘nuevopassword’) WHERE User=’’;

Mysql> FLUSH PRIVILEGES; → Cuenta anónima

Mysql> UPDATE mysql.user SET PASSWORD = PASSWORD (‘nuevopassword’)

WHERE User=’root’;

Mysql> FLUSH PRIVILEGES; → Cuenta root

Luego de actualizar las contraseñas directamente en la tabla user empleando UPDATE, se le debe indicar al servidor que relea las tablas de privilegios con FLUSH PRIVILEGES. De otro modo, los cambios no tendrán efecto hasta que se reinicie el servidor.

Otra opción es eliminar las cuentas anónimas, con la siguiente sentencia:

Shell> mysql –u root

Mysql> DELETE FROM mysql.user WHERE User=’’;

Mysql> FLUSH PRIVILEGES;

De esta manera se reforzará la seguridad y el acceso a la base de datos. Así como se crean y modifican cuentas de usuario, uno de los privilegios que tienen la cuenta root es la creación de nuevas cuentas de usuario, utilizando la sentencia GRANT para crear y REVOKE para eliminar cuentas de usuarios.

Además del software de MySQL se necesitan los permisos para crear la base de datos y sus tablas, estos permisos los debe otorgar el administrador de MySQL (root), por medio del siguiente comando:

GRANT ALL ON colegio_bd.* TO laura@localhost IDENTIFIED BY "secreto"

Este comando lo que permite es el acceso completo a la base de datos *colegio_bd* y a sus tablas al usuario *laura* cuando se conecta desde el localhost (la misma máquina donde se está ejecutando el servidor), asignándole una contraseña para acceder llamada "secreto".

Como vimos anteriormente en el momento de confirmar la instalación de MySQL, para conectarse a la base de datos lo podemos hacer mediante el siguiente comando:

mysql –h *host_name* –u *user name* –p
Enter password: ******* (tecleará la contraseña la cual no se reflejará en pantalla)

En caso de tener un nombre de usuario y una contraseña el comando sería el siguiente:

mysql –u *laura* –p
Enter password: ******* (secreto)

Para quitar o eliminar una cuenta de usuario con sus privilegios se utiliza la sentencia DROP USER:

DROP USER user;

Los comandos GRANT y REVOKE permiten a los administradores de sistemas crear cuentas de usuario MySQL, darles permisos y quitarlos de las cuentas. Los permisos pueden darse en varios niveles:

- Nivel global

  Los permisos globales se aplican a todas las bases de datos de un servidor dado. Estos permisos se almacenan en la tabla mysql.user. GRANT ALL ON *.* y REVOKE ALL ON *.* otorgan y quitan sólo permisos globales.

- Nivel de base de datos

  Los permisos de base de datos se aplican a todos los objetos en una base de datos dada. Estos permisos se almacenan en las tablas mysql.db y mysql.host. GRANT ALL ON db_name.* y REVOKE ALL ON db_name.* otorgan y quitan sólo permisos de bases de datos.

- Nivel de tabla

  Los permisos de tabla se aplican a todas las columnas en una tabla dada. Estos permisos se almacenan en la tabla mysql.tables_priv . GRANT ALL ON db_name.tbl_name y REVOKE ALL ON db_name.tbl_name, otorgan y quitan permisos sólo de la tabla.

- Nivel de columna

  Los permisos de columna se aplican a columnas en una tabla dada. Estos permisos se almacenan en la tabla mysql.columns_priv . Usando REVOKE, se deben especificar las mismas columnas a las que se otorgaron los permisos.

Una vez conectados podemos consultar las bases de datos creadas o existentes en el manejador, ó crear una propia con sus tablas y datos. Para esto debemos conocer la sintaxis de las sentencias que nos permitirán realizarlo.

1.5 Lenguaje Estructurado de Consulta SQL

Es un lenguaje para acceder a la información almacenada en Bases de Datos relacionales. SQL fue la evolución de SEQUEL, que surgió como solución al concepto formulado por el Sr. Codd de Bases de Datos Relacionales en 1970. Permite expresar operaciones aritméticas, combinatorias y lógicas (entre otras) con los datos almacenados. El lenguaje SQL nos sirve para:

- Consultar y actualizar datos almacenados.
- Definir y destruir objetos de la Base de Datos (generalmente un ABD).
- Conceder y denegar autorizaciones para usar esos objetos (generalmente un ABD).

1.5.1. Componentes de una sentencia SQL

- Palabras predefinidas (SELECT, FROM, WHERE, INTO,...)
- Nombres de tablas y columnas.
- Constantes (numéricas y alfanuméricas).
- Signos delimitadores.

Por ejemplo:

```
SELECT  nom_alumno, edad_alumno  FROM alumnos WHERE id_alumno = 15
```

Palabras predefinidas: SELECT, FROM, WHERE.
Nombres de tablas y columnas: nom_alumno, edad_alumno, alumnos, id_alumno.
Constantes: 15.
Signos delimitadores: = .

1.5.2. Tipos de sentencias SQL

1. Sentencias LMD (en inglés, Data Manipulation Language, abreviado DML)

SELECT: sentencia de consulta.
INSERT: sentencia de inserción de datos.
UPDATE: sentencia para actualizar datos.
DELETE: sentencia para borrar datos.

2. Sentencias LDD (en inglés, Data Definition Language, abreviado DDL)

CREATE: sentencia para la creación de objetos de la base de datos.
DROP: sentencia para la eliminación de objetos de la base de datos.

3. Sentencias LCD (en inglés, Data Control Language, abreviado DCL)

GRANT: sentencia para otorgar permisos y privilegios.
REVOKE: sentencia que revoca los permisos y privilegios otorgados.

1.5.3. Tipos de Datos

Prácticamente todo MySQL tiene que ver con datos, ya que el propósito del sistema de administración de una base de datos, es por definición administrar datos. Por tanto hay 3 grandes tipos de datos:

1. Numéricos (enteros, decimales, en coma flotante).
2. Alfanuméricos (cadena de caracteres).
3. De tiempo (fecha y hora).

Cada uno de los cuales tiene un tipo específico para MySQL, por ejemplo 45,8 es un número y "Eduardo" es una cadena. Estos tipos de datos se declaran cuando se emite la sentencia CREATE TABLE en donde se declara el tipo de cada columna como parte de la tabla. Por ejemplo:

```
CREATE TABLE mi_tabla
        ( Int_col INT,          /* columna evaluada como entero*/
         Str_col CHAR(20),      /*columna evaluada como cadena*/
        Date_col DATE)          /*columna evaluada como fecha*/
```

En la tabla 1 se muestran los tipos de datos que soporta MySQL:

TIPOS NUMÉRICOS	
Nombre del tipo	Significado
TINYINT	Entero muy pequeño (1 byte).
SMALLINT	Entero pequeño (2 bytes).
MEDIUMINT	Entero mediano (3 bytes).
INT	Entero estándar (4 bytes).
BIGINT	Entera larga (8 bytes).
DOUBLE	Numero doble de precisión de coma flotante (8 bytes).
DECIMAL	Numero de coma flotante, representado como una cadena.
FLOAT	Numero único de precisión de coma flotante (4 bytes).
TIPOS CARÁCTER (CADENA)	
CHAR	Cadena de caracteres de longitud fija.
VARCHAR	Cadena de caracteres de longitud variable.
TINYBLOB	Objeto Binario Largo muy pequeño.
BLOB	Objeto Binario Largo pequeño.
MEDIUMBLOB	Objeto Binario Largo mediano.
LONGBLOB	Objeto Binario Largo.
TINYTEXT	Cadena de texto muy pequeña.
TEXT	Cadena de texto pequeña.
MEDIUMTEXT	Cadena de texto mediana.
LONGTEXT	Cadena de texto larga.
ENUM	Una enumeración; columnas a las que se puede asignar un miembro de numeración.
SET	Un conjunto; columnas a las que se pueden asignar múltiples conjuntos de miembros.

TIPO DE FECHA Y TIEMPO	
DATE	Valor de fecha en formato AAAA-MM-DD
TIME	Valor de hora en formato hh:mm:ss
DATETIME	Valor de fecha y hora en fomato AAAA-MM-DD hh:mm:ss
TIMESTAMP	Valor de lapso de tiempo en formato AAAAMMDDhhmmss
YEAR	Valor de año en formato AAAA

Tabla 1. Tipos de datos que soporta MySQL.

Estos son los tipos de datos que podemos utilizar en las sentencias de SQL, a partir de ellos podemos definir los tipos de columna que se almacenarán en las tablas de la base de datos. Una vez que tenemos una visión general de los requisitos para crear la base de datos, se procede a crearla.

1.6 Creación, eliminación y selección de la base de datos

1.6.1 Creación de una base de datos

Usar una base de datos incluye varios pasos:

- Crear (inicializar) la base de datos.
- Crear las tablas de la base de datos.
- Manipular las tablas, insertando, recuperando, modificando o borrando datos.

Para crear una base de datos nueva, se debe realizar la conexión con el servidor usando *mysql* y luego emitir la sentencia CREATE DATABASE que especifica el nombre de la base de datos, el cual no debe existir ya, y tener los privilegios necesarios para crearla.

```
mysql> CREATE DATABASE escuela
```

1.6.2 Eliminación de una base de datos

Eliminar una base de datos es tan fácil como crearla, pero hay que tener en cuenta que cuando se elimina también desaparecen todas las tablas que contiene, por eso es de cuidado la ejecución de la sentencia que realiza esta acción, ya que una vez eliminada la base de datos no se podrá recuperar, solo si el administrador de la base tiene un respaldo de ella.

La sentencia para eliminar una base de datos es:

```
mysql> DROP DATABASE escuela
```

1.6.3 Selección de una base de datos

Para seleccionar una base de datos y convertirla en la base de datos por defecto (actual) para una conexión determinada a un servidor, se ejecuta la siguiente sentencia:

```
mysql> USE escuela
```

El seleccionar una base de datos no significa que no se puedan usar las tablas de otra base ya existente, se puede hacer utilizando esta sentencia cuantas veces sea necesario para determinar qué base de datos se quiere manipular en ese momento.

1.7 Creación, eliminación e indexación y de tablas

1.7.1 Creación de tablas

La creación de tablas se realiza mediante la sentencia CREATE TABLE la cual especifica como mínimo el nombre de la tabla y la lista de sus columnas, adicionalmente se colocan las clausulas de restricción, integridad y claves, pero para un primer ejemplo, basta con escribir los requisitos mínimos, como lo muestra la siguiente sintaxis:

```
CREATE TABLE mi_tabla
        (nombre CHAR(20),
        edad INT NOT NULL,
        peso INT,
        sexo ENUM (´F´,´M´));
```

Cuando se crea una tabla se especifica el tipo de dato de cada una de sus columnas, se debe elegir el tipo de dato adecuado para no ocupar espacio innecesario en memoria, ya que de lo contrario no se almacenaran de manera óptima los datos y se destinará más espacio en memoria de lo necesario. Otro punto importante en el momento de la creación de tablas es el tipo de motor de almacenamiento que utilizará la tabla.

Un motor de almacenamiento es una parte esencial de un SMBD, se encarga de crear, recuperar, actualizar y borrar los datos de una base de datos. Su importancia radica en el modo en que se almacenan los datos y las ventajas que se puede obtener de cada uno de ellos. (DuBois, 2001).

Los motores de almacenamiento que tiene disponibles MySQL se muestran en la tabla 2.

Engine	Support	Comment
MyISAM	DEFAULT	Default engine as of MySQL 3.23 with great performance
MEMORY	YES	Hash based, stored in memory, useful for temporary tables
InnoDB	YES	Supports transactions, row-level locking, and foreign keys
BerkeleyDB	NO	Supports transactions and page-level locking
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)
EXAMPLE	NO	Example storage engine
ARCHIVE	YES	Archive storage engine
CSV	YES	CSV storage engine
ndbcluster	DISABLED	Clustered, fault-tolerant, memory-based tables
FEDERATED	YES	Federated MySQL storage engine
MRG_MYISAM	YES	Collection of identical MyISAM tables
ISAM	NO	Obsolete storage engine

Tabla 2. Motores de almacenamiento de MySQL.

La elección de algún motor de almacenamiento en especial depende de cuatro funcionalidades clave:

- Tipos de datos: aunque la mayoría son comunes hay algunos específicos que pueden ser decisivos bajo determinadas circunstancias.

- Bloqueo de datos: la forma en la que el motor protege un dato que está siendo modificado para evitar problemas de acceso concurrente a los datos y mantener la integridad referencial.

- Indexado: las diferentes técnicas de indexado influyen de manera drásticamente en el rendimiento de una base de datos.

- Transacciones: brindan fiabilidad a los datos mientras se realizan operaciones, permite utilizar los datos y sólo permite guardarlos cuando se comprueba que las otras condiciones que pudiesen requerirse se han cumplido.

Otra característica importante que debe considerarse es la denominada ACID, propiedad de una base de datos para realizar transacciones seguras. Así pues *ACID* define a un sistema de gestión de bases de datos que puede realizar transacciones seguras, es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español. Cada una de estas propiedades se describe a continuación:

- Atomicidad: es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias. Es decir, al realizar una operación se llevan a cabo varios pasos, entonces ocurren todos o ninguno.

- Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

- Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Asegurando que al realizarse dos transacciones sobre la misma información nunca generará ningún tipo de error.

- Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema. (DuBois, 2001).

1.7.1.1 Motores de almacenamiento

Los motores de almacenamiento que tiene MySQL para definir las tablas de sus bases de datos son:

1) *MyISAM*: Se basa en el antiguo ISAM, pero con muchas mejoras, es el motor que usa MySQL por defecto. Es una buena combinación entre funcionalidad y rendimiento aunque carece de algunas características interesantes.

   Características más importantes:

   - Límite de 2^32 registros
   - Máximo de 64 índices por tabla
   - Máximo de 16 columnas por índice
   - Los datos son independientes de la máquina y el sistema operativo
   - Permite campos índice como NULL
   - BLOB3 y TEXT pueden ser índices
   - Permite un gran tamaño en las tablas (hasta 256TB)
   - No soporta transacciones
   - Bloquea los datos a nivel de tabla
   - No permite “claves ajenas”

El potencial de este motor es la rapidez de las operaciones de lectura (consultas SELECT), razón por la que MySQL es tan popular en la web, ya que la mayoría de las operaciones que se realizan son de este tipo, además de no hacer comprobaciones de integridad referencial contribuyen a su velocidad.

```
CREATE TABLE pruebaMyISAM (
        codigo varchar(5) default NOT NULL,
        descripcion varchar(255) default NULL,
        PRIMARY KEY (codigo)
        ) ENGINE=MyISAM;
```

2) *MERGE*: Permite combinar varias tablas de igual estructura en una única tabla, pudiendo así realizar consultas sobre una tabla que nos devuelve datos de varias.

Características más importantes:

- Límite de 2^32 registros
- Las tablas "base" deben ser MyISAM
- Bloqueo a nivel de tabla
- No tiene índices, usa los de las tablas "base" (salvo FULLTEXT)
- La lectura es más lenta al tener que ir consultando la clave en cada una de las tablas subyacentes
- No permite REPLACE
- No soporta transacciones
- En su creación no comprueba que las tablas que usa existan y tengan una estructura idéntica

Una de sus funcionalidades puede ser, que parte de una tabla muy grande y otras más pequeñas y, al unirlas con MERGE, permitirnos trabajar con ellas como si fuesen una sola.

Ejemplo de creación de una tabla a partir de otras dos:

```
CREATE TABLE t1 (
        a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
         message CHAR(20));
CREATE TABLE t2 (
        a INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        message CHAR(20));
INSERT INTO t1 (message) VALUES ('Testing'),('table'),('t1');
INSERT INTO t2 (message) VALUES ('Testing'),('table'),('t2');
CREATE TABLE total (
        a INT NOT NULL AUTO_INCREMENT,
        message CHAR(20), INDEX(a))
        TYPE=MERGE UNION=(t1,t2) INSERT_METHOD=LAST;
```

3) *MEMORY (HEAP)*: Guarda todos los datos en memoria, de forma que si se cae el servidor o reiniciamos MySQL se pierden los datos, aunque la estructura de las tablas se guarda.

Características más importantes:

- Bloquea los datos a nivel de tabla
- Puede usar índices HASH
- No soporta BLOB ni TEXT
- No soporta transacciones
- Resulta extremadamente fácil perder los datos

Resultan útiles como tablas temporales para determinadas consultas ya que al estar en memoria y poder tener Hash como índice resultan extremadamente rápidas, es una buena elección cuando necesitamos realizar operaciones muy rápidas sobre conjuntos pequeños de datos.

4) *FEDERATED*: La novedad de este motor es que permite el acceso a una base de datos MySQL remota como si fuese local, en realidad tenemos una tabla local que representa a otra remota, ambas deben ser idénticas.

Características más importantes:

- Permite acceso a BBDD remotas
- MySQL no instala este motor por defecto
- No soporta transacciones
- No contempla el bloqueo de datos
- No permite ALTER

Ejemplo de tabla en la que se le indica la dirección de los datos:

```
CREATE TABLE federated_table (
        id int(20) NOT NULL auto_increment,
        name varchar(32) NOT NULL default '',
        other int(20) NOT NULL default '0',
        PRIMARY KEY (id),
        KEY name (name),
        KEY other_key (other)
        )
        ENGINE=FEDERATED
        DEFAULT CHARSET=latin1
        COMMENT='mysql://root@remote_host:9306/federated/test_table';
```

5) *ARCHIVE*: Se utiliza básicamente para almacenar grandes cantidades de datos sin índices en muy poco espacio, ya que los comprime con zlib alcanzando un nivel de ahorro de espacio considerable.

Características más importantes:

- Gran compresión de los datos
- Sólo permite INSERTS y SELECTS
- Bloquea los datos a nivel de registro
- Almacena los datos en un buffer hasta que los comprime e inserta
- No soporta transacciones

Este motor resulta especialmente útil para el almacenamiento de históricos o logs ya que suelen ocupar gran cantidad de espacio y no es necesario modificarlos con posterioridad.

6) *CSV*: Almacena la información utilizando el formato de valores separados por comas (comma-separated values), de forma que cada tabla es un fichero que contiene los datos. No soporta indexado y su fin principal es permitir exportar los datos de forma que puedan ser importados fácilmente por algunas suites ofimáticas.

Características más importantes:

- Útil para exportar e importar datos
- No soporta indexación ni transacciones

7) *BLACKHOLE*: El sorprendente uso de este motor es no almacenar los datos sino crear un log con la consulta SQL utilizada. Como no almacena ningún dato lógicamente no soporta índices, ni transacciones. Su principal utilidad es mantener un servidor esclavo que mantenga un log del sistema principal.

8) *InnoDB*: Está considerado como uno de los motores más avanzados para el almacenamiento de datos en MySQL. Tiene un soporte completo de transacciones (es ACID compliant), permite el bloqueo de datos a nivel de registro permitiendo gran flexibilidad a la hora de utilizar las tablas, controla la integridad referencial, permite claves ajenas y tiene un sistema de recuperación de caídas. No obstante su potencia radica en su mecanismo de indexación y cache de los registros pues mantiene una caché de índices y datos en memoria y en disco proporcionando un muy alto rendimiento.

Características más importantes:

- ACID compliant
- Permite claves ajenas y transacciones, soporte de integridad referencial
- Bloqueo de datos a nivel de registro y no bloquea la lectura durante los selects (mejora la concurrencia)
- Sistema de recuperación de caídas
- Cambiar la ubicación de la base de datos/tabla es complicado
- Una tabla no puede tener más de 1000 columnas
- El tamaño de sus logs debe ser inferior a 4GB
- El tamaño máximo para una tabla es de 64TB
- No permite índices de FULLTEXT
- No mantiene una contador interno de registros (select count(*) from tabla) ya que es lento al tener recorrer todo el índice)

Ejemplo de uso de transacciones:

```
CREATE TABLE CUSTOMER (A INT, B CHAR (20), INDEX (A))
ENGINE=InnoDB;
Query OK, 0 rows affected (0.00 sec)
BEGIN;
Query OK, 0 rows affected (0.00 sec)
        INSERT INTO CUSTOMER VALUES (10, 'Heikki');
        Query OK, 1 row affected (0.00 sec)
COMMIT;
Query OK, 0 rows affected (0.00 sec)
SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)
        INSERT INTO CUSTOMER VALUES (15, 'John');
        Query OK, 1 row affected (0.00 sec)
ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
SELECT * FROM CUSTOMER;
```

De acuerdo a las necesidades de cada usuario se determina que motor de almacenamiento es el más adecuado para el tipo de datos que contendrá la tabla. Teniendo en cuenta que lo que se busca es hacer más eficiente el sistema. (Hinz, Lead, DuBois, Stephens, Bedford, & Russell, 2010).

1.7.2 Eliminación de tablas

Eliminar una tabla es mucho más fácil que crearla, ya que solamente se menciona la tabla sin necesidad de especificar su contenido. La sentencia que realiza esta acción es la siguiente:

DROP TABLE nombre_tabla

Se pueden eliminar varias tablas con la misma sentencia, únicamente hay que mencionarlas, separándolas con comas.

DROP TABLE nombre_tabla1, nombre_tabla2, nombre_tabla3,

1.7.3 Creación y eliminación de índices

Los índices son la mejor herramienta para aumentar la velocidad de acceso a los contenidos de las tablas, en particular cuando se hacen consultas que impliquen uniones entre varias de ellas.

Existen cuatro tipos de índices que podemos utilizar en MySQL: de clave primaria, únicos, de texto completo, y ordinarios.

1) *Índice primario*: estos índices se crean al definir en la creación de las tablas una clave primaria, estas claves primarias no pueden ser valores nulos, por tanto siempre que se crea una PRIMARY KEY se crea un índice primario. Se puede crear más de un índice, solo hay que separarlos por comas. La siguiente sintaxis muestra la creación de un índice:

   Podemos crear el índice cuando se crea la tabla, usando la palabra INDEX seguida del nombre del índice y columnas a indexar.

   INDEX nombre_indice (columna_indexada, columna_indexada2,…)

2) *Índice único*: es aquel que no permite almacenar dos valores iguales. Y su sintaxis es la siguiente:

   CREATE UNIQUE INDEX nombre_indice ON nombre_tabla (lista_columnas)

3) *Índices de texto completo*: estos índices permiten realizar búsquedas de palabras, y se pueden crear sobre columnas tipo CHAR, VARCHAR o TEXT. Una vez creado puedes hacer búsquedas del tipo:

   SELECT * FROM nombre_tabla WHERE MATCH (nombre_indice_fulltext) AGAINST (´palabra_a_buscar');

   Este tipo de índices tiene algunas limitaciones: solo busca palabras completas, no indexa palabras de menos de cuatro letras, no indexa columnas de menos de tres filas, ni palabras que aparezcan en la mitad o más de las filas. Las palabras separadas por guiones se cuentan como dos palabras.

4) *Índices ordinarios*: estos índices no tienen restricciones en cuanto a valores idénticos o nulos, si se crean este tipo de índices sobre columnas CHAR y VARCHAR se puede delimitar el numero de caracteres sobre los que queremos indexar, indicándose entre paréntesis después del nombre de la columna:

   ALTER TABLE libros ADD INDEX idx_autor (nombre(10), apellidos (10));

Al crear índices debemos tener presente algunas consideraciones para poder utilizarlos de manera óptima y no dañar las transacciones, los índices se actualizan cada vez que se modifica la columna o columnas que utiliza. Por ello no es aconsejable usar índices en columnas que serán de uso frecuente en operaciones de escritura (INSERT, UPDATE, DELETE).

Tampoco tendría sentido crear índices sobre columnas cuando cualquier SELECT sobre ellos va a devolver una gran cantidad de resultados; por ejemplo una columna booleana que admita los valores Yes/No. En fin, tampoco es necesario usar índices en tablas demasiado pequeñas, ya que en estos casos no hay ganancia de rapidez frente a una consulta normal. Los índices ocupan espacio, incluso más que la tabla de datos, por tanto es importante conocer cuáles serían las ventajas o desventajas de crearlos. (DuBois, 2001).

1.8 Consultas

1.8.1 Recuperación de registros

Una de las principales funciones en una base de datos es la recuperación de registros, y se hace mediante la sentencia SELECT, una de las más utilizadas en el lenguaje SQL, la sintaxis básica de la esta sentencia es la siguiente:

SELECT lista_selección	Que columnas elegir.
FROM lista_tablas	De dónde seleccionar filas.
WHERE restricción_primaria	Que condiciones deben cumplir las filas.
GROUP BY grupo_columnas	Como se agruparán los resultados.
ORDER BY columnas_ordenar	Como clasificar los resultados.
HAVING restricción_secundaria	Condiciones secundarias que deben cumplir las filas.
LIMIT contador	Limitar los resultados.

La sintaxis básica para la recuperación de registros de una tabla es:

SELECT nombre_columna FROM nombre_tabla WHERE condición_primaria;

Las demás clausulas pueden ser opcionales dependiendo del tipo especifico de datos que se quieran recuperar, se verán algunos ejemplos de consultas más comunes, tomando en cuenta que este tema no es el punto central del desarrollo del presente trabajo, pero de igual forma es importante tener conocimiento acerca de cómo se realizan las consultas en MySQL.

Algunas de las consultas más utilizadas son las que combinan más de una tabla, partiremos de las más sencillas para entender la metodología, posteriormente cada usuario debe adaptar todos los elementos con los que cuenta MySQL para recuperar lo que necesita de las tablas.

Para poder explicar cómo realizar consultas, utilizaremos la tabla ALUMNOS y recuperaremos registros utilizando la sentencia SELECT:

Alumnos

Matricula-alumno	Nombre_alumno	Tutor	Despacho-Tutor
1022	Luis	García	412
2345	Daniela	Díaz	216
3456	Alfredo	González	546
4321	Rodolfo	García	412

Registros

Matricula-alumno	Num-clase
1022	101-07
1023	143-01
1024	159-02
4321	201-01
4322	211-02
4323	214-01

1. SELECT * FROM Alumnos;

Con esta consulta se recuperan todos los datos de la tabla Alumnos, es decir nos va a desplegar una lista de todos los alumnos, con el nombre de su tutor y el número del despacho del tutor.

2. SELECT Nombre_alumno FROM Alumnos WHERE tutor = García;

Con esta consulta nos mostrará los nombres de los alumnos que tengan al tutor de apellido García, es decir el resultado serían los siguientes 2 registros:

Nombre_alumno
Luis
Rodolfo

3. SELECT A.Nombre_alumno, A.Tutor, R.Num_clase
   FROM Alumnos A, Registros R
   WHERE A.Matricula_alumno = R.Matricula_alumno
   ORDER BY Tutor, Nombre_alumno, Num_clase;

El resultado de esta consulta sería una combinación (JOIN) de ambas tablas, referenciadas por una llave primaria que es Matricula_alumno, la cual servirá para hacer la comparación y extraer todos los datos de los identificadores que son iguales entre ambas tablas, ordenándolos de la siguiente manera:

Tutor	Nombre_alumno	Num_clase
García	Luis	101-07
García	Rodolfo	201-01

Existen muchas otras consultas que pueden realizarse, todo depende de las necesidades de cada usuario, la sintaxis básica para realizarlas es la misma que el primer ejemplo, las clausulas opcionales se colocarán si así lo requiere la consulta, estas consultas pueden ser las más sencillas o las más complejas todo depende de los requisitos que se quieren satisfacer en la organización para la cual fue creada la base de datos.

C A P I T U L O III

1. Análisis y diseño de la base de datos de información escolar para una Institución Particular.

1.1 Antecedentes de la Organización

La organización para la cual se llevo a cabo el presente proyecto es una Institución Particular llamada "Grupo Educativo Los Ángeles", ubicado en el municipio de Huehuetoca Estado de México, en una de las principales colonias del municipio, su ubicación geográfica se presenta en la figura 1.

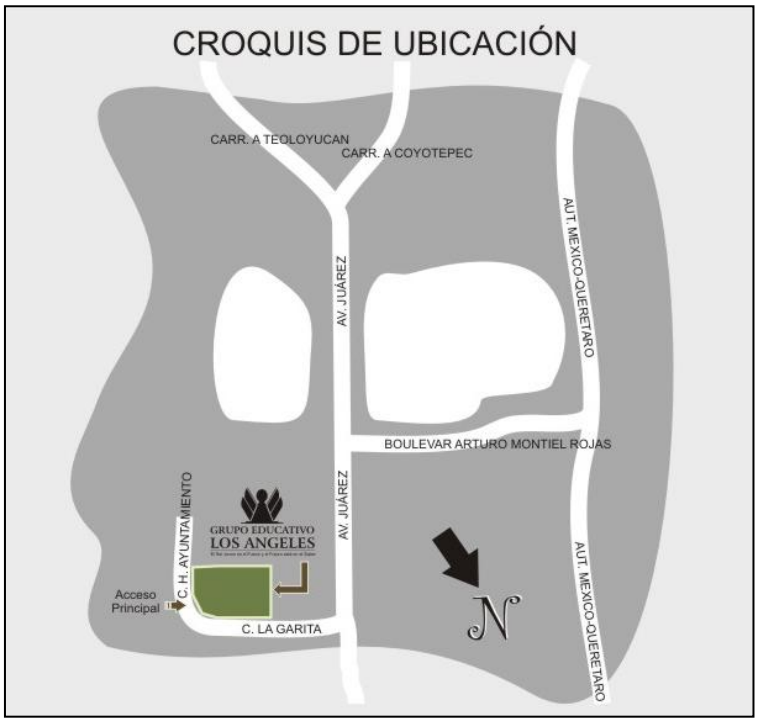


Figura 1. Ubicación geográfica de la Institución.

La Institución ha crecido paulatinamente desde su fundación en 1992, en la actualidad cuenta con 5 niveles de educación: Preescolar, Primaria, Secundaria, Preparatoria y Licenciatura.

- En 1994 se crea el Jardín de Niños Mexicano Los Ángeles
- En 1997 nace primaria y secundaria bajo el nombre de Colegio Mexicano Los Ángeles
- En 2001 se logra la incorporación de la Escuela Preparatoria Los Ángeles con una carrera técnica en Computación Fiscal Contable.
- En 2006 se incorpora el Jardín de Niños Los Ángeles, plantel San Miguel Jagüeyes.
- En 2007 se obtiene el RVOE de la Licenciatura en Administración de Negocios, siendo la primera Institución de Nivel Superior en el municipio de Huehuetoca.

- En 2008 se agregan dos licenciaturas más en Ciencias de la Educación y Derecho. Las tres bajo el nombre de Centro Universitario Los Ángeles.

Este crecimiento se ha visto reflejado en su matrícula, habiendo un incremento considerable, al igual que en las labores docentes y administrativas. Después de esta breve reseña y de considerar ciertos puntos claves se opto por apoyar en la realización de este proyecto para mejorar y actualizar los mecanismos de gestión de la información escolar.

1.2 Estudio previo y plan de trabajo

Los proyectos relacionados a sistemas se inician por muchas causas, pero dos de sus principales razones son: (1) experimentar en algún problema que lleve a una solución de sistemas y (2) reconocer oportunidades para hacer mejoras en un sistema ya implementado. En cualquiera de los dos casos es muy importante escuchar cuales son las necesidades de los usuarios, empleados o personas que están directamente involucrados con el desarrollo o mejora del sistema, ya que son la parte fundamental por la cual se realiza, además de ser fuentes externas valiosas para la localización de problemas. Para esto se realizaron algunas entrevistas al personal que estará directamente involucrado con el sistema, ver ANEXO I.

Tomando como referencia los resultados de las entrevistas y que las tareas de gestión de la información escolar se trabajan mediante archivos aislados, se tomo la iniciativa de proponer un sistema basado en computadora (base de datos) que pudiera gestionar todas las tareas que se venían realizando en cada uno de los niveles relacionadas con la administración de la información de los alumnos, como su registro, sus asignaturas y sus calificaciones, con el objetivo de agilizar las tareas y disminuir tiempos de respuesta para los usuarios del Colegio, evitando la duplicidad de actividades por tener la información de los alumnos en diversos archivos y en diferentes equipos de computo, aunque cada nivel es independiente, la mayoría de los alumnos siguen una trayectoria escolar de un nivel a otro, formando así un histórico, el cual es importante tener actualizado y sin errores.

La tarea de recabar toda la información personal y académica de los alumnos es muy importante pero requiere de tiempo, ya que son datos muy delicados, la manera en la que ha venido ejecutándose es la siguiente:

1: Los padres o tutores registran los datos personales del alumno en una tarjeta de inscripción o Carnet, cuando realizan la inscripción al nivel correspondiente. Estos datos son cotejados posteriormente con la documentación requerida.

2: Estas tarjetas junto con los expedientes de los alumnos son proporcionadas al encargado de control escolar en cada nivel educativo, verificando que todos los datos proporcionados sean correctos de acuerdo a los documentos solicitados en el momento de la inscripción.

3: Posteriormente se hace el vaciado en archivos de Excel, el formato de cada uno de ellos depende de las necesidades del nivel.

4: Se lleva a cabo la creación de formatos, plantillas o reportes individuales a partir de la información capturada de los alumnos.

En el ANEXO II se muestran algunos de los formatos que se utilizan para la captura de datos y para la emisión de plantillas o reportes.

En el siguiente organigrama de la figura 2, el recuadro de línea punteada enmarca las áreas de oportunidad de mejora que se pretenden satisfacer.

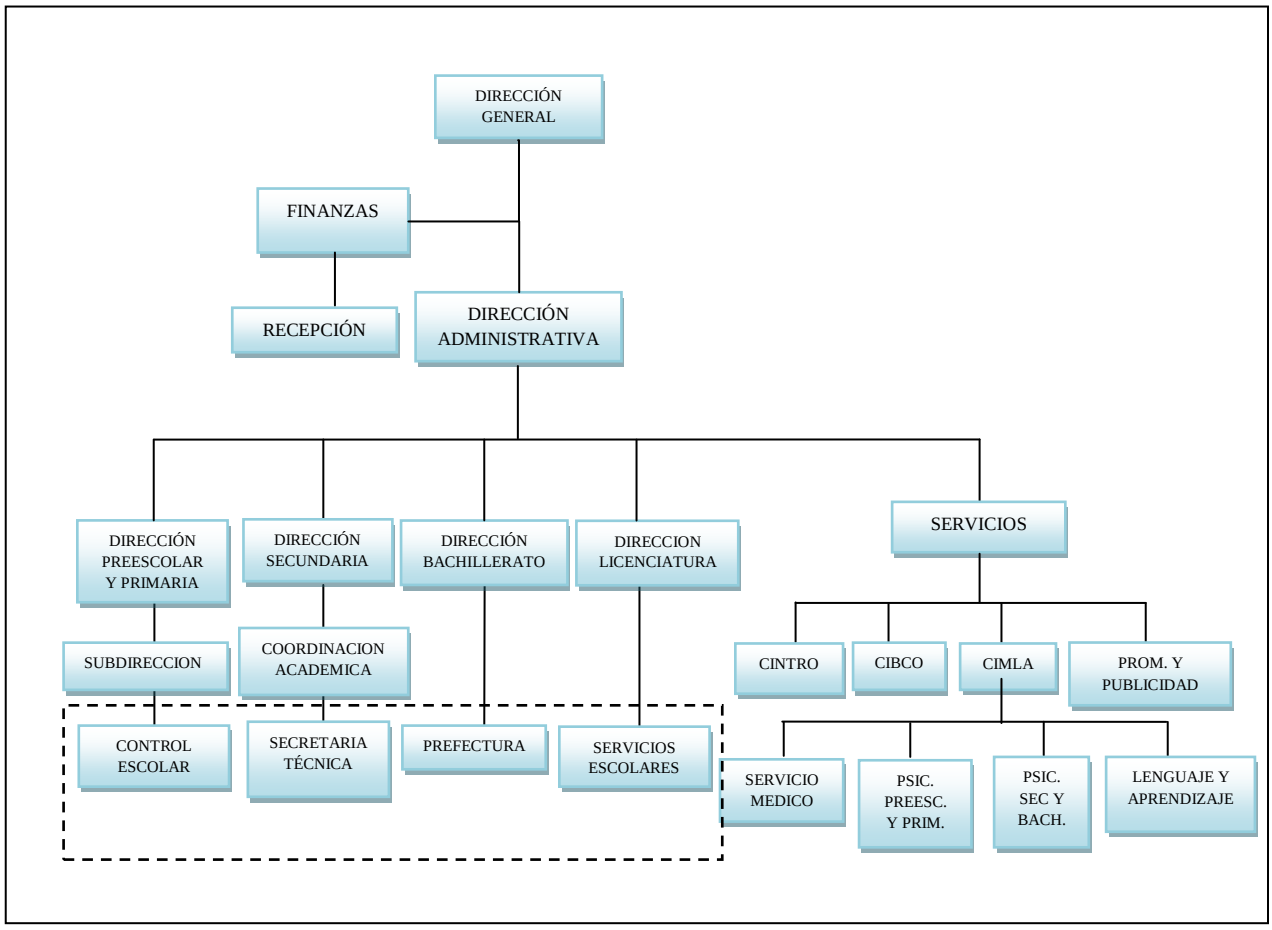


Figura 2. Organigrama de la Institución.

Como se puede observar las áreas en las que se pretende dar solución mediante un sistema a varios problemas, son las áreas que controlan la gestión de la información personal y académica de los alumnos en cada nivel (Control Escolar de Preescolar y Primaria, Secretaria Técnica de Secundaria, Prefectura de Bachillerato y Control Escolar de Licenciatura).

Uno de los objetivos de la creación de este sistema es tener almacenada toda la información personal y académica de los alumnos, la cual es de suma importancia tanto para el Colegio como para los lineamientos que se deben cumplir ante la instancia reguladora del sistema educativo que en este caso es la Supervisión Escolar. Con ello se busca tener un mejor manejo de la información, concentrarla en un solo archivo y distribuirla a cada uno de los niveles correspondientes, a través de un sistema, evitando así la redundancia de datos, tareas innecesarias y duplicidad de actividades que llevan consigo pérdida de tiempo.

Las necesidades que se desean cubrir son las siguientes:

- Se necesita un banco de datos para organizar los datos de los alumnos inscritos, altas y bajas que se realicen a lo largo del ciclo escolar, ya que actualmente se hace de forma manual o en tablas de Excel que terminan siendo insuficientes para manipular grandes cantidades de información.

- Almacenar las calificaciones de cada uno de los alumnos e inasistencias, generando los promedios generales y anuales correspondientes.

- Que solo se realice una sola captura de datos y no haya duplicidad de información ni de actividades en otras áreas.

- Edición de plantillas, reportes y formatos que necesiten cada una de las áreas que tendrán acceso a ella.

- Dar un mejor tiempo de respuesta a los usuarios finales.

Cabe mencionar que los datos que se manejan dentro de la institución se deben tratar de manera muy delicada, es decir no cualquier persona que este laborando dentro del Colegio puede tener acceso a ella, solo y exclusivamente las áreas de control escolar de cada nivel, así como la Dirección y Subdirección. Cada uno de ellos deberá tener acceso a los registros de sus alumnos con todos sus datos personales y sus calificaciones. Las calificaciones son de suma importancia ya que de ellas depende el paso de un nivel a otro, su promedio anual y general de aprovechamiento. La captura de ellas la deberá realizar el docente que imparta la asignatura o materia, proporcionándole una clave de acceso o *password*.

Para poder iniciar la realización de este proyecto se necesito de los siguientes requisitos previos:

- Autorización y respaldo del Director General.
- Estimación del tiempo dispuesto para el desarrollo del sistema.
- El sistema se considero práctico en cuestión de recursos tanto para el que lo desarrolla como para la organización.
- El proyecto se considero como una oportunidad de mejora en la gestión de la información escolar.

Todos estos criterios deben tomarse en cuenta para poder dar inicio al proyecto y ver si es factible. Esta factibilidad debe ser valorada en tres formas: operacional, técnica y económicamente, si cumple las tres formas puede merecer un desarrollo posterior. Los datos para el estudio de factibilidad fueron recolectados por medio de las entrevistas realizadas a los usuarios directamente relacionados con el problema u oportunidad detectada (Ver ANEXO I).

Determinando que el sistema era factible por cumplir con lo siguiente:

- Es una oportunidad de mejora para la actual administración de información.
- La tecnología disponible puede satisfacer las necesidades de los usuarios.
- La persona destinada a llevar a cabo el proyecto tiene disponibilidad de tiempo.
- Los costos de tiempo, de estudio del sistema, del tiempo de empleados, de hardware y de software fueron valorados y no sobrepasan las ganancias a largo plazo.
- El personal que labora actualmente manifestó la necesidad de implementar un mecanismo nuevo que pudiera reducir todas las tareas repetitivas en las cuales se empleaba demasiado tiempo.

1.2.1 Planificación y calendarización de las actividades

Una vez determinadas las necesidades que se tienen que satisfacer y contar con la autorización para realizar el proyecto, deben planearse las actividades, ya que son un punto importante y necesario para estimar tiempos, espacios disponibles, personal adecuado, tareas a realizar, etc. Además de planear debemos controlar todas estas actividades, ya que es importante hacer la comparación de la planeación de proyecto con la evolución actual, para poder terminar el trabajo en el tiempo estimado y adecuadamente. Hay ocasiones en las que se tiene que recalendarizar debido a que hay retroalimentación y debe ser tomada en cuenta para un mejor desarrollo del sistema. Para lograr esto debemos tener en cuenta todas las tareas que se realizarán a lo largo del proyecto, sin excluir ninguna, todas son importantes y deben detallarse para lograr estimar el tiempo que requerirá y no pasar por alto su realización.

Esta calendarización se realizo mediante un Diagrama de Gantt, con ayuda del software Microsoft Project 2007, el cual permite de forma fácil y sencilla estimar los tiempos de cada tarea o actividad. En el ANEXO III, podemos ver el Diagrama de Gantt resultante de las tareas y actividades del proyecto.

Cabe mencionar que en un primer momento se tomaron en cuenta algunas tareas y actividades considerando llevar a cabo el proyecto hasta la implementación, por cuestiones laborales y termino de contrato, no se concluyo hasta esa etapa, por lo que solo se dio termino a la creación, carga de datos y consultas en la base de datos.

1.3 Recolección y análisis de requisitos

Una vez realizadas las entrevistas, hacer observaciones del funcionamiento de la institución, examinar documentos, calendarizar actividades y tener la experiencia de haber trabajado en uno de los niveles educativos, se lleva a cabo la recolección y análisis de requisitos:

a) Requisitos de datos de los usuarios del sistema:

- El Colegio está organizado por niveles, cada nivel debe tener un identificador *id_nivel* que los distinguirá de los demás. Donde aparecerá el nombre descriptivo de cada uno de ellos, el Director del nivel, su clave de centro de trabajo y el turno correspondiente.

- El sistema deberá tener almacenados a los alumnos de nuevo ingreso, a los que ya están inscritos en el plantel desde el nivel de Preescolar hasta Licenciatura de cada ciclo escolar, así como bajas y altas que se presenten a lo largo del periodo escolar.

- El registro de inscripción debe tener los siguientes datos personales de los alumnos: Nombre del Alumno con apellidos, Sexo, CURP, estado escolar, nivel que cursa o al que ingresa, grado y grupo en el que estará inscrito, Fecha de nacimiento, Edad, Talla, Peso, Grupo Sanguíneo, Nombre de los padres y ocupación, Dirección la cual está compuesta por País, Estado, Municipio, Código Postal, Teléfono, Fecha de alta y Fecha de baja.

- El estado escolar de los alumnos se identificará por dos valores nuevo ingreso o repetidor, identificando con esto si ha cursado los niveles en tiempo y forma.

- Los directores de cada nivel se identificarán mediante un *id_director*, almacenando su Nombre Completo y el nivel que dirigen.

- Las calificaciones de los alumnos deberán estar relacionadas con el *id_alumno*, el periodo de evaluación, las asignaturas, la calificación asignada y las faltas totales obtenidas.

- Cada periodo de evaluación se identificará de acuerdo al nivel que se curse, ya que puede ser bimestral, semestral o cuatrimestral.

- Las asignaturas se clasificarán por medio de un *id_asignatura* de acuerdo a cada nivel, ya que son diferentes, así como el docente que las imparte.

- Los docentes se identificarán mediante un *id_docente*, almacenando su nombre completo y la carrera que tienen como profesión.

b) Usuarios que interactúan con el sistema:

- Director de cada nivel: tendrá el privilegio de consultar los datos personales y académicos de los alumnos, alguna otra área dentro del plantel o la supervisión bajo la cual está regida.

- Personal de Control Escolar: serán los encargados del manejo de gran parte de la información académica y personal de los alumnos de su nivel.

- Docentes: tendrán acceso a la Base de Datos para la captura de sus evaluaciones bimestrales de las asignaturas que impartan.

c) Requisitos de Hardware y Software:

En cuanto a los requerimientos de equipo y programas se hará uso de los equipos con los que cuenta el Colegio Particular, tomando en cuenta sus características y analizando los requisitos mínimos para instalar MySQL. Una de las condiciones por parte del Director General, fue trabajar con el equipo que hay en la Institución ya que no se invertiría en la compra de equipo nuevo. De acuerdo a lo anterior se trabajo con el equipo disponible adecuándolo a las necesidades.

Por tanto para la creación e instalación de la base de datos en MySQL se cuenta con el siguiente equipo que tiene como características:

- Sistema Operativo Windows XP (uno de los SO soportados por MySQL), por tanto es factible utilizarlo.

- Por el costo se trabajara con un manejador de bases de datos libre: MySQL versión 5.1.45, posteriormente se dará la justificación de su elección.

- 10 Computadoras disponibles para el proyecto con Procesador AMD 2.1 Ghz, memoria RAM de 1GB, Disco duro de 100 GB, Quemador DVD y CD, monitor 17", teclado y mouse. Las cuales cuentan con las características mínimas requeridas para la instalación del manejador y la creación de la base de datos.

- Cable UTP 100 Mb/s. Es adecuado para una red local como la que se maneja en la Institución, que tenga pocos nodos, un presupuesto limitado y una conectividad simple.

- El ambiente será multiusuario ya que accederán a él aproximadamente 50 usuarios internos.

- Se manejará bajo un esquema Centralizado ya que solo se tendrá implementado en la Institución.

El equipo con el que se cuenta, tiene las características antes mencionadas, por tanto haciendo un análisis de los requisitos sugeridos es factible proceder a la creación e instalación de la base de datos.

1.4 Diseño Conceptual de la Base de Datos de Información Escolar

Hay varias estrategias a seguir para realizar el diseño: de abajo a arriba, de arriba a abajo, de dentro a fuera y la estrategia mixta. Se eligió la estrategia *de abajo a arriba* partiendo de todos los atributos y agrupándolos en entidades y relaciones. Esta estrategia es apropiada cuando la base de datos es simple, con pocos atributos, como lo son la mayoría de las entidades de la base de datos. (Castaño & Piattini Velthuis, Fundamentos y modelos de bases de datos, 1999).

A partir de las especificaciones de requisitos de datos podemos empezar a formar el esquema conceptual de la base de datos, definiendo las entidades y los atributos que las conformarán. Para su definición se colocará el nombre de la entidad seguida de la palabra colegio y los atributos deben hacer referencia a los valores que los conforman.

1.4.1 Diagrama Entidad-Relación de la base de datos

- La entidad *alumnos_colegio* se compone de los siguientes atributos: *id_alumno, Nombre_alumno, Apellidopaterno_alumno, Apellidomaterno_alumno, fecha_nacimiento, sexo (que se manejará como M para mujer y H para hombre), CURP, edo_escolar (nuevo ingreso ó repetidor), Nombre_padre, Apellidopaterno_padre, Apellidomaterno_padre, Ocupación_padre, talla, peso, tipo de sangre (que puede ser A+, A- ,B+, B-, O+, O-, AB+, AB-), ciclo_escolar (2009-2010), nivel_educativo, grado, grupo, fecha_alta y fecha_baja*. Algunas características descriptivas adicionales son el atributo multivaluado *teléfono*, el atributo derivado *edad* y el atributo compuesto *Dirección* con los atributos *país_alumno, edo_alumno, municipio_alumno, colonia, código, calle y numero_casa*.

- La entidad *nivel_educativo* se compone de *id_nivel, nivel, nombre_nivel*, clave del centro de trabajo ó *CCT* y *turno*.

- La entidad *direc_colegio* se compone de *id_director, nombre_director, apepat_dir, apemat_dir* y el nivel que dirigen *dir_nivel*.

- La entidad *periodo_evaluacion* se compone de *id_periodo* y el *per_eval* que pueden ser (Bimestral, Semestral o Cuatrimestral).

- La entidad *calif_colegio* se compone de *id_alumno, periodo_eval, materias, calificación (la cual adopta valores enteros de 5 a 10) y faltas*, teniendo como atributo derivado *promedio_bimestral ó promedio_anual*.

- La entidad *asignaturas_colegio* contiene el *id_asignatura, nombre_asignatura*.

- La entidad *docente_colegio* se compone de *id_docente, nom_docente, apepat_docente, apemat_docente* y *titulo.*

- La entidad *asignatura_docente* se compone de las llaves primarias (id_asignatura) *asignatura*, (id_nivel) *nivel_asignatura* y (id_docente) *docente_asignatura*.

De la misma manera describiremos las relaciones que las conforman:

- *Ingresa*, una relación muchos a uno entre alumnos y nivel educativo, donde muchos alumnos ingresan a un nivel educativo.

- *Dirige*, una relación uno a uno entre directores y nivel educativo, donde un director puede estar solo a cargo de un nivel.

- *Compone*, una relación muchos a uno entre asignaturas y nivel educativo, donde varias asignaturas componen un nivel.

- *Cursa*, una relación uno a muchos entre alumno y asignaturas, donde un alumno cursa varias asignaturas.

- *Obtiene*, una relación uno a uno entre asignaturas y calificaciones, donde a cada asignatura se le asigna una calificación.

- *Corresponde*, una relación muchos a uno entre calificaciones y periodo de evaluación, ya que por cada periodo de evaluación se asignan calificaciones a todas las asignaturas.

- *Imparte*, una relación uno a muchos entre docente y asignaturas, donde un docente puede impartir varias asignaturas y cada una de ellas pertenece a un determinado nivel.

Una vez descritas las entidades, atributos y conjunto de relaciones derivadas del modelo conceptual, se hace la representación mediante un diagrama E-R mostrado en la figura 3.

1.5 Elección de MySQL como Sistema Manejador de Bases de Datos

Ahora es el momento de hacer la elección del manejador (SMBD) que se utilizará para la creación de la base de datos. El tipo de manejador que se utilizará será relacional, es MySQL 5.1.45, está elección se derivo de algunas consideraciones tanto de la organización como del estudio de factibilidad. Algunas ventajas que se tomaron en cuenta fueron las siguientes:

- MySQL software es Open Source.
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación (Soporta gran variedad de Sistemas Operativos).
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Su conectividad, velocidad, y seguridad hacen que MySQL Server sea altamente apropiado para acceder a bases de datos en Internet
- El software MySQL usa la licencia GPL

- Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
- Seguridad, en forma de permisos y privilegios, determinados usuarios tendrán permiso para consulta o modificación de determinadas tablas. Esto permite compartir datos sin que peligre la integridad de la base de datos o protegiendo determinados contenidos.
- Potencia, SQL es un lenguaje muy potente para consulta de bases de datos, usar un motor nos ahorra una enorme cantidad de trabajo.

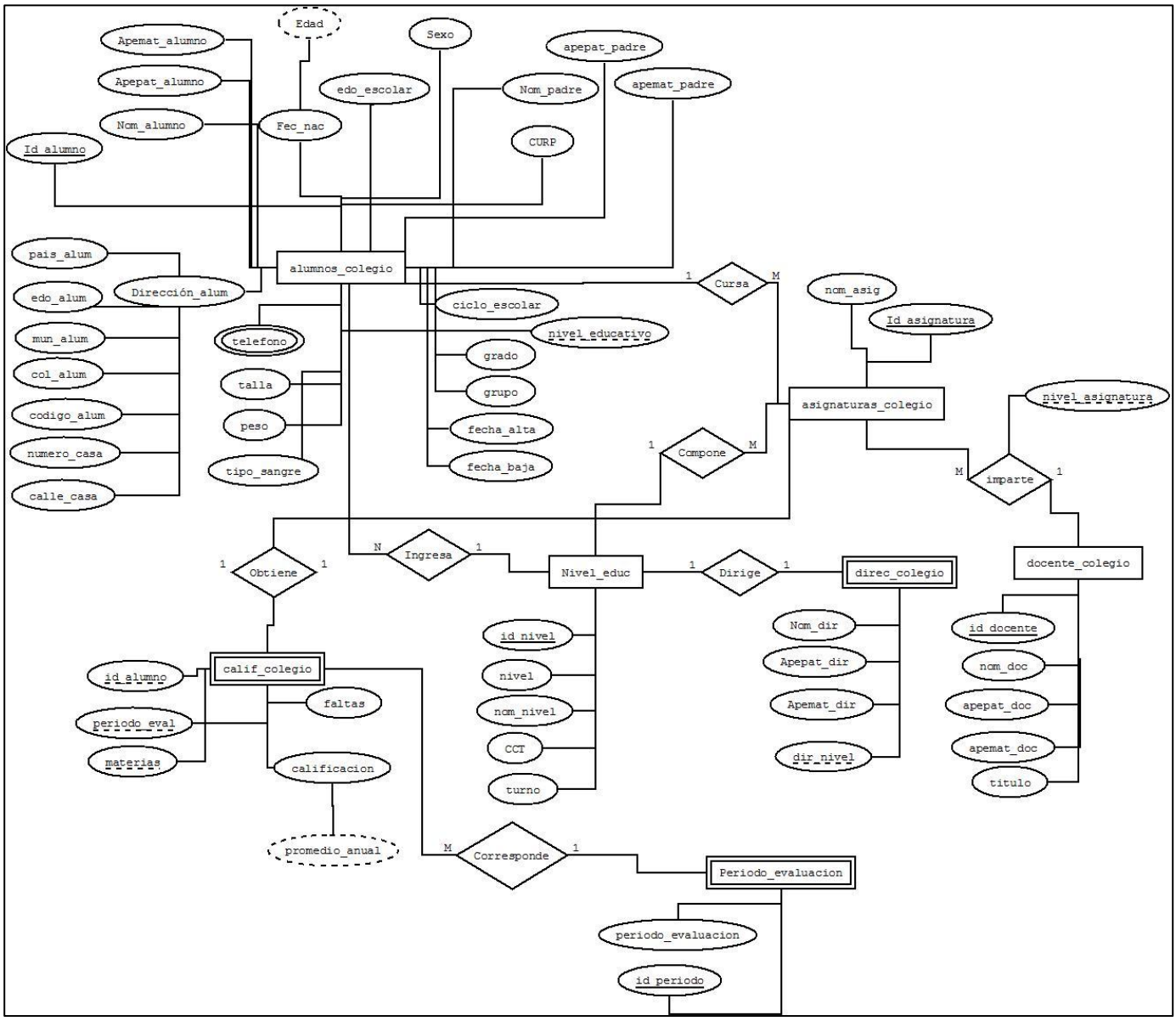


Figura 3. Diagrama E/R de la base de datos de información escolar de alumnos.

- Portabilidad, SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas.
- Tiene escalabilidad ya que es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
- MySQL está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs (en inglés: *Aplication Programming Interface*), para muchas plataformas diferentes.
- Permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.
- Es multihilo, con lo que puede beneficiarse de sistemas multiprocesador.
- Permite manejar multitud de tipos para columnas.
- Permite manejar registros de longitud fija o variable.
- Es fácil de usar. (DuBois, 2001).

Estas son algunas de las ventajas de usar MySQL, a partir de ellas se definió que era una buena opción para el proyecto, entre las más consideradas, el ser un software libre y sin costo, ya que no se conto con el apoyo económico para adquirir otro software, no dejando a un lado todas las demás que se enumeraron anteriormente.

1.6 Diseño Lógico de la base de datos de Información Escolar

1.6.1 Transformación del Diagrama E/R a tablas

Parte del Diseño Lógico es la transformación del diagrama E/R a tablas, esto se hará bajo el modelo relacional, en el cual representaremos los datos en una estructura de tablas bidimensionales, almacenando toda la información de la base de datos, conservando la semántica, la redundancia y su fácil comprensión.

Cada entidad representará una tabla, y los atributos de cada entidad serán los campos que la conformen, identificando aquellos que son claves primarias, ajenas o únicas, así como los que no son principales. De la misma forma se transformarán las relaciones (N:M, 1:N) creando una tabla que se identifique por las claves primarias de las entidades que la conforman, y que se convertirán en claves ajenas de las entidades involucradas. A partir del diagrama E/R, se desprende la transformación a tablas bajo el modelo relacional, como se muestra en la figura 4.

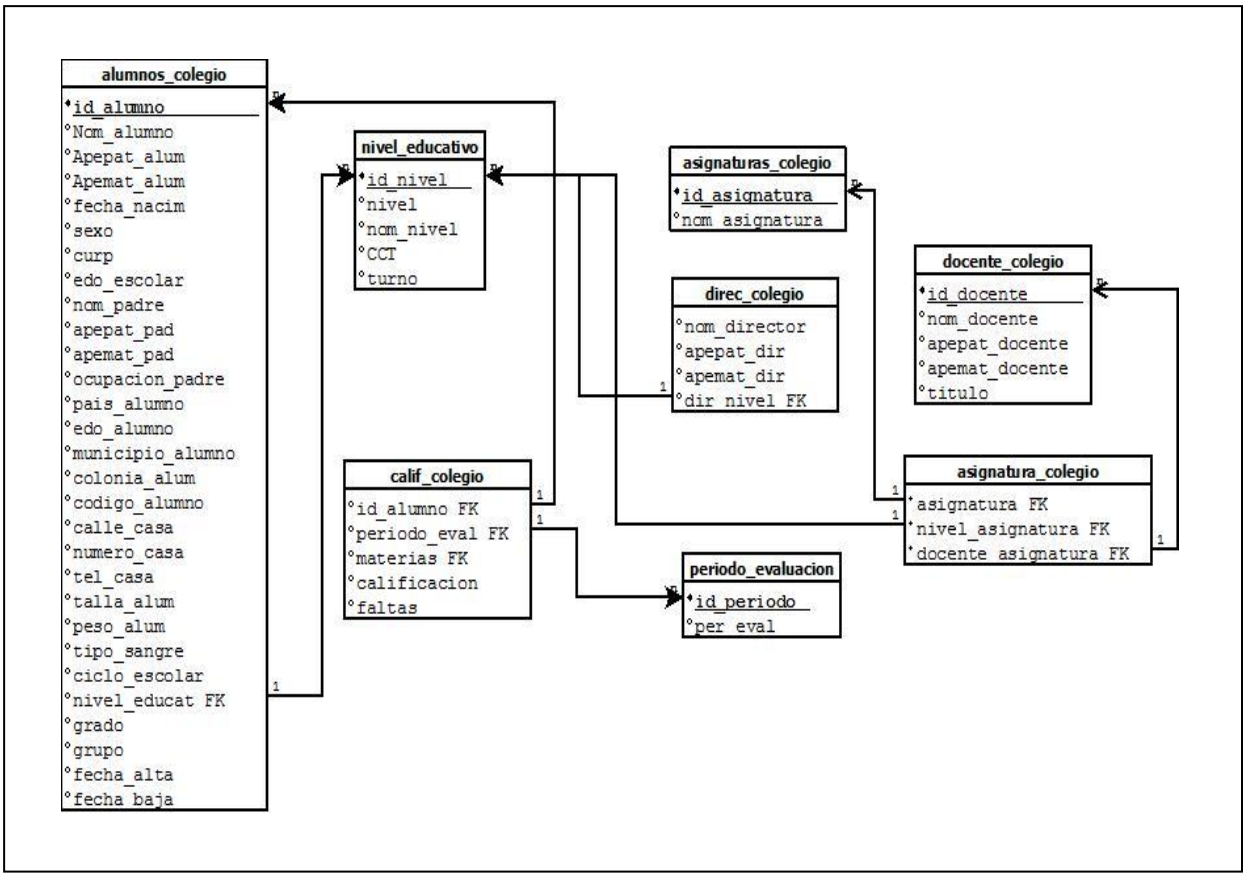


Figura 4. Modelo relacional de la base de datos *'colegio'*.

Posteriormente lo trasladaremos a un lenguaje lógico, en este caso el lenguaje a utilizar será SQL que forma parte de MySQL, con el cual se empezará la creación de las estructuras de la base de datos. Manejaremos la línea de comandos del cliente de MySQL, así como algunas herramientas graficas como MySQL Query Browser y MySQL Administrator, que nos servirán para ejecutar sentencias SQL y servicios de monitoreo tanto de la base de datos como del sistema.

1. Creación de la base de datos y tablas: mediante el Lenguaje de Definición de Datos LDD, se definirán los elementos de la base de datos y las tablas, es decir se creara el diccionario de datos, donde se encuentran los datos referentes a los datos del sistema (base de datos utilizada, nombre de las tablas, nombre de las columnas, tipos de datos, longitud de columna, asignación de claves, etc.).

   Las sentencias que se utilizarán para la creación de la base y las tablas serán CREATE (creación de objetos) y DROP (eliminación de objetos). Primero se eliminarán en caso de existir para no duplicar o tener problemas al realizar la creación, utilizando la sentencia DROP, en caso de no ser así se crearán sin ninguna dificultad.

Al determinar las claves se usará la opción CASCADE, la cual borra o actualiza el registro en la tabla padre y automáticamente borra o actualiza los registros coincidentes en la tabla hija. Tanto ON DELETE CASCADE como ON UPDATE CASCADE están disponibles en MySQL 5.1. Una consideración importante es que entre dos tablas, no se deben definir varias cláusulas ON UPDATE CASCADE que actúen en la misma columna en la tabla padre o hija.

El motor de almacenamiento con el cual se crearán las tablas es con el Engine InnoDB que soporta MySQL 5.1.45 y que es asignado por default.

Casi todos los valores de las tablas son NOT NULL, esto significa que no pueden quedar vacios estos campos, deben contener información, ya que en caso contrario se genera un error en el momento de la carga de datos. Esto fue determinado así ya que la administración escolar y las políticas de la organización así lo requieren.

Las tablas que contienen llaves primarias (PRIMARY KEY) tienen un valor extra que es *auto_increment*, esto significa que se crea una secuencia automáticamente, en la cual podemos definir el inicio de la secuencia o simplemente iniciar desde 1. Los tipos de datos y su longitud fue seleccionada y determinada de entre todos aquellos que soporta MySQL, pero también de acuerdo a la información que contienen ya que utilizar un dato con mayor o menor espacio es innecesario y consume recursos del equipo. Tomaremos como ejemplo la tabla *nivel_educativo*:

```
Create table nivel_educativo
(id_nivel smallint(2)  not null auto_increment,
nivel varchar(20) not null,
nom_nivel varchar(30)  not null,
CCT varchar(15) not null,
turno varchar(10) not null,
primary key (id_nivel));
```

Para esta tabla en particular se determino que la llave primaria *id_nivel* sería de tipo *smallint (2)*, no puede ser nula y generará una secuencia de números, la razón es la siguiente: la lista de niveles con los que cuenta la escuela no exceden los 2 dígitos, es decir analizando no se pueden tener más de 99 niveles educativos, por eso se asigno *smallint* ya que ocupa un espacio en memoria de 2 bytes suficientes para satisfacer la secuencia. Los demás valores son de tipo *varchar,* debido a que la longitud de la información para cada campo es variable, así de acuerdo a los caracteres que se almacenan corresponderá el espacio ocupado en memoria (n + 1 byte).

Una vez determinadas algunas características de las tablas, se realizará la transformación a lenguaje lógico estándar (SQL): la base de datos que gestionará la información escolar de los alumnos de la Institución lleva como nombre *colegio*. El código para la creación de las estructuras de la base de datos se muestra en el ANEXO IV.

De esta manera determinamos las estructuras de los objetos de la base de datos (base y tablas), como se muestra en la figura 5, una vez creados automáticamente se genera el diccionario de datos, el cual es de suma importancia ya que cada vez que se requiera manipular los datos la base tiene que checar primero en el diccionario de datos para realizar cualquier acción.

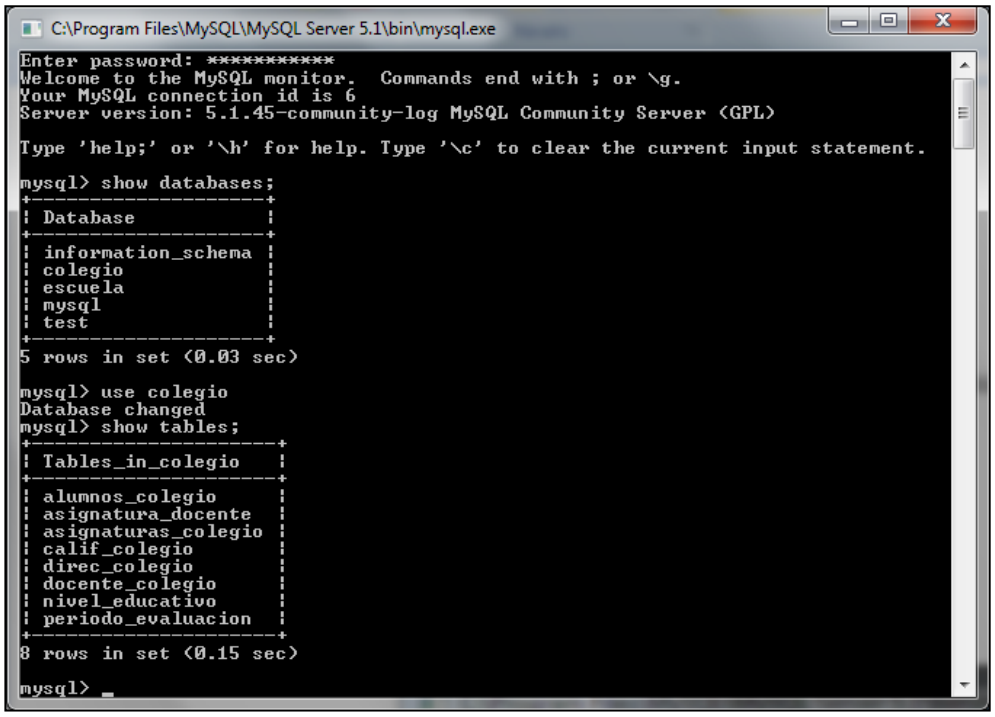


Figura 5. Creación de la base de datos *colegio* y las tablas que la conforman.

INFORMATION_SCHEMA es la base de datos de información, que almacena información acerca de todas las otras bases de datos que mantiene el servidor MySQL . Hay varias tablas que conforman esta base, cada una de ellas tiene información relativa con las tablas de la base de datos que se especifique, es decir esta base es el diccionario o directorio de datos.

En la figura 6 se muestra una parte del diccionario de datos de la tabla *alumnos_colegio*, sustraído de la base de datos INFORMATION_SCHEMA. El diccionario de datos completo se puede ver en el ANEXO IV.

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	alumnos_colegio	id_alumno	int	int(6)	PRI	auto_increment
colegio	alumnos_colegio	Nom_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	Apepat_alum	varchar	varchar(20)		
colegio	alumnos_colegio	Apemat_alum	varchar	varchar(20)		
colegio	alumnos_colegio	fecha_nacim	date	date		
colegio	alumnos_colegio	sexo	varchar	varchar(2)		
colegio	alumnos_colegio	curp	varchar	varchar(30)		
colegio	alumnos_colegio	edo_escolar	varchar	varchar(14)		
colegio	alumnos_colegio	nom_padre	varchar	varchar(20)		
colegio	alumnos_colegio	apepat_pad	varchar	varchar(20)		
colegio	alumnos_colegio	apemat_pad	varchar	varchar(20)		
colegio	alumnos_colegio	ocupacion_padre	varchar	varchar(20)		
colegio	alumnos_colegio	pais_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	edo_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	municipio_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	col_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	cod_alum	varchar	varchar(6)		
colegio	alumnos_colegio	calle	varchar	varchar(20)		
colegio	alumnos_colegio	numero	varchar	varchar(4)		
colegio	alumnos_colegio	telefono	varchar	varchar(20)		

Figura 6. Diccionario de datos de la base de datos '*colegio*'.

La descripción de cada una de las tablas de la base de datos *colegio* nos proporciona un panorama de cómo quedaron estructuradas y definidas, en la figura 7 se describe la tabla *alumnos_colegio*, en la cual podemos ver las columnas o atributos que la forman, el tipo de dato de la columna, las claves primarias y foráneas, así como algunas restricciones, esta misma estructura es la que tienen las demás tablas de la base de datos '*colegio*'.

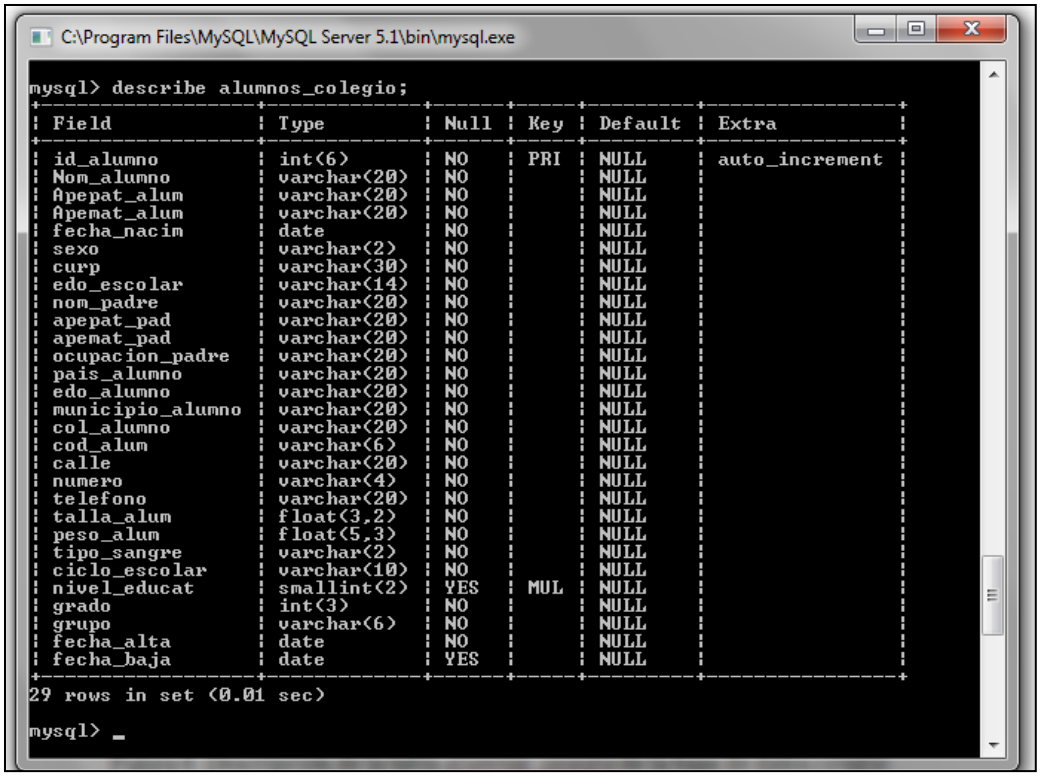


Figura 7. Descripción de la tabla *alumnos_colegio* de la base de datos *colegio*.

Es importante dentro del diseño de una base de datos tener bien estructuradas las tablas, utilizando valores atómicos en los campos, evitando dependencias funcionales y transitivas en algunos atributos, para formar entidades independientes, dejando solamente los atributos que dependen de la llave primaria. Con ello se evitará la redundancia y cumpliremos la regla de que solo un valor de un atributo particular se asocia con un único y especifico valor de otro atributo. En caso de no ser así, se puede recurrir a la normalización de tablas, estrategia mediante la cual se evitan todos los problemas mencionados anteriormente. Durante el diseño de la base de datos se tomaron en cuenta todos estos aspectos para no realizar la normalización posteriormente, aun así se podrían realizar cambios una vez que se hagan pruebas con la base de datos.

1.7 Diseño físico de la base de datos de Información Escolar.

El diseño de una base de datos se descompone en tres etapas: diseño conceptual, lógico y físico. La etapa del diseño lógico es independiente de los detalles de implementación y dependiente del tipo de SMBD que se vaya a utilizar. La salida de esta etapa es el esquema lógico global y la documentación que lo describe. Todo ello es la entrada para la etapa que viene a continuación, el diseño físico.

Mientras que en el diseño lógico se especifica qué se guarda, en el diseño físico se especifica cómo se guarda. Para ello, se debe conocer muy bien toda la funcionalidad del SMBD concreto que se vaya a utilizar y también el sistema informático sobre el que éste va a trabajar. El diseño físico no es una etapa aislada, ya que algunas decisiones que se tomen durante su desarrollo, por ejemplo el mejorar las prestaciones, pueden provocar una reestructuración del esquema lógico.

Para eso debemos tomar en cuenta y conocer la arquitectura de MySQL que tiene como característica más notable el separar el motor de almacenamiento (que se encarga de los detalles de entrada-salida y representación de la información en memoria secundaria) del resto de los componentes de la arquitectura. Es decir, el diseño del gestor está preparado para que se pueda cambiar el gestor de almacenamiento. Esto permite incluso crear nuevos motores de almacenamiento especializados para ciertas tareas o tipos de aplicaciones.

En la figura 8, se muestra una visión abstracta de la arquitectura lógica de MySQL. En ella se hace una división entre los componentes que conforman el servidor, las aplicaciones cliente que lo utilizan y las partes del sistema operativo en las que se basa el almacenamiento físico.

1.7.1 Arquitectura de MySQL

Las utilidades y herramientas de MySQL son los programas y aplicaciones que se incluyen con la distribución del gestor, o que pueden instalarse como aplicaciones adicionales. Estas incluyen las herramientas de backup, el navegador de consultas MySQL Query Browser, las aplicaciones administrativas de interfaz gráfico y la herramienta de diseño MySQL Workbench, entre otras.

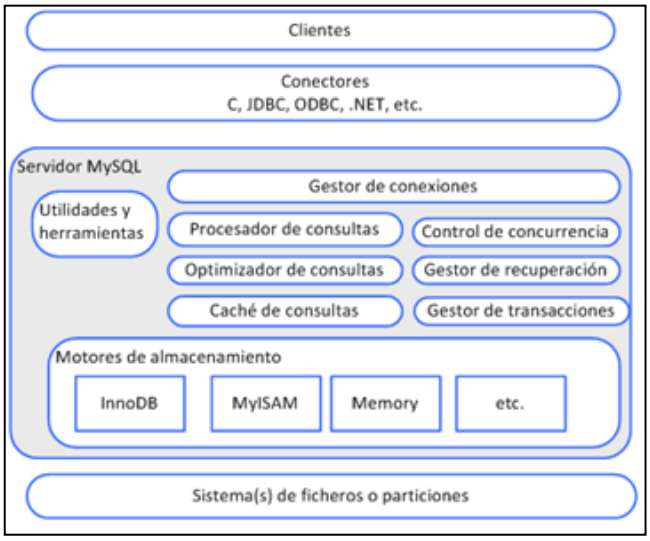


Figura 8. Arquitectura de MySQL.

1.7.2 Elección del motor de almacenamiento InnoDB

En consecuencia, una primera tarea del diseño físico en MySQL es la de decidir el motor de almacenamiento más apropiado.

Los elementos que puede implementar un motor de almacenamiento son los siguientes:

- Concurrencia. Es responsabilidad del motor implementar una política de bloqueos (o no implementar ninguna). Una estrategia de bloqueos por fila permite una mayor concurrencia, pero también consume más tiempo de procesamiento en aplicaciones en las que la concurrencia no es realmente grande.
- Soporte de transacciones. No todas las aplicaciones necesitan soporte de transacciones.
- Comprobación de la integridad referencial, declarada como restricciones en el DDL de SQL.
- Almacenamiento físico, incluyendo todos los detalles de la representación en disco de la información.
- Soporte de índices. Dado que la forma y tipo de los índices depende mucho de los detalles del almacenamiento físico, cada motor de almacenamiento proporciona sus propios métodos de indexación (aunque algunos como los árboles B casi siempre se utilizan).

- Cachés de memoria. La eficiencia de los cachés de datos en memoria depende mucho de cómo procesan los datos las aplicaciones. MySQL implementa cachés comunes en el gestor de conexiones y la caché de consultas, pero algunos motores de almacenamiento pueden implementar cachés adicionales.
- Otros elementos para ayudar al rendimiento, como puede ser el uso de múltiples hilos para operaciones paralelas o mejoras de rendimiento para la inserción masiva (Castaño & Piattini Velthuis, Fundamentos y modelos de bases de datos, 1999).

No hay una receta única que permita definir el motor de almacenamiento. La selección debe hacerse una vez que tenemos el modelo lógico de la base de datos y conocemos los requisitos de rendimiento y no funcionales de la aplicación o aplicaciones que se van a construir.

La sentencia SHOW ENGINES nos muestra la lista de motores en MySQL, incluyendo el motor por defecto y los que no están disponibles con la configuración actual. El resultado para MySQL 5.1. 45 es el mostrado en la figura 9. Cuando se instala MySQL en Windows usando el MySQL Configuration Wizard, InnoDB es el motor de almacenamiento por defecto en lugar de MyISAM.

Engine	Support	Comment	Transactions	XA	Savepoints
MyISAM	YES	Default engine as of MySQL 3.23 with great performance	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	NULL	NULL	NULL
InnoDB	DEFAULT	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
ARCHIVE	YES	Archive storage engine	NO	NO	NO
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO

Figura 9. Lista de motores de almacenamiento soportados por MySQL 5.1.45

Al realizar la creación de las tablas de la base de datos ‘colegio’ con el Engine InnoDB se dota a MySQL de un motor de almacenamiento transaccional (conforme a ACID) con capacidades de *commit* (confirmación), *rollback* (cancelación) y recuperación de fallas. InnoDB se diseñó para obtener el máximo rendimiento al procesar grandes volúmenes de datos. Probablemente ningún otro motor de bases de datos relacionales en disco iguale su eficiencia en el uso de CPU.

La decisión de utilizar este tipo de Engine, fue precisamente por los beneficios que proporciona, entre ellos están:

1. Recuperación automática ante fallas, es decir en caso de que el servicio se interrumpa de forma anormal InnoDB completa las transacciones que quedaron incompletas. Por lo tanto la información se mantiene contenida hasta el preciso instante en el que ocurrió la falla.

2. Se mantiene una integridad referencial permitiendo definir llaves foráneas entre tablas para asegurarse de que un registro no puede ser eliminado de una tabla si aún está siendo referenciado por otra. Como ocurre en casi todas las tablas de la base de datos, esto nos garantiza la integridad de los datos, manteniendo la relación que existe entre ellos y evitando perdidas de información.

3. Permite el bloqueo a nivel de filas para mejorar de manera impresionante el rendimiento. Es decir evita la inserción en tablas donde el índice está ocupado por una consulta.

4. SELECT sin bloqueo, el motor InnoDB usa una técnica conocida como multi-versioning que elimina la necesidad de hacer bloqueos en consultas SELECT muy simples. Ya no será necesario molestarse porque una simple consulta de solo lectura está siendo bloqueada por otra consulta que está haciendo cambios en una misma tabla.

5. Si falla una actualización, todos los cambios se deshacen. (Con tablas no transaccionales, todos los cambios son permanentes.), de esta manera se da la pauta para corregir el error y no dañar los datos.

6. Proporcionan mejor concurrencia para tablas que tienen varias actualizaciones concurrentes con lecturas.

7. Los datos se guardan en disco: un fichero para la definición de la tabla (.frm), y un tablespace para guardar conjuntamente datos e índices. El tablespace puede consistir en uno o más ficheros, o incluso una partición entera en disco.

8. Podemos especificar cómo crecen los tablespaces en el fichero de configuración (figura 10), modificando los valores y eliminando el comentario (#). Por ejemplo:

[mysqld] innodb_data_file_path = ibdata1:100M;ibdata2:100M:autoextend:max:500M.

```
# Uncomment the following if you are using InnoDB tables
#innodb_data_home_dir = C:\mysql\data/
#innodb_data_file_path = ibdata1:10M:autoextend
#innodb_log_group_home_dir = C:\mysql\data/
# You can set .._buffer_pool_size up to 50 - 80 %
# of RAM but beware of setting memory usage too high
#innodb_buffer_pool_size = 16M
#innodb_additional_mem_pool_size = 2M
# Set .._log_file_size to 25 % of buffer pool size
#innodb_log_file_size = 5M
#innodb_log_buffer_size = 8M
#innodb_flush_log_at_trx_commit = 1
#innodb_lock_wait_timeout = 50
```

Figura 10. Especificación de crecimiento de los *tablespaces* en el fichero de configuración de MySQL.

9. Necesita más espacio en disco y memoria que MyISAM para guardar los datos (unas tres veces más de espacio en disco, y mucha RAM para las memorias temporales consiguiendo así un rendimiento óptimo).

10. Para concluir, es una buena elección porque necesitamos transacciones, restricciones en claves foráneas, y escrituras simultáneas.

Al crear las tablas con el Engine InnoDB, MySQL siempre crea un fichero .frm para guardar la definición de tablas y columnas. El índice y datos de la tabla se almacenan en *tablespaces* en uno o más ficheros, en función del tipo de tabla. El servidor crea el fichero .frm por encima del nivel de almacenamiento del motor. La figura 11 muestra los ficheros .frm de la base de datos 'colegio'.

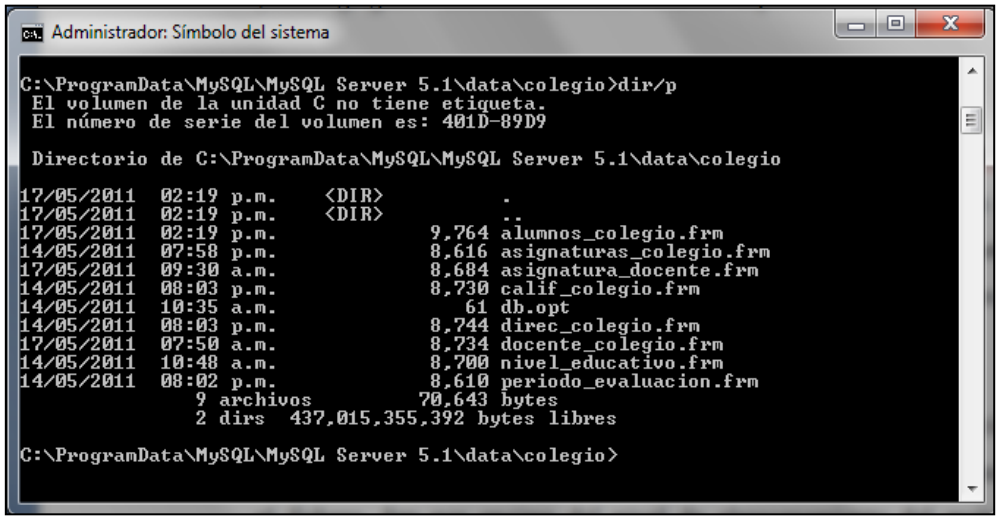


Figura 11. Ficheros .frm de la base de datos 'colegio'.

Los conectores son bibliotecas en diferentes lenguajes de programación que permiten la conexión (remota o local) con servidores MySQL y la ejecución de consultas. Por ejemplo, el conector ODBC (en inglés, *Open Database Connectivity*) en equipos con Windows como es el caso, permite tener acceso a una amplia gama de bases de datos o fuentes de datos. ODBC es una API estándar que permite conexiones a servidores de base de datos SQL. ODBC usualmente se emplea cuando se requiere la independencia de la base de datos o acceso simultáneo a diferentes fuentes de datos.

La gestión de conexiones es la responsable de mantener las múltiples conexiones de los clientes. Un gestor de conexiones inexistente o laxo simplemente crearía una conexión por cada cliente conectado. No obstante, las conexiones consumen recursos de máquina, y crearlas y destruirlas son también procesos costosos. Por eso, el gestor de conexiones de MySQL puede configurarse para limitar el número de conexiones concurrentes, y también implementa un *pool de conexiones*. Estas especificaciones las podemos controlar desde el archivo de configuración de MySQL, en la parte que se muestra a continuación:

```
# The maximum amount of concurrent sessions the MySQL server will
# allow. One of these connections will be reserved for a user with
# SUPER privileges to allow the administrator to login even if the
# connection limit has been reached.
max_connections=100
```

La idea es que muchas aplicaciones abren una conexión y la mantienen abierta y ociosa durante mucho tiempo (por ejemplo, durante toda la sesión de un usuario, que de vez en cuando se levanta para diferentes tareas como tomar café), y solo de vez en cuando se utiliza un hilo de ejecución para ejecutar una consulta, que además, típicamente tarda como mucho unos milisegundos. No tiene sentido mantener una conexión ociosa para cada usuario. De aquí proviene la idea de los pools de conexiones: hay un número de conexiones disponibles, y cada vez que una aplicación hace una solicitud, se le asigna una conexión del pool que no esté ocupada.

Dado que las conexiones consumen recursos, es mejor limitar este número que llevar a una carga excesiva en el servidor, que podría acabar en una caída del sistema o un comportamiento impredecible. El gestor de conexiones también se ocupa de la autentificación de los usuarios. La autentificación por defecto se basa en el nombre de usuario, la máquina desde la que se conecta y el password.

Cada vez que una consulta llega al gestor de MySQL, se analiza sintácticamente y se produce una representación intermedia de la misma. A partir de esa representación, MySQL toma una serie de decisiones, que pueden incluir el determinar el orden de lectura de las tablas, el uso de ciertos índices, o la re-escritura de la consulta en una forma más eficiente. Existe la posibilidad de utilizar ciertas cláusulas en las consultas para ayudar al optimizador en su tarea, o bien podemos pedirle al servidor ciertas "explicaciones" sobre cómo ha planificado nuestras consultas, para entender mejor su funcionamiento. MySQL implementa un caché de consultas, donde guarda consultas y sus resultados enteros. De este modo, el procesador de consultas, antes de plantear la optimización, busca la consulta en la caché, para evitar realizar el trabajo en el caso de que tenga suerte y encuentre la consulta en la caché.

El control de concurrencia en un gestor de bases de datos es simplemente el mecanismo que se utiliza para evitar que lecturas o escrituras simultáneas a la misma porción de datos terminen en inconsistencias o efectos no deseados. El mecanismo que se utiliza para controlar este acceso es el de los bloqueos (*locks*). La idea es muy simple, cada vez que una aplicación quiere acceder a una porción de los datos, se le proporciona un bloqueo sobre los mismos. Lógicamente, varias aplicaciones que quieran leer simultáneamente no tienen ningún problema en hacerlo, de modo que para la lectura se proporcionan bloqueos compartidos (*shared locks*). Sin embargo, varios usuarios escribiendo o uno escribiendo simultáneo con algunos realizando lecturas pueden producir problemas. Por eso, para la escritura se proporcionan bloqueos exclusivos (*exclusive locks*).

1.7.3 Representación física de la base de datos

El almacenamiento de los datos, su recuperación y las velocidades de procesamiento son importantes, independientemente de qué tan buenas sean una aplicación y las estructuras de base de datos que utilice, si la aplicación es muy lenta y resulta inaceptable para el ambiente de negocio, entonces será un total fracaso. Por tanto podemos decir que el desempeño de una base de datos se puede ver afectado de forma negativa por una amplia variedad de factores como: la necesidad de uniones, las cuales pueden ser una solución a la necesidad de integración de los datos, calcular totales, operación que puede repetirse una y otra vez afectando el desempeño, grandes volúmenes de datos, requieren un cuidado especial para su almacenamiento y recuperación, llaves primarias difíciles de manejar, datos relacionados dispersos en el disco, la tarea de recuperación será más lenta que si estuvieran almacenados físicamente juntos en el disco, realizar demasiadas operaciones para el acceso a los datos puede ocasionar un cuello de botella, así mismo otorgar acceso liberal a los datos también perjudica su desempeño y puede ser un riesgo para la seguridad.

Por todo lo anterior es importante mejorar el desempeño del ambiente durante el tiempo de corrida, ya sea agregando índices, haciendo cambios importantes en las estructuras de las tablas o en su defecto en las especificaciones de configuración.

1.7.3.1 Cambios en los parámetros de configuración inicial.

A pesar de estar totalmente integrado con el servidor MySQL, el motor de almacenamiento InnoDB mantiene su propio pool de almacenamiento intermedio para tener un cache de datos e índices en la memoria principal. InnoDB almacena sus tablas e índices en un espacio de tablas *'tablespace'*, el cual puede consistir de varios ficheros (o particiones de disco).

Dos recursos basados en disco muy importantes que gestiona el motor de almacenamiento InnoDB son sus ficheros de datos de espacios de tablas y sus ficheros de registro (log). Si no se especifican opciones de configuración para InnoDB, MySQL crea en el directorio de datos de MySQL un fichero de datos de 10MB (autoextensible) llamado ibdata1 y dos ficheros de registro (log) de 5MB llamados ib_logfile0 y ib_logfile1, como se muestra en la figura 12.

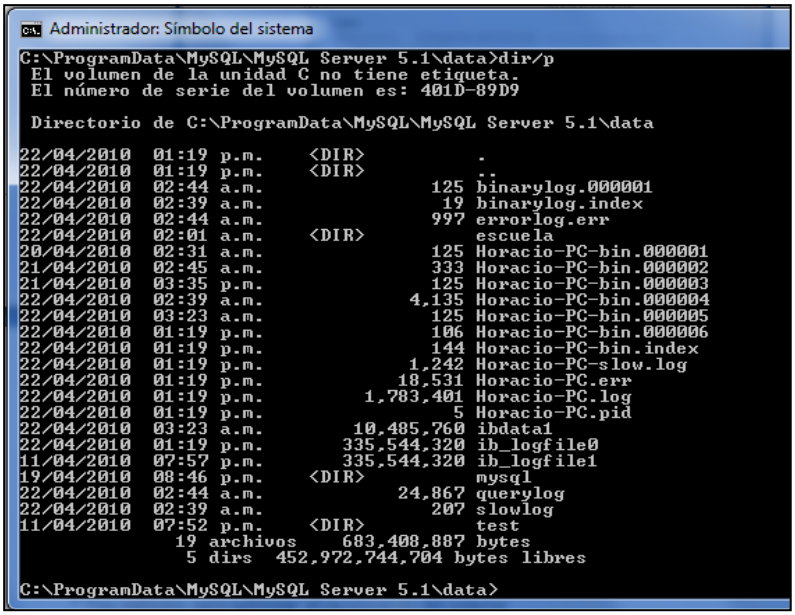


Figura 12. Ficheros de datos y de registro (log) de MySQL.

Para configurar los ficheros de espacio de tablas de InnoDB, debe utilizarse la opción innodb_data_file_path en el fichero my.ini, en Windows, o por medio de la interfaz grafica de MySQL Administrator. Por ejemplo, la siguiente es una configuración que creará explícitamente un espacio de tablas con las mismas características que el predeterminado, en el archivo de configuración my.ini:

 # Uncomment the following if you are using InnoDB tables
#innodb_data_home_dir = C:\mysql\data/
innodb_data_file_path = ibdata1:10M:autoextend
#innodb_log_group_home_dir = C:\mysql\data/
 # You can set .._buffer_pool_size up to 50 - 80 %
# of RAM but beware of setting memory usage too high
#innodb_buffer_pool_size = 16M
#innodb_additional_mem_pool_size = 2M
 # Set .._log_file_size to 25 % of buffer pool size
innodb_log_file_size = 5M
#innodb_log_buffer_size = 8M
#innodb_flush_log_at_trx_commit = 1
#innodb_lock_wait_timeout = 50

De esta manera se configura un único fichero de 10MB llamado ibdata1 el cual es autoextensible. No se suministra la ubicación del fichero, por lo tanto, el directorio predeterminado es el directorio de datos de MySQL. El tamaño del fichero se especifica empleando como sufijo las letras M o G para indicar unidades de MB o GB.

A continuación se dará la sintaxis para configurar un espacio de tablas que contiene un fichero de datos de tamaño fijo de 50MB llamado ibdata1 y un fichero autoextensible de 50MB llamado ibdata2, ambos en el directorio de datos:

Innodb_data_file_path=ibdata1:50M;ibdata2:50M:autoextend

Todo lo anterior está relacionado con las especificaciones que podemos realizar en el archivo de configuración. Otra de las acciones que podemos realizar para mejorar el rendimiento de nuestro sistema es la creación de índices, el uso de tablas temporales, la asignación adecuada de tipos de datos en las tablas, y por supuesto una de las más recomendadas, utilizar la versión más reciente de MySQL disponible.

1.7.3.2 Uso de tablas temporales

Cuando estamos trabajando con tablas muy grandes, suele suceder que ocasionalmente necesitemos ejecutar algunas consultas sobre un pequeño subconjunto de una gran cantidad de datos. En vez de ejecutar estas consultas sobre la tabla completa y hacer que MySQL encuentre cada vez los pocos registros que necesitamos, puede ser mucho más rápido seleccionar dichos registros en una tabla temporal y entonces ejecutar nuestras consultas sobre esta tabla.

Crear una tabla temporal es tan sencillo como agregar la palabra TEMPORARY a una sentencia típica CREATE TABLE, la sintaxis es la siguiente:

```
CREATE TEMPORARY TABLE tabla_temp
(
  campo1 tipoDato,
  campo2 tipoDeDato,
  ...
);
```

Una tabla temporal existe mientras dure la conexión a MySQL. Cuando se interrumpe la conexión MySQL remueve automáticamente la tabla y libera el espacio que ésta usaba. Nosotros podemos por supuesto eliminar esta tabla mientras estamos conectados a MySQL. Si una tabla nombrada tabla_temp ya existe en nuestra base de datos al momento de crear una tabla temporal con el mismo nombre, la tabla temporal oculta a la tabla no temporal.

MySQL también permite especificar que una tabla temporal sea creada en memoria si dicha tabla se declara del tipo MEMORY:

```
CREATE TEMPORARY TABLE tabla_temp
(
  campo1 tipoDato,
  campo2 tipoDeDato,
  ...
) TYPE = MEMORY;
```

Ya que las tablas del tipo MEMORY son almacenadas en memoria, las consultas sobre estas tablas son ejecutadas mucho más rápido que en las tablas en disco no temporales. Por tal motivo podemos duplicar la tabla que se quiera utilizar definiéndole o cambiándole el tipo de *Engine=Memory*, para realizar solamente consultas, ya que una vez que se pare el servicio de MySQL los datos de esta tabla desaparecerán. La sintaxis para duplicar la tabla con todo y su contenido es la siguiente:

*CREATE TEMPORARY TABLE nom_tabla_nueva AS SELECT * FROM nom_tabla_existente;*

ALTER TABLE nom_tabla_existente ENGINE=MEMORY;

Ahora lo único que queda es probar si con las tablas temporales nuestras consultas se ejecutan más rápidamente que usando la tabla que contiene una gran cantidad de datos, si no es así y los datos están bien indexados puede que las tablas temporales no sean de mucha utilidad.

Se crea la tabla temporal con el tipo Engine = Memory para probar si las consultas son más rápidas. La consulta que se realiza en la tabla alumnos tiene un tiempo de ejecución de (.0110 s) ver figura 13, y la consulta que se realiza en la tabla temporal tiene un tiempo de ejecución de (.0084 s) ver figura 14, con esto comprobamos que el uso de tablas temporales es útil al realizar consultas con gran cantidad de datos.

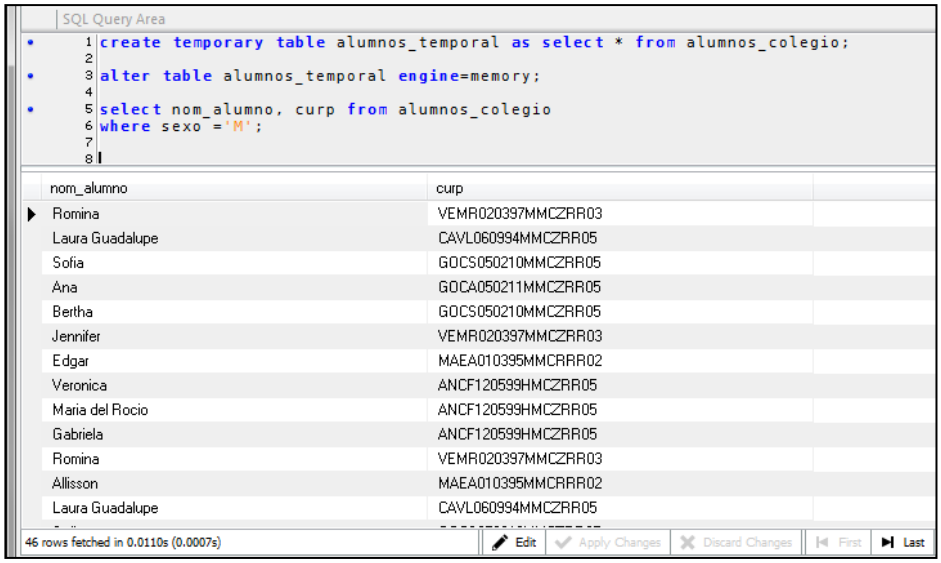


Figura 13. Consulta realizada en tabla no temporal (*alumnos_colegio*).

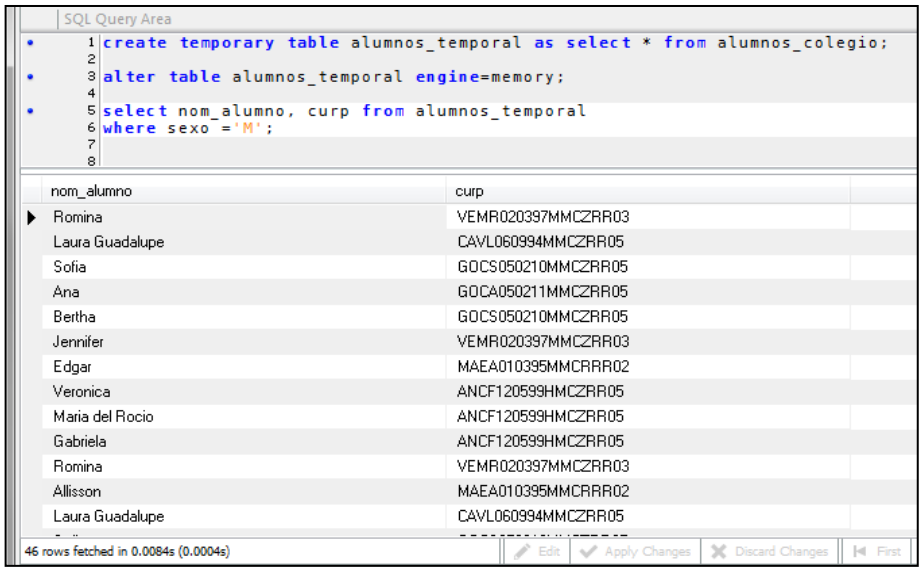


Figura 14. Consulta realizada en tabla temporal (*alumnos_temporal*).

1.7.3.3 Creación de índices

Los índices son un sistema especial que utilizan las bases de datos para mejorar su rendimiento global. Dado que los índices hacen que las consultas se ejecuten más rápido, podemos estar incitados a indexar todas las columnas de nuestras tablas. Sin embargo, lo que tenemos que saber es que el usar índices tiene un precio. Cada vez que hacemos un INSERT, UPDATE, REPLACE, o DELETE sobre una tabla, MySQL tiene que actualizar cualquier índice en la tabla para reflejar los cambios en los datos.

El usar o no índices depende de manera simple, de qué tipo de consultas ejecutamos y que tan frecuentemente lo hacemos, además de muchas otras cosas. La razón para tener un índice en una columna es para permitirle a MySQL que ejecute las búsquedas tan rápido como sea posible (y evitar los escaneos completos de tablas). Podemos pensar que un índice contiene una entrada por cada valor único en la columna, en el índice, MySQL debe contar cualquier valor duplicado. Estos valores duplicados decrementan la eficiencia y la utilidad del índice. Así que antes de indexar una columna, debemos considerar que porcentaje de entradas en la tabla son duplicadas. Si el porcentaje es demasiado alto, seguramente no veremos alguna mejora con el uso de un índice.

Otra cosa a considerar es qué tan frecuentemente los índices serán usados. MySQL puede usar un índice para una columna en particular si dicha columna aparece en la cláusula WHERE en una consulta. Si muy rara vez usamos una columna en una cláusula WHERE, seguramente no tiene mucha sentido indexar dicha columna. De esta manera, probablemente sea más eficiente sufrir el escaneo completo de la tabla las raras ocasiones en que se use esta columna en una consulta, que estar actualizando el índice cada vez que cambien los datos de la tabla.

Ante la duda, no tenemos otra alternativa que probar. Siempre podemos ejecutar algunas pruebas sobre los datos de nuestras tablas con y sin índices para ver como obtenemos los resultados más rápidamente. Lo único a considerar es que las pruebas sean lo más realistas posibles. Estas pruebas se realizarán en el apartado de Análisis de Consultas, en el cual se considerarán todos los puntos analizados en el Diseño Físico.

1.7.3.4 Cambios en las estructuras de las tablas

Nombres de la base de datos, tablas y columnas: El último paso del diseño de la base de datos es adoptar determinadas convenciones de nombres. Aunque MySQL es muy flexible en cuanto a la forma de asignar nombre a las bases de datos, tablas y columnas, se deben considerar algunas reglas.

- Utilizar caracteres alfanuméricos.
- Limitar los nombres a menos de 64 caracteres (es una restricción de MySQL).
- Utilizar el guión bajo (_) para separar palabras.
- Utilizar palabras en minúsculas.
- Los nombres de las tablas deberían ir en plural y los nombres de las columnas en singular.
- Utilizar las letras ID en las columnas de clave primaria y foránea.
- En una tabla, colocar primero la clave primaria seguida de las claves foráneas.
- Los nombres de los campos deben ser descriptivos de su contenido.
- Los nombres de los campos deben ser unívocos entre tablas, excepción hecha de las claves.

La figura 15, muestra la descripción de la tabla *nivel_educativo* y sus columnas, tomando en cuenta las consideraciones antes mencionadas.

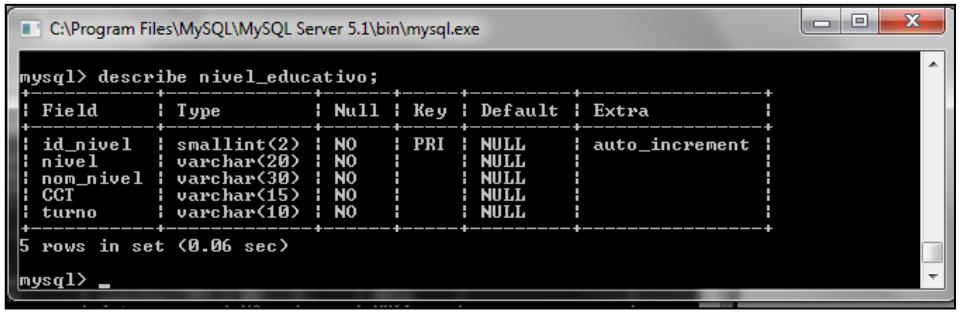


Figura 15. Descripción de la tabla *nivel_educativo* y sus columnas

Lo más importante de todas estas consideraciones es que la nomenclatura utilizada en nuestras bases de datos sea coherente y consistente con el fin de minimizar la posibilidad de errores al momento de crear una aplicación de bases de datos.

Almacenar solo la información necesaria: Algunas veces pensamos que agregar campos a las tablas de una base de datos una vez que han sido creadas es demasiado difícil, así que nos vemos impulsados a definir tantas columnas como se pueda. Bueno, esto simplemente es un concepto erróneo, ya que en MySQL podemos usar el comando ALTER TABLE para modificar la definición de una tabla en cualquier momento para que se adecue a nuestras necesidades cambiantes.

Por ejemplo, si en algún momento dado se necesitará agregar una columna de costo de colegiatura a nuestra tabla *nivel_educativo* (para obtener una estimación de cuanto es el ingreso por nivel de acuerdo a los alumnos inscritos), se podría hacer lo siguiente:

ALTER TABLE nivel_educativo ADD costo_nivel INTEGER;

La tabla con las modificaciones realizadas es la mostrada en la figura 16.

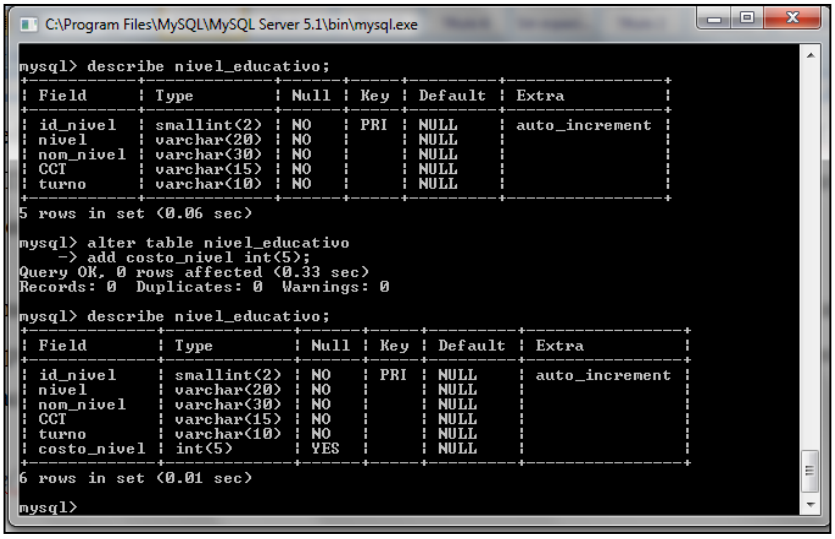


Figura 16. Modificaciones en las estructuras de las tablas.

Pedir solo lo necesario y ser explicito: Igual que decir "almacenar sólo lo necesario", esto puede parecer un poco más de sentido común, sin embargo, esto no suele ser considerado muy a menudo. Porque cuando una aplicación está en desarrollo los requerimientos suelen cambiar, de tal forma que muchas de las búsquedas terminan pareciéndose a esto:

*SELECT * FROM docente_colegio;*

(Seleccionar todo de la tabla *docente_colegio*)

Obtener todas las columnas de una tabla es simplemente lo más conveniente que podemos hacer cuando no estamos seguros de qué campos necesitamos. Sin embargo, a medida que las tablas crecen y cambian, esto puede convertirse en un problema de rendimiento. A la larga es mucho mejor tardarnos un tiempo extra después de nuestro desarrollo inicial y decidir exactamente qué es lo que necesitamos en nuestras búsquedas y especificar las columnas de forma explícita:

SELECT nom_docente, FROM docente_colegio;

Seleccionar el tipo de dato apropiado: Una vez identificadas todas las tablas y columnas que necesita la base de datos, debemos determinar el tipo de dato de cada campo. Existen tres categorías principales que pueden aplicarse prácticamente a cualquier aplicación de bases de datos:

- Texto
- Números
- Fecha y hora

Cada uno de éstos presenta sus propias variantes, por lo que la elección del tipo de dato correcto no sólo influye en el tipo de información que se puede almacenar en cada campo, sino que afecta al rendimiento global de la base de datos. A continuación se listan algunos consejos que se deben considerar para elegir un tipo de dato adecuado para las tablas:

- Identificar si una columna debe ser de tipo texto, numérico o de fecha. Valores eminentemente numéricos como códigos postales o cantidades monetarias deben tratarse como campos de texto si decidimos incluir sus signos de puntuación, pero obtendremos mejores resultados si los almacenamos como números y solucionamos la cuestión del formato de alguna otra forma.

- Elegir el subtipo más apropiado para cada columna. Los campos de longitud fija (como CHAR) son generalmente más rápidos que los de longitud variable (como VARCHAR), aunque ocupan más espacio en disco.

- El tamaño de cada campo debe restringirse al mínimo en función de cuál pudiera ser la entrada más grande. Por ejemplo, si el valor en una columna de tipo entero es menor de mil, lo mejor es configurar esta columna como un SMALLINT de tres dígitos sin signo (lo que permite exactamente 999 valores distintos).

- Configurar la longitud máxima para las columnas de texto y numéricas, así como otros atributos.

- El factor más importante es siempre ajustar al máximo la información de cada campo en lugar de usar siempre tipos TEXT e INT genéricos (e ineficientes).

Tomando en cuenta estas consideraciones se realizaran cambios en las estructuras de las tablas de la base de datos 'colegio'. Relacionadas directamente con la selección del tipo de dato más adecuado para algunos campos.

1) La tabla de datos alumnos tiene en la llave primaria un tipo de dato int(6), se cambiará por un mediumint (6), el campo de sexo se cambiará por ENUM ('H','M') reconociéndose 'H' para hombre y 'M' para mujer. Hay columnas que tienen valores fijos y se podrían declarar como char, pero no se pueden alternar tipos char y varchar en una tabla, por tanto se decide trabajar con el tipo varchar que aunque no es tan rápido como char para recuperar datos, ocupa menos espacio en memoria.

   Lo primero que tenemos que hacer es identificar el nombre de la llave foránea que se asocia con la llave primaria *id_alumno* de la tabla *alumnos_colegio* que se va a modificar, lo podemos hacer con el comando:

   SHOW CREATE TABLE calif_colegio;

   Desplegando la pantalla de la figura 17:

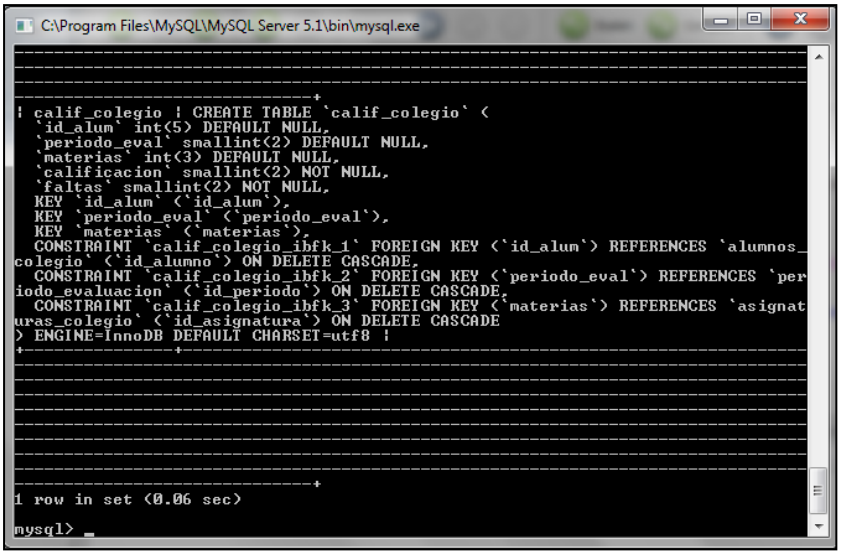


Figura 17. Pantalla del comando SHOW CREATE TABLE *calif_colegio*.

Una vez identificada (*calif_colegio_ibfk_1*)la eliminamos para poder hacer el cambio en la llave primaria, ya que de lo contrario no se podría realizar la acción debido a la integridad referencial que existe entre ellas. La eliminación la realizamos con el siguiente comando:

ALTER TABLE calif_colegio DROP FOREIGN KEY calif_colegio_ibfk_1;

Posteriormente se hace el cambio de tipo de dato de la llave primaria con el siguiente comando:

ALTER TABLE calif_colegio MODIFY id_alumno mediumint(6) not null auto_increment;

La figura 18 muestra las modificaciones realizadas en la tabla *alumnos_colegio*, la figura 19 muestra la tabla original y la figura 20 la tabla modificada.

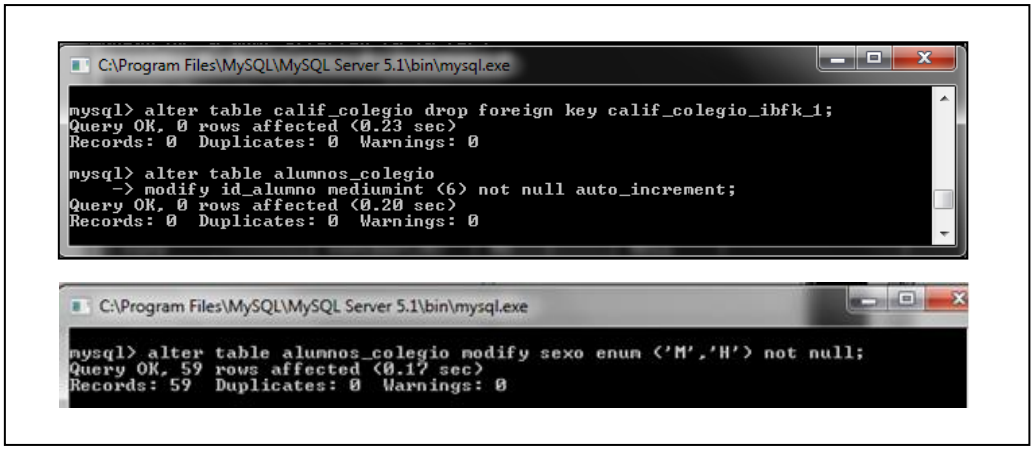


Figura 18. Modificaciones de la tabla *alumnos_colegio*.

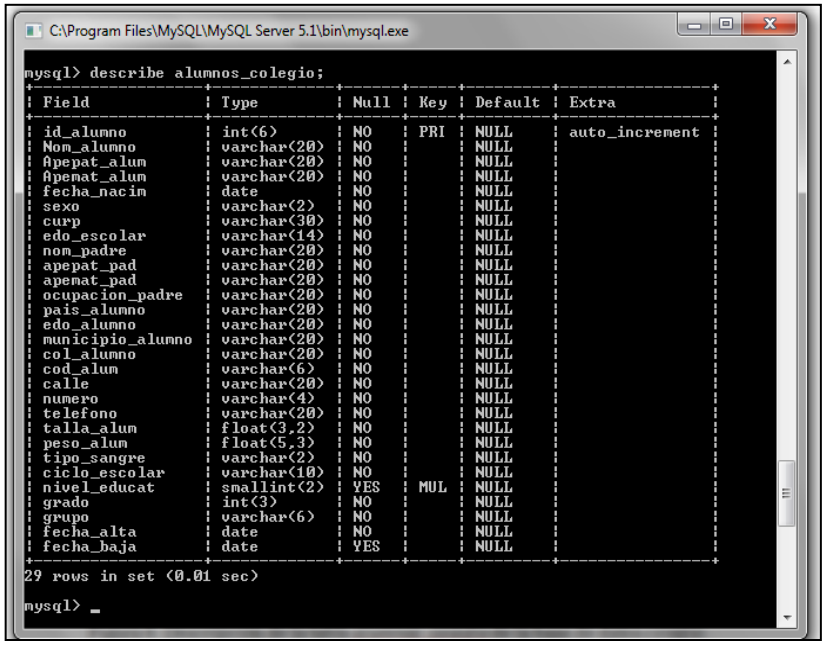


Figura 19.Tabla *alumnos_colegio* sin modificaciones.

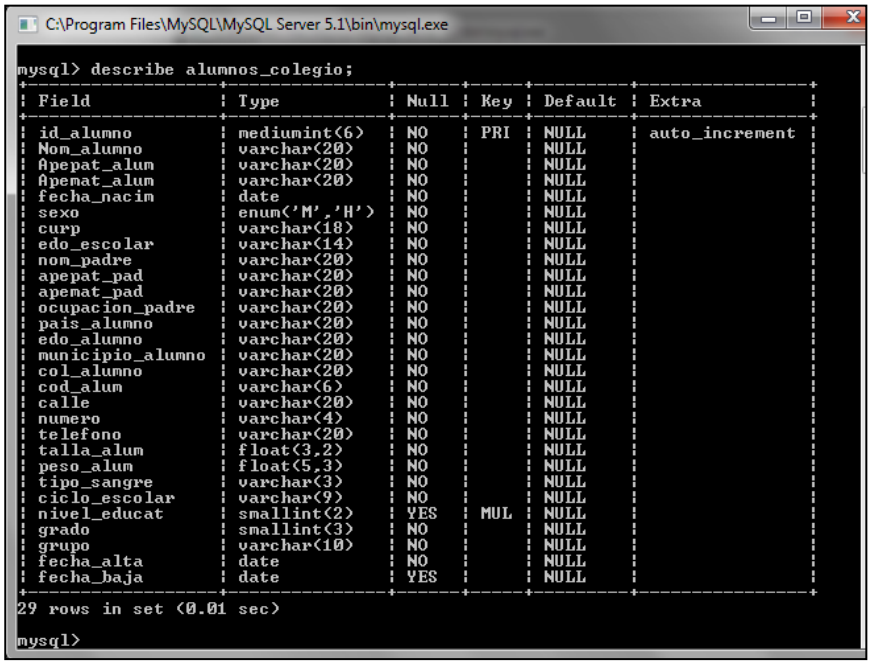


Figura 20. Tabla *alumnos_colegio* modificada.

Una vez modificado el tipo de dato en la llave primaria, procedemos a modificarlo en la llave foránea y asignamos nuevamente la condición de llave foránea a la columna con sus restricciones, ver figura 21.

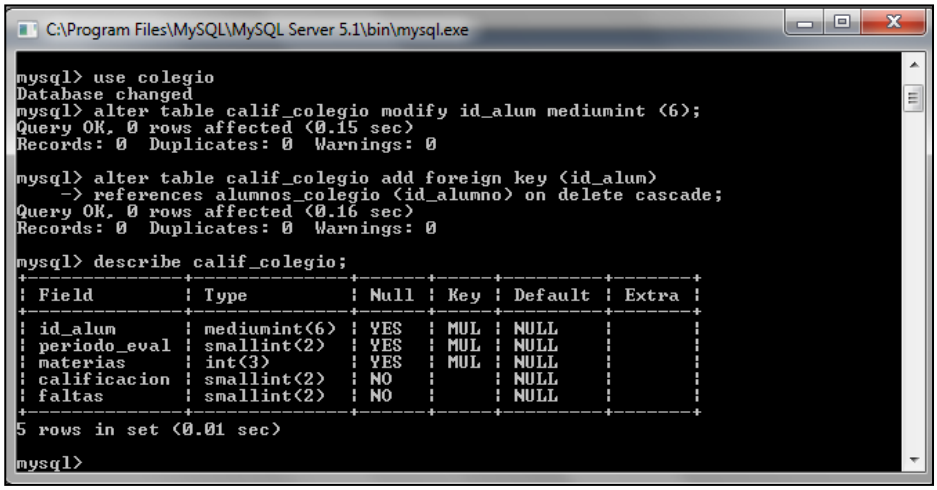


Figura 21. Modificación y asignación de la llave foránea en la tabla *calif_colegio*.

Otro de los campos que se modificará, es el de *grado* el cual se cambiará por un *smallint (3).* Para tener como resultado final la tabla de la figura 20.

2) El mismo procedimiento se realizará con la tabla *asignaturas_colegio* en la cual cambiaremos el tipo de dato de la llave primaria *id_asignatura int(3)* de la tabla original (figura 22), por *mediumint (5)*, eliminando primero las llaves foráneas asociadas (figura 23). De igual manera se cambiarán los tipos de datos de sus claves foráneas asociadas que se encuentran en la tabla *calif_colegio* y *asignatura_docente* (figura 24). Así la figura 25, muestra la llave primaria modificada.

```
mysql> use colegio;
Database changed
mysql> describe asignaturas_colegio;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id_asignatura | int(3)      | NO   | PRI | NULL    | auto_increment |
| nom_asig      | varchar(80) | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

mysql> _
```

Figura 22. Tabla original de *asignaturas_colegio*.

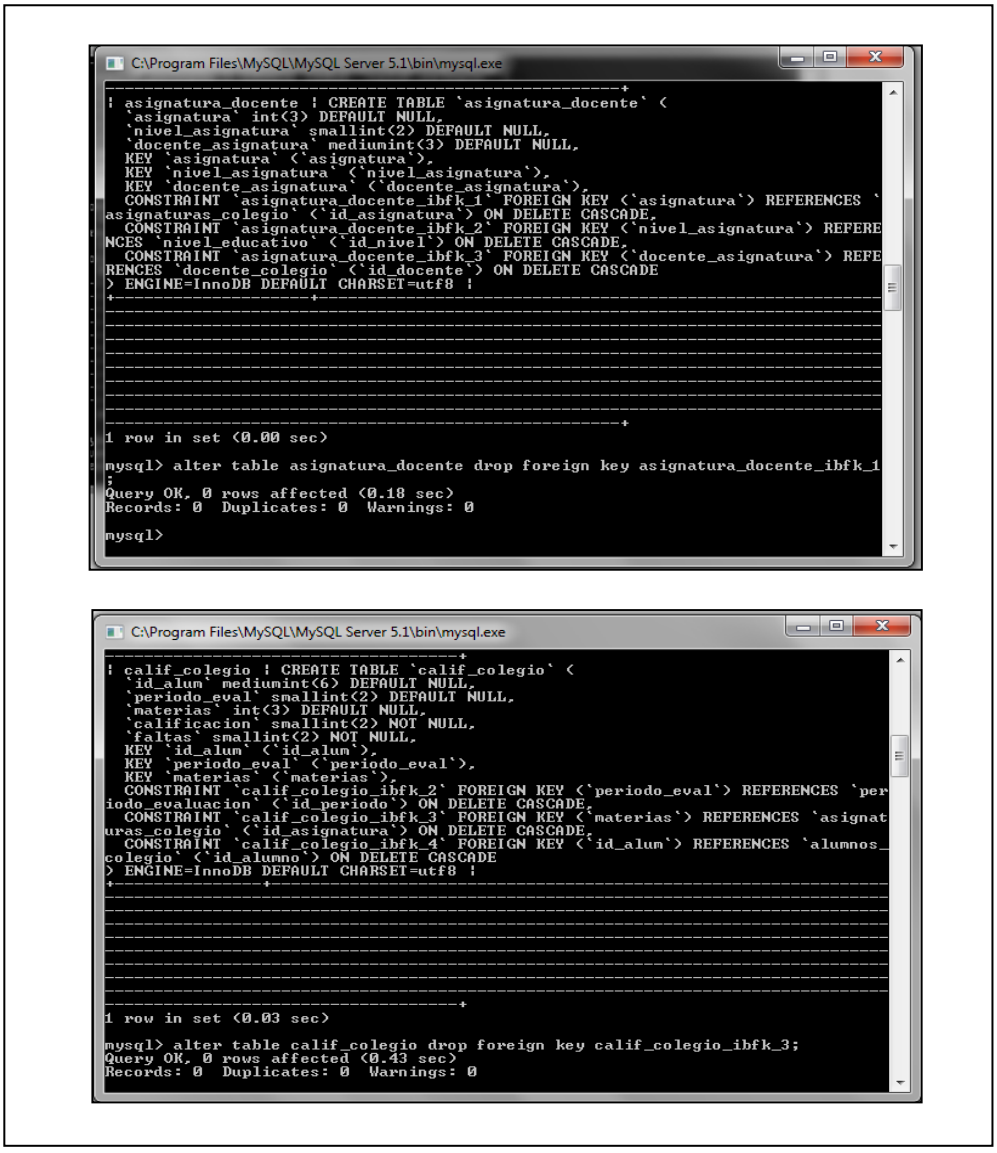


Figura 23. Eliminación de las llaves foráneas en la tabla *asignatura_docente y calif_colegio.*

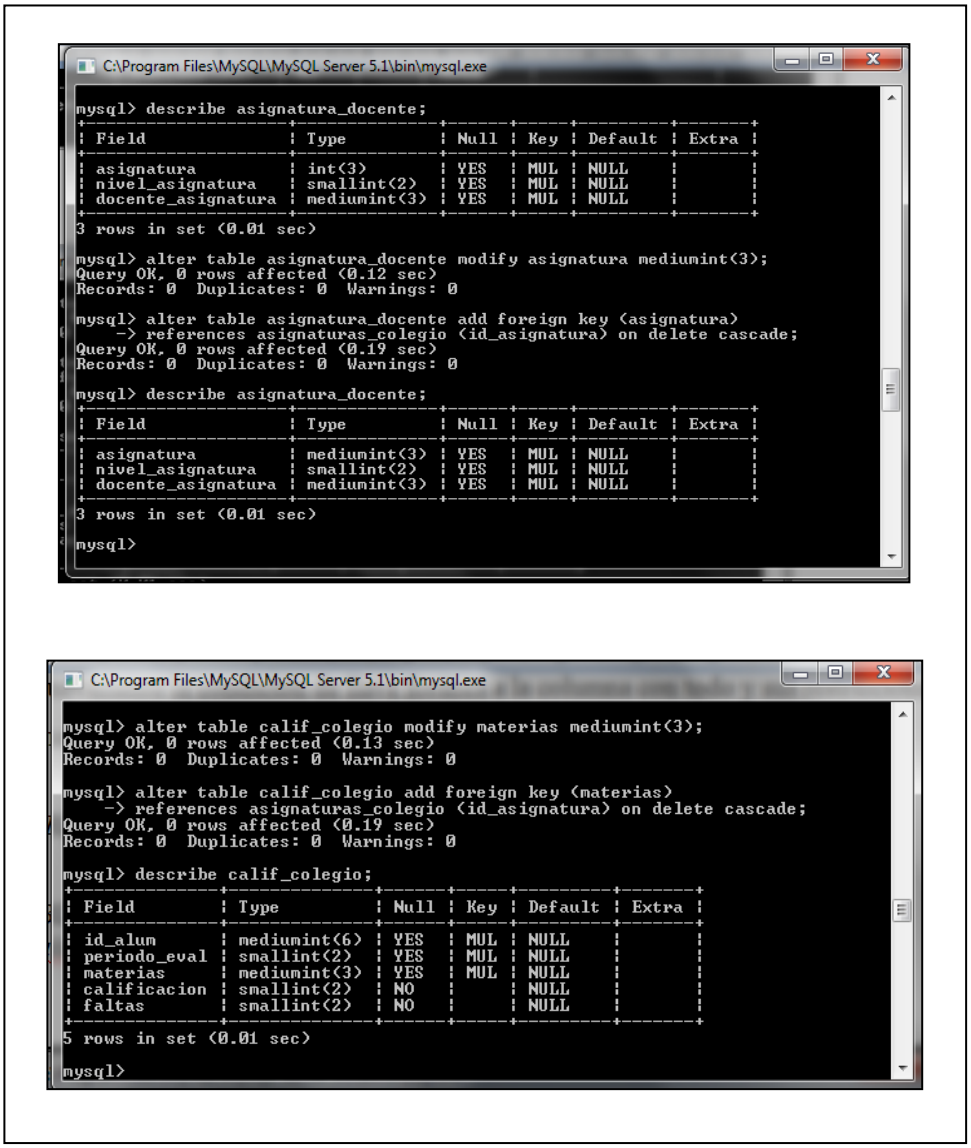


Figura 24. Modificación de los tipos de datos de las llaves foráneas en las tablas *asignatura_docente* y *calif_colegio.*

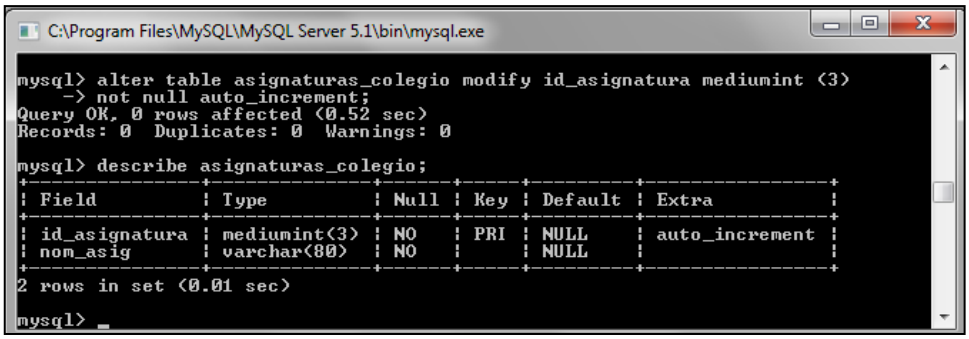


Figura 25. Llave primaria de la tabla *asignaturas_colegio* modificada.

1.8 Carga de datos a la Base de Datos

Una vez que se tiene la estructura de la base de datos creada, lo siguiente para poder utilizarla y crear consultas en ella, es que la base contenga datos. Para llevar a cabo este paso se utiliza la inserción de datos en una sentencia SQL, utilizando sentencias simples en donde se indica que datos y en que tablas de MySQL se van a insertar.

La sentencia a utilizar es la siguiente:

INSERT INTO nom_tabla VALUES (valor_columna1, valor_columna2, valor_columna3);

Los datos que se insertaron quedaron previamente definidos cuando se realizo el diseño conceptual de la base de datos y se definieron los requisitos datos del sistema.

Las sentencias que a continuación se muestran en la figura 26, se realizaron con la herramienta grafica de MySQL Query Browser, son inserciones de datos a las tablas de la base de datos creada.

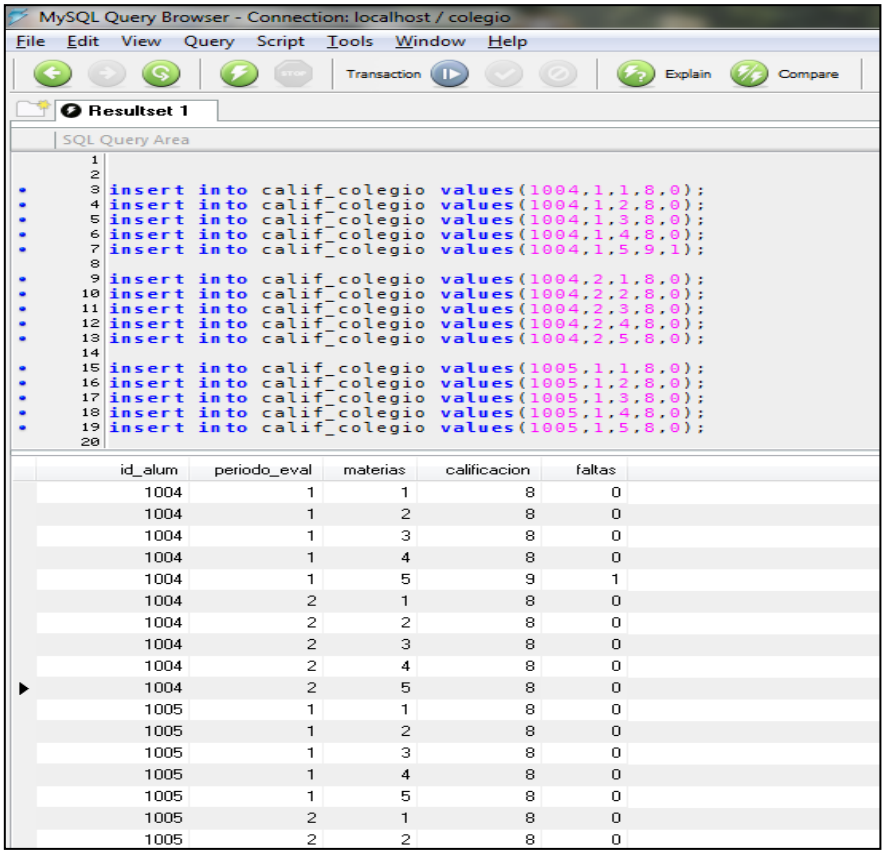


Figura 26. Inserción de datos a las tablas de la Base de Datos *'colegio'*.

En el ANEXO VI, se muestran las tablas de la Base de Datos con sus respectivos datos, las cuales servirán para realizar las consultas que requiere el sistema y que se tratarán en el siguiente apartado.

1.9 Pruebas y mantenimiento del sistema

El sistema de información debe probarse antes de utilizarlo. El costo es menor si se detectan los problemas antes de la entrega del sistema. En un principio, se hace una serie de pruebas, con datos tipo, para identificar las posibles fallas del sistema, más adelante, se utilizarán los datos reales.

El mantenimiento del sistema y de su documentación empieza justamente en esta etapa y después, esta función se realizará de forma rutinaria a lo largo de toda la vida del sistema. Las actividades de mantenimiento son una parte fundamental para las organizaciones que cuentan con una Base de Datos, ya que llegan a simplificar importantes sumas de dinero. Sin embargo, el costo del mantenimiento disminuye de manera importante cuando el analista aplica procedimientos sistemáticos en el desarrollo de los sistemas.

Para realizar las pruebas del sistema se crearán algunas consultas de acuerdo a los requisitos de los usuarios del sistema para detectar fallas o en su caso hacer las modificaciones pertinentes. Antes de crear las consultas debemos tomar en cuenta algunas características importantes de cómo realiza internamente este proceso el SMBD.

Las consultas a la base de datos son el típico punto de degradación de una aplicación tras varios meses o años de funcionamiento. Salvo que nuestra aplicación realice operaciones masivas de inserción de registros, debemos preocuparnos sólo de las operaciones de SELECT. Es en este tipo de operaciones donde notaremos como, poco a poco, se va degradando nuestra aplicación.

Este análisis es importante debido a que nos permite determinar la actividad del usuario de manera individual verificando las tendencias como las transacciones de cada usuario, aislar los problemas de performance cuando sea posible, identificar cuellos de botella (bottlenecks) y determinar si se puede mejorar el performance. Algunos métodos que se pueden realizar para conseguir lo siguiente son:

- Mejorar la estructura de las tablas.
- Mejorar la configuración del servidor MySQL.
- Mejorar las consultas.

Los dos primeros puntos se analizaron y trataron de cubrir en el Diseño Físico de la base de datos creada, con la finalidad de realizar mejoras en cuanto a cómo se guardarán los datos. El último punto se ejemplificará de mejor manera al hacer algunas consultas a la base de datos.

Las consultas se realizarán con ayuda de la herramienta grafica de MySQL Query Browser para visualizar de mejor manera las pantallas. Cabe mencionar y aclarar que se usarán datos tipo o ficticios para la realización de las pruebas aunque manteniendo cierta relación con lo reales, solo se utilizará un 20% de los datos que se pudieran llegar a almacenar en la base de datos, ya que a partir de este porcentaje se puede estimar cual será su rendimiento.

1) Esta consulta nos devuelve los alumnos de un nivel determinado, ordenados por grado, grupo y apellido paterno al principio (figura 27).

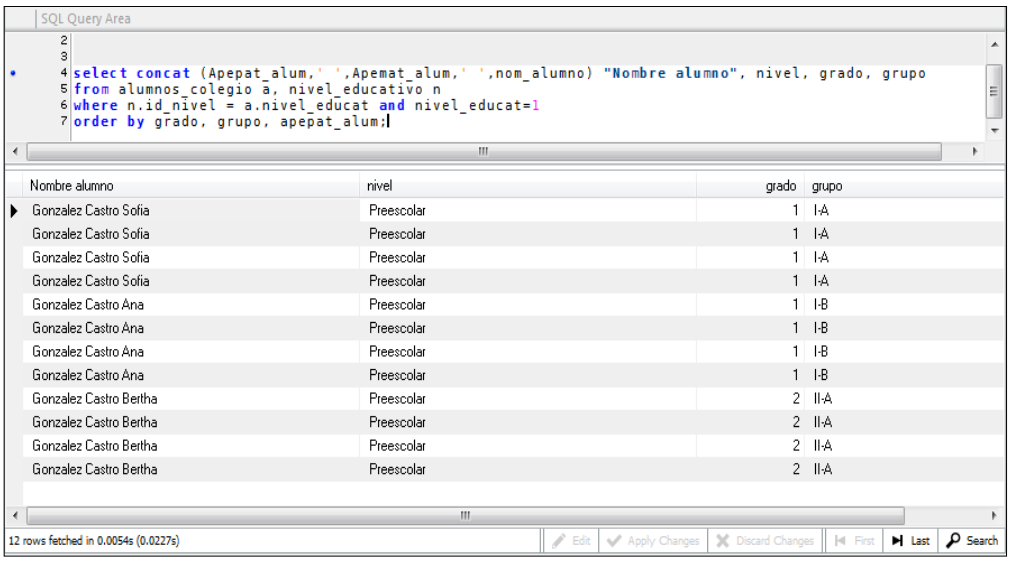


Figura 27. Consulta 1.

2) Esta consulta devuelve la edad de los alumnos y el sexo, ordenados por nivel y el grado que cursan cada uno de ellos (figura 28).

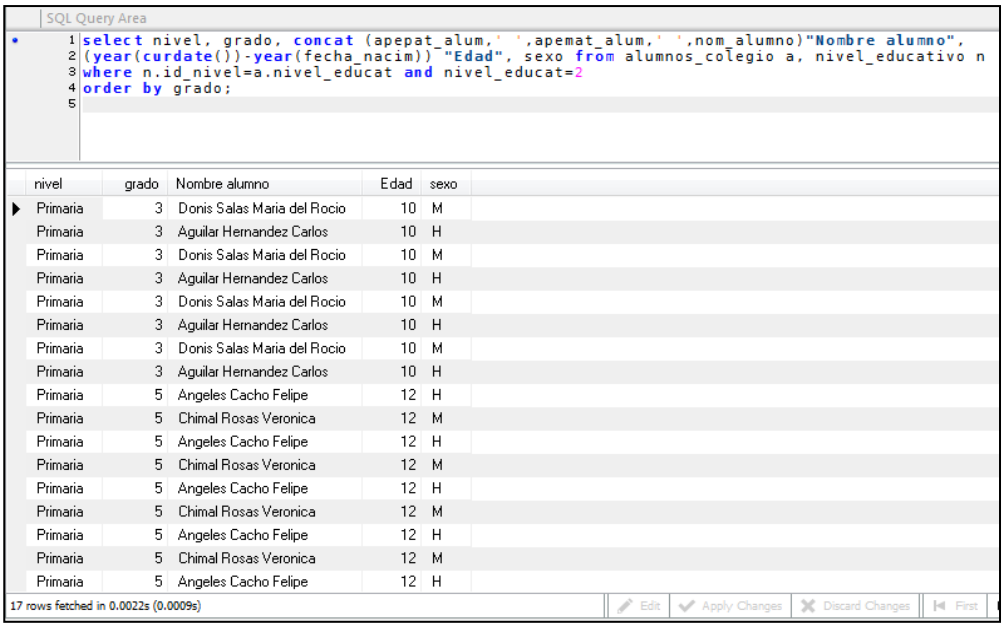


Figura 28. Consulta 2.

3) Esta consulta devuelve el nombre del alumno y su CURP, ordenados por nivel y el grado que cursan cada uno de ellos (figura 29).

```
SQL Query Area
1 select nivel,grado, concat (apepat_alum,' ',apemat_alum,' ',nom_alumno)"Nombre alumno",
2 curp from alumnos_colegio a, nivel_educativo n
3 where n.id_nivel=a.nivel_educat and nivel_educat=3
4 order by grado;
5
```

nivel	grado	Nombre alumno	curp
Secundaria	2	Velazquez Morales Romina	VEMR020397MMCZRR03
Secundaria	2	Martinez Escobedo Allisson	MAEA010395MMCRRR02
Secundaria	2	Pinto Oropeza Jennifer	VEMR020397MMCZRR03
Secundaria	2	Mendoza Casas Edgar	MAEA010395MMCRRR02
Secundaria	2	Velazquez Morales Romina	VEMR020397MMCZRR03
Secundaria	2	Martinez Escobedo Allisson	MAEA010395MMCRRR02
Secundaria	2	Pinto Oropeza Jennifer	VEMR020397MMCZRR03
Secundaria	2	Mendoza Casas Edgar	MAEA010395MMCRRR02
Secundaria	2	Velazquez Morales Romina	VEMR020397MMCZRR03
Secundaria	2	Velazquez Morales Romina	VEMR020397MMCZRR03
Secundaria	2	Martinez Escobedo Allisson	MAEA010395MMCRRR02
Secundaria	2	Pinto Oropeza Jennifer	VEMR020397MMCZRR03
Secundaria	2	Mendoza Casas Edgar	MAEA010395MMCRRR02
Secundaria	2	Velazquez Morales Romina	VEMR020397MMCZRR03
Secundaria	2	Martinez Escobedo Allisson	MAEA010395MMCRRR02
Secundaria	2	Mendoza Casas Edgar	MAEA010395MMCRRR02
Secundaria	2	Pinto Oropeza Jennifer	VEMR020397MMCZRR03
Secundaria	3	Gonzalez Garcia Gabriela	ANCF120599HMCZRR05
Secundaria	3	Padilla Gamero Miguel	ANCF120599HMCZRR05
Secundaria	3	Gonzalez Garcia Gabriela	ANCF120599HMCZRR05
Secundaria	3	Gonzalez Garcia Gabriela	ANCF120599HMCZRR05

25 rows fetched in 0.0061s (0.0010s) Edit Apply Changes Discard Changes First

Figura 29. Consulta 3.

4) Esta consulta devuelve las calificaciones y faltas de cada una de las materias por periodo de evaluación de los alumnos de un determinado nivel (figura 30).

```
SQL Query Area
1 select concat (apepat_alum,' ',apemat_alum,' ',nom_alumno)
2 "Nombre alumno", periodo_eval, nom_asig "asignaturas", calificacion, faltas
3 from alumnos_colegio a, calif_colegio c, periodo_evaluacion p,
4 asignaturas_colegio ac
5 where p.id_periodo= c.periodo_eval
6 and a.id_alumno=c.id_alum
7 and ac.id_asignatura =c.materias;
```

Nombre alumno	periodo_eval	asignaturas	calificacion	faltas
Gonzalez Castro Sofia	1	Español Kin	8	0
Gonzalez Castro Sofia	1	Matematicas Kin	8	0
Gonzalez Castro Sofia	1	Ambiente kin	8	0
Gonzalez Castro Sofia	1	Escritura kin	8	0
Gonzalez Castro Sofia	1	Inglés Kin	9	1
Gonzalez Castro Sofia	2	Español Kin	8	0
Gonzalez Castro Sofia	2	Matematicas Kin	8	0
Gonzalez Castro Sofia	2	Ambiente kin	8	0
Gonzalez Castro Sofia	2	Escritura kin	8	0
Gonzalez Castro Sofia	2	Inglés Kin	8	0
Gonzalez Castro Ana	1	Español Kin	8	0
Gonzalez Castro Ana	1	Matematicas Kin	8	0
Gonzalez Castro Ana	1	Ambiente kin	8	0
Gonzalez Castro Ana	1	Escritura kin	8	0
Gonzalez Castro Ana	1	Inglés Kin	8	0
Gonzalez Castro Ana	2	Español Kin	8	0
Gonzalez Castro Ana	2	Matematicas Kin	8	0
Gonzalez Castro Ana	2	Ambiente kin	8	0

45 rows fetched in 0.0073s (0.0009s) Edit Apply Changes Discard Changes

Figura 30. Consulta 4.

5) Esta consulta devuelve los promedios y las faltas totales de cada uno de los alumnos de acuerdo al periodo de evaluación determinado (figura 31).

SQL Query Area

```
select id_alumno as Matricula,concat (apepat_alum,' ',apemat_alum,' ',nom_alumno) as Nombre,
periodo_eval as Bimestre, avg(calificacion) as Promedio,
sum(faltas) as Faltas
from alumnos_colegio a, calif_colegio c, periodo_evaluacion p,
asignaturas_colegio ac
where p.id_periodo= c.periodo_eval
and a.id_alumno=c.id_alum
and ac.id_asignatura =c.materias
and periodo_eval=1
group by id_alumno;
```

matricula	Nombre	Bimestre	Promedio	Faltas
1004	Gonzalez Castro Sofia	1	8.2000	1
1005	Gonzalez Castro Ana	1	8.0000	0
1006	Gonzalez Castro Bertha	1	8.0000	0
1018	Gonzalez Castro Sofia	1	8.0000	0
1019	Gonzalez Castro Ana	1	8.0000	0
1020	Gonzalez Castro Bertha	1	8.0000	0
1032	Gonzalez Castro Sofia	1	8.0000	0

7 rows fetched in 0.0021s (0.0012s)

Figura 31. Consulta 5.

6) Esta consulta devuelve los nombres de los docentes así como las asignaturas que imparten en cada uno de los niveles (figura 32).

SQL Query Area

```
select concat (apepat_doc,' ',apemat_doc,' ',nom_docente) as Docente,
nom_asig as asignatura, nivel
from docente_colegio d, asignatura_docente ad, asignaturas_colegio ac, nivel_educativo n
where n.id_nivel=ad.nivel_asignatura
and d.id_docente=ad.docente_asignatura
and ac.id_asignatura =ad.asignatura
order by apepat_doc;
```

Docente	asignatura	nivel
Castañeda Diaz Milton3	Educacion Fisica Sec	Secundaria
Castañeda Diaz Milton4	Proceso administrativo	Lic. Adm de Negocios
Contreras DonJuan Gonzalo2	Matematicas I Sec	Secundaria
Contreras DonJuan Gonzalo3	Metodología de la investigación	Lic. Adm de Negocios
Contreras DonJuan Mario2	Ciencias Prim	Primaria
Contreras DonJuan Mario3	Proyectos Institucionales Prep	Preparatoria
Mendoza Cruz Rosalba3	Ciencias III Sec	Secundaria
Mendoza Cruz Rosalba4	Derecho laboral	Lic. Cien de la Educ
Mendoza Cruz Rosalba2	Español Kin	Preescolar
Rivera Garcia Horacio3	Comunicacion y Sociedad Prep	Preparatoria
Rivera Garcia Horacio4	Investigación de mercados	Lic. Cien de la Educ

11 rows fetched in 0.0051s (0.0009s)

Figura 32. Consulta 6.

7) Esta consulta devuelve los nombres de los directores de cada nivel educativo (figura 33).

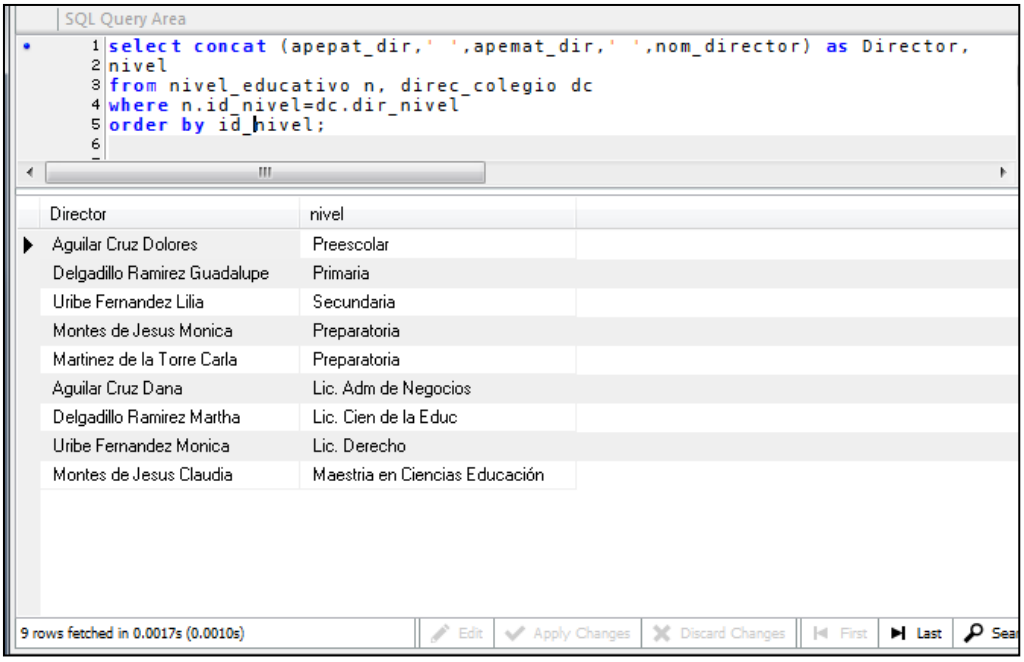


Figura 33. Consulta 7.

La creación de índices tratada en el diseño físico se puede aplicar en este momento para hacer consultas mucho más rápidas, teniendo los siguientes beneficios:

- Es menos costoso insertar, modificar y eliminar en un índice que hacerlo en toda una tabla ordenada.
- Puedo tener varios índices en una tabla, lo que permite ordenar la tabla por varios criterios a la vez.
- Los índices ocupan espacio en disco y memoria, y las operaciones de inserción, modificación y borrado serán un poco más lentas (habrá que actualizar índices), por lo que no debemos indexar todas las columnas, sino sólo las más usadas en búsquedas.
- Cuanto más pequeño es el índice más rápidas serán las operaciones.

MySQL nos ofrece diferentes tipos de índices de los cuales podemos elegir el que más nos convenga por ejemplo:

- Parciales: en lugar de indexar todo el campo solo se determina una parte de este campo.
- Multicolumna: varias columnas pueden ser indexadas y parcialmente también.
- Clustered: que son característicos de las tablas InnoDB y están definidos por la Primary Key, es decir el índice esta incrustado en los datos y se almacenan ordenados por clave primaria, es decir una vez encontrada la entrada del índice ya se han encontraron los datos.

- Full-text: que se crean sobre las palabras de un campo de alguna tabla, esta opción proporciona soporte a búsquedas textuales.

Podemos añadirlos a una tabla después de crearla:

ALTER TABLE nombre_tabla ADD INDEX nombre_indice (columna_indexada);

Si queremos eliminar un índice:

ALTER TABLE tabla_nombre DROP INDEX nombre_indice ;

Los índices permiten mayor rapidez en la ejecución de las consultas a la base de datos tipo SELECT ... WHERE. La regla básica para crear tus índices es hacerlo sobre aquellas columnas que vayas a usar con una cláusula WHERE, y no crearlos con aquellas columnas que vayan a ser objeto de un SELECT:

SELECT nom_alumno from alumnos_colegio WHERE nivel= 'Secundaria';

En este ejemplo, la columna *nivel* es buena candidata a un índice; *nom_alumno,* no.

Otra regla básica es que son mejores candidatas a indexar aquellas columnas que presentan muchos valores distintos, mientras que no son buenas candidatas las que tienen muchos valores idénticos, como por ejemplo sexo (H o M) porque cada consulta implicará siempre recorrer prácticamente la mitad del índice.

Algunas sentencias se ejecutan mucho más eficientemente si hay índices:

```
SELECT concat (apepat_dir,' ',apemat_dir,' ',nom_director) as Director, nivel
FROM nivel_educativo n, direc_colegio dc
WHERE n.id_nivel=dc.dir_nivel
ORDER BY id_nivel;
```

Para este caso la llave primaria *id_nivel* es el índice incrustado.

Sin índices -> producto cartesiano + filtrado
Con índices -> A + (filtrado A.a = B.b) + (filtrado B.b = C.c)

Las tablas InnoDB utilizan índices B-Tree incrustados. Esto las hace muy rápidas en la búsqueda por clave primaria. Además, al tratarse del índice, es frecuente que la página con la clave y los datos ya se encuentren en la caché del disco.

En la mayoría de las consultas las columnas más empleadas en la clausula *WHERE* son el *id_alumno* y el *id_nivel*, que por consecuencia son las llaves primarias de cada una de sus respectivas tablas, analizando lo anterior no se considera necesaria la creación de algún otro índice, por el momento, ya que todo dependerá de las necesidades de los usuarios y de la mejora del tiempo de respuesta que se requiera una vez que empiecen a trabajar con el sistema.

En cuanto al mantenimiento del sistema, es una tarea que se realiza constantemente a lo largo de toda la vida activa de la base de datos, una herramienta que nos proporciona la interfaz gráfica de MySQL, es MySQL Administrator por medio del cual podemos monitorear el funcionamiento de la Base de Datos, aunque podemos hacer uso de algún otro software que nos proporcione mayor seguridad, pero realizando algunas búsquedas casi ninguno es gratuito, por tanto se aprovechará la herramienta de administración de MySQL Administrator para realizar esta importante tarea, ver figura (34).

Entre las tareas que nos permite hacer MySQL Administrator están:

- Iniciar o detener el servicio.
- Modificar las variables de inicio del archivo de configuración de MySQL de acuerdo a las necesidades que vaya teniendo el sistema.
- Asignar usuarios y privilegios.
- Controlar y supervisar los servicios de conexión.
- Supervisar la información del cliente de MySQL.
- Podemos visualizar todas las acciones sobre la Base de Datos, archivos (logs).
- Hacer respaldos y restauraciones de la Base de Datos, si ocurre alguna falla.
- Visualizar los catálogos de tablas, índices, vistas y procedimientos de la Base de Datos.

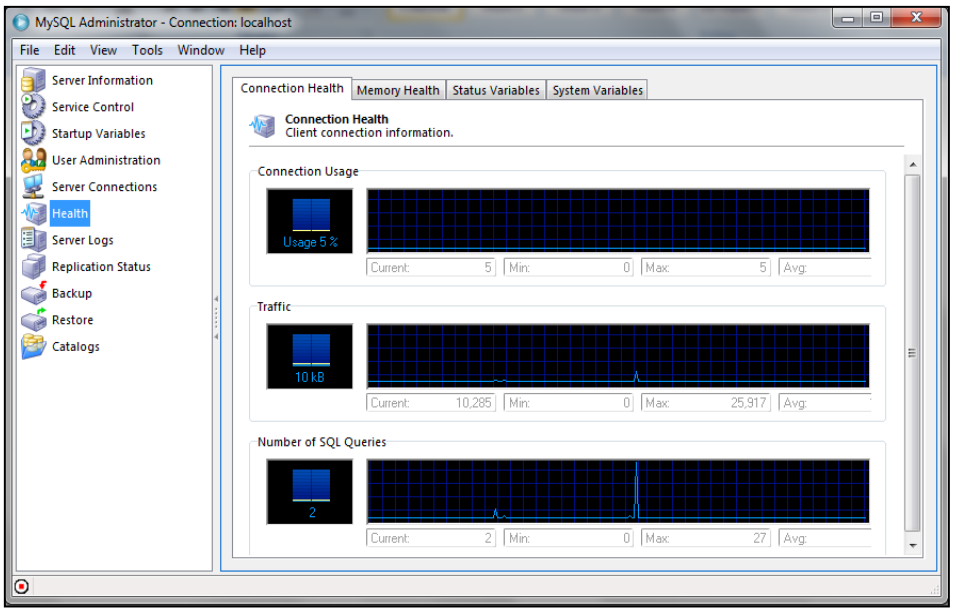


Figura 34. Herramienta grafica MySQL Administrator.

1.10 Controles de Seguridad (Cuentas de usuarios)

El control de seguridad en una base de datos es fundamental, de esta manera determinamos que usuarios podrán acceder a ella y que privilegios tendrán sobre cada tabla de la base de datos.

Se podría posponer la aplicación de contraseñas hasta más tarde, para que no sea necesario ingresarlas mientras se desarrollan tareas adicionales de configuración o prueba. Sin embargo, hay que asegurarse de establecerlas antes de poner la instalación en trabajo de producción real.

Una cuenta en MySQL está compuesta por un nombre de usuario, el equipo o equipos desde los que se puede conectar al servidor y una contraseña. Hay algunas diferencias entre los usos de nombres de usuario y contraseña que hacen MySQL y el sistema operativo que son importantes y que se deben considerar. Por ejemplo:

- Los nombres de usuario, que usa MySQL para autentificación, y los que usa Windows o cualquier otro sistema operativo, no tienen nada que ver, el tratar de acceder con el mismo nombre de usuario es solo por conveniencia, por no olvidarlo o no confundirse. No hay conexión entre estos nombres de usuario se usa uno para acceder al sistema operativo y otro diferente o en su defecto igual (aunque no recomendable) para acceder a MySQL.

- Nombre de usuarios en MySQL pueden tener una longitud máxima de 16 caracteres. Este límite está especificado y determinado en los servidores y clientes MySQL, por tanto tratar de modificarlo en las tablas en la base de datos mysql no funcionará. Cada uno de los sistemas operativos define el tamaño de los nombres de usuario, por tanto son independientes uno del otro.

Por tanto se cambiará el password de los usuarios creados por default en el SMBD y aquellos que se pudieran heredar por el Sistema Operativo se bloquearan para evitar problemas de seguridad y accesos no autorizados por gente que tiene conocimiento de estas cuentas, determinaremos los roles y privilegios de cada uno de los usuarios de la Base de Datos. Para una mejor administración de los usuarios serán asignados los privilegios de acuerdo a las actividades que desempeñen dentro del sistema, ver tabla 1.

PRIVILEGIOS DE USUARIOS

Usuario	Privilegio
DBA (usuario root)	Todos los privilegios.
Director General de la escuela (nombre_dirgral)	Conectarse y consultar algunas vistas de la base de datos *'colegio'*.
Encargados de la administración de información personal y académica de los alumnos de cada nivel (Control Escolar de Preescolar y Primaria, Secretaria Técnica de Secundaria, Prefectura de Bachillerato y Servicios Escolares de Licenciatura) (nom_encargado_1), (nom_encargado_2), (nom_encargado_3), (nom_encargado_n).	Conexión, selección, borrado, inserción y actualización de datos en las tablas *alumnos_colegio* y *calif_colegio*.
Docentes de cada asignatura (nombre_docente_1), (nombre_docente_2), (nombre_docente_3), (nombre_docente_n).	Conexión, consulta e inserción de datos sobre la tabla de calificaciones *calif_colegio*.

Tabla 1. Privilegios de los usuarios de la base de datos *'colegio'*.

1) Se crearán las cuentas de cada usuario, evitando que sean compartidas para que se pueda realizar una auditoría a cada usuario como política de seguridad.

2) La autenticación se realizará mediante password, el cual deberá ser mínimo de 8 caracteres incluyendo letras, números y caracteres especiales, cambiándose cada mes sin repetir por lo menos las 10 últimas utilizadas y no utilizar palabras comunes o conocidas. Haciendo hincapié a los usuarios del sistema de no tener a la vista de todos estas claves que son confidenciales y que pueden poner en riesgo la seguridad de la Base de Datos permitiendo un acceso no autorizado.

3) Las funciones de cada uno de los usuarios de la Base de Datos deben de ser concretas de acuerdo a su rol, para no entrar en conflicto con las otras áreas.

La creación de usuarios y asignación de privilegios la podemos realizar desde la línea de comandos de mysql o desde la interfaz grafica MySQL Administrator, de igual forma podemos definir el número máximo de querys, de actualizaciones y de conexiones que pueden tener los usuarios, ver figura 35.

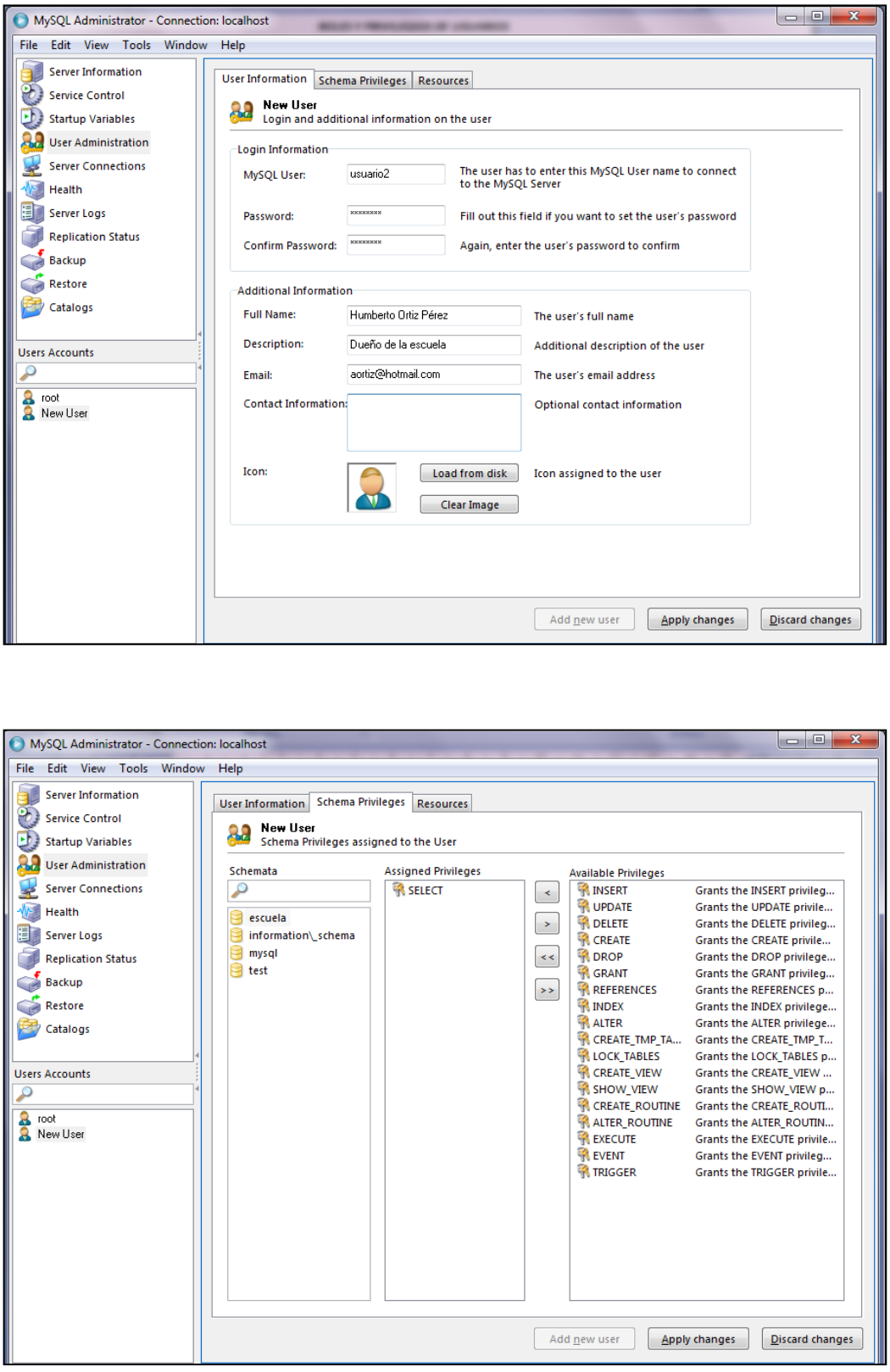


Figura 35. Creación de cuentas de usuario y asignación de privilegios.

1.11 Implementación y evaluación del sistema.

Esta última etapa del desarrollo del sistema, no se pudo concluir de manera satisfactoria, debido a la falta de apoyo y autorización por parte del Director General de la Institución, lamentablemente el proyecto quedo de alguna manera truncado, pero el trabajo realizado garantiza en gran parte que si no se logro implementar en la organización para la cual fue creada, se puede implementar en otra, considerando las necesidades de la misma o adecuando lo que se tiene, para poder finalizar y ver concluido el trabajo.

Aunque la evaluación del sistema se plantea como parte integrante de la última etapa del ciclo de desarrollo de los sistemas, realmente la evaluación toma parte en cada una de las etapas. Uno de los criterios fundamentales que debe satisfacerse, es que el futuro usuario utilice el sistema desarrollado.

C O N C L U S I O N E S

Podemos concluir que diseñar eficientemente una Base de Datos nos asegura el buen funcionamiento de ella, si en determinado momento no se definieron de manera correcta los parámetros, se puede realizar un proceso correctivo con el fin de tener en un estado 'óptimo' a la Base de Datos, lo cito así porque lo óptimo dependerá de cada negocio en el que se implemente el sistema.

Una vez detectados cuales son los procesos preventivos y reactivos que se tienen que hacer, se deben monitorear los recursos tanto de la Base de Datos como del Sistema, lo cual puede hacerse con herramientas adicionales especializadas en ello, que soportan diferentes tipos de SMBD o manualmente con las bitácoras diarias, teniendo todo documentado, para que en caso de realizarse algún cambio no se tenga ninguna duda de lo que ya se realizo para solucionarlo.

Otro de los aspectos que se debe considerar importante en todos los sistemas es el análisis de las consultas, lo cual puede tener un impacto de riesgo en la disponibilidad del servicio, por tanto es imprescindible realizarlo y aplicar las mejores estrategias en las estructuras de la Base, para obtener una mejor respuesta.

Algo que no se debe pasar por alto es la política de seguridad lo cual garantiza la integridad de los datos, el equipo, los objetos de la Base de Datos, ya que el activo más importante para toda organización que maneja información, son los datos. Con esto limitamos a los usuarios a no provocar daños en el sistema teniendo acceso solo a la información que necesitan.

Lamentablemente por razones personales y laborales, no se pudo dar término a este proyecto, quedando inconclusa la implementación de la base de datos en la Institución para la cual se diseño, dejando como experiencia personal que el desarrollo de un proyecto se debe valorar en la medida del tiempo empleado por parte del factor humano así como de los recursos materiales, no obstante el hecho de no haberse implementado no quiere decir que todo el trabajo realizado no sea importante o no se pueda emplear para otro proyecto o adecuarlo a otra organización, ya que se tienen las bases para realizar cualquier otro sistema.

Con todos estos puntos se da por terminado el análisis y diseño del trabajo en cuestión, que en lo particular me satisface dejando un aprendizaje significativo no solo en el ámbito de las bases de datos, sino en el desarrollo profesional y laboral de mi carrera.

ANEXO I

ENTREVISTAS REALIZADAS A LOS USUARIOS

ENTREVISTAS PARA LA IMPLEMENTACION DE UN SISTEMA DE PROCESAMIENTO Y ADMINISTRACION DE INFORMACION ESCOLAR

NOMBRE: Ma. Dolores Aguilar Cruz FECHA: 4/09/09

AREA: Dirección Primaria PUESTO: Directora

1. ¿Considera que la escuela debe migrar a otra forma de administrar y procesar la información escolar que recibe? ¿Por qué?
   Sí, urge para no capturar tantos datos repetidos veces.
2. ¿De qué manera se lleva el registro de todos los alumnos que se inscriben a la escuela?
   Manualmente y después en formatos de computadora.
3. ¿Si tuviera la posibilidad de implementar un sistema que le facilitará toda la información escolar que necesidades tendría que cubrir?
   todas, pero las más importantes, serían el evitar trabajar doble o repetir actividades, tener al momento todo lo solicitado, etc.
4. ¿Qué información tendría que estar almacenada en este sistema?
   Datos del alumno, Calificaciones, datos del docente, los más importantes.
5. ¿Cuáles departamentos considera que deben estar involucrados en el sistema?
   Caja, Contabilidad, Recursos Humanos, Control Escolar y Direcciones.
6. ¿Qué departamentos estarían involucrados con su área?
   Control Escolar.
7. ¿Qué información debe ser compartida con todos los usuarios del sistema?
   los datos personales del alumno y tal vez sus calificaciones para determinar su avance académico.
8. ¿Qué personal tendría el manejo directo de este sistema?
   Caja, Contabilidad, Control Escolar.
9. ¿Qué reportes, plantillas o documentos necesita que el sistema emita?
   Todos los que se puedan emitir o desglosar de toda la información personal o académica de los alumnos.
10. ¿Cuál es la disponibilidad que la escuela tiene para invertir en un proyecto que actualice el procesamiento de la información escolar? ¿Presupuesto aproximado?
    Esa determinación concierne a Dirección General.
11. ¿Qué ventajas traería para la escuela la adquisición de un sistema como el que se menciona?
    Una escuela de calidad y a la vanguardia con las técnicas informáticas.

Realizadas por: LIZETH BERENICE GARCIA PEREZ

ENTREVISTAS PARA LA IMPLEMENTACION DE UN SISTEMA DE PROCESAMIENTO Y ADMINISTRACION DE INFORMACION ESCOLAR

NOMBRE: Mónica Montes de Jesús FECHA: 4/09/09

AREA: Preparatoria PUESTO: Directora

1. ¿Considera que la escuela debe migrar a otra forma de administrar y procesar la información escolar que recibe? ¿Por qué?

   Si, porque lo necesita.

2. ¿De qué manera se lleva el registro de todos los alumnos que se inscriben a la escuela?

   Manual y luego en la PC.

3. ¿Si tuviera la posibilidad de implementar un sistema que le facilitará toda la información escolar que necesidades tendría que cubrir?

   Optimizar actividades de captura y manipulación de datos.

4. ¿Qué información tendría que estar almacenada en este sistema?

   Datos del alumno, Calificaciones, estadisticas que se pudieran generar de esta información.

5. ¿Cuáles departamentos considera que deben estar involucrados en el sistema?

   Los más importantes: Direcciones, Control Escolar, Caja.

6. ¿Qué departamentos estarían involucrados con su área?

   Control Escolar

7. ¿Qué información debe ser compartida con todos los usuarios del sistema?

   Datos de los alumnos.

8. ¿Qué personal tendría el manejo directo de este sistema?

   Caja, Contabilidad, Control Escolar y tal vez Recursos Humanos.

9. ¿Qué reportes, plantillas o documentos necesita que el sistema emita?

   Todos los que faciliten el trabajo administrativo.

10. ¿Cuál es la disponibilidad que la escuela tiene para invertir en un proyecto que actualice el procesamiento de la información escolar? ¿Presupuesto aproximado?

    No puedo dar respuesta a esta pregunta, concierne directamante con Dirección General.

11. ¿Qué ventajas traería para la escuela la adquisición de un sistema como el que se menciona?

    Muchas, entre las mejores un mejor servicio.

Realizadas por: LIZETH BERENICE GARCIA PEREZ

ENTREVISTAS PARA LA IMPLEMENTACION DE UN SISTEMA DE PROCESAMIENTO Y ADMINISTRACION DE INFORMACION ESCOLAR

NOMBRE: Olvera Martinez Martha FECHA: 28/Agosto/2009

AREA: Contabilidad PUESTO: Encargada

1. ¿Considera que la escuela debe migrar a otra forma de administrar y procesar la información escolar que recibe? ¿Por qué?
   Si, por el crecimiento que ha tenido en cuestión de los niveles escolares, además de dar mejor calidad de servicio al plantel.
2. ¿De qué manera se lleva el registro de todos los alumnos que se inscriben a la escuela?
   De forma manual y posteriormente con algunos formatos hechos por la propia encargada de esta actividad.
3. ¿Si tuviera la posibilidad de implementar un sistema que le facilitará toda la información escolar que necesidades tendría que cubrir?
   - Dar un servicio de calidad a toda la comunidad
   - Evitará duplicidad de actividades
   - Un manejo eficaz y rápido de la información (manipulación datos)
4. ¿Qué información tendría que estar almacenada en este sistema?
   - Información de alumnos, docentes, situación académica alumnos, pagos, y edición de formatos como resultado de toda esta información.
5. ¿Cuáles departamentos considera que deben estar involucrados en el sistema?
   Recursos Humanos, Control Escolar, Direcciones de c/nivel, Contabilidad, Caja, Centro de Computo.
6. ¿Qué departamentos estarían involucrados con su área?
   Recursos Humanos y tal vez Pagos (Caja).
7. ¿Qué información debe ser compartida con todos los usuarios del sistema?
   Situación de pago de los alumnos, estadísticas cuantitativas y cualitativas del desempeño académico de los alumnos.
8. ¿Qué personal tendría el manejo directo de este sistema?
   personal de control escolar, cajo, recursos humanos, contabilidad y de centro computo para las publicaciones en la pag. web.
9. ¿Qué reportes, plantillas o documentos necesita que el sistema emita?
   Para el área de contabilidad 8 recibos de nómina, concentrado de ingresos y egresos por mes.
10. ¿Cuál es la disponibilidad que la escuela tiene para invertir en un proyecto que actualice el procesamiento de la información escolar? ¿Presupuesto aproximado?
    La mejor disponibilidad, el costo estaría a consideración del proyecto que se presente y que satisfaga las necesidades.
11. ¿Qué ventajas traería para la escuela la adquisición de un sistema como el que se menciona?
    - Prestigio, Calidad de servicio, Reducción de actividades duplicadas, Optimización del tiempo de respuesta en trámites a la comunidad interna y externa.

Realizadas por: LIZETH BERENICE GARCIA PEREZ

ENTREVISTAS PARA LA IMPLEMENTACION DE UN SISTEMA DE PROCESAMIENTO Y ADMINISTRACION DE INFORMACION ESCOLAR

NOMBRE: Carmen Martinez de la Torre FECHA: 3/09/09

AREA: Dirección Administrativa. PUESTO: Directora Adm.

1. ¿Considera que la escuela debe migrar a otra forma de administrar y procesar la información escolar que recibe? ¿Por qué?

   Si, porque el crecimiento de la escuela ya lo requiere.

2. ¿De qué manera se lleva el registro de todos los alumnos que se inscriben a la escuela?

   Manual y después lo vacían en formato de computadora.

3. ¿Si tuviera la posibilidad de implementar un sistema que le facilitará toda la información escolar que necesidades tendría que cubrir?

   Reducir el tiempo de actividades que se duplicarán por varias áreas.

4. ¿Qué información tendría que estar almacenada en este sistema?

   Toda la personal y académico de los alumnos, ademas de lo relacionado a los pagos colegiatura y a docentes.

5. ¿Cuáles departamentos considera que deben estar involucrados en el sistema?

   Todos, aunque solo algunos van a manipular directamente los datos y otros solo consultarán la información deseada.

6. ¿Qué departamentos estarían involucrados con su área?

   Mi área solo seña de consulta, no trabajaña directamente con los datos.

7. ¿Qué información debe ser compartida con todos los usuarios del sistema?

   Los datos personales del alumno.

8. ¿Qué personal tendría el manejo directo de este sistema?

   Control escolar, Caja, Contabilidad, Recursos Humanos, Direcciones de cada nivel.

9. ¿Qué reportes, plantillas o documentos necesita que el sistema emita?

   Recibos de colegiatura, de docentes, expedientes personales, boletas, Calificaciones parciales, Constancias, Kardex, etc.

10. ¿Cuál es la disponibilidad que la escuela tiene para invertir en un proyecto que actualice el procesamiento de la información escolar? ¿Presupuesto aproximado?

    Estará relacionada con el presupuesto del proyecto y los requerimientos del mismo.

11. ¿Qué ventajas traería para la escuela la adquisición de un sistema como el que se menciona?

    Una escuela de mejor calidad de servicio a toda la comunidad.

Realizadas por: LIZETH BERENICE GARCIA PEREZ

ENTREVISTAS PARA LA IMPLEMENTACION DE UN SISTEMA DE PROCESAMIENTO Y ADMINISTRACION DE INFORMACION ESCOLAR

NOMBRE: Nadia Morales Morales FECHA: 4/09/09

AREA: Control Escolar Primaria PUESTO: Encargada

1. ¿Considera que la escuela debe migrar a otra forma de administrar y procesar la información escolar que recibe? ¿Por qué?
   Si, para facilitar todos los procedimientos de la captura de la información escolar.
2. ¿De qué manera se lleva el registro de todos los alumnos que se inscriben a la escuela?
   Manual, y posteriormente se vacían a formatos hechos por el área misma.
3. ¿Si tuviera la posibilidad de implementar un sistema que le facilitará toda la información escolar que necesidades tendría que cubrir?
   Optimización de recursos, de actividades, de tiempos de respuesta para brindar un mejor servicio.
4. ¿Qué información tendría que estar almacenada en este sistema?
   Datos personales y academicos del alumno, Calificaciones parciales y finales, Datos personales del docente. etc.
5. ¿Cuáles departamentos considera que deben estar involucrados en el sistema?
   Control Escolar, Direcciones de c/nivel, Caja, Recursos Humanos, Contabilidad, etc.
6. ¿Qué departamentos estarían involucrados con su área?
   Caja y Direcciones de c/nivel.
7. ¿Qué información debe ser compartida con todos los usuarios del sistema?
   Primordialmente los datos del alumno, ya que la demás información es propia de cada área o nivel.
8. ¿Qué personal tendría el manejo directo de este sistema?
   Caja, Contabilidad, Control Escolar, Direcciones de cada nivel y tal vez Centro de Compúto.
9. ¿Qué reportes, plantillas o documentos necesita que el sistema emita?
   Listas de asistencia, Boletas, Kardex, Constancias, Carnet de datos personales, recibos de pago de colegiatura y pago a docentes.
10. ¿Cuál es la disponibilidad que la escuela tiene para invertir en un proyecto que actualice el procesamiento de la información escolar? ¿Presupuesto aproximado?
    Eso lo determina el área de Dirección General y el area de Contabilidad.
11. ¿Qué ventajas traería para la escuela la adquisición de un sistema como el que se menciona?
    De un mejor servicio a toda la comunidad en general.

Realizadas por: LIZETH BERENICE GARCIA PEREZ

ANEXO II

BOLETA INTERNA DE CALIFICACIONES

Colegio Mexicano Los Angeles
SECUNDARIA PARTICULAR 0344
C.C.T. 15PES0833G
TURNO MATUTINO

TARJETA DE CALIFICACIONES
Ciclo Escolar 2008 – 2009

Nombre del alumno: **ADAME HERNANDEZ JESSICA MADAI**

Grado: **1°** Grupo: **A**

Bimestre: **PRIMERO**

Asignatura	Español	Matemáticas	Ciencias I	Geografía	Inglés	Ed. Física	Computación	Artes	Asignatura Estatal	Orientación	Promedio
Calificación	7	9	8	9	8	8	10	6	9	B	8,22
Inasistencia				1	1				1		

Huehuetoca Estado de México a 15 de octubre de 2008.

Lic. Maritza del Rocío Díaz Pulido
DIRECTORA SECUNDARIA

Part. No. 0344 "Colegio Mexicano Los Angeles"

Lista de Alumnos — Ciclo Escolar 2008-2009

NIVEL SECUNDARIA — TERCER GRADO

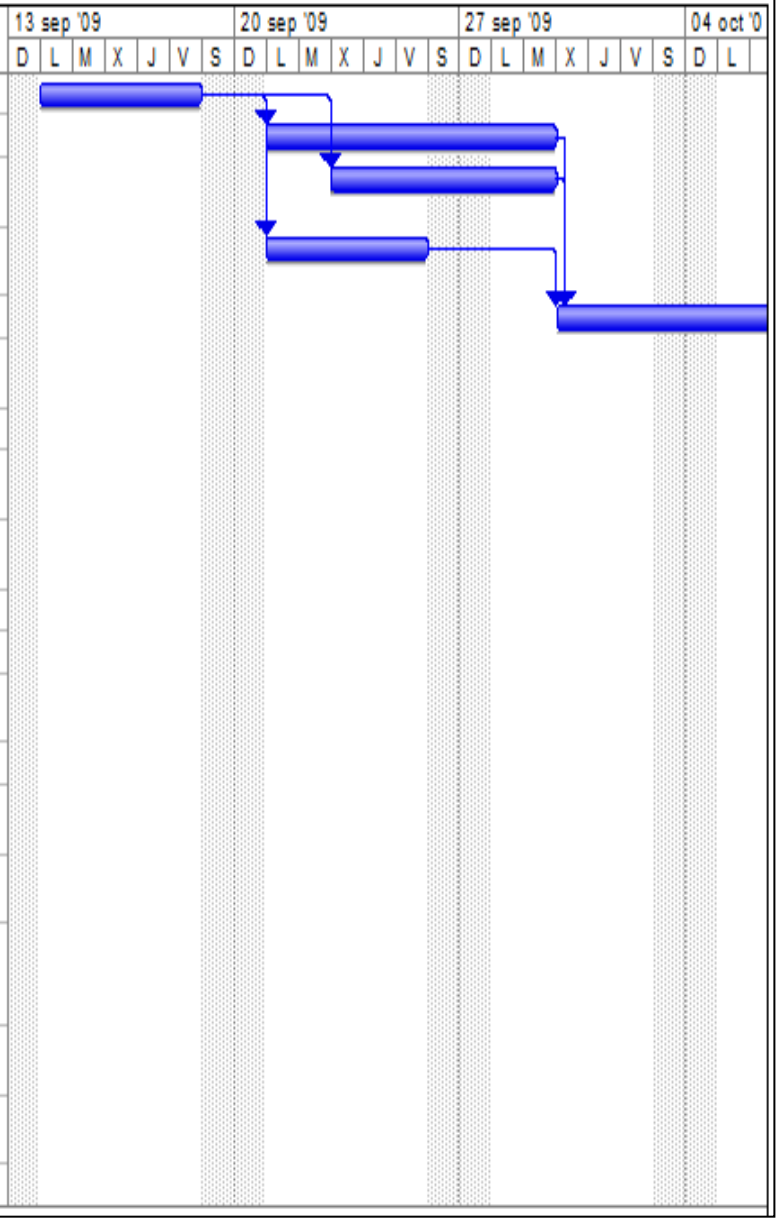
N.P.	NOMBRE DEL ALUMNO
1	CAMPOS ISLAS ALINE ITZEL
2	CANO VELAZQUEZ LAURA GUADALUPE
3	CHAVEZ TEJADA GABRIELA
4	CORONA FERIA SANDRA BELINDA
5	CUELLAR GRAJALES ARIADNA XIMENA
6	FLORES OLIVARES STEVE DYLAN
7	FUERTE VILLEGAS ULISES RAFAEL
8	GARCIA CRISTOBAL GUADALUPE
9	GARCIA GONZALEZ ILSE DANIELA
10	GONZALEZ SANTILLAN GABRIELA
11	HERNANDEZ GARRIDO ALLIN
12	LIRA VILLEGAS ALEJANDRO
13	LLANOS LANDA MIRIAM
14	LOPEZ PLIEGO EFREN
15	MARQUEZ DE LA SANCHA MEUSSA
16	MARTINEZ LUNA ABIGAHIL
17	MARTINEZ PORTILLO BRAYAN ARIEL
18	PAREDES VARGAS ARIANI MARIT
19	PINEDA LEDESMA CARLOS
20	QUIÑONES MORENO CARLA MARIA
21	REYNA GONZALEZ TATIANA LEILANI
22	RODRIGUEZ ALVAREZ LUIS DAVID
23	TREJO GONZALEZ VERONICA LIZBETH
24	VALENTINO ALVA JUAN
25	VAZQUEZ PEREZ LILIAN MARLENE
26	VELAZQUEZ PINEDA CYNTHIA LIZBETH
27	VELAZQUEZ ROJAS JOSE DE JESUS
28	YAÑEZ FLORES FERNANDO
29	ZARAGOZA ABARCA ALLISON JANETH

CONCENTRADO DE EVALUACIONES

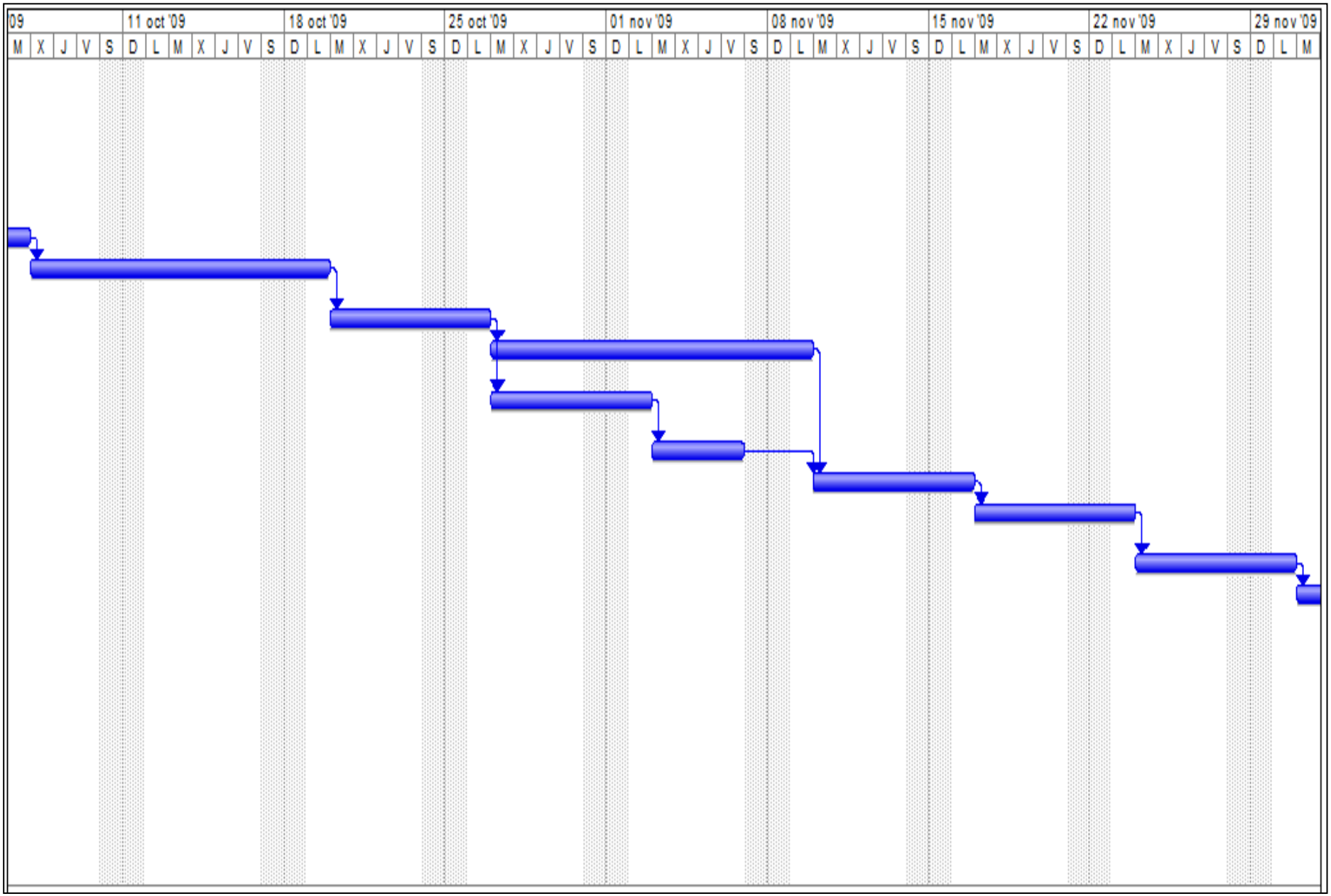
N.P.	NOMBRE DEL ALUMNO	ESPAÑOL	ESP	MATEMATICAS	MAT.	CIENCIAS	CIEN.	GEOGRAFIA	GEOGR.	INGLES	ING.	ED. FISICA	ED. FIS.	COMPUTACION	COMP	ARTES	ARRT	ASIGNATURA ESTATAL	ASIGN	ORIENTACION	ORIENT.	PROMEDIO	PROM.
1	AVILES ARRIAGA RICARDO	6		6		7	2	5	1	5	2	8		8		6		5	1	M	2	6.22	6,22
2	BUSTOS RUIZ DAMIAN	10		10		10		9		9		10		10		8		9		B	1	9.44	9.44
3	CABALLERO ASTORGA ZAIRA YANEL	9		6	1	8		7		8		10		10		9		7		B	2	8.22	8.22
4	CASTILLO TAPIA PAMELA	9		9		8		9		8		10		9		10		8		B	2	8.89	8.89
5	CONTRERAS RODRIGUEZ JUAN FERNANDO	10		6		9		5	1	7	2	10		9		8		5	1	R	3	7.67	7.67
6	CORTES GARCIA JUAN JAVIER	8		9		9		8		5		10		10		10		10	1	B		8.78	8.78
7	CRUZ ELIZALDE RAQUEL	10		6	1	7		7	2	6	2	10		9		8		7		R	3	7.78	7.78
8	HERNANDEZ MEDINA KARLA EVELYN	6	3	8	1	6		7		6		10		9		8		6	4	M	3	7.33	7.33
9	LABASTIDA SANCHEZ RUBEN ABRAHAM	6		7		8		6	1	5		10		8		7		7	1	R	2	7.11	7.11
10	LABASTIDA VELAZQUEZ RAUL BRYAN	6		7		7		6	1	7	2	10		8		7		5		M	1	7.00	7
11	LEON BRITO RODRIGO DANIEL	9		9		9		8		8		10		8		8		6	1	R		8.33	8.33
12	LOPEZ CABRERO MAGNOLIA	8		6	1	7		6		5		10		9		8		7		R	1	7.33	7.33
13	LOPEZ MENDOZA DIANA LAURA	8		6		10		7	1	7		10		9		9		8	1	R		8.22	8.22
14	MARTINEZ NAVARRO MIROSLAVA WENDOLINE	10		9		10		9		8		10		10		9		8		B	1	9.22	9.22
15	MARTINEZ RODRIGUEZ ABISAY	10	2	8		9		9	1	8	1	10		9		9		7		B	2	8.78	8.78
16	MENENDEZ MARTINEZ KARLA JESSICA	8		9		10		8		9		10		10		9		8	1	B	1	9.00	9
17	NOGUEZ FLORES MIGUEL ANGEL	10		10		10		9		7		10		10		10		8		B	1	9.33	9.33
18	PINEDA CHAVARRIA JORGE SAUL	7		10		9		6		5		10		8		7		7		M		7.67	7.67
19	RENDON SALES XOCHITL TLANETZIN	7		8		8		7		6		10		9		10		7		R	2	8.00	8
20	ROMERO GARCIA BRANDON ENRIQUE	7		7		8		7	1	6		10		7		8		6	2	M		7.33	7,33
21	SORIANO ALCANTARA MARIO ANTONIO	9		9		9		9		6		10		8		7		7	1	B	1	8.22	8.2
22	SUAREZ HERNANDEZ EMMANUEL ESAU	8	1	7		8		5	1	5	1	10		8		6		6	4	R	2	7.00	7
23	TORRES VASQUEZ ARMANDO																						
24	TREJO AYALA MARIA DE GUADALUPE	5	2	6		7		5		5		10		8		8		5	1	M		6.56	6.56
25	VARELA RAMIREZ FERNANDA	5	2	6		8		6	1	6	1	10		8		8		6	3	R		7.00	7
26	VAZQUEZ CURIEL JENNIFER	6		6		10		6		5		10		8		8		6		R	1	7.22	7.22
27	VELAZQUEZ VELAZQUEZ FRANCISCO JAVIER	5	4	5	1	5		5	2	5	5	8	2	5		6		5	5	M	4	5.44	5.44
28	VILLEGAS ISLAS MIGUEL ANGEL	8		6	1	8		6	1	7		10		9		8		6	1	M	2	7.56	7.56
29																							
30																							
31																							
32																							

ANEXO III
DIAGRAMA DE GANTT

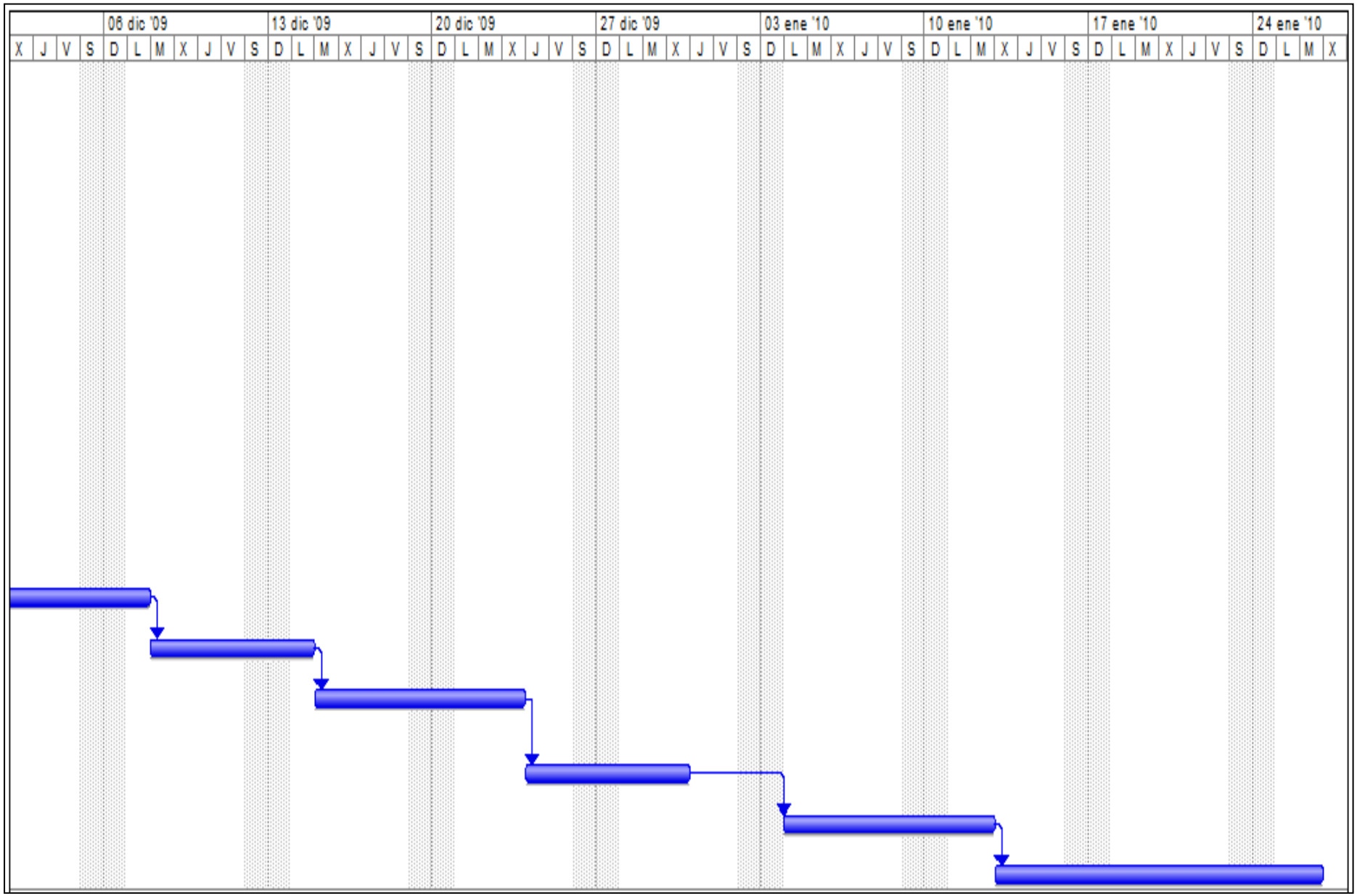
Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Análisis de la situación actual	5 días	lun 14/09/09	vie 18/09/09	
2	Recopilación de la información	7 días	lun 21/09/09	mar 29/09/09	1
3	Entrevistas a los posibles usuarios del sistema	5 días	mié 23/09/09	mar 29/09/09	1
4	Observaciones del funcionamiento de la organización	5 días	lun 21/09/09	vie 25/09/09	1
5	Identificación de necesidades	5 días	mié 30/09/09	mar 06/10/09	3,4,2
6	Determinar los requisitos de los usuarios y requerimientos del sistema	9 días	mié 07/10/09	lun 19/10/09	5
7	Estudio de factibilidad del sistema	5 días	mar 20/10/09	lun 26/10/09	6
8	Diseño conceptual de la Base de Datos	10 días	mar 27/10/09	lun 09/11/09	7
9	Elección del SMBD (Sistema Manejador de Base de Datos)	5 días	mar 27/10/09	lun 02/11/09	7
10	Instalación del SMBD elegido	4 días	mar 03/11/09	vie 06/11/09	9
11	Diseño Logico de la Base de Datos	5 días	mar 10/11/09	lun 16/11/09	8,10
12	Creación de la Base de Datos y tablas que la componen	5 días	mar 17/11/09	lun 23/11/09	11
13	Diseño Físico de la Base de Datos	5 días	mar 24/11/09	lun 30/11/09	12
14	Cambios en las estructuras de las tablas	5 días	mar 01/12/09	lun 07/12/09	13
15	Carga de datos tipo (ficticios) a la Base de Datos	5 días	mar 08/12/09	lun 14/12/09	14
16	Pruebas de validación mediante las consultas más consultadas, para detección de fallas	7 días	mar 15/12/09	mié 23/12/09	15
17	Definición del mantenimiento de la Base de Datos	5 días	jue 24/12/09	mié 30/12/09	16
18	Implementación y documentación del sistema (manual de operación)	7 días	lun 04/01/10	mar 12/01/10	17
19	Capacitación a los usuarios finales	10 días	mié 13/01/10	mar 26/01/10	18



CONTINUACIÓN



CONTINUACIÓN



ANEXO IV

LENGUAJE LOGICO ESTANDAR (SQL) DE LA CREACION DE LA BASE DE DATOS

```
DROP DATABASE IF EXISTS colegio;
CREATE DATABASE IF NOT EXISTS colegio;
USE colegio;

SELECT 'CREATING DATABASE STRUCTURE' as 'INFO';

DROP TABLE IF EXISTS

nivel_educativo,
alumnos_colegio,
asignaturas_colegio,
periodo_evaluacion,
direc_colegio,
calif_colegio,
docente_colegio,
asignatura_docente,


Create table nivel_educativo
(id_nivel smallint(2)  not null auto_increment,
nivel varchar(20) not null,
nom_nivel varchar(30)  not null,
CCT varchar(15) not null,
turno varchar(10) not null,
primary key (id_nivel));


Create table alumnos_colegio
(id_alumno int(6) not null auto_increment,
Nom_alumno varchar(20) not null,
Apepat_alum varchar(20) not null,
Apemat_alum varchar(20) not null,
fecha_nacim date  not null,
sexo varchar(2) not null,
curp varchar(18) not null,
edo_escolar varchar (14)not null,
nom_padre varchar(20) not null,
apepat_pad varchar(20) not null,
apemat_pad varchar(20) not null,
ocupacion_padre varchar(20) not null,
pais_alumno varchar(20) not null,
edo_alumno varchar(20) not null,
municipio_alumno varchar(20) not null,
col_alumno varchar(20) not null,
```

cod_alum varchar(6) not null,
calle varchar(20) not null,
numero varchar(4) not null,
telefono varchar(20) not null,
talla_alum float(3,2) not null,
peso_alum float(5,3) not null,
tipo_sangre varchar(3) not null,
ciclo_escolar varchar(9) not null,
nivel_educat smallint(2),
grado int(3) not null,
grupo varchar(6) not null,
fecha_alta date not null,
fecha_baja date,
key(nivel_educat),
foreign key (nivel_educat) references nivel_educativo (id_nivel) on delete cascade,
primary key(id_alumno));

Create table asignaturas_colegio
(id_asignatura int(3) not null auto_increment,
nom_asig varchar(80) not null,
primary key (id_asignatura));

Create table periodo_evaluacion
(id_periodo smallint(2) not null auto_increment,
per_eval varchar(20) not null,
primary key(id_periodo));

Create table direc_colegio
(id_director smallint(2) not null auto_increment,
nom_director varchar(20) not null,
apepat_dir varchar(20) not null,
apemat_dir varchar(20) not null,
dir_nivel smallint(2),
key (dir_nivel),
foreign key (dir_nivel) references nivel_educativo (id_nivel) on delete cascade,
primary key(id_director));

Create table calif_colegio
(id_alum int(5),
periodo_eval smallint (2),
materias int(3),
calificacion smallint (2) not null,
faltas smallint (2) not null,
key (id_alum),
key (periodo_eval),
key (materias),
foreign key (id_alum) references alumnos_colegio (id_alumno) on delete cascade,
foreign key (periodo_eval) references periodo_evaluacion (id_periodo) on delete cascade,
foreign key (materias) references asignaturas_colegio (id_asignatura) on delete cascade);

Create table docente_colegio
(id_docente int(3) not null auto_increment,
nom_docente varchar(50) not null,
apepat_doc varchar(20) not null,
apemat_doc varchar(20) not null,
titulo varchar(50),
primary key(id_docente));

Create table asignatura_docente
(asignatura int(3),
nivel_asignatura smallint (2),
docente_asignatura int(3),
key (asignatura),
key (nivel_asignatura),
key (docente_asignatura),
foreign key (asignatura) references asignaturas_colegio (id_asignatura) on delete cascade,
foreign key (nivel_asignatura) references nivel_educativo (id_nivel) on delete cascade,
foreign key (docente_asignatura) references docente_colegio (id_docente) on delete cascade);

ANEXO V

DICCIONARIO DE DATOS DE LA BASE DE DATOS ‘colegio’

Tabla *alumnos colegio*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	alumnos_colegio	id_alumno	mediumint	mediumint(6)	PRI	auto_increment
colegio	alumnos_colegio	Nom_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	Apepat_alum	varchar	varchar(20)		
colegio	alumnos_colegio	Apemat_alum	varchar	varchar(20)		
colegio	alumnos_colegio	fecha_nacim	date	date		
colegio	alumnos_colegio	sexo	enum	enum('M','H')		
colegio	alumnos_colegio	curp	varchar	varchar(18)		
colegio	alumnos_colegio	edo_escolar	varchar	varchar(14)		
colegio	alumnos_colegio	nom_padre	varchar	varchar(20)		
colegio	alumnos_colegio	apepat_pad	varchar	varchar(20)		
colegio	alumnos_colegio	apemat_pad	varchar	varchar(20)		
colegio	alumnos_colegio	ocupacion_padre	varchar	varchar(20)		
colegio	alumnos_colegio	pais_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	edo_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	municipio_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	col_alumno	varchar	varchar(20)		
colegio	alumnos_colegio	cod_alum	varchar	varchar(6)		
colegio	alumnos_colegio	calle	varchar	varchar(20)		
colegio	alumnos_colegio	numero	varchar	varchar(4)		
colegio	alumnos_colegio	telefono	varchar	varchar(20)		
colegio	alumnos_colegio	talla_alum	float	float(3,2)		
colegio	alumnos_colegio	peso_alum	float	float(5,3)		
colegio	alumnos_colegio	tipo_sangre	varchar	varchar(3)		
colegio	alumnos_colegio	ciclo_escolar	varchar	varchar(9)		
colegio	alumnos_colegio	nivel_educat	smallint	smallint(2)	MUL	
colegio	alumnos_colegio	grado	smallint	smallint(3)		
colegio	alumnos_colegio	grupo	varchar	varchar(10)		
colegio	alumnos_colegio	fecha_alta	date	date		
colegio	alumnos_colegio	fecha_baja	date	date		

Tabla *asignatura_docente*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	asignatura_docente	asignatura	mediumint	mediumint(3)	MUL	
colegio	asignatura_docente	nivel_asignatura	smallint	smallint(2)	MUL	
colegio	asignatura_docente	docente_asignatura	mediumint	mediumint(3)	MUL	

Tabla *asignaturas_colegio*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	asignaturas_colegio	id_asignatura	mediumint	mediumint(3)	PRI	auto_increment
colegio	asignaturas_colegio	nom_asig	varchar	varchar(80)		

Tabla *calif_colegio*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	calif_colegio	id_alum	mediumint	mediumint(6)	MUL	
colegio	calif_colegio	periodo_eval	smallint	smallint(2)	MUL	
colegio	calif_colegio	materias	mediumint	mediumint(3)	MUL	
colegio	calif_colegio	calificacion	smallint	smallint(2)		
colegio	calif_colegio	faltas	smallint	smallint(2)		

Tabla *direc_colegio*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	direc_colegio	id_director	smallint	smallint(2)	PRI	auto_increment
colegio	direc_colegio	nom_director	varchar	varchar(20)		
colegio	direc_colegio	apepat_dir	varchar	varchar(20)		
colegio	direc_colegio	apemat_dir	varchar	varchar(20)		
colegio	direc_colegio	dir_nivel	smallint	smallint(2)	MUL	

Tabla *docente_colegio*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	docente_colegio	id_docente	mediumint	mediumint(3)	PRI	auto_increment
colegio	docente_colegio	nom_docente	varchar	varchar(20)		
colegio	docente_colegio	apepat_doc	varchar	varchar(20)		
colegio	docente_colegio	apemat_doc	varchar	varchar(20)		
colegio	docente_colegio	titulo	varchar	varchar(50)		

Tabla *nivel_educativo*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	nivel_educativo	id_nivel	smallint	smallint(2)	PRI	auto_increment
colegio	nivel_educativo	nivel	varchar	varchar(20)		
colegio	nivel_educativo	nom_nivel	varchar	varchar(30)		
colegio	nivel_educativo	CCT	varchar	varchar(15)		
colegio	nivel_educativo	turno	varchar	varchar(10)		

Tabla *periodo_evaluacion*

table_schema	table_name	column_name	data_type	column_type	column_key	extra
colegio	periodo_evaluacion	id_periodo	smallint	smallint(2)	PRI	auto_increment
colegio	periodo_evaluacion	per_eval	varchar	varchar(20)		

ANEXO VI

CARGA DE DATOS A LA BASE DE DATOS ‘colegio’

Tabla *alumnos_colegio*

SQL Query Area

```
1 select * from alumnos_colegio;
2
```

id_alumno	Nom_alumno	Apepat_alum	Apemat_alum	fecha_nacim	sexo	curp	edo_escolar	nom_padre	apepat_pad	apemat_pad	ocupacion_padre	pais_alumno	edo_alumno
1000	Felipe	Angeles	Cacho	1999-12-05	H	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1001	Romina	Velazquez	Morales	1997-03-02	M	VEMR020397MMCZRR03	nuevo ingreso	Pablo	Velazquez	Corsa	Doctor	México	Estado de México
1003	Laura Guadalupe	Cano	Velazquez	1994-09-06	M	CAVL060994MMCZRR05	nuevo ingreso	Pedro	Cano	Soto	Empleado	México	Estado de México
1004	Sofia	Gonzalez	Castro	2005-10-15	M	GOCS050210MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1005	Ana	Gonzalez	Castro	2005-11-12	M	GOCA050211MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1006	Bertha	Gonzalez	Castro	2005-02-13	M	GOCS050210MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1007	Jennifer	Pinto	Oropeza	1997-03-15	M	VEMR020397MMCZRR03	nuevo ingreso	Pablo	Velazquez	Corsa	Doctor	México	Estado de México
1008	Edgar	Mendoza	Casas	1995-05-25	M	MAEA010395MMCRRR02	repetidor	Anabel	Escobedo	Chavez	Maestra	México	Estado de México
1009	Veronica	Chimal	Rosas	1999-05-26	M	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1010	Maria del Rocio	Donis	Salas	2001-01-23	M	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1011	Carlos	Aguilar	Hernandez	2001-08-27	H	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1012	Gabriela	Gonzalez	Garcia	1995-03-15	M	ANCF120599HMCZRR05	nuevo ingreso	Ruben	Contreras	Mendez	Contador	México	Estado de México
1013	Miguel	Padilla	Gamero	1995-05-06	H	ANCF120599HMCZRR05	nuevo ingreso	Ruben	Contreras	Mendez	Contador	México	Estado de México
1014	Felipe	Angeles	Cacho	1999-12-05	H	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1015	Romina	Velazquez	Morales	1997-03-02	M	VEMR020397MMCZRR03	nuevo ingreso	Pablo	Velazquez	Corsa	Doctor	México	Estado de México
1016	Allisson	Martinez	Escobedo	1995-02-12	M	MAEA010395MMCRRR02	repetidor	Anabel	Escobedo	Chavez	Maestra	México	Estado de México
1017	Laura Guadalupe	Cano	Velazquez	1994-09-06	M	CAVL060994MMCZRR05	nuevo ingreso	Pedro	Cano	Soto	Empleado	México	Estado de México
1018	Sofia	Gonzalez	Castro	2005-10-15	M	GOCS050210MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1019	Ana	Gonzalez	Castro	2005-11-12	M	GOCA050211MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1020	Bertha	Gonzalez	Castro	2005-02-13	M	GOCS050210MMCZRR05	nuevo ingreso	Hector	Gonzalez	Vergara	Empleado	México	Estado de México
1021	Jennifer	Pinto	Oropeza	1997-03-15	M	VEMR020397MMCZRR03	nuevo ingreso	Pablo	Velazquez	Corsa	Doctor	México	Estado de México
1022	Edgar	Mendoza	Casas	1995-05-25	M	MAEA010395MMCRRR02	repetidor	Anabel	Escobedo	Chavez	Maestra	México	Estado de México
1023	Veronica	Chimal	Rosas	1999-05-26	M	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1024	Maria del Rocio	Donis	Salas	2001-01-23	M	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1025	Carlos	Aguilar	Hernandez	2001-08-27	H	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1026	Gabriela	Gonzalez	Garcia	1995-03-15	M	ANCF120599HMCZRR05	nuevo ingreso	Ruben	Contreras	Mendez	Contador	México	Estado de México
1027	Miguel	Padilla	Gamero	1995-05-06	H	ANCF120599HMCZRR05	nuevo ingreso	Ruben	Contreras	Mendez	Contador	México	Estado de México
1028	Felipe	Angeles	Cacho	1999-12-05	H	ANCF120599HMCZRR05	nuevo ingreso	Gonzalo	Angeles	Contreras	Militar	México	Estado de México
1029	Romina	Velazquez	Morales	1997-03-02	M	VEMR020397MMCZRR03	nuevo ingreso	Pablo	Velazquez	Corsa	Doctor	México	Estado de México

Tabla *asignatura_docente*

SQL Query Area

```
2
3 select * from asignatura_docente;
4
```

asignatura	nivel_asignatura	docente_asign...
1	1	106
8	2	108
14	3	109
19	3	110
30	3	111
42	4	112
46	4	113
58	6	114
60	6	115
72	7	116
81	7	117

11 rows fetched in 0.0023s (0.0004s)

Tabla *asignaturas_colegio*

SQL Query Area

```
5 select * from asignaturas_colegio;
6
```

id_asignatura	nom_asig
1	Español Kin
2	Matematicas Kin
3	Ambiente kin
4	Escritura kin
5	Inglés Kin
6	Español Prim
7	Matematicas Prim
8	Ciencias Prim
9	Sociedad Prim
10	Inglés Prim
11	Civismo Prim
12	Educacion Fisica Prim
13	Español I Sec
14	Matematicas I Sec
15	Ciencias I Sec
16	Civica I Sec
17	Ingles Sec
18	Geografia I Sec
19	Educacion Fisica Sec
20	Artes Sec

105 rows fetched in 0.0039s (0.0006s)

Tabla *calif_colegio*

SQL Query Area

```
7 select * from calif_colegio;
8
```

id_alum	periodo_eval	materias	calificacion	faltas
1004	1	1	8	0
1004	1	2	8	0
1004	1	3	8	0
1004	1	4	8	0
1004	1	5	9	1
1004	2	1	8	0
1004	2	2	8	0
1004	2	3	8	0
1004	2	4	8	0
1004	2	5	8	0
1005	1	1	8	0
1005	1	2	8	0
1005	1	3	8	0
1005	1	4	8	0
1005	1	5	8	0
1005	2	1	8	0
1005	2	2	8	0
1005	2	3	8	0
1005	2	4	8	0
1005	2	5	8	0
1006	1	1	8	0
1006	1	2	8	0
1006	1	3	8	0
1006	1	4	8	0
1006	1	5	8	0

45 rows fetched in 0.0031s (0.0005s) Edit Apply Changes

Tabla *direc_colegio*

```
SQL Query Area
9  select * from direc_colegio;
10
```

nom_director	apepat_dir	apemat_dir	dir_nivel
Dolores	Aguilar	Cruz	1
Guadalupe	Delgadillo	Ramirez	2
Lilia	Uribe	Fernandez	3
Monica	Montes	de Jesus	4
Carla	Martinez	de la Torre	5
Dana	Aguilar	Cruz	6
Martha	Delgadillo	Ramirez	7
Monica	Uribe	Fernandez	8
Claudia	Montes	de Jesus	9

9 rows fetched in 0.0097s (0.0009s)

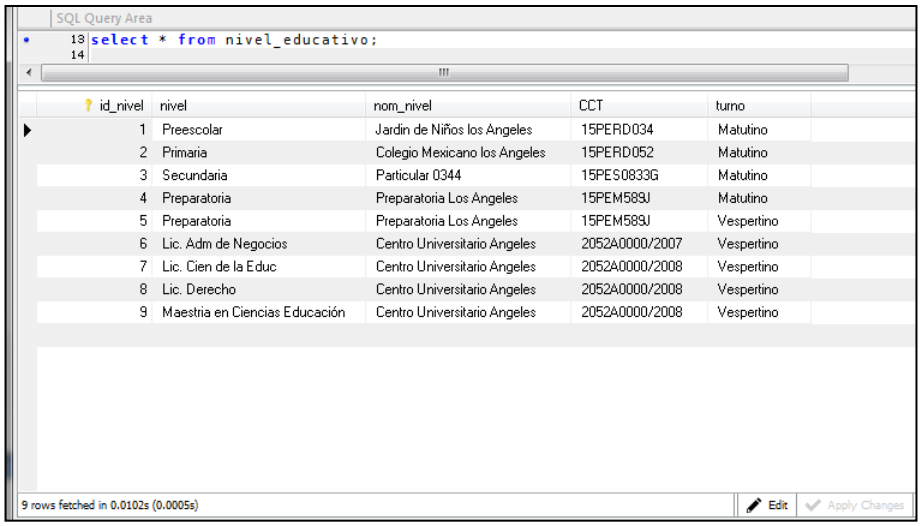
Tabla *docente_colegio*

```
SQL Query Area
11 select * from docente_colegio;
12
```

id_docente	nom_docente	apepat_doc	apemat_doc	titulo
100	Milton	Castañeda	Diaz	Lic. Ciencias Educación
101	Rosalba	Mendoza	Cruz	Lic. Matematicas
102	Horacio	Rivera	Garcia	Lic. Pedagogía
103	Mario	Contreras	DonJuan	Ing. Industrial
104	Gonzalo	Contreras	DonJuan	Biologo
105	Milton2	Castañeda	Diaz	Ing. petroquimica
106	Rosalba2	Mendoza	Cruz	Lic. Sistemas
107	Horacio2	Rivera	Garcia	Lic. Educación Básica
108	Mario2	Contreras	DonJuan	Medico Cirujano
109	Gonzalo2	Contreras	DonJuan	Medico Cirujano
110	Milton3	Castañeda	Diaz	Ing. petroquimica
111	Rosalba3	Mendoza	Cruz	Lic. Sistemas
112	Horacio3	Rivera	Garcia	Lic. Educación Básica
113	Mario3	Contreras	DonJuan	Medico Cirujano
114	Gonzalo3	Contreras	DonJuan	Medico Cirujano
115	Milton4	Castañeda	Diaz	Ing. petroquimica
116	Rosalba4	Mendoza	Cruz	Lic. Sistemas
117	Horacio4	Rivera	Garcia	Lic. Educación Básica
118	Mario4	Contreras	DonJuan	Medico Cirujano
119	Gonzalo4	Contreras	DonJuan	Medico Cirujano
120	Milton5	Castañeda	Diaz	Ing. petroquimica
121	Rosalba5	Mendoza	Cruz	Lic. Sistemas

67 rows fetched in 0.0101s (0.0008s) Edit

Tabla *nivel_educativo*

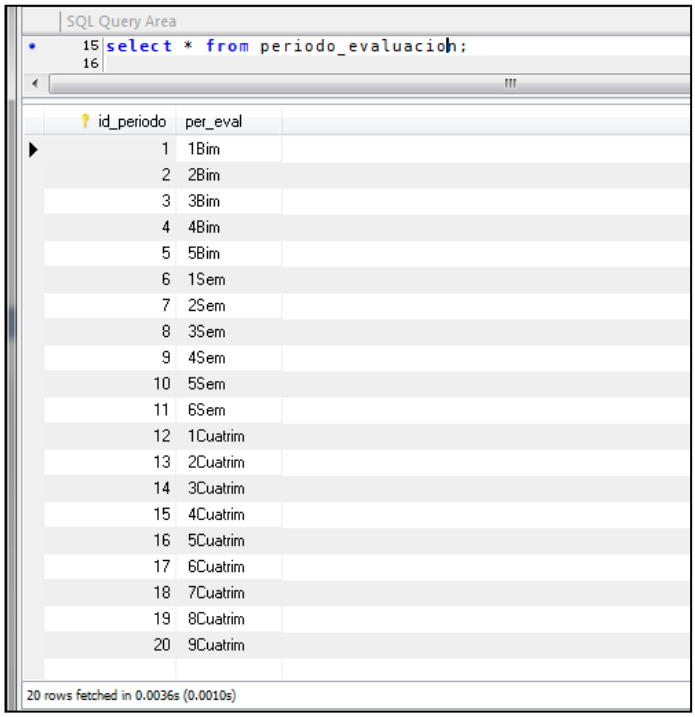


```
SQL Query Area
13 select * from nivel_educativo;
14
```

id_nivel	nivel	nom_nivel	CCT	turno
1	Preescolar	Jardin de Niños los Angeles	15PERD034	Matutino
2	Primaria	Colegio Mexicano los Angeles	15PERD052	Matutino
3	Secundaria	Particular 0344	15PES0833G	Matutino
4	Preparatoria	Preparatoria Los Angeles	15PEM589J	Matutino
5	Preparatoria	Preparatoria Los Angeles	15PEM589J	Vespertino
6	Lic. Adm de Negocios	Centro Universitario Angeles	2052A0000/2007	Vespertino
7	Lic. Cien de la Educ	Centro Universitario Angeles	2052A0000/2008	Vespertino
8	Lic. Derecho	Centro Universitario Angeles	2052A0000/2008	Vespertino
9	Maestria en Ciencias Educación	Centro Universitario Angeles	2052A0000/2008	Vespertino

9 rows fetched in 0.0102s (0.0005s) Edit Apply Changes

Tabla *periodo_evaluacion*



```
SQL Query Area
15 select * from periodo_evaluacion;
16
```

id_periodo	per_eval
1	1Bim
2	2Bim
3	3Bim
4	4Bim
5	5Bim
6	1Sem
7	2Sem
8	3Sem
9	4Sem
10	5Sem
11	6Sem
12	1Cuatrim
13	2Cuatrim
14	3Cuatrim
15	4Cuatrim
16	5Cuatrim
17	6Cuatrim
18	7Cuatrim
19	8Cuatrim
20	9Cuatrim

20 rows fetched in 0.0036s (0.0010s)

GLOSARIO

Acceso rápido: El sistema debe poder responder a las consultas lo más rápido posible. Esto requiere algoritmos de búsqueda rápidos.

Aislamiento: Es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.

Almacenamiento no transaccional: Este tipo de almacenamiento sólo puede obtener y mostrar los datos de una base de datos, al igual que una lista de los CD o los libros en una biblioteca.

Almacenamiento transaccional: Es un tipo de sistema de información diseñado para recolectar, almacenar, modificar y recuperar todo tipo de información que es generada por las transacciones en una organización.

APIs: (Interfaz para programas de aplicación) Una serie de reglamentos y acuerdos que nos definen la manera en cómo llamar determinado servicio desde cierto programa.

Árboles B+: En un árbol-B+, en contraste respecto un árbol-B, toda la información se guarda en las hojas. Los nodos internos sólo contienen claves y punteros. Todas las hojas se encuentran en el mismo, más bajo nivel. Los nodos hoja se encuentran unidos entre sí como una lista enlazada para permitir búsqueda secuencial. El número máximo de claves en un registro es llamado el orden del árbol-B+. El mínimo número de claves por registro es la mitad del máximo número de claves. Por ejemplo, si el orden de un árbol-B+ es n, cada nodo (exceptuando la raíz) debe tener entre n/2 y n claves. El número de claves que pueden ser indexadas usando un árbol-B+ está en función del orden del árbol y su altura.

Atomicidad: Es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

Atributo compuesto: Son atributos que se pueden dividir en sus componentes, pudiendo formar jerarquías.

Atributo derivado: el valor para este tipo de atributo se puede derivar de los valores de otros atributos o entidades relacionados. Por ejemplo del atributo fecha_nacimiento se puede derivar la edad, restándola a la fecha actual.

Atributo multivaluado: son atributos que tienen límites inferior y superior en el número de valores para una entidad.

Backup: (copia de seguridad) es una medida de precaución para el cuidado de nuestra información el crear un respaldo o copia de seguridad, de tal manera que si por una u otra razón se borra o se pierde el original, podremos utilizar la copia. Nos evitará problemas el tener siempre una copia de nuestro trabajo.

Base de Datos: Cualquier conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora, diseñado para facilitar su mantenimiento y acceso de una forma estándar. Los datos suelen aparecer en forma de texto, números o gráficos.

Base de Datos Relacional: Tipo de base de datos o sistema de administración de bases de datos, que almacena información en tablas (filas y columnas de datos) y realiza búsquedas utilizando los datos de columnas especificadas de una tabla para encontrar datos adicionales en otra tabla.

BBDD: (siglas para una base de datos), es un conjunto ordenado de información almacenado de forma que se pueda acceder a la misma fácil y rápidamente.

Bottlenecks: Un cuello de botella es un fenómeno por el cual se limita el rendimiento o la capacidad de todo un sistema por un número único o limitado de componentes o de los recursos. El cuello de botella término se toma de los "activos son el agua" metáfora. Como el agua se vierte de una botella, la tasa de salida está limitada por el ancho del conducto de salida, es decir, cuello de botella. Al aumentar la anchura del cuello de botella se puede aumentar la velocidad a la que el agua sale del cuello a diferentes frecuencias. Como componentes de un sistema de limitación se refiere a veces como puntos de cuello de botella.

B-tree: Utiliza una estructura de árbol cuyos nodos tienen un arreglo de valores. Se puede conseguir sobre esta estructura como "Árbol B" en los libros de estructuras de datos. Es ideal para en los WHERE usaremos búsquedas ordenadas usando operadores > < >= <=. Normalmente es el tipo de índice por defecto.

B-Tree incrustados (clustered): Esto quiere decir que el índice se almacena junto a los datos. Los registros se ordenan físicamente por su clave primaria. Esto hace que las búsquedas por clave primaria sean muy rápidas, porque al leer el índice, ya se está accediendo a los datos.

Buffer: es una ubicación de la memoria en un Disco o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada.

Cable UTP: El cable UTP es un sistema de cableado estructurado consiste de una infraestructura flexible de cables que puede aceptar y soportar sistemas de computación y de teléfono múltiples.

Caché del disco: se tiene memoria RAM configurada para tener páginas de datos (sectores o bloques) del disco. El gestor de ficheros mira primero en la caché para ver si está el sector que se desea leer o escribir, y sólo si no lo encuentra es cuando va a disco. Por el principio de localidad de las referencias, hay una gran probabilidad de que lo que se necesita esté ya en la caché al ir a buscarlo.

Campos: Un campo es identificado por un rótulo numérico que se define en la FDT de la base de datos. A diferencia de los campos.

Cardinalidad: Forma como cada elemento de la entidad participa de la relación (Mínima, Máxima). El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M".

Cifrar: El cifrado es un método que permite aumentar la seguridad de un mensaje o de un archivo mediante la codificación del contenido, de manera que sólo pueda leerlo la persona que cuente con la clave de cifrado adecuada para descodificarlo. Por ejemplo, si realiza una compra a través de Internet, la información de la transacción (como su dirección, número de teléfono y número de tarjeta de crédito) suele cifrarse a fin de mantenerla a salvo. Use el cifrado cuando desee un alto nivel de protección de la información.

Clave foránea (Foreign Key FK): es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

Clave primaria ó PRIMARY KEY: es una clave única elegida entre todas las candidatas que define unívocamente a todos los demás atributos de la tabla, para especificar los datos que serán relacionados con las demás tablas. Sólo puede existir una clave primaria por tabla y ningún campo de dicha clave puede contener valores NULL.

Clave única ó UNIQUE: Cada tabla puede tener uno o más campos cuyos valores identifican de forma única cada registro de dicha tabla, es decir, no pueden existir dos o más registros diferentes cuyos valores en dichos campos sean idénticos. Este conjunto de campos se llama clave única.

Cliente/Servidor: La arquitectura cliente-servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Clúster: Referido a un disco de almacenamiento, es cada uno de los sectores en los que se divide físicamente. (Unidad de asignación). Forma de direccionamiento usada por el IPv6, en la que se asigna una dirección a un grupo de computadoras; un datagrama enviado a la dirección puede entregarse a cualquiera de las computadoras del grupo.

Clustering: Agrupamiento. Las técnicas de clustering permiten el crecimiento de un sistema mediante la adición de procesadores o CPUs a la unidad primitiva.

Código fuente: El código fuente de un programa informático (o software) es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está descrito por completo su funcionamiento.

Colisión: suceso que ocurre en una red CSMA/CD cuando dos estaciones intentan transmitir de manera simultánea. Las señales se interfieren y obligan a las dos estaciones a retroceder e intentar de nuevo.

Commit: En el contexto de la Ciencia de la computación y la gestión de datos, commit (acción de cometer) se refiere a la idea de consignar un conjunto de cambios "tentativos, o no permanentes". Un uso popular es al final de una transacción de base de datos.

Compiladores: Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior (típicamente lenguaje de máquina). De esta manera un programador puede diseñar un programa en un lenguaje mucho más cercano a cómo piensa un ser humano, para luego compilarlo a un programa más manejable por una computadora.

Concurrencia: la concurrencia es la propiedad de los sistemas que permiten que múltiples procesos sean ejecutados al mismo tiempo, y que potencialmente puedan interactuar entre sí.

Conector ODBC: sirve para que desde un sistema Microsoft Windows se pueda acceder a una base de datos MySQL.

Consistencia: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

Consulta: En base de datos, una consulta es el método para acceder a los datos en las bases de datos. Con las consultas se puede modificar, borrar, mostrar y agregar datos a una base de datos. Para esto se utiliza un lenguaje de consultas.

DBA: administrador de base de datos, es la persona responsable de los aspectos ambientales de una base de datos.

Dependencia funcional: es una extensión del concepto de función para n dominios. Informalmente, una dependencia funcional ocurre cuando el valor de una tupla sobre un conjunto de atributos X determina unívocamente el valor de otro conjunto de atributos Y. Esto significa que, si existen dos tuplas que coincidan en los valores para X, entonces deben coincidir en los valores para Y.

Dependencias transitivas: se dice que un conjunto B depende de forma transitiva de otro conjunto A, si existe un conjunto Z que depende funcionalmente de A y B depende funcionalmente de Z. Se denota A Z B.

Diagrama de Gantt: El diagrama de Gantt, gráfica de Gantt o carta Gantt es una popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. A pesar de que, en principio, el diagrama de Gantt no indica las relaciones existentes entre actividades, la posición de cada tarea a lo largo del tiempo hace que se puedan identificar dichas relaciones e interdependencias. Fue Henry Laurence Gantt quien, entre 1910 y 1915, desarrolló y popularizó este tipo de diagrama en Occidente.

Diagrama Entidad-Relación (DER): es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información así como sus interrelaciones y propiedades.

Diccionario de datos: es un conjunto de metadatos que contiene las características lógicas y puntuales de los datos que se van a utilizar en el sistema que se programa, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño. En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

Distribución binaria: Es una distribución compilada y es la manera de facilitar más las cosas a los instaladores de la distribución de módulos: para los usuarios de Linux basado en RPM, es un RPM binario; para los usuarios de Windows, un instalador ejecutable; para los usuarios de Debian, un paquete Debian, etc.

Dominio: conjunto de valores admisibles para un elemento de una relación.

Durabilidad: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Edgar Frank Codd: Científico informático inglés (23 de agosto de 1923 - 18 de abril de 2003), conocido por sus aportes a la teoría de bases de datos relacionales.

Ejecutable: En informática, un ejecutable o archivo ejecutable, es tradicionalmente un archivo binario cuyo contenido se interpreta por el ordenador como un programa. Generalmente, contiene instrucciones en código máquina de un procesador en concreto, pero también puede contener bytecode que requiera un intérprete para ejecutarlo. un archivo ejecutable contiene mucha información que no es parte del programa en sí: recursos como textos e imágenes, requisitos del entorno de ejecución, información simbólica y de depuración, u otra información que ayude al sistema operativo a ejecutar el programa.

Engine InnoDB: InnoDB es una tecnología de almacenamiento de datos de código abierto para la base de datos MySQL, incluido como formato de tabla estándar en todas las distribuciones de MySQL AB a partir de las versiones 4.0. Su característica principal es que soporta transacciones de tipo ACID y bloqueo de registros e integridad referencial. InnoDB ofrece una fiabilidad y consistencia muy superior a MyISAM, la anterior tecnología de tablas de MySQL, si bien el mejor rendimiento de uno u otro formato dependerá de la aplicación específica.

Entidad: Objeto del mundo real sobre el que queremos almacenar información (Ej: una persona). Las entidades están compuestas de *atributos* que son los datos que definen el objeto (para la entidad persona serían DNI, nombre, apellidos, dirección,...). De entre los atributos habrá uno o un conjunto de ellos que no se repite; a este atributo o conjunto de atributos se le llama *clave* de la entidad, (para la entidad persona una clave seria DNI). En toda entidad siempre hay al menos una clave que en el peor de los casos estará formada por todos los atributos de la tabla.

Escalabilidad: es la capacidad de mejorar recursos para ofrecer una mejora (idealmente) lineal en la capacidad de servicio. La característica clave de una aplicación es que la carga adicional sólo requiere recursos adicionales en lugar de una modificación extensiva de la aplicación en sí.

Estándar ANSI/ISO SQL: Es una definición del estándar del leguaje de consulta de base de datos. Este te permite hacer consulta sobre una base de datos, también te permite modificar y registrar nuevos datos. En general te permite manejar los datos de una base de datos. La mayoría de motores de bases de datos lo implementa SQL server, Oracle, Sybase, Postgres, etc.

Estructura lógica: La estructura lógica hace referencia a la idea sobre cómo están organizados los datos sin hacer mención a la forma ni método de almacenamiento, ni tampoco a los métodos físicos de acceso a los datos.

Estructuras de almacenamiento: son el esqueleto de la base de datos, ellas contendrán los datos de nuestro sistema de acuerdo a ciertas características de tipo y forma.

Exclusive locks*:* Cuando una instrucción modifica los datos, la transacción mantiene un bloqueo *exclusivo* en los datos que impide que otras transacciones accedan a los datos. Este bloqueo se mantiene en su lugar hasta que la transacción libera los problemas de bloqueo con un commit o rollback.

Factibilidad: se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señalados. Generalmente la factibilidad se determina sobre un proyecto.

Fichero desordenado: en estos ficheros los registros no siguen un orden específico.

Ficheros de agrupamiento: varios ficheros intercalan sus registros de modo que estos queden agrupados por el valor de algún campo que tienen en común.

Ficheros de la base de datos: colección de información relacionada.

Ficheros del sistema operativo: proporcionan el mecanismo para el almacenamiento masivo de programas y datos, tanto del propio sistema operativo como de todos los usuarios del ordenador.

Ficheros dispersos (hashing): se calcula aplicando cierta función sobre uno de sus campos, el campo de dispersión. El acceso a los datos es muy rápido sólo si se busca con la condición de igualdad sobre el campo de dispersión. La dispersión se puede utilizar a nivel interno (RAM), como una estructura de datos de un programa, o bien a nivel externo para ficheros en disco.

Gestor de buffer: esta soportado por un gestor de ficheros el cual controla y gestiona la asignación de espacio en disco junto con sus estructuras de datos asociadas.

Gestor de conexiones: responsable de mantener las múltiples conexiones de los clientes.

Gestor de ficheros: ó administrador de archivos, gestor de archivos o explorador de archivos (del inglés *file manager*) es una aplicación informática que provee acceso a archivos y facilita el realizar operaciones con ellos, como copiar, mover o eliminar archivos donde el usuario lo quiera ubicar.

Grado: Número de entidades que participan en la relación, puede ser unitaria, binaria o ternaria.

Inconsistencia: Ocurre cuando existe información contradictoria o incongruente en la base de datos.

Independencia física: El nivel físico puede ser modificado independientemente del nivel conceptual. Esto significa que el usuario no puede ver todos los componentes de hardware de la base de datos, que es simplemente una estructura transparente para representar la información almacenada.

Independencia lógica: El nivel conceptual debe poder modificarse sin alterar el nivel físico. En otras palabras, el administrador de la base de datos debe poder introducir mejoras sin afectar la experiencia de los usuarios.

Índice: el índice de una base de datos es una estructura de datos que mejora la velocidad de las operaciones, permitiendo un rápido acceso a los registros de una tabla en una base de datos. Al aumentar drásticamente la velocidad de acceso, se suelen usar sobre aquellos campos sobre los cuales se hacen frecuentes búsquedas. El índice tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice dicho elemento para, una vez encontrado, devolver el registro que se encuentre en la posición marcada por el índice.

Índice Full Text: Los índices de texto completo son del tipo FULLTEXT, se usan en tablas del tipo MyISAM, y pueden contener uno o más campos del tipo CHAR, VARCHAR y TEXT. Un índice de texto completo está diseñado para facilitar y optimizar la búsqueda de palabras clave en tablas que tienen grandes cantidades de información en campos de texto.

Índice HASH: Utiliza algo parecido a las tablas hash para ubicar fácilmente registros dado un valor de los campos en el índice. Es ideal para cuando en los WHERE estaremos preguntando por un valor específico.

Índice ordinario: es un índice que no es primario y permite valores duplicados (a menos que los campos hayan sido especificados como UNIQUE).

Índice primario: normalmente para cada archivo de datos debe existir un índice cuya llave de indexación sea idéntica a su llave primaria. Este índice es llamado índice primario.

Índice secundario: es utilizado para reducir el tiempo de localización de una determinada información dentro de un archivo o para clasificar los registros del archivo de acuerdo con el orden necesario para la obtención de la información deseada.

Índice único: Un índice único actúa como una restricción en la tabla previniendo filas idénticas en el índice.

Integridad referencial: es un sistema de reglas que utilizan la mayoría de las bases de datos relacionales para asegurarse que los registros de tablas relacionadas son válidos y que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

Interfaz: es la conexión entre dos ordenadores o máquinas de cualquier tipo dando una comunicación entre distintos niveles. Interfaz también hace referencia al conjunto de métodos para lograr interactividad entre un usuario y una computadora. Una interfaz puede ser del tipo GUI, o línea de comandos, etc. También puede ser a partir de un hardware.

Interrelaciones: se entiende por interrelación a la asociación, vinculación o correspondencia entre entidades. Por ejemplo, entre la entidad "PROFESOR" y la entidad "CURSO" podemos establecer la relación "IMPARTE" porque el profesor imparte cursos.

Join: La sentencia join en SQL permite combinar registros de dos o más tablas en una base de datos relacional. En el Lenguaje de Consultas Estructurado (SQL), hay tres tipos de *JOIN*: interno, externo, y cruzado.

Laxo: flojo o falto de fuerza.

Lenguaje de consulta estructurado (SQL): El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre ella.

Lenguaje de Definición de Datos (LDD): El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

Lenguaje de Manipulación de Datos (LMD: Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

Lenguajes de alto nivel: Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de a la capacidad ejecutora de las máquinas.

Licencia GPL: Una de las más utilizadas es la *Licencia Pública General de GNU* (GNU GPL). El autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados para asegurarse de que todas las versiones modificadas del software permanecen bajo los términos más restrictivos de la propia GNU GPL.

Licenciamiento comercial: pueden ser adquiridas por cualquier empresa que desea hacer servicios con ANSYS con fines de lucro. Este tipo de licenciamiento no puede ser usado por Instituciones educativas y/o centros de investigación.

Licenciamiento libre: La Licencia de documentación libre de GNU o GFDL (*GNU Free Documentation License*) es una licencia copyleft para contenido libre, esta licencia, a diferencia de otras, asegura que el material licenciado bajo la misma esté disponible de forma completamente libre, pudiendo ser copiado, redistribuido, modificado e incluso vendido siempre y cuando el material se mantenga bajo los términos de esta misma licencia (GNU GFDL). En caso de venderse en una cantidad superior a 100 ejemplares, deberá distribuirse en un formato que garantice futuras ediciones (debiendo incluir para ello el texto o código fuente original). Dicha licencia fue diseñada principalmente para manuales, libros de texto y otros materiales de referencia e institucionales que acompañaran al software GNU. Sin embargo puede ser usada en cualquier trabajo basado en texto, sin que importe cuál sea su contenido.

Línea de comandos: es un método que permite a las personas dar instrucciones a algún programa informático por medio de una línea de texto simple.

Locks*:* El bloqueo es un dispositivo mecánico o electrónico de sujeción que se desprende de un objeto físico (como una llave, tarjeta de acceso, huella digital, RFID tarjeta o token de seguridad) o información de secreto (por ejemplo, un código de clave o contraseña), o una combinación de ambos.

Log: es un registro oficial de eventos durante un rango de tiempo en particular. Para los profesionales en seguridad informática es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (who, what, when, where y why) un evento ocurre para un dispositivo en particular o aplicación.

Memoria secundaria: es un tipo de almacenamiento masivo y permanente (no volátil), a diferencia de la memoria RAM que es volátil; pero posee mayor capacidad de memoria que la memoria principal, aunque es más lenta que ésta.

Microsoft Project 2007: Microsoft Project (o MSP) es un software de administración de proyectos desarrollado y comercializado por Microsoft el cual está diseñado para asistir a administradores de proyectos en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo. Asignas tiempos aproximados con costo por horas y responsables si las tareas tienen precedentes o corren en paralelo, asignas recursos y costos, el programa te saca graficas, Graficas de Gannt, Diagramas de Red, etc.

Modelo de Datos: es un conjunto de herramientas conceptuales para describir los datos, las relaciones entre ellos, su semántica y sus limitantes. Esa colección de conceptos incluye entidades, atributos y relaciones.

Modelo Relacional: Este modelo se está empleando con más frecuencia en la práctica, debido a la ventajas que ofrece sobre los dos modelos anteriores, entre ellas, el rápido entendimiento por parte de usuarios que no tienen conocimientos profundos sobre Sistemas de Bases de Datos.

Motor de almacenamiento: es una parte esencial de un SGDB puesto que se encarga de crear, recuperar, actualizar y borrar los datos de una base de datos.

Multihilo: una programación multihilo hace que los programas operan con mayor velocidad en sistemas de computadores con múltiples CPUs (sistemas multiprocesador o a través de grupo de máquinas) ya que los hilos del programa se prestan verdaderamente para la ejecución concurrente.

Multi-threaded: En sistemas operativos, un hilo de ejecución o subproceso es una característica que permite a una aplicación realizar varias tareas a la vez (concurrentemente). Los distintos hilos de ejecución comparten una serie de recursos tales como el espacio de memoria, los archivos abiertos, situación de autenticación, etc. Esta técnica permite simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

MySQL Administrator: es el nuevo software de administración de servidores de Bases de Datos de MySQL que ha creado MySQL AB. Se trata de un software multiplataforma, que por el momento se encuentra disponible para Linux y Microsoft Windows y que cuenta con un entorno gráfico de usuario muy intuitivo.

MySQL Query Browser: es una utilidad para trabajar con la base de datos MySQL. Es un editor de sentencias SQL visual, que además incorpora herramientas para optimizar las consultas. Dispone también de un editor de tablas y registros, que permite crear nuevas tablas o cambiar las existentes y la posibilidad de cambiar los registros, es decir, los datos almacenados en las tablas.

MySQL: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Nivel Conceptual: Es aquel en el que se definen las estructuras lógicas de almacenamiento y las relaciones que se darán entre ellas. Ejemplos comunes de este nivel son el diseño de los registros y las ligas que permitirán la conexión entre registros de un mismo archivo, de archivos distintos incluso, de ligas hacia archivos.

Nivel Físico: Es aquel en el que se determinan las características de almacenamiento en el medio secundario. Los diseñadores de este nivel poseen un amplio dominio de cuestiones técnicas y de manejo de hardware.

Nivel Lógico: El siguiente nivel más alto de abstracción describe que datos se almacenan en la base de datos y que relaciones existen entre esos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples en el nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. Los administradores de base de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.

Nodo raíz: nodo que no tiene padre.

Nodos padres: nodo que contiene un puntero al nodo actual.

Not null: Significa que la columna no puede tener valores nulos.

Null: (nulo) es un marcador especial usado en el lenguaje de consulta estructurado (SQL) para indicar que no existe un valor dentro de una base de datos.

Open Source: Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

Paging: procedimiento que realiza el sistema operativo transfiriendo páginas a disco para liberar memoria.

Password: Una contraseña o clave (en inglés *password*) es una forma de autentificación que utiliza información secreta para controlar el acceso hacia algún recurso. La contraseña normalmente debe mantenerse en secreto ante aquellos a quien no se les permite el acceso. Aquellos que desean acceder a la información se les solicita una clave; si conocen o no conocen la contraseña, se concede o se niega el acceso a la información según sea el caso.

Performance: Desempeño con respecto al rendimiento de una computadora, un dispositivo, un sistema operativo, un programa o una conexión a una red.

Plataformas: es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dicho sistema está definido por un estándar alrededor del cual se determina una arquitectura de hardware y una plataforma de software (incluyendo entornos de aplicaciones).

Pool de conexión: se denomina connection pool (agrupamiento de conexiones) al manejo de una colección de conexiones abiertas a una base de datos de manera que puedan ser reutilizadas al realizar múltiples consultas o actualizaciones.

Portabilidad: Se define como la característica que posee un software para ejecutarse en diferentes plataformas, el código fuente del software es capaz de reutilizarse en vez de crearse un nuevo código cuando el software pasa de una plataforma a otra. A mayor portabilidad menor es la dependencia del software con respecto a la plataforma.

Problemas de Integridad: Ocurre cuando no existe a través de todo el sistema procedimientos uniformes de validación para los datos.

Problemas de Seguridad: Se presentan cuando no es posible establecer claves de acceso y resguardo en forma uniforme para todo el sistema, facilitando así el acceso a intrusos.

Programas cliente: El cliente es una aplicación informática o un computador que accede a un servicio remoto en otro computador, conocido como servidor, normalmente a través de una red de telecomunicaciones.

Programas de aplicación: una aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo. Esto lo diferencia principalmente de otros tipos de programas como los sistemas operativos (que hacen funcionar al ordenador), las utilidades (que realizan tareas de mantenimiento o de uso general), y los lenguajes de programación (con el cual se crean los programas informáticos).

Propagación de clave: Se trata de un concepto que se aplica a interrelaciones N:1 ó 1:1, que nos ahorra la creación de una relación.

RDBMS: Un RDBMS es un Sistema Administrador de Bases de Datos Relacionales. RDBMS viene del acrónimo en inglés *Relational Data Base Management System*. Los RDBMS proporcionan el ambiente adecuado para gestionar una base de datos.

Redundancia controlada: El DBMS debe poder evitar la redundancia de datos siempre que sea posible, tanto para minimizar los errores como para prevenir el desperdicio de memoria.

Redundancia: Esta se presenta cuando se repiten innecesariamente datos en los archivos que conforman la base de datos.

Relación: una relación o vínculo entre dos o más entidades describe alguna interacción entre las mismas. Por ejemplo, una relación entre una entidad "Empleado" y una entidad "Sector" podría ser "trabaja_en", porque el empleado trabaja en un sector determinado.

Relaciones 1-1: Las entidades que intervienen en la relación se asocian una a una (Ej: la entidad HOMBRE, la entidad MUJER y entre ellos la relación MATRIMONIO).

Relaciones 1-n: Una ocurrencia de una entidad está asociada con muchas (n) de otra (Ej: la entidad EMPERSA, la entidad TRABAJADOR y entre ellos la relación TRABAJAR-EN).

Relaciones n-m: Cada ocurrencia, en cualquiera de las dos entidades de la relación, puede estar asociada con muchas (n) de la otra y viceversa (Ejemplo: la entidad ALUMNO, la entidad EMPRESA y entre ellos la relación MATRÍCULA).

Restricción: una limitación o una reducción ya sea natural o impuesta, según corresponda.

Root: Son cuentas de superusuario que pueden realizar cualquier tarea. Inicialmente las cuentas root no tienen contraseñas, de forma que cualquier persona puede conectarse al servidor MySQL como root sin una contraseña y recibirá todos los privilegios.

SCBD: es el software encargado de interactúa con los programas de aplicación de los usuarios para recuperar los datos de la base de datos.

Seguridad de los datos: El DBMS debe poder administrar los derechos de acceso a los datos de cada usuario.

Semántica: se refiere a los aspectos del significado, sentido o interpretación del significado de un determinado elemento, símbolo, palabra, expresión o representación formal. En principio cualquier medio de expresión (lenguaje formal o natural) admite una correspondencia entre expresiones de símbolos o palabras y situaciones o conjuntos de cosas que se encuentran en el mundo físico o abstracto que puede ser descrito por dicho medio de expresión.

Sentencia SQL: procedimiento para iniciar el motor de base de datos.

Shared locks*:* Cuando una instrucción lee los datos sin realizar ninguna modificación, su transacción obtiene un *bloqueo compartido* en los datos.

Sistema de información: es el término general utilizado para la estructura global que incluye todos los mecanismos para compartir datos que se han instalado.

SMBD (en inglés: Data Base Management System) SISTEMA DE MANEJO DE BASE DE DATOS: Consiste de una base de datos y un conjunto de aplicaciones (programas) para tener acceso a ellos.

SSBD: es el software encargado de manipular los archivos de datos necesarios para almacenar los datos dentro de la base de datos.

Suites ofimáticas: Una suite ofimática o suite de oficina es una recopilación de programas, los cuales son utilizados en oficinas y sirve para diferentes funciones como crear, modificar, organizar, escanear, imprimir, etc. archivos y documentos. Son ampliamente usados en varios lugares, ya que al ser eso (una recopilación), hace que sea asequible adquirir toda la suite, que programa por programa, lo cual es más complejo, al tener que conseguir programa por programa, y en caso del software pagado, más caro.

Swapping: técnica mediante la cual se intercambia un proceso que está en memoria por otro que no lo está. Para esto se hace uso de un área de memoria conocida como de intercambio (swap).

Tablas de referencia (*lookup*): son listas de valores, cada uno de los cuales tiene un código. Por ejemplo puede haber una tabla de referencia para los tipos de inmueble, con las descripciones de estos tipos y un código asociado.

Tablas hash: Una tabla hash o mapa hash es una estructura de datos que asocia *llaves* o *claves* con *valores*. La operación principal que soporta de manera eficiente es la *búsqueda*: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un *hash*, un número que la tabla hash utiliza para localizar el valor deseado.

Tablas temporales: permite la creación de tablas temporales, visibles exclusivamente en la sesión abierta, y guardar datos entre consultas. La utilidad de las tablas temporales se limita a consultas complejas que deben generar resultados intermedios que debemos consultar (hacer 'join' con ellas) varias veces o en consultas separadas.

Tasa de transferencia: corresponde a la velocidad media con que los datos son transferidos desde la red del ISP al usuario conectado a éste, durante períodos de tiempo determinados, medida en bits por segundo y presentada en tres parámetros: promedio, máxima, mínima.

TCP/IP: El modelo TCP/IP, describe un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que una computadora pueda comunicarse en una red. TCP/IP provee conectividad de extremo a extremo especificando como los datos deberían ser formateados, direccionados, transmitidos, enrutados y recibidos por el destinatario. Existen protocolos para los diferentes tipos de servicios de comunicación entre computadoras.

Tiempo de respuesta: El tiempo de respuesta se define como el tiempo que pasa desde que se envía una comunicación y se recibe la respuesta.

Transacciones: Una transacción en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Tupla: En algunos lenguajes y especialmente en la teoría de bases de datos, una tupla se define como una función finita que *mapea* (asocia unívocamente) los nombres con algunos valores.

Uso compartido de datos: El DBMS debe permitir que múltiples usuarios accedan simultáneamente a la base de datos.

Valor atómico: significa "indivisible", es decir, cada atributo debe contener un único valor del *dominio*. Los atributos, en cada tabla de una base de datos 1FN (primera forma normal), no pueden tener listas o arrays (arreglos) de valores, ya sean del mismo dominio o de dominios diferentes.

Verificación de integridad: Los datos deben ser internamente coherentes y, cuando algunos elementos hacen referencia a otros, estos últimos deben estar presentes.

Vistas: una vista es un objeto de la base de datos que se define mediante una SELECT que agrupa o selecciona un conjunto de datos.

REFERENCIAS

Berzal Galiano, F. (s.f.). *Cursos de Informática.* Recuperado el 24 de Febrero de 2010, de Cursos de Informática: http://elvex.ugr.es/idbis/db/docs/design/1-process.pdf

Castán Salinas, A. (Febrero de 2006). *XTEC Xarxa Telemática Educativa de Catalunya.* Recuperado el Febrero de 2010, de Guía rápida de administración de MySQL: http://www.xtec.net/~acastan/textos/Administracion%20de%20MySQL.html

Castaño, A. d., & Piattini Velthuis, M. G. (1993). *Concepción y Diseño de Bases de Datos: Del modelo E/R al modelo relacional.* España: RA-MA.

Castaño, A. d., & Piattini Velthuis, M. G. (1999). *Fundamentos y modelos de bases de datos.* México: Alfaomega Grupo Editor.

Castillo, C. (28 de Marzo de 2008). *Tejedores de la Web.* Recuperado el 22 de Septiembre de 2009, de Sistemas de Información II. Modelo Entidad-Relación: http://www.tejedoresdelweb.com/wiki/images/c/c7/Basesdatos_teo3_modelo_er.pdf

Cuadra, D., Castro, E., Iglesias, A. M., Martínez, P., Calle, F. J., de Pablo, C., y otros. (2008). *Desarrollo de Bases de Datos.* México: Editorial Alfaomega Ra-Ma.

DuBois, P. (2001). *Edición Especial MySQL.* Madrid: Pearson Educación.

Gilfillan, I. (2003). *La Biblia de MySQL.* Madrid: Anaya Multimedia.

Hinz, S., Lead, T., DuBois, P., Stephens, J., Bedford, A., & Russell, J. (2010). *MySQL The world's most popular open source database*. Recuperado el 14 de Noviembre de 2009, de MySQL The world's most popular open source database: http://dev.mysql.com/doc/index-about.html

Jaque Barbero, M. (6 de Abril de 2007). *Ilke Benson Ingenieria de Software Libre.* Recuperado el Febrero de 2010, de Manual de Supervivencia del Administrador de MySQL: http://www.ilkebenson.com/articulos/mysql2.pdf

Jaque Barbero, M. (24 de Julio de 2008). *Ilke Benson Ingenieria de Software Libre.* Recuperado el Febrero de 2010, de Optimización de MySQL: http://www.ilkebenson.com/articulos/mysql2.pdf

Kendall, K. E., & Kendall, J. E. (1997). *Análisis y Diseño de Sistemas.* España: Prentice Hall.

L. Gillenson, M. (2006). *Administración de Bases de Datos.* México: Editorial Limusa Wisley.

López Quijado, J. (2001). *Domine PHP y MySQL. Programación din´mica en el lado del servidor.* México: Alfa Omega Ra-Ma.

Lucas Gómez, A., Romera García, P., Fraile Dotes, M. V., Argente del Castillo, F. J., & Alfaro Pesa, A. (1993). *Diseño y Gestión de Sistemas de Bases de Datos.* España: Paraninfo.

M. Kroenke, D. (1996). *Procesamiento de Bases de Datos.* México: Ed. Prentice Hall.

Marqués Andrés, M. M. (10 de febrero de 2001). *Universidad Ujimel.* Recuperado el 22 de septiembre de 2009, de Apuntes de ficheros y bases de datos: http://www3.uji.es/~mmarques/f47/apun/apun.pdf

Martín Escofet, C. (2008). *OpenCourseWare.* Recuperado el 24 de febrero de 2010, de OpenCourseWare: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02149.pdf

Navathe, S. B., & Elmesn, R. A. (2002). *Fundamentos de Sistemas de Bases de Datos.* EE.UU: Addison Wesley Pearson Educación.

ORACLE. (2011). *MySQL*. Recuperado el Febrero de 2010, de MySQL 5.1 Reference Manual: http://dev.mysql.com/doc/refman/5.1/en/index.html

Pastor López, O., & Blesa Pons, P. (2000). *Gestión de Bases de Datos.* España: Editorial Servicio de Publicaciones Departamento de Sistemas Informáticos y Computación .

Pozo Coronado, S. (Mayo de 2005). *Scribd.* Recuperado el Febrero de 2010, de MySQL con Clase Gestión de bases de datos: http://es.scribd.com/doc/4671122/Curso-MySQL

Riordan, R. M. (2000). *Diseño de base de datos relacionales con Access y SQL Server.* México: Editorial Mc Graw-Hill.

Rivero Cornelio, E., Alonso Martínez, I., & Martínez Fuentes, L. (2005). *Bases de Datos Relacionales: Fundamentos y Diseño Lógico.* España: R.B Servicios Editoriales S.L.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *Fundamentos de Bases de Datos.* España: Mc Graw Hill.

Squire, E. (1984). *Introducción al Diseño de Sistemas.* España: Fondo Educativo Interamericano.

Whitten, J. L., & Bentley, L. D. (2008). *Análisis de Sistems Diseño y Métodos.* España: Mc Graw Hill.

Whitten, J. L., Bentley, L. D., & Barlow, V. M. (1996). *Análisis y Diseño de Sistemas de Información.* España: Mc Graw Hill.