UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

BASE DE DATOS

INGENIERIA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

UNIREMINGTON
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

AUTOR

Luisa María Jiménez Ramos
Ingeniera de Sistemas – Especialista en Gerencia Informática.
luisa.jimenez@uniremington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

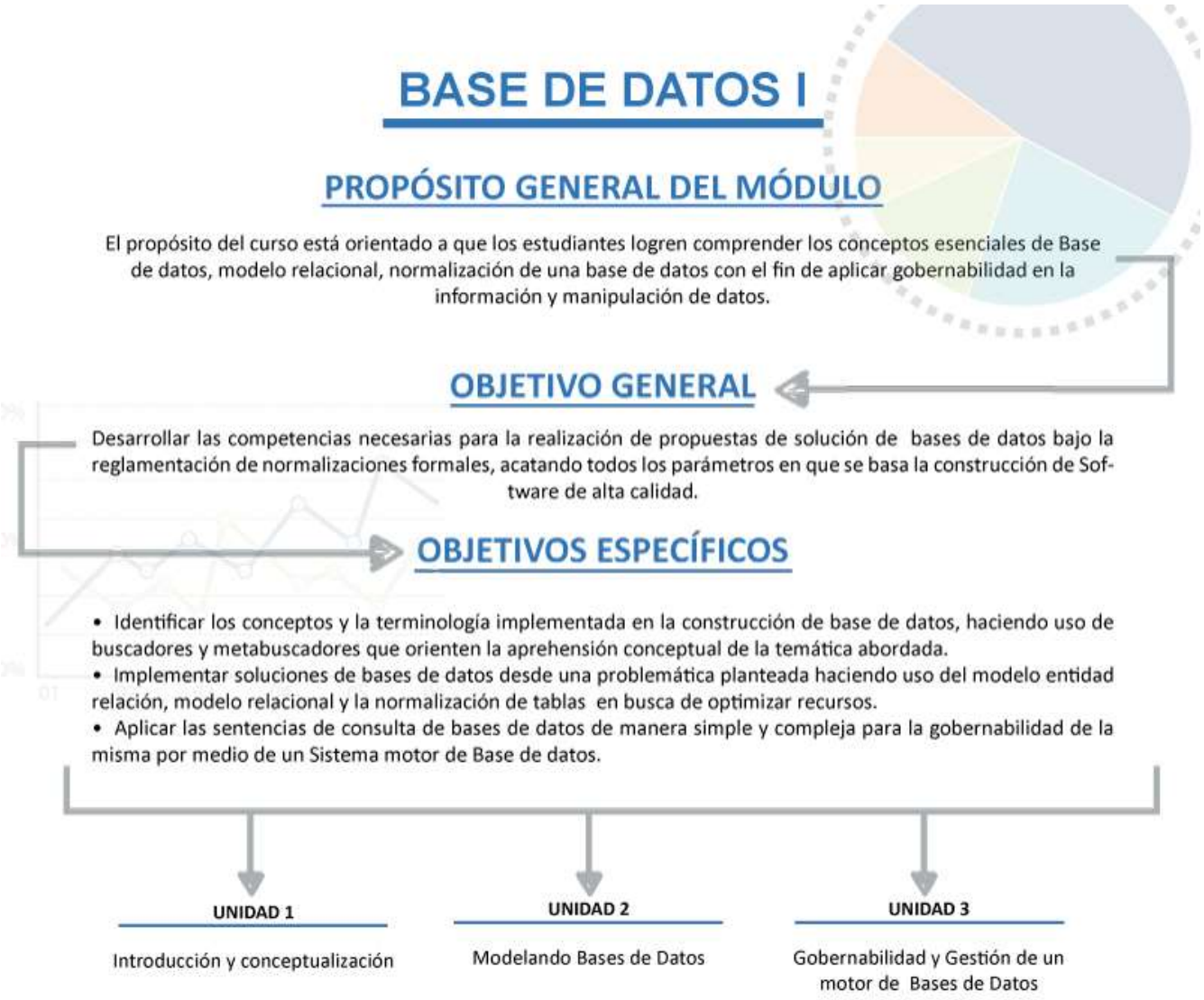
Personal de la Unidad CUR-Virtual
EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011.
Segunda versión. Marzo de 2012
Tercera versión. noviembre de 2015
Cuarta versión 2016

TABLA DE CONTENIDO

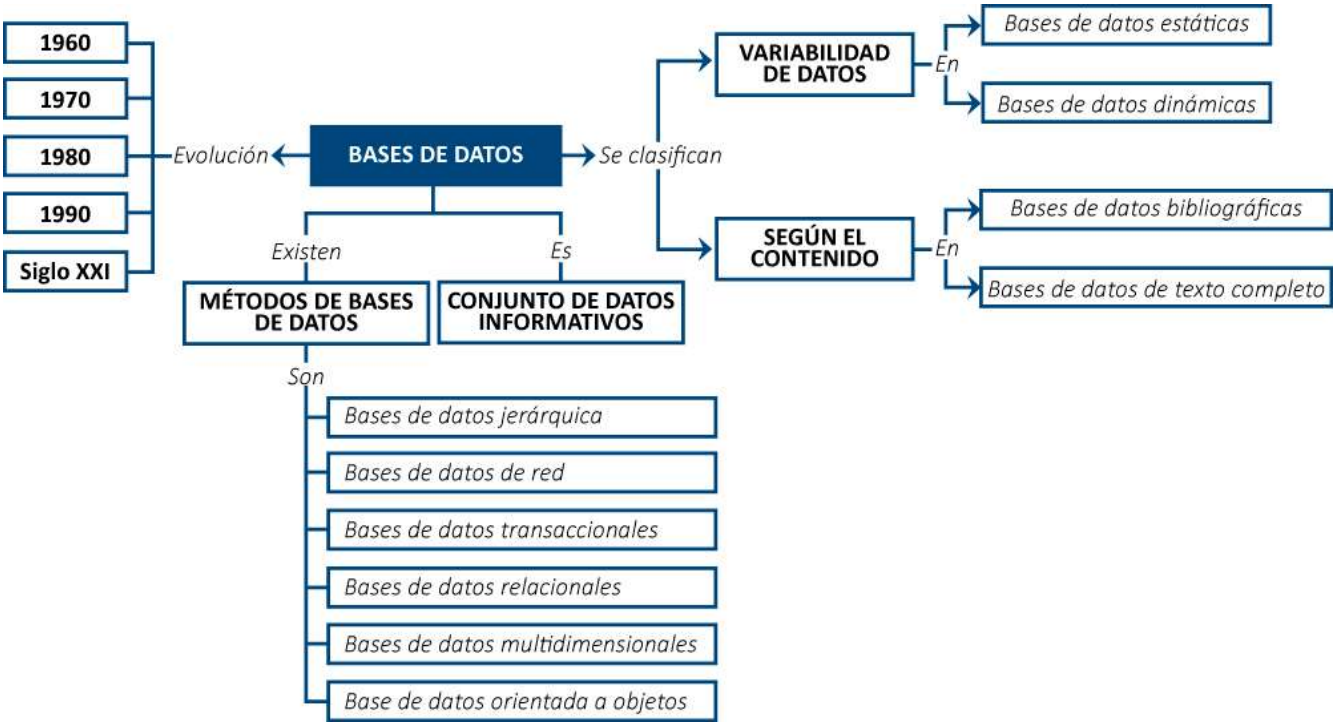
Pág.

1 MAPA DE LA ASIGNATURA



2 INTRODUCCIÓN Y CONCEPTUALIZACIÓN

2.1 TEMA 1 DEFINICIÓN, HISTORIA Y CONCEPTOS CLAVES DE LAS BASES DE DATOS.



Conceptos Básicos:

1. **Bases de Datos**: Conjunto de datos informativos organizados en un mismo contexto para su uso y vinculación.

2. **Modelos de Bases de Datos**: una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores.

2.1.1 DEFINICIÓN

Una base de datos se define como:

El **conjunto** de **datos informativos organizados** en un mismo contexto para su **uso** y vinculación. Se le llama **base de datos** a **los bancos de información** que contienen **datos relativos** a diversas temáticas y **categorizados** de **distinta manera**, pero que comparten entre sí algún tipo de **vínculo** o **relación** que busca **ordenarlos** y **clasificarlos** en conjunto.

2.1.2 HISTORIA

La historia de las bases de datos está determinada por su **evaluación** en **cada década** y por **los principales actores** de **cada momento**. Actualmente, las bases de datos están teniendo **un impacto decisivo** sobre el creciente uso de las computadoras y el apoyo que estas ofrecen a las organizaciones como factor de decisión. Las Décadas más importantes son 1960, 1970, 1980, 1990 y Siglo XXI.

Historia de la base de datos Enlace

http://dryvalleycomputer.com/index.php/bases-de-datos/introduccion/45-historia-de-las-bases-de-datos

2.1.3 TALLER DE ENTRENAMIENTO HISTORIA Y EVOLUCION BASES DE DATOS.

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente la pregunta propuesta y dar respuesta a ella.

1. Después de revisar el video y el enlace propuesto sobre la historia y evolución de las bases de datos, determine los actores y aportes más importantes realizados en cada década.

2.1.4 TIPOS DE BASES DE DATOS

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al contexto que se esté manejando, la utilidad de las mismas o las necesidades que satisfagan.

A. SEGÚN LA VARIABILIDAD DE LOS DATOS	
1. Bases de datos estáticas	Son bases de datos **únicamente de lectura**, utilizadas primordialmente para almacenar **datos históricos** que posteriormente se pueden utilizar para: - **Estudiar el comportamiento** de un **conjunto de datos** a través del tiempo, - **Realizar proyecciones**, - **Tomar decisiones** y - Realizar **análisis de datos** para **inteligencia empresarial.**
2. Bases de dato dinámicas	Son bases de datos donde **la información almacenada** se **modifica con el tiempo**, permitiendo **operaciones** como: - **Actualización,** - **Borrado y edición de datos, además de** - **Las operaciones fundamentales de consulta.** Un ejemplo, puede ser **la base de datos** utilizada en un **sistema de información** de un supermercado.
B. SEGÚN EL CONTENIDO	
	Sólo contienen un **subrogante** (**representante**) de **la fuente primaria**, que permite **localizarla**. Un **registro**

1. Bases de datos Bibliográficas	**típico** de **una base de datos bibliográfica** contiene información de una determinada publicación sobre: - **El autor,** - **Fecha de publicación,** - **Editorial,** - **Título,** - **Edición, entre otros.** Puede contener **un resumen** o **extracto** de la publicación original, pero **nunca el texto completo**, porque si no, estaríamos en presencia de **una base de datos a texto completo**. Como su nombre lo indica, el contenido son **cifras** o **números**. Por ejemplo, **una colección** de **resultados de análisis** de **laboratorio**, entre otras.
2. Bases de datos de texto completo	**Almacenan** las **fuentes primarias**, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

Tipos de bases de datos Enlace

Enlace: http://www.netronycs.com/clasificacion_de_base_datos.html

2.1.5 MODELOS DE BASES DE DATOS

Además de la clasificación por **la función** de **las bases de datos**, éstas también se pueden clasificar de acuerdo a su **modelo de administración de datos**. Un modelo de datos es básicamente una **"descripción"** de algo conocido como **contenedor de datos** (algo en donde se **guarda la información**), así como de **los métodos** para **almacenar** y **recuperar información** de esos **contenedores**. Los modelos de datos **no son cosas físicas**: son **abstracciones** que permiten **la implementación** de **un sistema eficiente** de base de datos; por lo general se refieren a **algoritmos** y **conceptos matemáticos**.

BASES DE DATOS	CARACTERÍSTICAS
1. Base de datos jerárquica	En este modelo **los datos** se organizan en forma de **árbol invertido** (algunos dicen **raíz**), en donde un **nodo padre de información** puede tener **varios hijos**. **El nodo** que **no tiene padres** es llamado **raíz**, y a **los nodos** que **no tienen hijos** se los conoce como **hojas**. Las **bases de datos jerárquicas** son especialmente útiles en el caso de **aplicaciones** que **manejan** un **gran volumen de información** y **datos muy compartidos** permitiendo **crear estructuras estables** y de **gran rendimiento**. Una de **las principales limitaciones** de este modelo es su **incapacidad** de **representar eficientemente** la **redundancia de datos**.
2. Base de datos de red	Éste es un modelo **ligeramente distinto** del jerárquico; su diferencia fundamental es **la modificación del concepto de nodo**: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).
3. Base de datos transaccionales	Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, estas bases son muy poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción e industrial, es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto la redundancia y duplicación de información no es un problema como con las demás bases de datos, por lo general para poderlas aprovechar al máximo permiten algún tipo de conectividad a bases de datos relacionales.

4. Base de datos relacionales	Éste es el modelo utilizado en la actualidad para representar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).
5. Base de datos multidimensionales	Son bases de datos ideadas para desarrollar aplicaciones muy concretas, como creación de Cubos OLAP. Básicamente no se diferencian demasiado de las bases de datos relacionales (una tabla en una base de datos relacional podría serlo también en una base de datos multidimensional), la diferencia está más bien a nivel conceptual; en las bases de datos multidimensionales los campos o atributos de una tabla pueden ser de dos tipos, o bien representan dimensiones de la tabla, o bien representan métricas que se desean aprender
6. Base de datos orientada a objetos	Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de 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.

2.1.6 TALLER DE ENTRENAMIENTO TIPOS DE BASES DE DATOS.

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente las preguntas propuestas y dar respuesta a ellas.

1. Establezca la diferencia entre bases de dato dinámica y base de dato estática.

2. Enuncie la importantica que tienen los tres tipos de bases de datos vistas anteriormente.

2.1.7 SISTEMA DE GESTIÓN DE BASE DE DATOS (DBMS)

Sistema de gestion de base de datos Enlace

Enlace: http://users.dsic.upv.es/~jorallo/docent/BDA/castella/tema3_4x1.pdf

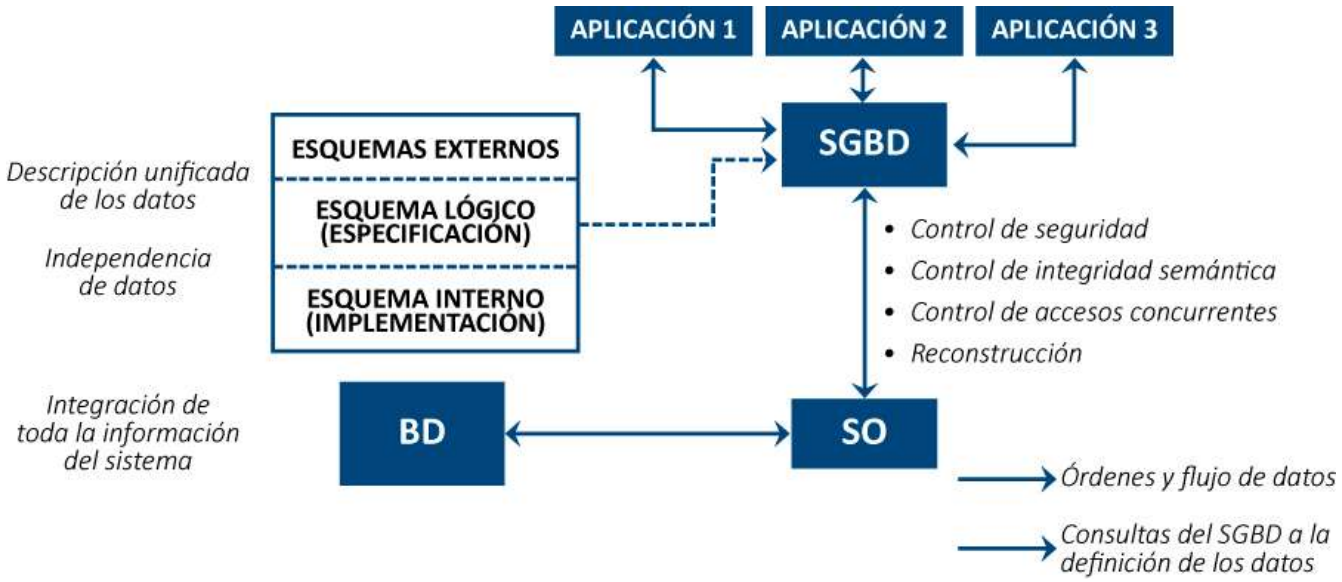
Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Los usuarios pueden acceder a la información usando herramientas específicas de interrogación y de generación de informes, o bien mediante aplicaciones al efecto.

Estos sistemas también proporcionan métodos para mantener la **integridad de los datos**, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe. Permiten presentar la información de la base de datos en variados formatos. La mayoría **incluyen** un generador **de informes**. También

pueden incluir un módulo gráfico que permita presentar la información con gráficos y tablas. Hay muchos tipos distintos según cómo manejen los datos y muchos tamaños distintos de acuerdo a si operan en computadoras personales y con poca memoria o grandes sistemas que funcionan en mainframes con sistemas de almacenamiento especiales.

En el proceso de construcción de una base de datos se almacenan los datos en algún medio de almacenamiento permitiendo hacer las transacciones básicas con ellos tales como:

1. **Funciones de consulta.**



2. **Funciones de actualización.**

3. **Funciones de Eliminación entre otros.**

Los sistemas de gestión de bases de datos tienen una clasificación de la siguiente manera:

Sistemas de Gestión de Bases de Datos Enlace

2.1.8 TALLER DE ENTRENAMIENTO SISTEMA DE GESTION DE BASES DE DATOS

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente las preguntas propuestas y dar respuesta a ellas.

Con Sus propias palabras explique en que consiste la gestión de bases de datos y describa su importancia.

2.2 TEMA 2 MOTORES DE BASES DE DATOS

Para el manejo y gestión de bases de datos existen varios motores que se usan dependiendo el tamaño y tipo, además de la integridad que se exige, pues hay algunas bases de datos que requieren mayor seguridad que otras todo esto depende para que están siendo usadas.

El Motor de base de datos es el **servicio principal** para almacenar, procesar y proteger los datos. El Motor de base de datos proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa.

Use Motor de base de datos para crear bases de datos relacionales para el procesamiento de transacciones en línea o datos de procesamiento analíticos en línea. Se pueden crear tablas para almacenar datos y objetos de base de datos como índices, vistas y procedimientos almacenados para ver, administrar y proteger los datos. Puede usar SQL Server Management Studio para administrar los objetos de bases de datos y SQL Server Profiler para capturar eventos de servidor.

los mejores motores de bases de datos Enlace

Enlace: https://technet.microsoft.com/es-es/library/ms187079(v=sql.105).aspx

2.2.1 MYISAM

Es el mecanismo de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales MySQL. Este tipo de tablas están basadas en el formato ISAM pero con nuevas extensiones. En las últimas versiones de MySQL, el motor InnoDB está empezando a reemplazar a este tipo de tablas por su capacidad de ejecutar transacciones de tipo ACID y bloqueo de registros e integridad referencial.

Cada tabla de tipo **MyISAM** se guarda en tres archivos. Los archivos tienen el nombre de la tabla y una extensión que indica el tipo de archivo:

1. Frm **almacena la definición de la tabla**
2. **MYD (MyData) contiene los registros de la tabla**
3. **MYI (MyIndex) contiene los índices de la tabla**

Para especificar que deseas usar el tipo de tablas MyISAM, se indica con la opción ENGINE al crear la tabla o modificarla, por ejemplo:

CREATE TABLE t (i INT) ENGINE = MYISAM;

La principal característica de este tipo de almacenamiento es la gran velocidad que obtiene en las consultas, ya que no tiene que hacer comprobaciones de la integridad referencial, ni bloquear las tablas para realizar las operaciones por la ausencia de características de atomicidad. Este tipo de tablas está especialmente indicado para sistemas que no tienen un número elevado de **inserciones** como pueden ser las páginas web.

2.1.2. InnoDB

Es un mecanismo de almacenamiento de datos de código abierto para la base de datos MySQL, incluido como formato de tabla estándar en todas las distribuciones deMySQL 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 dependerán de la aplicación específica.**

2.1.3. MYSQL

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 ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

2.1.4. POSTGRESQL

PostgreSQL es un Sistema de gestión de bases de datos relacional orientado a objetos y libre, publicado bajo la licencia PosgreSQL , similar a la BSD o la MIT. Como muchos otros proyectos de código abierto, el desarrollo de **PostgreSQL no es manejado por una empresa y/o persona**, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados pororganizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

2.1.5. ORACLE

Oracle Database es un sistema de gestión de base de datos de tipo objeto-relacional (ORDBMS, por el acrónimo en inglés deObject-Relational Data Base Management System), desarrollado por Oracle Corporation.

Se considera a **Oracle Database como uno de los sistemas de bases de datos más completos**, destacando: soporte detransacciones, estabilidad, escalabilidad, y soporte multiplataforma. Su dominio en el mercado de servidores empresariales había sido casi total hasta que recientemente tiene la competencia delMicrosoft SQL Server y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Las siguientes direcciones corresponden a videos que te ayudarán a la comprensión y el mejor aprovechamiento de la unidad y que pueden ser bajados de youtube.

MOTORES DE BASES DE DATOS Enlace

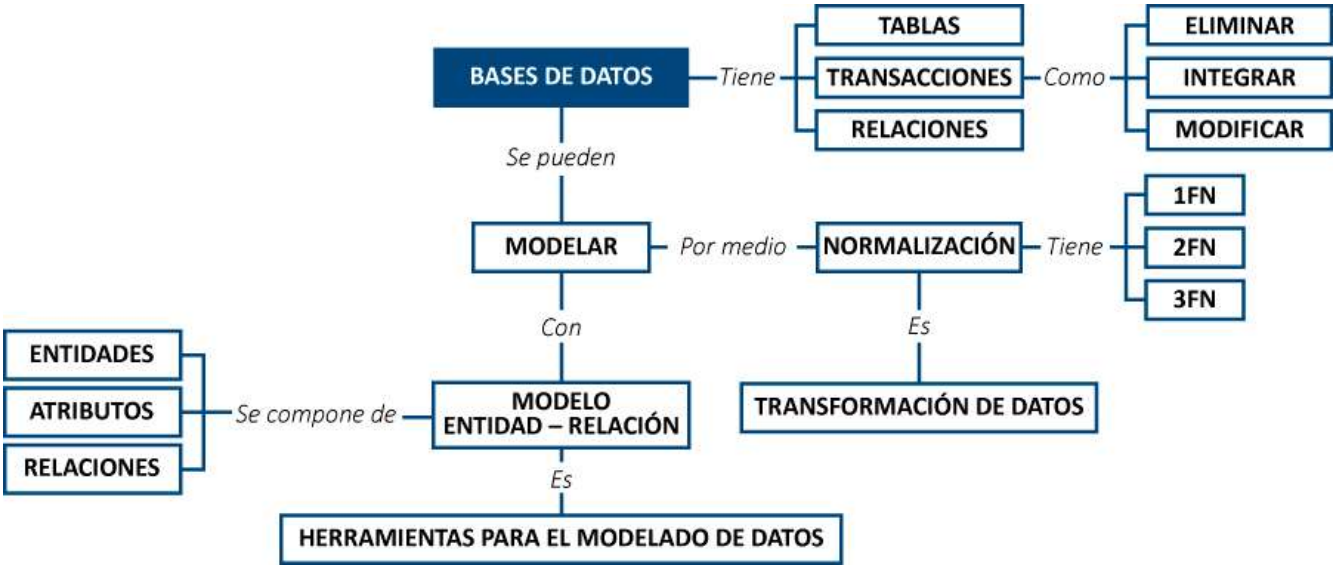
https://www.youtube.com/user/Oracle

2.2.2 TALLER DE ENTRENAMIENTO MOTORES DE BASES DE DATOS

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente las preguntas propuestas y dar respuesta a ellas.

Realice un paralelo entre los dos motores de Mysql más utilizados (MyISAM e InnODB) y ubique sus ventajas y desventajas**.**

3 UNIDAD 2 MODELANDO BASES DE DATOS



Conceptos Básicos:

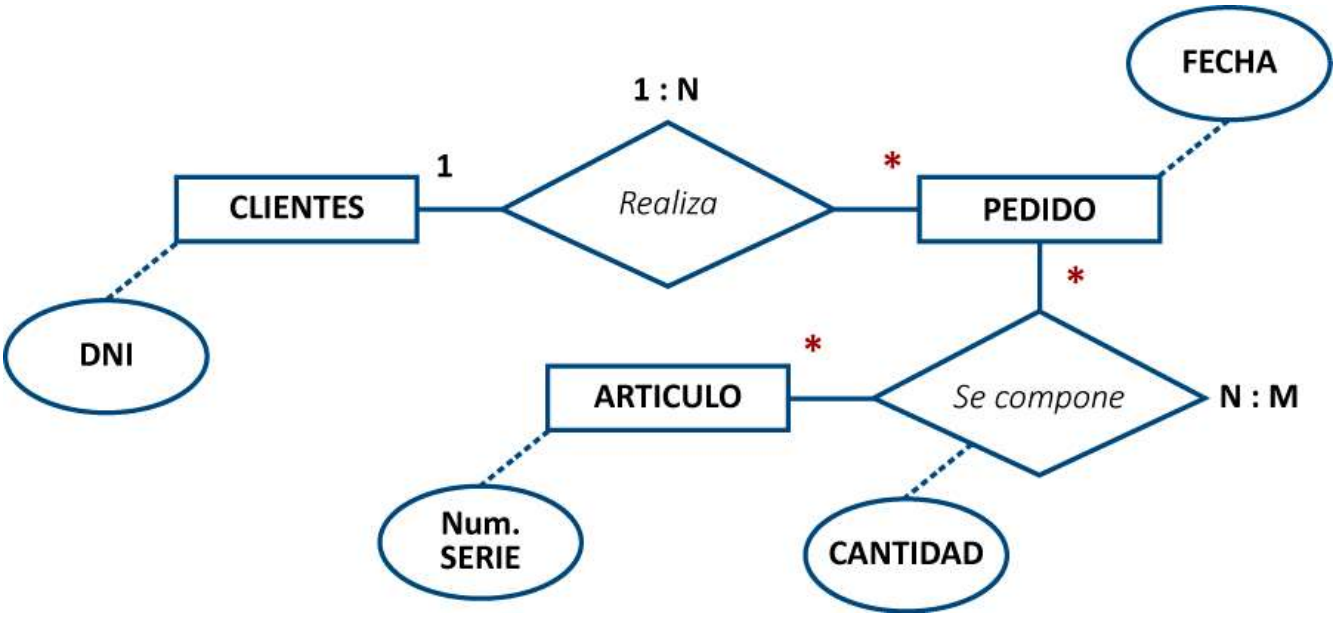
1. **Modelos de Bases de Datos**: una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores.

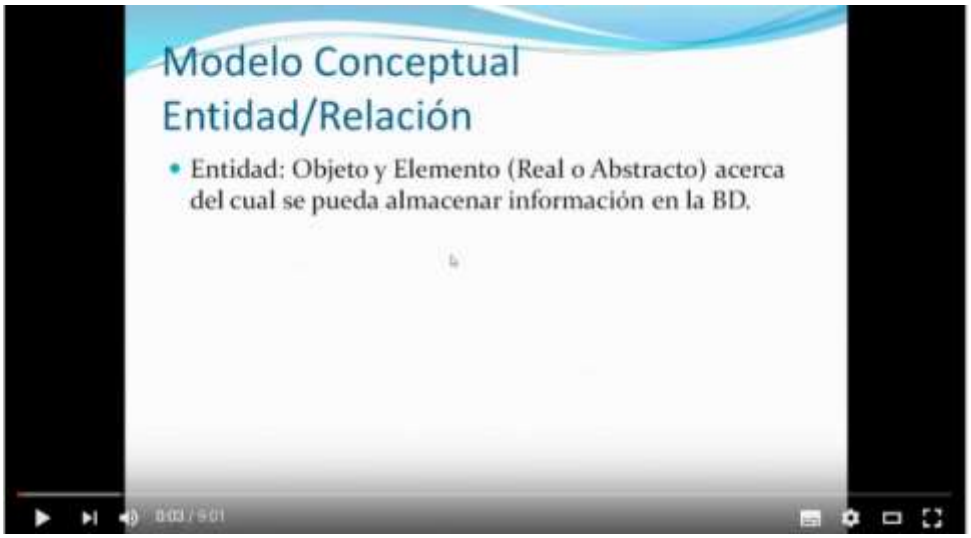
2. **Modelo Entidad - Relación:** una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

3. **Normalización:** proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

3.1 TEMA 1 MODELO ENTIDAD RELACIÓN

Se define como un diagrama o modelo entidad-relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.





Modelo Entidad Relacion de Base de Datos [Enlace]

El modelo entidad relación se compone de unos elementos tales como:

1. **Entidad:** Las entidades representan *cosas* u *objetos* (ya sean reales o abstractos), que se diferencian claramente entre sí. Su representación gráfica es la siguiente:

2. **Atributos:** Los atributos definen o identifican las características de entidad (**es el contenido de esta entidad**). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos

atributos pueden ser de distintos tipos (numéricos, texto, fecha...). Su representación gráfica es la siguiente:

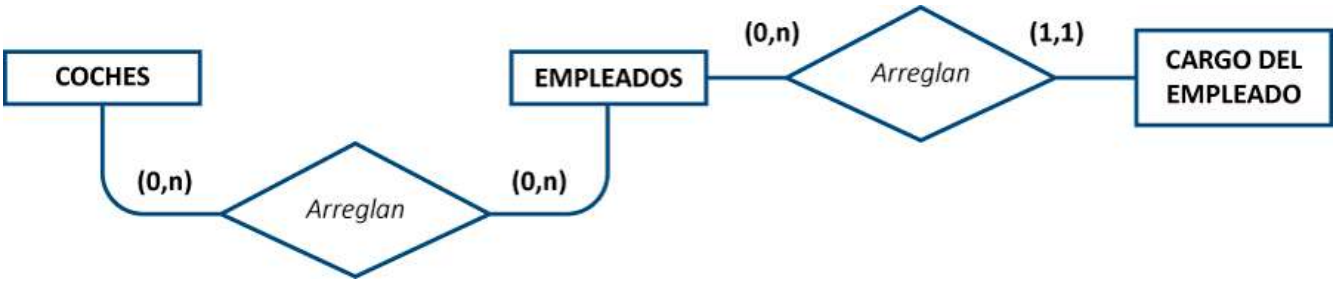
1. **Relación:** Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable. Su representación gráfica es la siguiente:

3.1.1 EJERCICIO DE APRENDIZAJE 1. MODELO ENTIDAD RELACIÓN:

Se tiene un taller mecánico en donde se manejan coches, empleados y el cargo del empleado, es importante tener en cuenta que para los autos se requiere el número del chasis, matricula, Numero de identificación del propietario, marca y modelo. Por otro lado para los empleados se requiere nombre, Numero de identificación y cargo.

La solución a la situación planteada se resuelve de esta forma:

1. Se definen las entidades: Coches, Empleados, Cargo del empleado.
2. Se definen los atributos de las entidades: para coches número del chasis, matricula, Numero de identificación del propietario, marca y modelo; Para empleados nombre, Numero de identificación y cargo. Y para cargo tipo de cargo.
3. Se establecen relaciones como Arreglan y es en el taller.
4. Se definen relaciones de carnalidad
5. Se procede a realizar el diagrama.



3.2 EJERCICIO DE APRENDIZAJE 2. MODELO ENTIDAD RELACIÓN:

Se requiere una base de datos para una pequeña empresa debe contener información acerca de clientes, artículos y pedidos. Hasta el momento se registran los siguientes datos en documentos varios:

1. Para cada cliente: Número de cliente (único), Direcciones de envío (varias por cliente), Saldo, Límite de crédito (depende del cliente, pero en ningún caso debe superar los 3.000.000 pts), Descuento.

2. Para cada artículo: Número de artículo (único), Fábricas que lo distribuyen, Existencias de ese artículo en cada fábrica, Descripción del artículo.

3. Para cada pedido: Cada pedido tiene una cabecera y el cuerpo del pedido. La cabecera está formada por el número de cliente, dirección de envío y fecha del pedido. El cuerpo del pedido son varias líneas, en cada línea se especifican el número del artículo pedido y la cantidad. Además, se ha determinado que se debe almacenar la información de las fábricas. Sin embargo, dado el uso de distribuidores, se usará: Número de la fábrica (único) y Teléfono de contacto. Y se desean ver cuántos artículos (en total) provee la fábrica. También, por información estratégica, se podría incluir información de fábricas alternativas respecto de las que ya fabrican artículos para esta empresa.

   Nota: Una dirección se entenderá como Nº, Calle, Comuna y Ciudad. Una fecha incluye hora. Se pide hacer el diagrama ER para la base de datos que represente esta información.

La solución a la situación planteada se resuelve de esta forma:

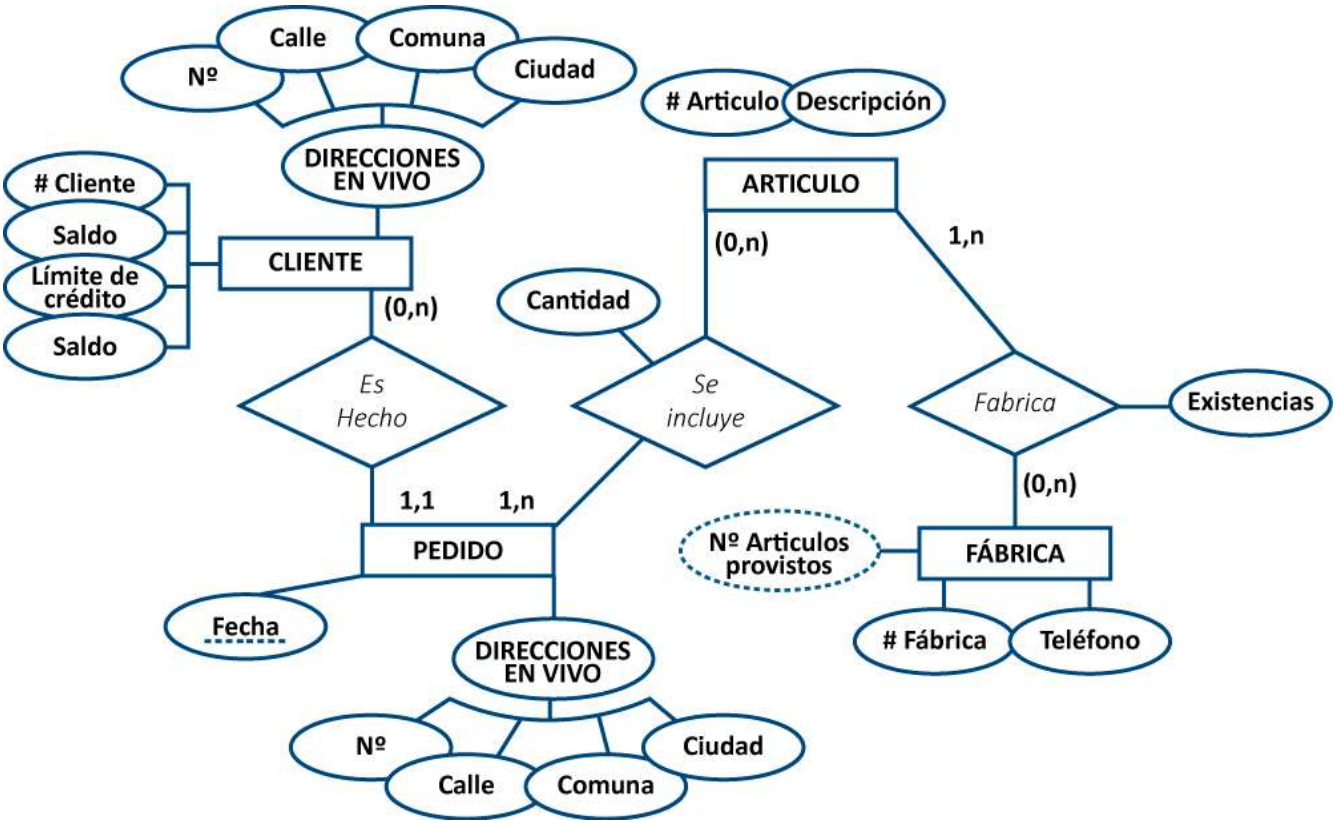
1. Se definen las entidades: Cliente, pedido, articulo fábrica.

2. Se definen los atributos de las entidades: para Cliente serian Descuento, límite de crédito, Saldo, #Cliente, Direcciones de Envío. Nótese que de este último atributo se desprenden otros atributos. Para Pedido se tiene fecha y Dirección de envío. Para artículo se tiene #articulo y descripción y para fabrica #de fábrica y teléfono.

3. Se establecen relaciones como Es hecho, Se incluye y fabrica.

4. Se definen relaciones de carnalidad

5. Se procede a realizar el diagrama.



Notas: - El Nº de artículos provistos es la suma de las existencias de cada artículo.

- Se podría almacenar una fábrica de la cual no se tengan artículos

En el siguiente enlace encuentra una serie de ejercicios resueltos que ayudaran a comprender de mejor forma el tema: *http://users.dcc.uchile.cl/~mnmonsal/BD/guias/g-modeloER.pdf*

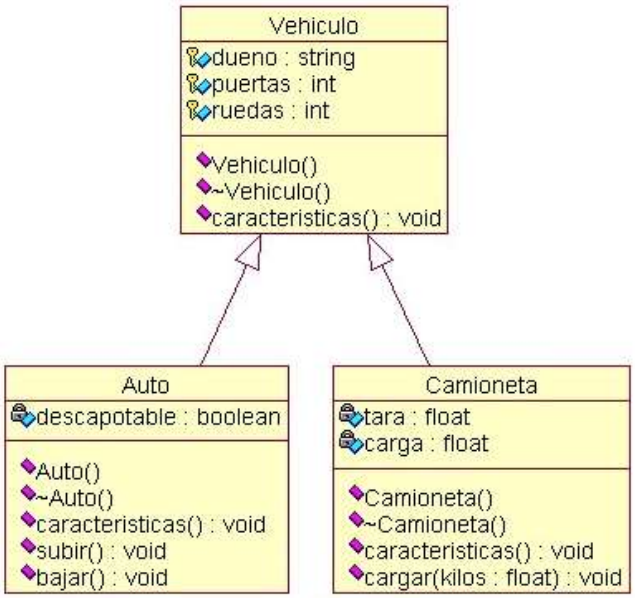
3.2.1 REPASO DIAGRAMA DE CLASE

Un diagrama de clases sirve **para visualizar** las relaciones entre **las clases** que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Veamos el siguiente ejemplo:



Fuente: : http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

La clase vehículo la cual tiene unos atributos como lo son: dueño, puertas y ruedas. Esta se relaciona con la clase Auto cuyos atributos son descapotable y se relaciona con la clase camioneta que tiene los atributos de tara y carga. Las clases Auto y Camioneta heredan de la clase vehículo.

Diagrama de clases Enlace

Enlace: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

3.2.2 TALLER DE ENTRENAMIENTO MODELO ENTIDAD RELACION

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente las situaciones propuestas y dar respuesta a ellas.

Crear un diseño entidad relación (estando prohibido utilizar símbolos del modelo extendido) que permita gestionar los datos de una biblioteca de modo que Las personas socias de la biblioteca disponen de un código de socio y además necesitar almacenar su dni, dirección, teléfono, nombre y apellidos La biblioteca almacena libros que presta a los socios y socias, de ellos se almacena su título, su editorial, el año en el que se escribió el libro, el nombre completo del autor (o autores), el año en que se editó y en qué editorial fue y el ISBN.

Necesitamos poder indicar si un volumen en la biblioteca está deteriorado o no Queremos controlar cada préstamo que se realiza almacenando la fecha en la que se realiza, la fecha tope para devolver (que son 15 días más que la fecha en la que se realiza el préstamo) y la fecha real en la que se devuelve el libro

Crear un diseño entidad/relación que permita modelar un sistema que sirva para gestionar una empresa que posee inmuebles. Para ello Se almacenan los clientes usando su DNI, Teléfono fijo, Móvil, Nombre y Apellidos. Se almacenan los trabajadores y se almacenan los mismos datos. Ocurre además que un trabajador puede ser un cliente (porque puede alquilar o comprar mediante la inmobiliaria) a veces

A cada cliente y trabajador se le asigna un código personal Los clientes pueden comprar pisos, locales o garajes. En los tres casos se almacena un código de inmueble (único para cada inmueble), los metros que tienen, una descripción y su dirección. Los pisos tienen un código especial de piso que es distinto para cada piso. En los locales se indica el uso que puede tener y si tienen servicio o no. De los garajes se almacena el número de garaje (podría repetirse en distintos edificios) y la planta en que se encuentra (para el caso de garajes que están en varias plantas). Los garajes además pueden asociarse a un piso y así cuando se alquile el piso se incluirá el garaje. La empresa prevé que podría haber inmuebles que podrían no ser ni locales, ni garajes, ni pisos Los inmuebles se pueden comprar. Incluso varias veces. Se asigna un código de compra cada vez que se haga, la fecha y el valor de la compra. La compra puede tener varios titulares. Cada inmueble se puede alquilar y en ese caso se asigna un número de alquiler por cada inmueble. Ese número se puede repetir en distintos inmuebles (es decir puede haber alquiler nº 18 para el inmueble 40 y el 35). Pero no se repite para el mismo inmueble. Al alquilar queremos saber el nombre del agente de la empresa que gestionó el alquiler así como a qué persona (solo una) estamos alquilando el inmueble. Cada pago de cada alquiler será almacenado en la base de datos, llevando el año, el mes y el valor del mismo.

3.2.3 NORMALIZACIÓN

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables**, son más fáciles de mantener**. También se puede entender la normalización como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataban de manipular los datos.

La normalización también hace las cosas fáciles de entender. Los seres humanos tenemos la tendencia de simplificar las cosas al máximo. Lo hacemos con casi todo, desde los animales hasta con los automóviles. Vemos una imagen de gran tamaño y la hacemos más simple agrupando cosas similares juntas. Las guías que la normalización provee crean el marco de referencia para simplificar una estructura de datos compleja. Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco. El proceso de normalización tiene un nombre y una serie de reglas para cada fase. Esto puede parecer un poco confuso al principio, pero poco a poco se va entendiendo el proceso, así como las razones para hacerlo de esta manera.

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización.

No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización, puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización. En la tabla siguiente se describe brevemente en que consiste cada una de las reglas, y posteriormente se explican con más detalle

Regla	Descripción
Primera Forma Normal (1FN)	Incluye la eliminación de todos los grupos repetidos.
Segunda Forma Normal (2FN)	Asegura que todas las columnas que no son llave sean completamente dependientes de la llave primaria (PK).
Tercera Forma Normal (3FN)	Elimina cualquier dependencia transitiva. Una dependencia transitiva es aquella en la cual las columnas que no son llave son dependientes de otras columnas que tampoco son llave.

Fuente: http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf

3.2.4 EJERCICIO DE APRENDIZAJE 3. NORMALIZACIÓN:

Tenemos una empresa pública donde los puestos de trabajo están regulados por el Estado, de modo que las condiciones salariales están determinadas por el puesto. Se ha creado el siguiente esquema relacional

EMPLEADOS(nss, nombre, puesto, salario, emails) con nss como clave primaria.

nss	nombre	puesto	salario	email
111	Juan Pérez	Jefe de Área	3000	juanp@ecn.es
111	Juan Pérez	Jefe de Área	3000	jefe2@ecn.es
222	José Sánchez	Administrativo	1500	jsanchez@ecn.es
333	Ana Díaz	Administrativo	1500	adiaz@ecn.es
333	Ana Díaz	Administrativo	1500	ana32@gmail.com

Primera Forma normal (1FN): Se elimina los grupos repetidos.

nss	nombre	puesto	salario	emails
111	Juan Pérez	Jefe de Área	3000	juanp@ecn.es; jefe2@ecn.es
222	José Sánchez	Administrativo	1500	jsanchez@ecn.es
333	Ana Díaz	Administrativo	1500	adiaz@ecn.es; ana32@gmail.com

Segunda Forma norma (2FN): Asegurar que todas las columnas que no son llave sean completamente dependientes de la llave primaria.

nss	nombre	puesto	salario	emails
111	Juan Pérez	Jefe de Área	3000	juanp@ecn.es; jefe2@ecn.es
222	José Sánchez	Administrativo	1500	jsanchez@ecn.es
333	Ana Díaz	Administrativo	1500	adiaz@ecn.es; ana32@gmail.com

Al aplicar la segunda forma normal nos quedaría:

nss	email
111	juanp@ecn.es
111	jefe2@ecn.es
222	jsanchez@ecn.es
333	adiaz@ecn.es
333	ana32@gmail.com

Tercera Forma Normal (3FN): Eliminar cualquier dependencia transitiva.

nss	nombre	puesto
111	Juan Pérez	Jefe de Área
222	José Sánchez	Administrativo
333	Ana Díaz	Administrativo

En los siguientes link encuentran mayor información del tema.

Normalización, de una base de datos Enlace

Enlace: http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf

http://cnx.org/contents/qtZsLi-X@1/Un-ejemplo-simple-de-normaliza

3.2.5 TALLER DE ENTRENAMIENTO NORMALIZACION

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente las situaciones propuestas y dar respuesta a ellas.

1. FACTURA DE COMPRA VENTA: La empresa COLOMBIAN SYSTEMS lo ha contratado como el "Ingeniero Encargado" para sistematizar la facturación. En la siguiente FACTURA DE COMPRA VENTA, usted debe analizar toda la información disponible y aplique el proceso de normalización, hasta llegar a la Tercera Forma Normal.
   Se pide realizar la respectiva justificación detallada de cada uno de los pasos que conduzcan al resultado final.

   Factura(NUM_FAC, FECHA_FAC, NOM_CLIENTE, DIR_CLIENTE, RIF_CLIENTE, CIUDAD_CLIENTE, TELEF_CLIENTE, CATEGORIA, COD_PROD, DESP_PROD, VAL_UNIT, CANT_PROD)

   Donde:

   NUM_FAC: Número de la factura de compra venta

   FECHA_FAC: Fecha de la factura de compra venta

   NOM_CLIENTE: Nombre del cliente

   DIR_CLIENTE: Dirección del cliente

   RIF_CLIENTE: Rif del cliente

   CIUDAD_CLIENTE: Ciudad del cliente

   TELEF_CLIENTE: Teléfono del cliente

   CATEGORIA: Categoría del producto

   COD_PROD: Código del producto

   DESCRIPCION: Descripción del producto

   VAL_UNIT: Valor unitario del producto

   CANT_PROD: Cantidad de productos q compra el cliente

   La llave primaria es Número de Factura de venta: NUM_FAC

2. EMPRESA DE ENVIO DE MERCANCIA: a continuación se agrupan todos los atributos que hacen parte de la base de datos para aplicarle las reglas de normalización. Donde se incluyen los nombres de los atributos con su significado

* GUIA_NO = Numero de Guia

* GUIA_FECHA= Fecha de la Guia

* GUIA_HORA= Hora de la Guia

* ORGN_RIF = Identificacion de Empresa Origen

* ORGN_NOM = Nombre de Empresa Origen

* ORGN_ACT = Actividad Comercial de Empresa Origen

* ORGN_CIUDAD= Ciudad de Empresa Origen

* ORGN_DIR = Direccion de Empresa Origen

* ORGN_TEL = Telefono de Empresa Origen

* ORGN_CEL = Celular de Empresa Origen

* DEST_ID = Identificacion del destinatario

* DEST_NOM = Nombre del destinatario

* DEST_COD_CIUDAD = Codigo de la ciudad del destinatario

* DEST_CIUDAD= Ciudad del destinatario

* DEST_DIR = Direccion del destinatario

* DEST_TEL = Telefono del destinatario

* DEST_KM = Distancia kilometraje de Ciudad origen a ciudad del destinatario

* CODIGO = Codigo del paquete

* TIPO = Tipo de paquete

* NOMBRE = Nombre del paquete

* DESCRIPCION = Descripción del paquete

* VALR_ FLETE = Valor del flete

3. Video club: En una tienda de video se necesita mantener información de alrededor de 3000 casetas cada uno de los casetes tiene asignado un número por cada `película se necesita conocer un titulo y categoría por ejemplo: comedia, suspenso, drama, acción, ciencia ficción, etc. Se mantienen algunas copias de muchas películas. Se le da a cada película una identificación y se mantiene seguimiento de lo que contiene cada casete.

Un casete puede venir en varios formatos y una película es grabada en un solo casete; frecuentemente las películas son pedidas de acuerdo a un actor especifico Tom Cruise y Demi More son los más populares es por esto que se debe mantener información de los actores que pertenecen a cada película.

No en todas las películas actúan artistas famosos, a los clientes de la tienda le gusta conocer datos como el nombre real del actor, y su fecha de nacimiento.

En la tienda se mantienen información solo de los actores que aparecen en las películas y que se tiene a disposición. Solo se alquila videos a aquellos que pertenecen al club de videos. Para pertenecer al club se debe tener un buen crédito. Por cada miembro del club se mantiene una ficha con su nombre, teléfono y dirección, cada miembro del club tiene asignado un número de membresía. Se desea mantener información de todos los casetes que un cliente alquila, cuando un cliente alquila un casete se debería conocer el nombre de la película, la fecha en la que se alquila y la fecha de devolución.

Se pide aplicar las reglas de normalización hasta la tercera forma normal, teniendo las siguientes entidades con sus respectivos atributos:

Alquiler (cod_alquiler, num_membresia, cod_cliente, nom_cliente, dir_cliente, telef_cliente, cod_cassette, fecha_alquiler, fecha_dev, valor_alquiler, cantidad)

Cassettte (cod_cassette, num_copias, formato, cod_pelicula, titulo, categoría, cod_actor, nom_actor, fechanac_actor, cod_tipo)

Donde:

cod_alquiler = Codigo del alquiler

num_membresia = Numero de membresia

cod_cliente = código del cliente

nom_cliente = nombre del cliente

dir_cliente = dirección del cliente

telef_cliente = teléfono del cliente

cod_cassette = código del cassette

fecha_alquiler = fecha del alquiler del al película

fecha_dev = fecha de devolución de la pelicula

valor_alquiler = valor del alquiler de la película

cantidad = cantidad de película alquilada

num_copias = números de copias de cassette

formato = formato del cassette

titulo = nombre de la película

categoría = categoría de la película

cod_actor = código del actor

nom_actor = nombre del actor

fechanac_actor = fecha de nacimiento del actor

cod_tipo = código del tipo de película.

4. Se presenta una base de datos de una biblioteca, aplicar las reglas de normalización simplificando hasta la tercera forma normal.

Prestamos_libro (codLibro, Titulo, Autor, Editorial, NombreLector, Fechadev)

codLibro	Titulo	Autor	Editorial	nombreLector	Fechadev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez Gómez, Juan	15/04/2005
1004	Visual Basic 5	E. Petroustsos	Anaya	Ríos Terán, Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca, René	16/04/2005
1006	Oracle University	Nancy Greenberg y Priya Nathan	Oracle Corp.	García Roque, Luis	20/04/2005
1007	Clipper 5.01	Ramalho	McGraw Hill	Pérez Gómez, Juan	18/04/2005

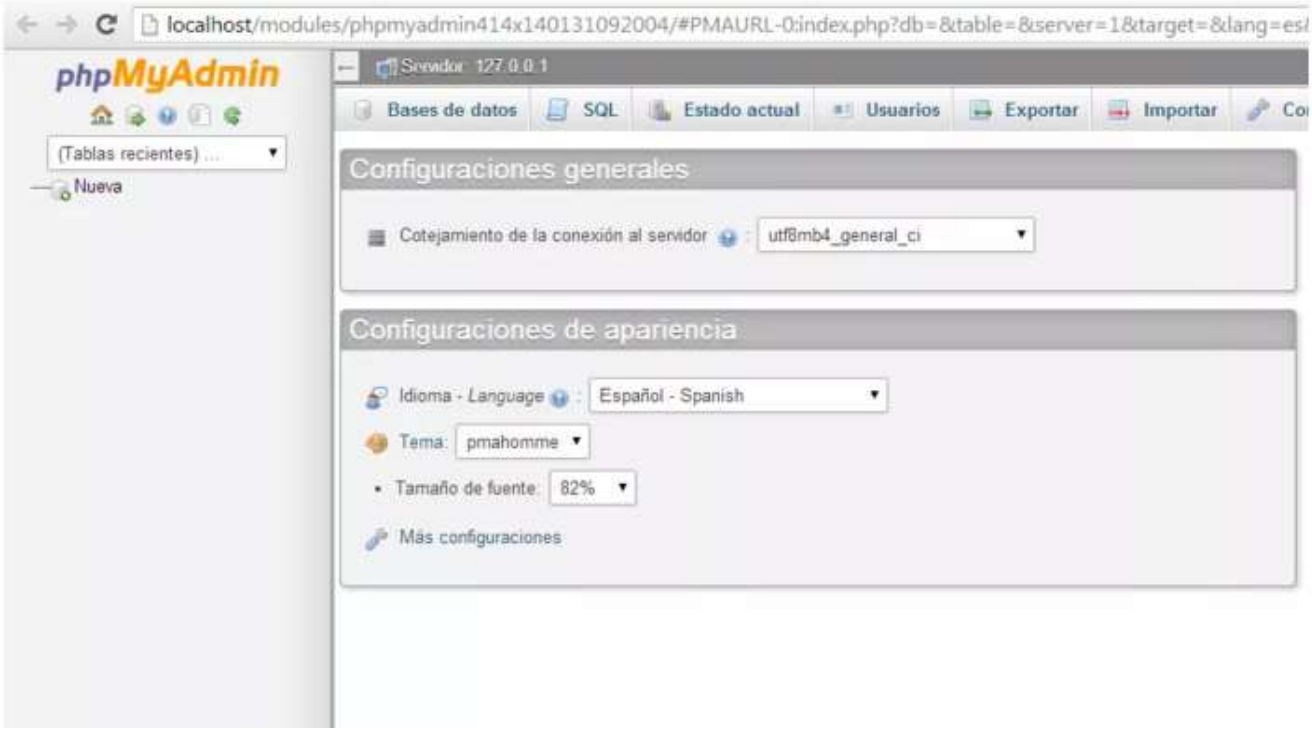
3.3 TEMA 2 ESTRUCTURA DE UNA BASE DE DATOS

En este momento se procede a establecer los pasos para creación de una base de datos con MySQL.

3.3.1 CREACIÓN DE UNA BASE DE DATOS.

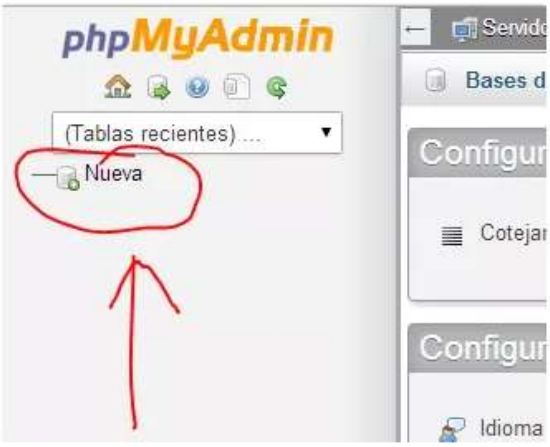
Para crear la base de datos usaremos PHPMyAdmin el cual **consiste en una interfaz web para la gestión de bases de datos.** Para poder usar PHPMyAdminvamos a descargar EasyPHP, este contenedor de programas contiene todas las herramientas necesarias para desarrollar una página web en un servidor local. Puedes descargar y saber más sobre EasyPHP en el tutorial que le dedicamos en Geeky Theory.

Una vez que tenemos instalado EasyPHP lo ejecutamos y escribimos en nuestro navegador la siguiente ruta: **http://localhost/home/index.php** Lo siguiente será pulsar sobre el botón "open" en "modules", nos aparecerá algo como esto:

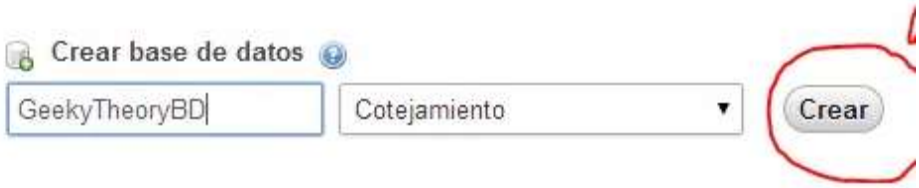


Fuente: Propia

Para crear una base de datos debemos pulsar sobre el botón de "**Nueva**":

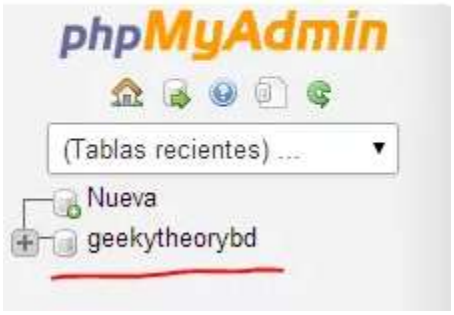


Escribimos el Nombre de la base de datos y le damos a crear:



Fuente: Propia

Podemos ver que ahora nos aparece una nueva base de datos en nuestro directorio de **PHPMyAdmin**:



Fuente: Propia

Es de gran importancia tener presente los tipos de datos que se manejan en una base de datos para esto mencionaremos lo siguiente: Según lo denotado en: http://www.desarrolloweb.com/articulos/1054.php

Para crear una base de datos existe otra forma adicional de crearla y es de comandos con sintaxis SQL, como veremos a continuación:

3.3.2 EJERCICIO DE APRENDIZAJE 4. CREAR UNA BASE DE DATOS

El primer paso es ingresar a Mysql haciendo clic en **inicio y se digita cmd**, luego de esto se digita el siguiente comando: **mysql-uroot-padmin**, si es la contraseña digitada por el usuario y se presiona enter y en ese momento ya estamos dentro del motor de bases de datos.

Es importante tener en cuenta que si está utilizando wampserver nos debemos asegurar que los servicios estén activos y arriba, para esto se sigue la siguiente ruta:

En la consola D.O.S. se digita **cd.**.y entrer, luego **cd..** y enter, en ese momento se está en la raíz del disco duro.

C:/cd wamp/ bin/mysql/mysql5.1.30/bin
Luego se digita: **mysql –uroot y enter,** allí ya e está en la consola de mysql.

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos:

1. **Tipos Numéricos**: Existen tipos de datos numéricos, que se pueden dividir en dos grandes grupos, los que están en coma flotante (con decimales) y los que no.

   TinyInt: es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255

   Bit ó Bool: un número entero que puede ser 0 ó 1

   SmallInt: número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.

   MediumInt: número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a16777215.

   Integer, Int: número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295

   BigInt: número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.

   Float: número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

   xReal, Double: número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308

   Decimal, Dec, Numeric: Número en coma flotante desempaquetado. El número se almacena como una cadena.

2. **Tipos Fecha**: A la hora de almacenar fechas, hay que tener en cuenta que Mysql no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes está comprendido entre 0 y 12 y que el día esta comprendido entre 0 y 31.

   Date: tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia

   DateTime: Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-dia horas:minutos:segundos

   TimeStamp: Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037.

   Time: almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'

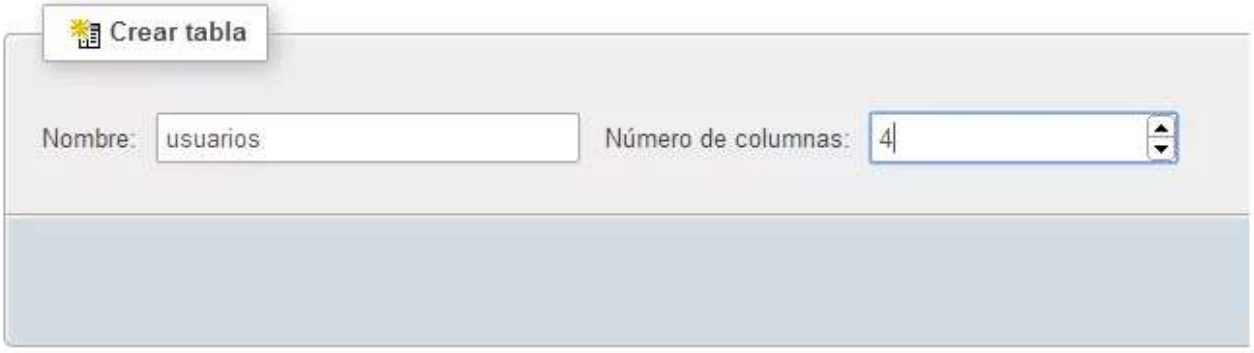
   Year: almacena un año. El rango de valores permitidos va desde el año 1901 al año 2155. El campo puede tener tamaño dos o tamaño 4 dependiendo de si queremos almacenar el año con dos o cuatro dígitos.

3. Tipos Cadena: Char(n): almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

   VarChar(n): almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

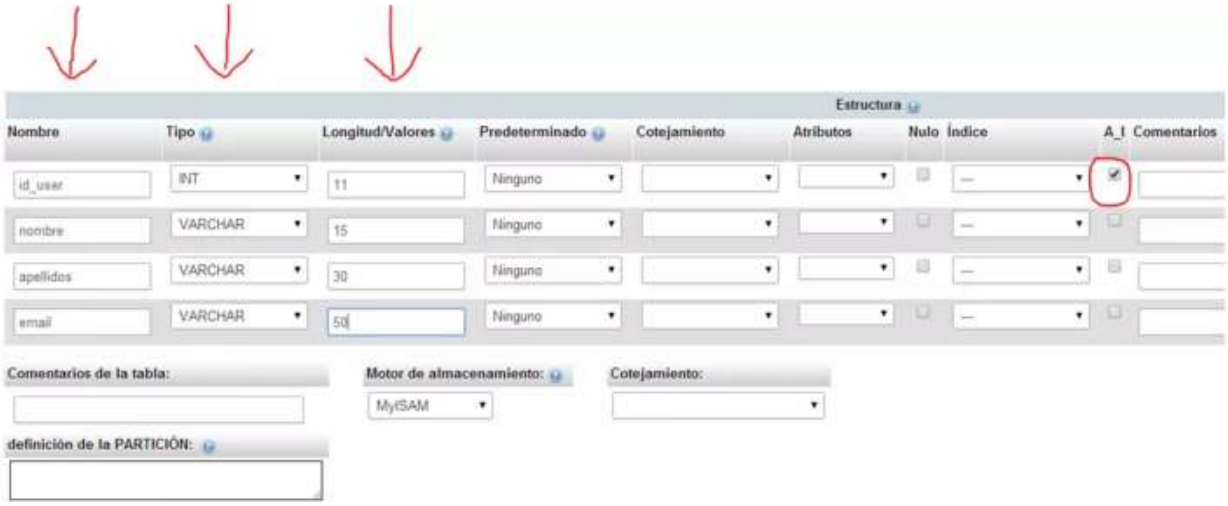
3.3.3 CREACIÓN DE TABLAS.

Pulsamos sobre nuestra base de datos y nos aparecerá un formulario para crear una tabla en nuestra base de datos. Siguiendo el ejemplo de nuestro tutorial vamos a añadir a nuestra base de datos una tabla llamada “usuarios” que va a tener 4 columnas:



Fuente: Propia

El siguiente paso será añadir las columnas en la base de datos, en nuestro ejemplo, vamos a crear un campo llamado "**id_user**" el cual será de tipo entero de**longitud 11** y **autoincrementable**. Seguidamente crearemos tres campos llamados**nombre, apellidos y email** consecutivamente con longitudes adecuadas, el tipo de estos tres campos será "**VARCHAR**". Se nos tiene que quedar algo como esto:



Fuente: Propia

Para dar más claridad del tema puede revisar los siguientes enlaces.

Crear una base de datos en mysql con phpmyadmin de wampserver Enlace

Enlace: https://geekytheory.com/java-php-mysql-ii-creacion-de-una-base-de-datos-en-mysql/

En modo consola la creación de una base de datos se realiza con la siguiente sintaxis:

mysql> create database banco;

Es decir se utiliza la sintaxis créate database y el nombre de la base de datos en este caso se denominó banco.

Para mostrar una base de datos se utiliza la siguiente expresión.

mysql> show database;

Luego de crear la base de datos se procede a la creación de tablas, para esto se usa la siguiente sintaxis. Siguiendo con el ejemplo de la base de datos banco seria:

mysql> create table empleado (cedula char(10) not null, nombre varchar(20), apellido varchar(20), fechanacimiento date, edad int(2));

Lo anterior se describe asi:

Nombre de la tabla: Empleado.

Campos de la tabla: Cedula, nombre, apellido, fechanacimento, edad.

Tipos de campo y tamaño: char(10), varchar(20), varchar(20), date, int(2).

3.3.4 RELACIÓN DE TABLAS.

Clave primaria: Primary key

Es una columna o un conjunto de columnas que identifican unívocamente a cada fila. Debe ser única, no nula y obligatoria. Como máximo, podemos definir una clave primaria por tabla. Esta clave se puede referenciar por una columna o columnas. Cuando se crea una clave primaria, automáticamente se crea un índice que facilita el acceso a la tabla.

Formato de restricción de columna:

CREATE TABLE NOMBRE_TABLA

(COL1 TIPO_DATO [CONSTRAINT NOMBRE_RESTRICCION] PRIMARY KEY

COL2TIPO_DATO
)[TABLESPACE ESPACIO_DE_TABLA];

Formato de restricción de tabla:

CREATE TABLE NOMBER_TABLA

(COL1 TIPO_DATO, COL2 TIPO_DATO,

[CONSTRAINT NOMBRERESTRICCION] PRIMARY KEY (COLUMNA [,COLUMNA]),

)[TABLESPACE ESPACIO_DE_TABLA];

Claves ajenas: Foreign Key:

Está formada por una o varias columnas que están asociadas a una clave primaria de otra o de la misma tabla. Se pueden definir tantas claves ajenas como se precise, y pueden estar o no en la misma tabla que la clave primaria. El valor de la columna o columnas que son claves ajenas debe ser: NULL o igual a un valor de la clave referenciada (regla de integridad referencial).

Formato de restricción de columna:

```
CREATE TABLE NOMBRE_TABLA
(COLUMNA1 TIPO_DATO
[CONSTRAINT NOMBRERESTRICCION]
REFERENCES NOMBRETABLA [(COLUMNA)] [ON DELETE CASCADE]
      )[TABLESPACE ESPECIO_DE_TABLA];
```

Formato de restricción de tabla:

```
   CREATE TABLE NOMBRE_TABLA
(COLUMNA1 TIPO_DATO,
COLUMNA2 TIPO_DATO,
…
[CONTRAINT NOMBRERESTRICCION]
FOREIGN KEY (COLUMNA [,COLUMNA])
  REFERENCES NOMBRETABLA [(COLUMNA [,
  COLUMNA])]
    [ON DELETE CASCADE],
)[TABLESPACE ESPACIO_DE_TABLA];
```

En resumen se establece así:

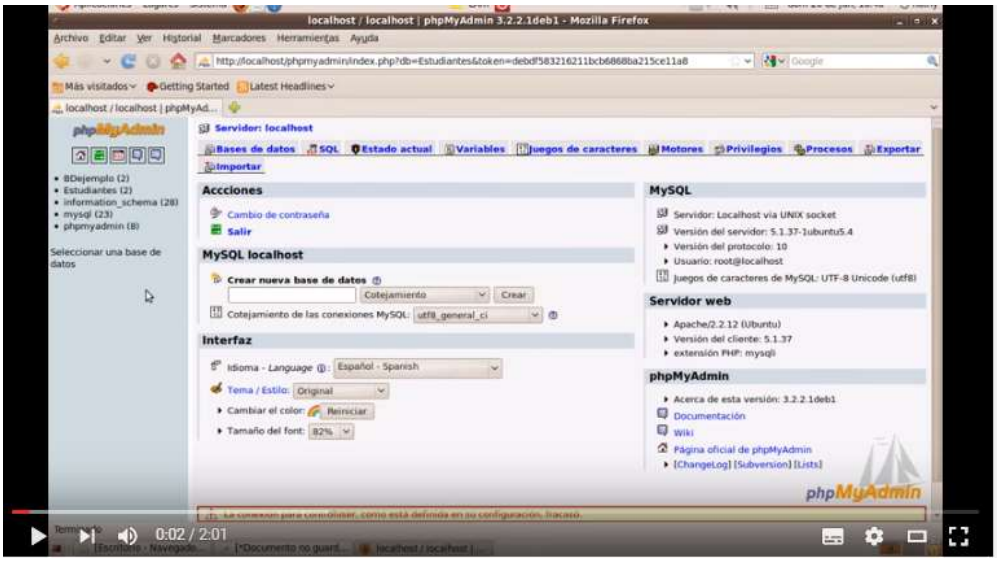
Restricciones de tabla

Restricción	Significado
primary key	Define la o las columnas que servirán como clave primaria. Las columnas que forman parte de la clave primaria deben de ser **not null**.
unique	Define las columnas en las que no pueden duplicarse valores. Serán las claves candidatas del modelo relacional.
foreign key (*columna*) **references** ***tabla*** (*columna2*)	Define que los valores de *columna* se permitirán sólo si existen en *tabla(columna2)*. Es decir, *columna* hace referencia a los registros de *tabla*, esto asegura que no se realicen referencias a registros que no existen.

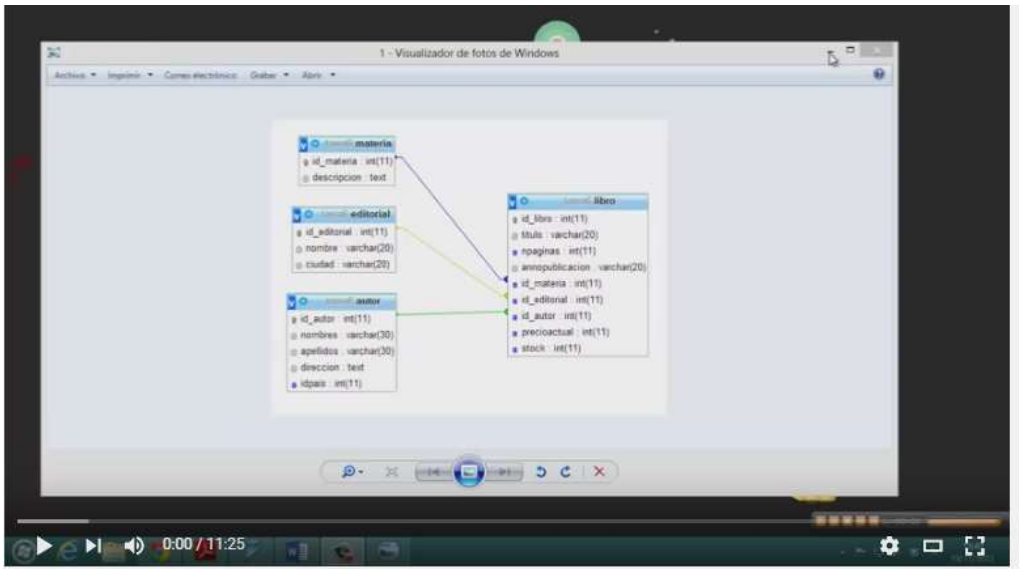
Índice: es **una estructura de datos que mejora la velocidad de las operaciones**, por medio de identificador único de cada fila de una tabla, permitiendo un rápido acceso a los registros de una tabla en una base de datos. Al aumentar drásticamente la velocidad de acceso, se suelen usar, sobre aquellos 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.

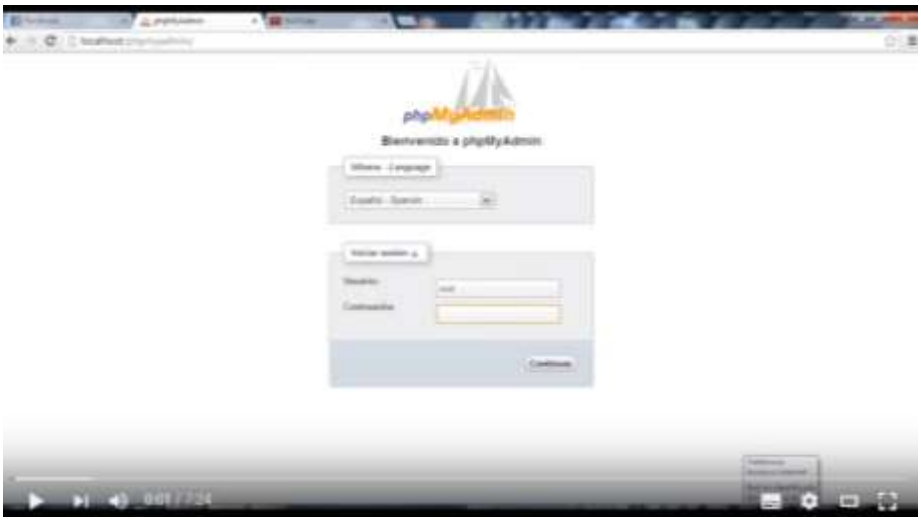
Para tener mayor claridad del tema se sugiérelos siguientes links.



Asignación de claves foráneas en MySQL Enlace



Relación de Tablas en MySQL Enlace



Relacionar dos tablas en MySQL Enlace

Retomando el ejemplo de la base de datos banco, para crear la clave primaria se utiliza la siguiente sintaxis:

mysql> create table **empleado** (cedula char(10) not null **primary key**, nombre varchar(20), apellido varchar(20), fechanacimiento date, edad int(2));

El campo que será clave primaria no debe tener tipos de dato null, por ende debe ser not null.

Modificación, Eliminación, Adición y cabios en las estructura de una tabla: Las tablas en una base de datos pueden estar sujetas a cambios como lo son la **modificación** un ejemplo de esto es agregar un campo a la tabla, eliminar un **campo a la tabla**, cambiar el nombre de la tabla, elimina llave primaria.

Se sugiere el siguiente link para dar claridad al tema:

phpMyAdmin: Creando tablas y campos Enlace

Enlace:http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf

Las tablas de una base de datos se componen de columnas con unos atributos definidos de la siguiente manera:

Atributos de columna

Atributo	Significado
null	Se permiten valores nulos, atributo por omisión si no se especifica lo contrario.
not null	No se permiten valores nulos.
default valor	Valor por omisión que se asigna a la columna.
auto_increment	El valor se asigna automáticamente incrementando en uno el máximo valor registrado hasta ahora. Se aplica sólo a las columnas marcadas como clave primaria.
primary key	Señala al campo como clave primaria, implícitamente también lo declara como **not null**.

3.3.5 EJERCICIO DE APRENDIZAJE 5. ADICIONAR UN CAMPO AL FINAL DE TABLA.

Para adicionar un campo al final de la tabla se utiliza la siguiente instrucción:

mysql> alter table banco add **cantidad_suc** int(3) not null; Es decir, a la base de datos banco se le agrega el campo cantidad_suc con tamaño 3 de tipo entero.

Para cambiar el nombre de una tabla de utiliza la siguiente instrucción:

mysql> alter table banco **rename** to cargo;

Para adicionar un campo al inicio de la tabla se utiliza la siguiente instrucción:

mysql> alter table cargo add **código_c** varchar(10) not null first;

Para adicionar un campo en cualquier lugar de la tabla se utiliza la siguiente instrucción:

mysql> alter table empleado add fecha_ingreso date not null after fechanacimiento;

Para eliminar un campo:

mysql> alter table empleado **drop** fechanacimiento;

Para eliminar una tabla:

mysql> **drop table** empledo;

Se sugiere visitar el siguiente enlace para ampliar el tema:

Enlace: http://moodle2.unid.edu.mx/dts_cursos_mdl/lic/TI/FB/AM/10/Modificacion_de_bd.pdf

3.3.6 TALLER DE ENTRENAMIENTO CREACION DE TABLAS Y ESTRUCTURAS

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente la situación propuesta y dar respuesta a ella.

1. Crear una base de datos denominada empresa.
    a. Crear una tabla llamada trabajadores con los campos código (int), nombre(char), dirección(varchar), edad(date).
    b. Crear una tabla llamada cargo con los campos nombre(char)
    c. Crear una tabla llamada nomina con los campos s_basico(int), deduciones (varchar), s_neto(int).

Realizar:

1. Agregar a la tabla cargo un campo del código del cargo.
2. Renombrar la tabla trabajadores por empleado.
3. Adicione a la tabla empleado el campo número de hijos antes del campo edad.

4. Eliminar la tabla nómina.

3.4 TEMA 3 INSTRUCCIONES DE MANIPULACIÓN DE DATOS.

Las instrucciones de **manipulación** de datos permiten lleva a cabio las **tareas de gestión** de la base de datos, es decir, eliminar datos, insertar, actualizar, importar **información** entre otras.

Para esto se hace necesario el uso de operadores lógicos, relacionales, aritméticos y especiales. En Mysql los más utilizados son:

Operaciones Relacionales:

=	Igual	**Is null / is not null** (si un valor es nulo o no).
<>	distinto	
>	Mayor que	**between:** "entre" si agregamos **not** antes de **between**, el valor se invierte.
<	Menor que	
>=	Mayor o igual que	**In:** Permite averiguar si el valor de un campo dado, está incluido en la lista de valores especificado.
<=	Menor o igual que	

Fuente: Propia

Operadores Lógicos:

and	Significa y
or	Significa y/o
xor	Significa o
not	Significa no (invierte el resultado)
()	parentesis

Fuente: Propia

Los operadores lógicos se utilizan para combinar condiciones.

Operadores Aritméticos:

Select: nos permite consultar los datos almacenados en una tabla de la base de datos.

Where: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admiten los operadores lógicos **AND** y **OR**.

+	Suma o adiciona
-	Resta o disminuye
*	Multiplica
/	Divide

Fuente: Propia

Operadores especiales:

Like	También es considerado un comando y se puede combinar con % para mayor precisión.
in	Es considerado un comando permite búsqueda de datos dispersos.
between	Se utiliza para buscar datos entre rangos.
having	Se utiliza para agrupar las búsquedas hechas con algunos de los anteriores como between

Fuente: Propia

Instrucción insert: La instrucción INSERT permite crear o insertar nuevos registros en una tabla, veamos su sintaxis con un ejemplo práctico, la inserción de un registro en la tabla ALUMNOS:

Un ejemplo es: Insert into ALUMNOS (ID_ALUMNO, NOMBRE, APELLIDOS, F_NACIMIENTO)
values (1 , 'Pablo' , 'Hernandaz Mata' , '1995-03-14')

Es decir, insertar en la tabla alumos cuyos campos son ID_ALUMNO, NOMBRE, APELLIDOS, F_NACIMIENTO los valores así: 1 , 'Pablo' , 'Hernandaz Mata' , '1995-03-14.

Instrucción Select: La instrucción SELECT se utiliza para obtener los registros de una tabla.

Su estructura es: SELECT * FROM categoría; esto significa que muestre lo que se encuentra en la tabla categoría.

Instrucción Update: La instrucción UPDATE permite actualizar registros de una tabla. Debemos por lo tanto indicar que registros se quiere actualizar mediante la cláusula WHERE, y que campos mediante la cláusula SET, además se deberá indicar que nuevo dato va a guardar cada campo.

Así por ejemplo supongamos que para el curso que carecía de profesor finalmente ya se ha decidido quien lo va a impartir, la sintaxis que permite actualizar el profesor que va a impartir un curso sería la siguiente:

update CURSOS set ID_PROFE = 2 where ID_CURSO = 5

Instrucción delete: La instrucción DELETE permite eliminar registros de una tabla, su sintaxis es simple, puesto que solo debemos indicar que registros deseamos eliminar mediante la cláusula WHERE. La siguiente consulta elimina todos los registros de la tabla mascotas que están de baja:

delete from MACOTAS where ESTADO = 'B'

Instrucción truncate: Quita todas las filas de una tabla sin registrar las eliminaciones individuales de filas. TRUNCATE TABLE es similar a la instrucción DELETE sin una cláusula WHERE; no obstante, TRUNCATE TABLE es más rápida y utiliza menos recursos de registros de transacciones y de sistema.

```
TRUNCATE TABLE
    [ { database_name .[ schema_name ] . | schema_name . } ]
    table_name
[ ; ]
```

Instrucción Limit: La cláusula LIMIT puede ser usada para acotar el número de filas devueltas por la sentencia SELECT. Toma uno o dos argumentos numéricos, que deben ser números enteros constantes no negativos.
Veamos un ejemplo de muestra para la sentencia SELECT LIMIT.

```
mysql> select * from student limit 2,5;
+--------+---------+-------+-----------------+---------+
| studid | name    | marks | address         | phone   |
+--------+---------+-------+-----------------+---------+
|      3 | michael |    75 | edinburgh       | 2598234 |
|      4 | jack    |    82 | victoria street | 2436821 |
|      5 | anne    |   100 | downing street  | 2634821 |
|      6 | steve   |    75 | downing street  | 2874698 |
|      7 | anne    |    80 | edinburgh       | 2569843 |
+--------+---------+-------+-----------------+---------+
5 rows in set (0.00 sec)
```

Fuente: http://deletesql.com/viewtopic.php?f=5&t=24

Aquí el primer argumento del ejemplo especifica el ajuste de la primera fila a devolver, y el segundo especifica el máximo número de filas a devolver. Por tanto devuelve las filas de la 3 a la 7 de la tabla student.

Backups y recuperación de información: Para realizar esta acción se utiliza el comando **dump.**

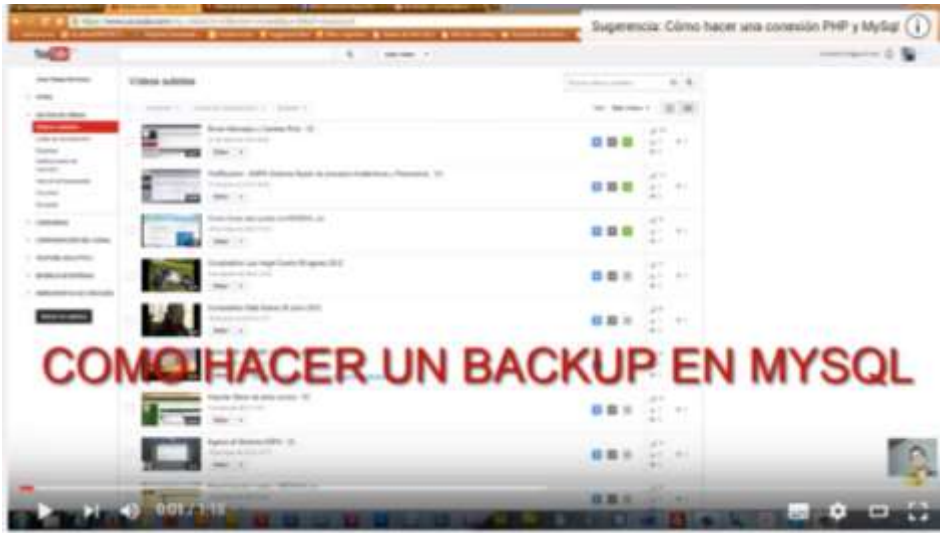
Un ejemplo de esto es el siguiente: retomando el ejemplo de la base de datos banco.

C:/wamp/bin/mysql/mysql5.1.30/bin>mysqldump –uroot –routines banco> k:/banco.sql

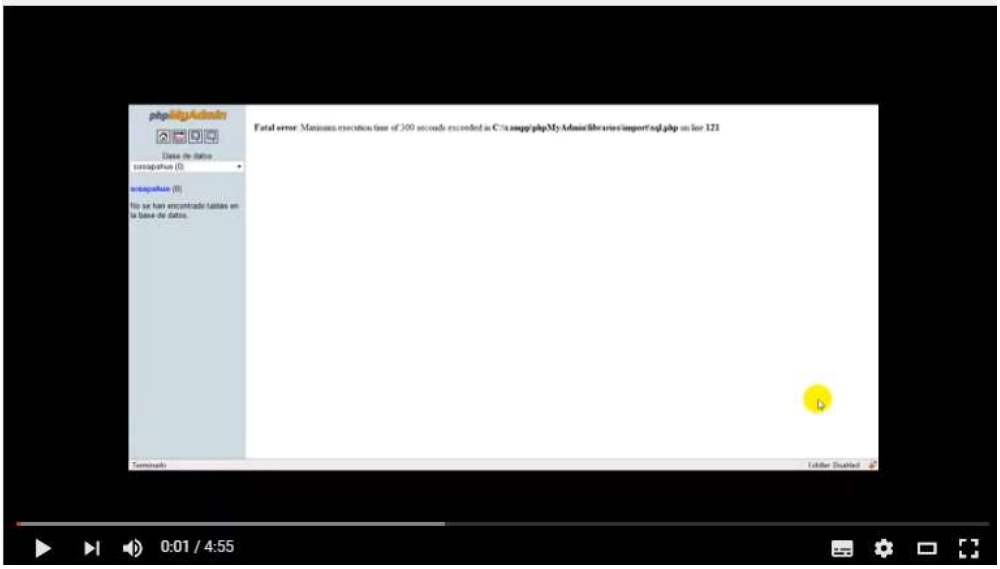
En los siguientes link encuentra ejemplos y videos de este tema para mayor claridad.

Enlace: http://deletesql.com/viewtopic.php?f=5&t=24

https://www.hscripts.com/es/tutoriales/mysql/selecto-limite.php



Como hacer un Backup o copia de seguridad en MySql Enlace



Restaurar base de datos mysql Enlace

3.4.1 TALLER DE ENTRENAMIENTO MANIPULACION DE DATOS

El siguiente taller de entrenamiento se propone para validar la aprehensión de los conceptos. Se debe leer detenidamente la pregunta propuestas y dar respuesta a ellas.

1. Construya una instrucción de inserción en la tabla CURSOS para guardar un nuevo curso de pintura y asígnele una clave que no entre en conflicto con la existentes, posteriormente construya la instrucción para eliminar de la tabla el registro que acaba de crear.

3.5 TEMA 4 INSTRUCCIONES PARA LA RECUPERACIÓN DATOS.

En esta unidad se desarrollaran instrucciones para llevar a cabo diferentes recuperaciones de datos es decir, consultas simples y complejas.

Función (Count, max, min, sum, avg, ect)

Count: Es utilizada para realizar operaciones de conteo. Veamos un ejemplo con una tabla denominad libros y esta contiene mucha información y se hace necesario saber la cantidad existente.

La sintaxis seria asi: Select count(*) from libros;

Esta función también se puede usar con la cláusula WHERE así:

Select count(*) from libros where editorial=planeta;

Max y Min: Para averiguar el valor máximo o mínimo de un campo usamos las funciones "max()" y "min()" respectivamente. Ejemplo, queremos saber cuál es el mayor precio de todos los libros:

Select max(precio) from libros;

Queremos saber cuál es el valor mínimo de los libros de "Rowling":

Select min(precio) from libros where autor like `% Rowling%`;

Sum: Retorna la suma de los valores que contiene el campo especificado. Por ejemplo, queremos saber la cantidad de libros que tenemos disponibles para la venta:

Select sum(cantidad) from libros;

También podemos combinarla con "where". Por ejemplo, queremos saber cuántos libros tenemos de la editorial "Planeta":

select sum(cantidad) from libros where editorial ='Planeta';

Avg: La función avg() retorna el valor promedio de los valores del campo especificado. Por ejemplo, queremos saber el promedio del precio de los libros referentes a "PHP":

select avg(precio) from libros where titulo like `%PHP%;

Estas funciones se denominan "funciones de agrupamiento" porque operan sobre conjuntos de registros, no con datos individuales.

Enlace: http://jjmojicamysql.blogspot.com.co/2012/04/34-funciones-de-agrupamiento-count-max.html

Las consultas en las bases de datos son la razón de ser de esta herramienta, dado que es la forma para la toma de decisiones de las personas que las usan, es por esto que para realízalas se debe tener en cuentas las siguientes instrucciones:

Instrucción Between: Between es una instrucción usada en consulta, con el fin de buscar entre rangos. Esta puede reemplazar en algunos aspecto el utilizar el mayor que (>=) y menor que (<=), esta instrucción se combina con where para llevar a cabo la consulta.

mysql> select * from producto where cod_producto between 350 and 600;

Esta consulta arrojará como resultado los registros de los productos que su código, esté entre 350 y 600.

In, esta instrucción sirve para llevar a cabo búsquedas con mucha más precisión, también se combina con la cláusula where.

In permite especificar valores y criterios dentro de la cláusula where, es utilizado básicamente para buscar registros que cumplan con más de un criterio, el operador in puede ser negado, para llevar a cabo la búsqueda de valores que por el contrario no cumplan con las especificaciones o criterios dados, para negar se utiliza not in.

mysql> select * from producto where cod_producto in (500,800);

En el ejemplo anterior, si se cambia el in por not in, el sistema mostrará en pantalla todos los productos excepto los que los que su código es 500 y 800.

mysql> select * from producto where cod_producto in (500,800);

Instrucción Distinct: tiene como función mostrar los registros que no están repetidos, filtrando el contenido de la tabla para sí mostrar únicamente los valores distintos. La instrucción distinct, debe ir combinada con el comando select.
Ejemplo:

Primero se ingresaran 3 nuevos registros, con nombres que ya existan en la tabla como: nevera, lavadora, teclado.

mysql> insert into product (cod_producto, nombre) values (210, 'nevera');

mysql> insert into product (cod_producto, nombre) values (310, 'lavadora');

mysql> insert into product (cod_producto, nombre) values (410, 'teclado');

La Instrucción distinct, se usa de la siguiente forma:

mysql> select distinct nombre from product;

Alias para las columnas y las tablas

Los **alias** con muy convenientes especialmente para dar un **nombre** efímero a una columna que se forma después de hacer una **consulta** o incluso para una tabla.

Para usarlo en una tabla se usa la siguiente instrucción:

mysql> select * from producto as mi_alias;

Recordemos que el alias **no le quita** ni le cambia el **nombre permanente** a la tabla, esto es temporal.

Para usar el alias en una columna o una consulta, se hace de la siguiente forma:

mysql> select nombre 'mi_alias' from producto;

En este ejemplo se está dando un nombre efímero o alias a la columna nombre de la tabla producto, esta acción a su vez **es una operación de consulta.**

mysql> select nombre 'mialias' from producto where cod_producto between 200 and 600;

Group by : El comando group by este nos permite agrupar registros de que tienen ciertas características y aunque se parece a un where con condiciones, este es muy diferente, group by debe combinarse con el comando select.

Como ejemplo se hará el ejercicio de saber la cantidad de visitantes que tenemos de cada ciudad, esto se podría hacer utilizando la cláusula where pero se tendría que hacer una consulta por cada ciudad, si se utiliza el group by entonces todo ese proceso puede hacerse con una sola consulta.

mysql> select ciudad, count(*) from visitantes group by ciudad;

Order By: La cláusula order by se combina con el comando select, para poder llevar a cabo el ordenamiento de una consulta, ya sea ascendente o descendente haciendo referencia a una columna especifica.

En este caso se procederá a ordenar la tabla visitantes tomando como referencia la columna nombre.

mysql> select * from visitantes order by nombre;

Si se desea ordenar en forma ascendente se escribe la sigla asc, después de la columna que se está tomando como referencia.

mysql> select * from visitantes order by nombre asc;

Por el contrario si se desea ordenar de forma descendente, se debe digitar la sigla desc, después de la columna que se está tomando como referencia.

mysql> select * from visitantes order by nombre desc;

Una tabla puede ordenarse por varias de sus columnas, es decir que una se ordene de forma ascendente y la otra de forma descendente.

mysql> select * from visitante order by ciudad asc , nombre desc;

En este caso si encuentra varias personas que visitaron la misma ciudad, se ubicaran de forma descendente, y por el contrario las ciudades se ubicarán de forma ascendente.

Having: La cláusula Having permite **seleccionar un grupo de registros** individuales al igual que la cláusula where, esta función es complementaria a la función group by la instrucción having, siempre se ubica al final de la sentencia. Su característica principal están que permite que se haga comparación con funciones sum, min max, count y avg

Ejemplo:

Si se tiene una agrupación como la anterior, por ciudades

mysql> select ciudad, count(*) from visitantes group by ciudad;

Y queremos que muestre solo las que tienen más de 3 personas; esto no puede hacerse con una comparación tradicional, en este caso se debe hacer uso de la cláusula having de la siguiente forma:

mysql> select ciudad, count(*) from visitantes group by ciudad having count(*)>=4;

En caso contrario si se buscan las ciudades tienen menos de 3 visitantes:

mysql> select ciudad, count(*) from visitantes group by ciudad having count(*)<3;

Vistas: Las vistas son registros de consultas que se hacen con determinado propósito. Las vistas comparten el mismo espacio que las tablas, por esta **razón no debe haber tablas con el mismo nombre que las vistas y viceversa.** Para crear una vista se utiliza el comando create; para el ejemplo tendremos en cuenta la tabla visitantes.

mysql> create view vista1 as select ciudad, count(*) from visitantes group by ciudad having count(*) between 2 and 3;

vista1 es el nombre de la vista y de select en adelante esta la consulta. Para invocar la vista, solo es necesario digitar:

mysql> select * from vista1;

Para llevar a cabo la **modificación de las vistas** creadas es necesario utilizar la sentencia **alter view**, pero además de modificar las Vistas, estas también pueden ser eliminadas

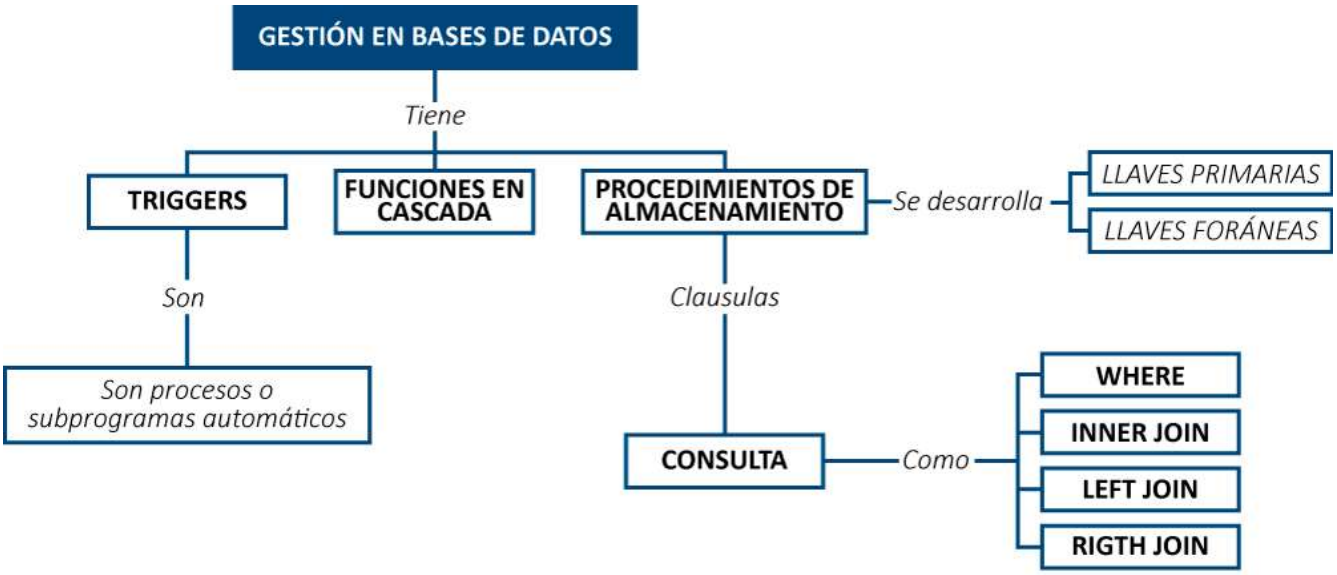
Este proceso se hace de la forma más sencilla:

mysql> drop view vista1;

Si se desea saber cuál fue la forma como está creada la vista estructuralmente, se hace una consulta de la siguiente manera:

mysql> show create view vista1;

4 UNIDA 3 GOBERNABILIDAD Y GESTION DE UN MOTOR DE BASES DE DATOS



Conceptos Básicos:

1. **Triggers:** Son procesos o sub programas automáticos, esto quiere decir que ese proceso va ligado a una tabla o una acción, estos se crean de forma parecida a los procedimientos almacenados.

4.1 TEMA 1 SISTEMA DE GESTIÓN DE BASES DE DATOS MYSQL

4.1.1 FUNCIONES EN CASCADA, RELACIONES Y PROCEDIMIENTO DE ALMACENAMIENTO.

Las funciones en cascada, permiten llevar a cabo tareas de actualización de varias tablas que dependen de otra, para ello en el momento de crear las tablas, es necesario especificar de forma puntual, que su función está definida en cascada, para que de tal forma se puedan actualizar los datos o eliminarlos.

Las principales funciones que se trabajan en cascada son Eliminar y Actualizar

Las funciones en cascada requieren trabajar con el motor InnoDB, el cual proporciona mayor integridad en los datos.

Para trabajar las funciones en cascada, algo llamado **integridad referencial** y también deben ser utilizadas las **claves foráneas** que permiten la relación en los datos.

Para ello se creará una base de datos nueva que se llamará **inscripción**

Se construirán dos tablas llamadas

Carrera y alumno

Bd: inscripción

Tabla: Carrera

Codcarrera: char(10) not null primary key

Nombre: char(30) not null

Tabla: alumno

Carnet: char(10) not null primary key

Nombre: char(40) not null

CodCarrera: char(10) not null foreign key

Estas tablas se deben crear utilizando el motor innodb

Las tablas que llevan integridad referencial en cascada y que utilizan el motor InnoDB, se construyen de la siguiente forma:

mysql> create database inscripción;

Después de crear la base de datos se procede a activarla.

mysql> use inscripción;

El paso a seguir es crear las tablas:

mysql> create table carrera(codcarrera char(10) not null primary key, nombre char(30) not null) engine=innodb;

mysql> create table alumno(carnet char(10) not null primary key, nombre char(40) not null, codcarrera char(10) not null, foreign key(codcarrera) references carrera(codcarrera) on delete cascade on update cascade) engine=innodb;

En la creación de la tabla alumno, como puede verse, la clave foránea(foreign key) es la columna codcarrera esta está haciendo referencia(references) a la columna codcarrera de la tabla carrera, y la parte que está en rojo, indica que los registros de esta tabla se actualizarán(update) y borraran(delete) en cascada(on cascade)

Se procede a ingresar la información, quedando las tablas de la siguiente forma:

```
mysql> select * from alumno;
+--------+-----------+------------+
| carnet | nombre    | codcarrera |
+--------+-----------+------------+
| 100    | lizet     | 10         |
| 200    | pablo     | 10         |
| 300    | lina      | 20         |
| 400    | luz       | 20         |
| 500    | nicolas   | 30         |
| 600    | alejandra | 30         |
| 700    | andrea    | 02         |
| 800    | emilio    | 02         |
+--------+-----------+------------+
8 rows in set (0.00 sec)

mysql> select * from carrera;
+------------+-------------+
| codcarrera | nombre      |
+------------+-------------+
| 02         | veterinaria |
| 10         | sistemas    |
| 20         | medicina    |
| 30         | turismo     |
+------------+-------------+
4 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Función Actualizar (Update)

Como su nombre lo dice **la tarea de esta función es actualizar los registros en cascada** a partir de los resultados ingresados en la tabla padre en este caso es la tabla carrera.

mysql> update carrera set codcarrera='03' where codcarrera='30';

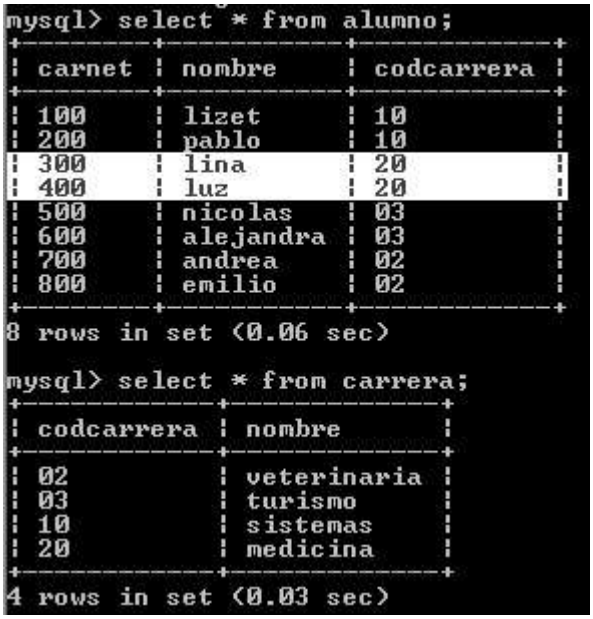
En esta instrucción se está cambiando el código de la carrera turismo (30) en la tabla padre (carrera), por el valor (03).

```
mysql> update carrera set codcarrera='03' where codcarrera='30';
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from alumno;
+--------+-----------+------------+
| carnet | nombre    | codcarrera |
+--------+-----------+------------+
| 100    | lizet     | 10         |
| 200    | pablo     | 10         |
| 300    | lina      | 20         |
| 400    | luz       | 20         |
| 500    | nicolas   | 03         |
| 600    | alejandra | 03         |
| 700    | andrea    | 02         |
| 800    | emilio    | 02         |
+--------+-----------+------------+
8 rows in set (0.00 sec)

mysql> select * from carrera;
+------------+-------------+
| codcarrera | nombre      |
+------------+-------------+
| 02         | veterinaria |
| 03         | turismo     |
| 10         | sistemas    |
| 20         | medicina    |
+------------+-------------+
4 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Función Eliminar (Delete)

La función delete en cascada le **permite al usuario llevar a cabo tareas de borrado de registros en las tablas hija**s si en la tablas padres se elimina la información principal, que alimenta las registros de las tablas hijas.

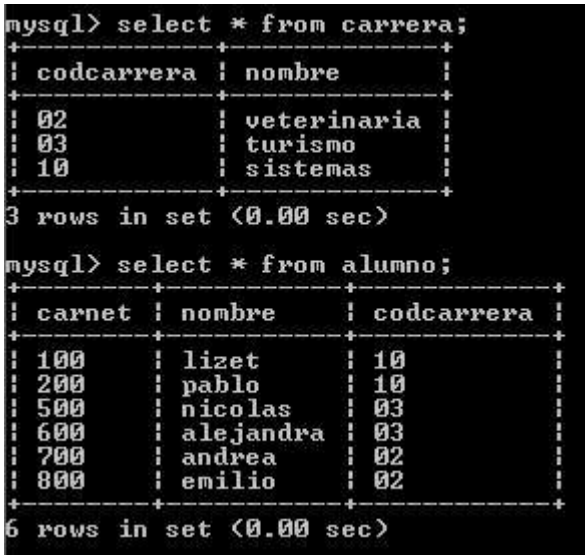


```
mysql> select * from alumno;
+--------+-----------+-------------+
| carnet | nombre    | codcarrera  |
+--------+-----------+-------------+
| 100    | lizet     | 10          |
| 200    | pablo     | 10          |
| 300    | lina      | 20          |
| 400    | luz       | 20          |
| 500    | nicolas   | 03          |
| 600    | alejandra | 03          |
| 700    | andrea    | 02          |
| 800    | emilio    | 02          |
+--------+-----------+-------------+
8 rows in set (0.06 sec)

mysql> select * from carrera;
+------------+-------------+
| codcarrera | nombre      |
+------------+-------------+
| 02         | veterinaria |
| 03         | turismo     |
| 10         | sistemas    |
| 20         | medicina    |
+------------+-------------+
4 rows in set (0.03 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Retomando el ejemplo anterior, se procederá a realizar el borrado en cascada, se eliminará la carrera identificada con el código '20', en este caso es medicina, y al eliminar la carrera medicina, automáticamente serán eliminados los alumnos que cursen esta carrera.

mysql> delete from carrera where codcarrera='20';



```
mysql> select * from carrera;
+------------+-------------+
| codcarrera | nombre      |
+------------+-------------+
| 02         | veterinaria |
| 03         | turismo     |
| 10         | sistemas    |
+------------+-------------+
3 rows in set (0.00 sec)

mysql> select * from alumno;
+--------+-----------+-------------+
| carnet | nombre    | codcarrera  |
+--------+-----------+-------------+
| 100    | lizet     | 10          |
| 200    | pablo     | 10          |
| 500    | nicolas   | 03          |
| 600    | alejandra | 03          |
| 700    | andrea    | 02          |
| 800    | emilio    | 02          |
+--------+-----------+-------------+
6 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

4.1.2 RELACIONES

Primero que todo para poder llevar a cabo la relaciones que puedan ser utilizadas con foreign key (clave foránea), se debe utilizar el motor InnoDB.

La relación está definida como una asociación establecida entre campos comunes, existen tres tipos de relaciones:

De uno a uno: "relaciona un único registro de la tabla principal con uno sólo de la tabla relacionada".

De uno a varios: "un único registro de la tabla principal se puede relacionar con varios de la tabla relacionada".

De vario a varios: "un registro de la tabla principal se relaciona con varios de la tabla relacionada y, además, un registro de la tabla relacionada se relaciona con varios de la tabla principal".

(Tomado de: http://www.adrformacion.com/curso/aplicacionesaccesxp/leccion2/RelacionesTablas.htm)

Para la relación entre tablas, es aconsejable nombrar de la misma forma los campos que se van a relacionar.

Para llevar a cabo la relación de las tablas, estas deben crearse antes con **integridad referencial** utilizando claves foráneas (foreign key)

left join: Un left join se usa para hacer coincidir **registros en una tabla** (izquierda) con otra (derecha).

En el caso en que un valor de la primera tabla, no encuentre coincidencia alguna en la segunda tabla (derecha), el sistema genera una fila adicional con todos los campos listos a null

(Tomado de: http://www.mysqlya.com.ar/temarios/descripcion.php?cod=58&punto=64)

La sintaxis es la siguiente: Tomando el ejemplo de la base de datos creada anteriormente llamada inscripción, y sus tablas alumno y carrera se digita lo siguiente:

mysql> select * from alumno left join carrera on carrera.codcarrera=alumno.codcarrera;

```
mysql> select * from alumno left join carrera on carrera.codcarrera=alumno.codcarrera;
+--------+-----------+-------------+-------------+--------------+
| carnet | nombre    | codcarrera  | codcarrera  | nombre       |
+--------+-----------+-------------+-------------+--------------+
| 100    | lizet     | 10          | 10          | sistemas     |
| 200    | pablo     | 10          | 10          | sistemas     |
| 500    | nicolas   | 03          | 03          | turismo      |
| 600    | alejandra | 03          | 03          | turismo      |
| 700    | andrea    | 02          | 02          | veterinaria  |
| 800    | emilio    | 02          | 02          | veterinaria  |
+--------+-----------+-------------+-------------+--------------+
6 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

La **cláusula where** también puede combinarse con el **left join** ello depende de la necesidad del usuario.

inner Join: El Inner join, igual que los demás join, sirven para hacer relaciones en tablas a la hora de hacer consultas.

Con el inner join, todos los registros que **no coinciden en las tablas** que Se van relacionar, son descartados, por esta razón el sistema simple mente arroja los que registros que coinciden, entregando una consulta más exacta.

Ejemplo:

Se creará una base de datos llamada arrendamientos

Tabla: inquilino

cedinq

nombre

mysql> create table inquilino(cedinq varchar(10) not null primary key, nombre char(40) not null);

Tabla: inmueble

codpro

dirección

valorarriendo

cedpro

mysql> create table inmueble(codpro varchar(10) not null, direccion char(50) not null, valorarriendo int not null, cedpro varchar(10) not null);

Tabla: contrato

codigo (auto_increment)

cedinq

fechacontrato

codpro

mysql> create table contrato(codigo int not null auto_increment primary key, ceding varchar(10) not null, fechacontrato date not null, codpro varchar(10) not null);

Tabla: propietario cedprop

nombre

mysql> create table propietario(cedpro varchar(10) not null primary key, nombre char(40) not null);

Luego se procede a ingresar la información, quedando las tablas de la siguiente forma:

```
mysql> select * from inquilino;
+---------+-------------------------+
| cedinq  | nombre                  |
+---------+-------------------------+
| 123444  | sandra betancur         |
| 234555  | maria fernanda vallejo  |
| 345666  | liliana valencia        |
| 456777  | eliana gonzalez         |
| 567888  | isabel garcia           |
| 678999  | ricardo bermudez        |
| 789000  | andres gomez            |
| 890111  | alberto perez           |
+---------+-------------------------+
8 rows in set (0.00 sec)

mysql> select * from propietario;
+----------+-----------------+
| cedprop  | nombre          |
+----------+-----------------+
| 111123   | gloria jimenez  |
| 222234   | rodrigo mena    |
| 333345   | guillermo rojas |
| 444456   | lina vallejo    |
| 555567   | maria lopez     |
+----------+-----------------+
5 rows in set (0.00 sec)

mysql> select * from inmueble;
+---------+--------------+-------------------+----------+
| codpro  | direccion    | valorararriendo   | cedprop  |
+---------+--------------+-------------------+----------+
| 100     | laureles     |            750000 | 444456   |
| 150     | poblado      |           1120000 | 444456   |
| 200     | centro       |            420000 | 111123   |
| 250     | bello        |            350000 | 111123   |
| 300     | buenos aires |            290000 | 222234   |
| 350     | floresta     |            425000 | 555567   |
| 400     | los colores  |            840000 | 555567   |
| 450     | laureles     |            680000 | 333345   |
| 500     | la milagrosa |            320000 | 222234   |
| 550     | envigado     |            556000 | 555567   |
| 600     | sabaneta     |            500000 | 555567   |
| 650     | itag         |            450000 | 444456   |
| 700     | envigado     |           1500000 | 111123   |
| 750     | poblado      |            935000 | 222234   |
+---------+--------------+-------------------+----------+
14 rows in set (0.00 sec)

mysql> select * from contrato;
+---------+---------+---------------+---------+---------+
| codigo  | cedinq  | fechacontrato | codpro  | tiempo  |
+---------+---------+---------------+---------+---------+
|       1 | 890111  | 1999-04-25    | 700     |      11 |
|       2 | 123444  | 2003-11-25    | 300     |       7 |
|       3 | 234555  | 2006-10-23    | 150     |       4 |
|       4 | 345666  | 1995-01-20    | 250     |      15 |
|       5 | 567888  | 2004-10-10    | 400     |       6 |
|       6 | 456777  | 2000-05-30    | 650     |      10 |
|       7 | 456777  | 2003-07-14    | 100     |       7 |
|       8 | 678999  | 2000-06-28    | 200     |      10 |
|       9 | 890111  | 1997-02-11    | 500     |      13 |
|      12 | 567888  | 2005-08-29    | 450     |       5 |
|      13 | 123444  | 2007-01-01    | 550     |       3 |
|      14 | 890111  | 2000-01-31    | 750     |      10 |
+---------+---------+---------------+---------+---------+
12 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Luego se procederá hacer una consulta en la cual se necesita saber cuántos inquilinos llevan más de 10 años.

mysql> select count(inquilino.nombre) from inquilino inner join contrato on contrato.cedinq=inquilino.cedinq where year(fechacontrato)<2000;

```
mysql> select count(inquilino.nombre) from inquilino inner join contrato
    -> on contrato.cediinq=inquilino.cediinq where year(fechacontrato)<2000;
+--------------------------+
| count(inquilino.nombre) |
+--------------------------+
|                        3 |
+--------------------------+
1 row in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Como puede verse, para poder realizar esta consulta, fue necesario utilizar el inner join, ya que este permite relacionar las tablas que se necesitaban para llevar a cabo la consulta y, además de eso, entrega la información de forma más precisa que el left join.

PISTAS DE APRENDIZAJE

Traer a la memoria:

Recuerde que: Tanto el inner join como el left join, el rigth join entre otros join, tienen como función permitir consultas mediante tablas relacionadas, pero todos arrojan resultados dependiendo de la necesidad del usuario, en pocas palabras: todos son para casos especiales.

Right Join: **El comando right join tiene una función similar a la de left join** con la única novedad que la búsqueda de coincidencias o de registros similares, la hace de forma contraria(inversa), en este caso busca valores coincidencia de valores desde la tabla de la derecha a la tabla de la izquierda.

Para establecer más claramente la diferencia entre left join y right join, se utilizará el mismo ejemplo que ha sido usado para left join:

mysql> select * from alumno right join carrera on carrera.codcarrera=alumno.codcarrera;

```
mysql> select * from alumno right join carrera on carrera.codcarrera=alumno.codcarrera;
+--------+----------+------------+------------+-------------+
| carnet | nombre   | codcarrera | codcarrera | nombre      |
+--------+----------+------------+------------+-------------+
| 700    | andrea   | 02         | 02         | veterinaria |
| 800    | emilio   | 02         | 02         | veterinaria |
| 500    | nicolas  | 03         | 03         | turismo     |
| 600    | alejandra| 03         | 03         | turismo     |
| 100    | lizet    | 10         | 10         | sistemas    |
| 200    | pablo    | 10         | 10         | sistemas    |
+--------+----------+------------+------------+-------------+
6 rows in set (0.06 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Como se puede observar, la función es la misma solo que el sistema busca de manera diferente entre los datos.
Natural Join: El natural join es utilizado para cuando los campos que cumplen la función de enlazar las tablas, tienen el mismo nombre.

Se utiliza de la siguiente forma:

mysql> select * from inquilino natural join contrato;

```
mysql> select * from inquilino natural join contrato;
+---------+-------------------------+--------+----------------+--------+---------+
| ceding  | nombre                  | codigo | fechacontrato  | codpro | tiempo  |
+---------+-------------------------+--------+----------------+--------+---------+
| 890111  | alberto perez           |      1 | 1999-04-25     | 700    |      11 |
| 123444  | sandra betancur         |      2 | 2003-11-25     | 300    |       7 |
| 234555  | maria fernanda vallejo  |      3 | 2006-10-23     | 150    |       4 |
| 345666  | liliana valencia        |      4 | 1995-01-20     | 250    |      15 |
| 567888  | isabel garcia           |      5 | 2004-10-10     | 400    |       6 |
| 456777  | eliana gonzalez         |      6 | 2000-05-30     | 650    |      10 |
| 456777  | eliana gonzalez         |      7 | 2003-07-14     | 100    |       7 |
| 678999  | ricardo bermudez        |      8 | 2000-06-28     | 200    |      10 |
| 890111  | alberto perez           |      9 | 1997-02-11     | 500    |      13 |
| 567888  | isabel garcia           |     12 | 2005-08-29     | 450    |       5 |
| 123444  | sandra betancur         |     13 | 2007-01-01     | 550    |       3 |
| 890111  | alberto perez           |     14 | 2000-01-31     | 750    |      10 |
+---------+-------------------------+--------+----------------+--------+---------+
12 rows in set (0.00 sec)
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

Como se puede observar el natural join hace una relación natural de las tablas, en este caso el campo de la coincidencia es cedinq.

Subconsultas: Las subconsultas en general consisten en hacer una consulta dentro de otra consulta para mostrar en pantalla una cantidad de información que no puede hacerse en una sola consulta**. La sub consulta** como tal no tiene sintaxis, esta **puede tener los select** que se deseen o se necesiten, en el lugar donde sean necesarias.

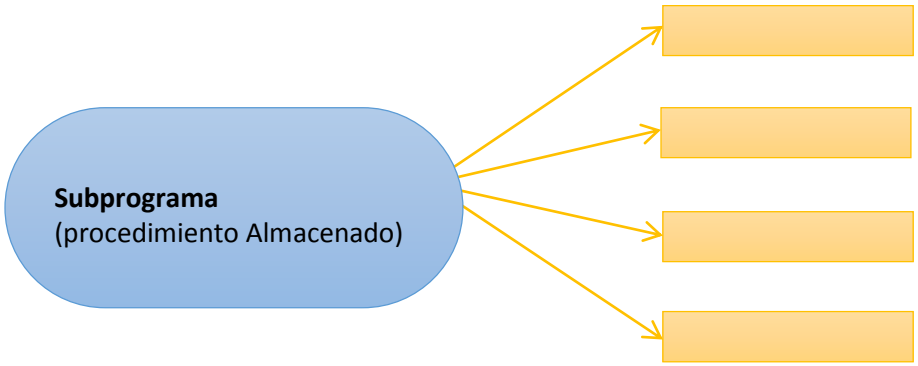
Se puede: **mysql> select (select) from tablas where (select);**

nota: se utiliza join solo cuando en el from van 2 tablas y se establece condición entre la tabla principal y las subconsultas.

Las subconsultas no son utilizadas únicamente para visualizar información también son utilizadas para actualizar, borrar e insertar.

mysql> update detallefactura set valor=(select valorunitario + (valorunitario * 0.23) from producto where detallefactura.codigo=producto.codigo);

Procedimientos almacenados: Los procedimientos almacenados son subprogramas, que permiten hacer ciertas labores especificadas pro el usuario o programador, en estos pueden ir sentencias que consten de un select, update, delete, insert, esto como ha sido nombrado anteriormente depende de la necesidad del usuario, de igual forma pueden ir subconsultas, las cuales constan de uno o varios select anidados.



Fuente: Propia

Los procedimientos almacenados tienen muchas ventajas algunas de estas son:

- Evita estar repitiendo instrucciones, gracias a que estas quedan grabadas, por lo que el usuario simplemente tiene que invocar al procedimiento almacenado, librándose a sí de estar escribiendo una consulta o instrucción muy larga.
- El procedimiento almacenado brinda mucha seguridad gracias a que evitará que el usuario se equivoque en la constante escritura de instrucciones, las cuales podrían arrojarle un resultado erróneo.
- Son muy eficientes para usuarios con poco conocimiento de SQL.
- Permite una gran facilidad de manejo, pues simplemente hay que invocar el procedimiento almacenado.

Condiciones

Generales: no tienen parámetros.

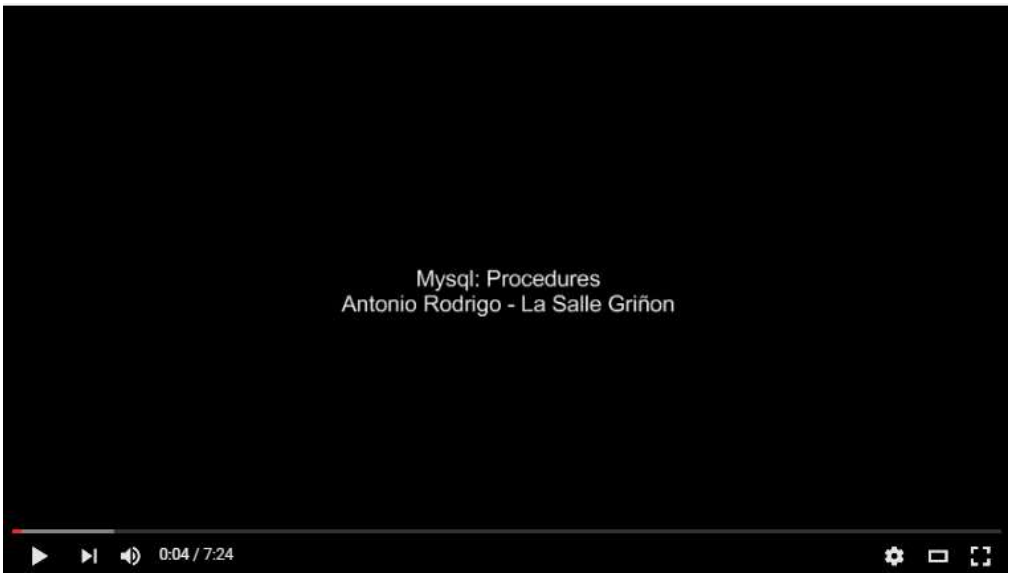
Específicos: Tienen parámetro o parámetros.

La instrucción delimiter anula temporalmente el ";", para que la instrucción no termine antes de tiempo.

Ejemplo:

```
mysql> delimiter  //

mysql> create procedure listar_producto()

-> begin

-> select * from producto order by articulo;

-> end

-> //

mysql> delimiter  ;
```

(Entre la palabra delimiter y el punto y coma (;), debe haber un espacio, de igual forma al principio entre la palabra delimiter y los dos signos de slash //).



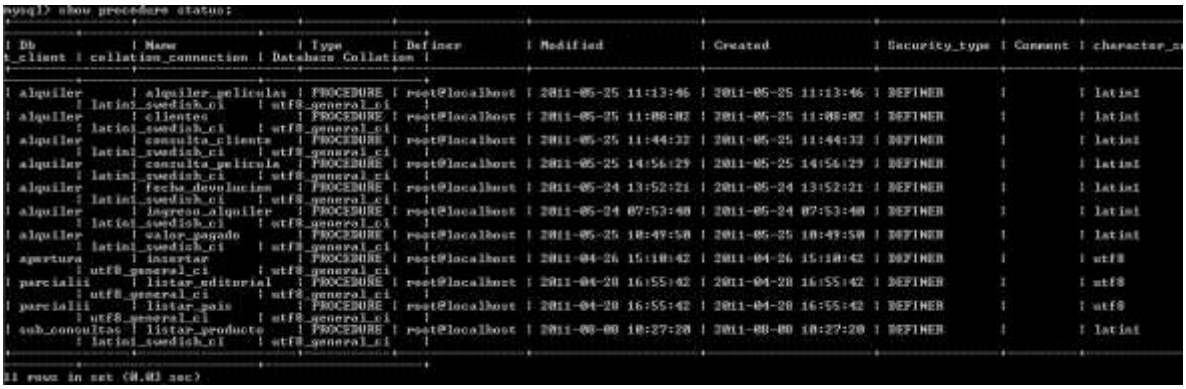
1Mysql: procedures o procedimientos almacenados - www.ofimaticaparatorpes.com Enlace

```
mysql> delimiter //
mysql> create procedure listar_producto()
    -> begin
    -> select * from producto order by articulo;
    -> end
    -> //
Query OK, 0 rows affected (0.05 sec)

mysql> delimiter ;
```

Fuente: Nelson Jovanny Isaza Morales. (2003).

mysql> show procedure status;

Esta sentencia se utiliza cuando se desea saber que procedimientos almacenados hay en la base de datos.

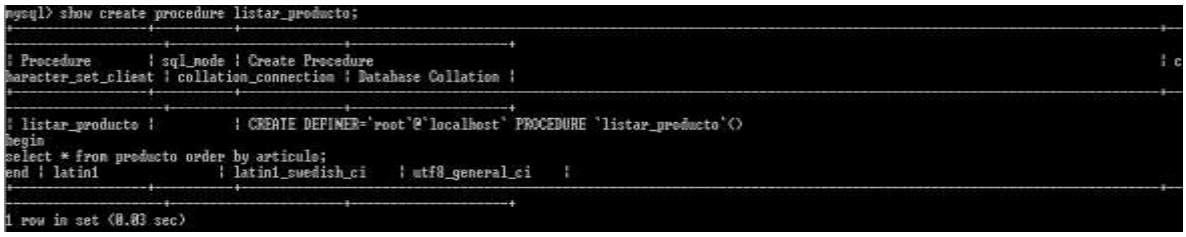


Fuente: Nelson Jovanny Isaza Morales. (2003).

Visualización de estructura de un procedimiento almacenado

mysql> show créate procedure listar_producto;

Esta instrucción se utiliza con el fin de saber cómo fue creado el procedimiento almacenado, con esta instrucción podremos visualizar la instrucción de consulta que tiene almacenado el procedimiento.

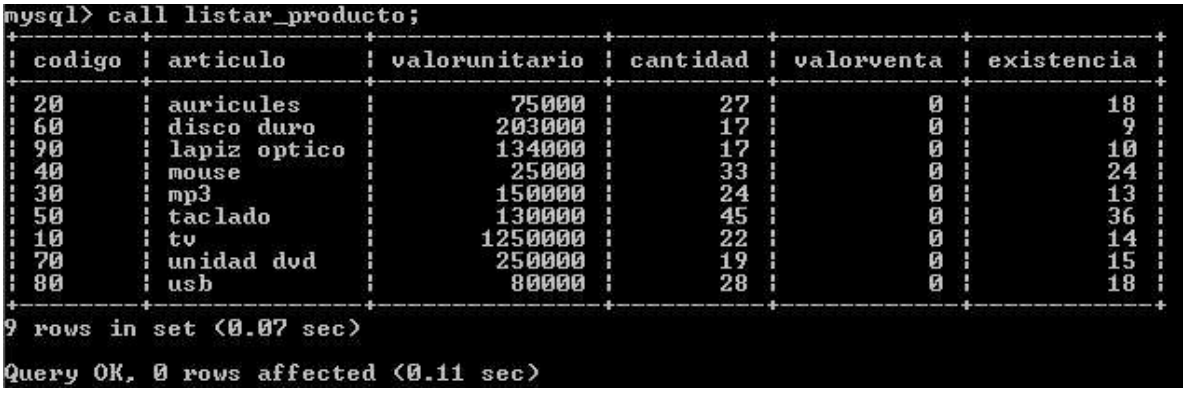


Fuente: Nelson Jovanny Isaza Morales. (2003).

Invocar el procedimiento almacenado que se ha creado:

mysql> call listar_producto;

De esta forma se llama un procedimiento almacenado que ha sido diseñado para mostrar información.



Fuente: Nelson Jovanny Isaza Morales. (2003).

Eliminar un procedimiento almacenado: **mysql> drop procedure listar_producto;**

Esta es la instrucción utilizada para llevar a cabo la eliminación de un procedimiento almacenado.

4.2 TEMA 2 TRIGGERS

Son procesos o sub programas automáticos, esto quiere decir que ese proceso va ligado a una tabla o una acción, estos se crean de forma parecida a los procedimientos almacenados.

Su propósito es **actualizar, eliminar o insertar datos,** que permitan actualizar varios registros de forma automática, por ello son llamados también disparadores.

Particularidad:

A diferencia de los procedimientos almacenados y los demás procesos vistos anteriormente, el trigger no permite el select, ya que el trigger es un proceso automático, por esta razón solo admite o tiene el propósito de insertar, actualizar y eliminar. El trigger no recibe parámetros

El trigger es **utilizado para actualizar cierta información de otras tablas**, como gracias a la acción de insertar información en alguna tabla principal o también en el momento de actualizar algún tipo de información en una tabla o un registro en particular, el trigger se aplica actualizando todos los registros que dependen de esta nueva información ingresada. Los triggers se construyen de la siguiente forma:

mysql> delimiter //

mysql> create trigger actualizar after insert on detallefactura

-> for each row

-> begin

-> update producto set existencia = (select sum(cantidad) from detallefactura where detallefactura.codigo=producto.codigo);

-> end

-> //

mysql> delimiter ;

Para visualizar todos los triggers que han sido utilizados en la base de datos que esta activada, ademas de una información detallada de cada trigger, se digita la siguiente instrucción:

mysql> select * from information_schema.triggers;

Este comando nos arroja información de cada trigger que compone la base de datos de una forma poco ordenada.

mysql> select * from information_schema.triggers\G;

Si adicionamos el símbolo backslash y la letra “G” mayúscula a la instrucción anterior, se podrá observar la información de cada trigger en la base de datos con mucho detalle y de forma más ordenada.

Para llevar a cabo el uso del trigger se aplica la siguiente instrucción:

mysql> insert into values('1010','10','3');

Para eliminar un trigger, es necesario utilizar la siguiente instrucción:

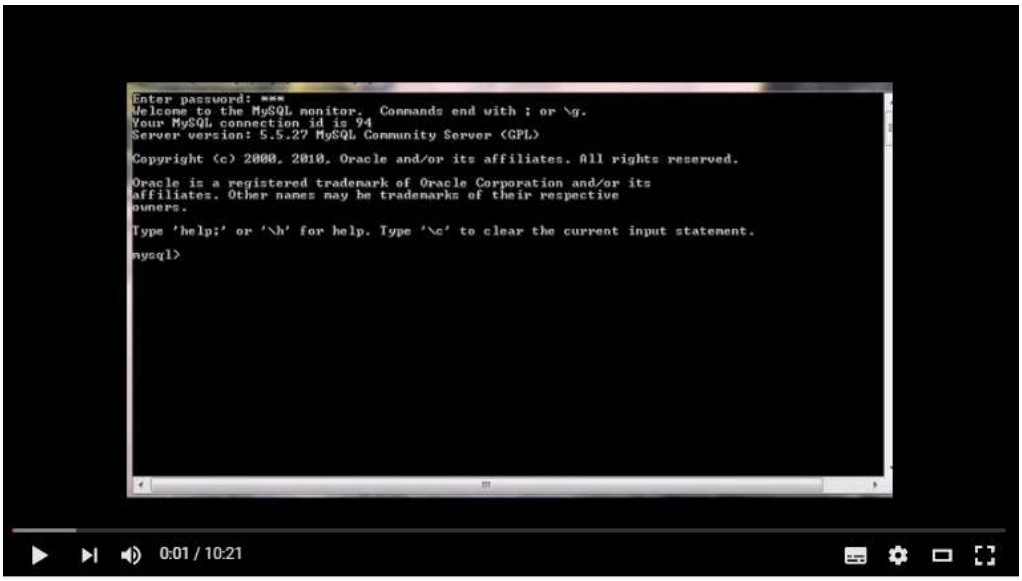
mysql> drop trigger actualizar;

En los siguientes link encontrara mayor explicación del tema y ejemplos:

Enlace: http://www.postgresql.org.es/node/244

http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=522

http://www.fpress.com/revista/num0206/art.htm



COMO HACER UN TRIGGER EN MYSQL Enlace

5 PISTAS DE APRENDIZAJE

Recuerda que

Para considerar una tabla como relación, hay que tener en cuenta que:

- Cada tabla debe tener un nombre único
- No pueden haber dos filas iguales, pues no están permitidos los duplicados
- Todos los datos de una columna deben ser del mismo tipo

Tener en cuenta

- Para poder trabajar con mysql, se debe tener instalado el servidor local **AppServ** o **WampServer**, existen más servidores pero estos son los más sencillos de utilizar, este servidor integra el paquete de mysql al momento de instalarlo entre otras herramientas que con el tiempo utilizaremos.

Tener en cuenta

- Es muy conveniente crear un modelo de entidad de relación para poder tener una clara expectativa o plan de cómo va a ser la estructura de nuestra base de datos. Especialmente cuando esta tiene muchas tablas que se relacionan y tablas que se comportan como relaciones.

Recordar que

- En la creación de tablas no es correcto nombrarlas de forma plural ejemplo: Nombre de la tabla empleado o ~~empleados~~, la forma correcta es empleado.

Recordar que

- Al cambiar el nombre de una tabla, el contenido en ella no cambia. Por otro lado pude versen afectados los procedimientos almacenados, ya que si estos hacen referencia a una tabla con otro nombre entonces podrían arrojar error al no encontrarla con el nombre que originalmente fueron creados para hacer referencia.

Recordar que

- A los campos tipo numérico no es necesario hacerles la reserva de memoria como el caso del tipo texto.
- Nombre de la tabla int(5) primary key… = nombre de la tabla int primary key …

No olvide que

- Al realizar las diferentes instrucciones se debe tener muy en cuenta las comas (,), el punto y coma al final (;), las **comillas sencillas** ('_ '), a la hora de insertar datos o crear alias, y no olvide poner mucho cuidado en la hora de escribir la sintaxis, para evitar errores.

Recuerde que

- En el momento de hacer copias de seguridad no es necesario poner punto y coma (;) al final de la instrucción.

No olvide que

- Cuando se refiere a una columna o una consulta, es necesario que el alias esté entre comillas sencillas y nunca comillas dobles.

Recuerde que

- Tanto el **inner join** como el **left join**, el **rigth join** entre otros **join**, tienen como función permitir consultas mediante tablas relacionadas, pero todos arrojan resultados dependiendo de la necesidad del usuario, en pocas palabras: todos son para casos especiales.

Recuerde que

- Por cada tabla solo pueden escribirse 3 triggers.

No olvide que

- Siempre portar una memoria usb en la cual tenga todos sus ejercicios

Recuerde que

- Para uso posterior de algunas sentencias, es bueno copiar las instrucciones debajo de cada pregunta en los diferentes ejercicios.

Recuerde

- Mantener copias de seguridad en la nube (internet) o en memorias usb, si es posible en ambos.

No olvide

- Poner una pequeña portada o sus datos personales en el correo en el cual entrega su trabajo de base de datos, en el cual debe ir las preguntas y como se resolvieron, y además de ello el archivo de base de datos.

6 GLOSARIO

Base de datos: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Base_de_datos)

Mysql: es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. 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. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/MySQL)

Modelos de Bases de Datos: una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Basesdedatos

Modelo Entidad - Relación: una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades. Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Basesdedatos

Normalización: proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener. Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Basesdedatos

Procedimiento Almacenado: es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Procedimiento_almacenado)

POO: (Programación Orientada a Objetos) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos)

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. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/SQL)

Trigger: procedimiento que se ejecuta cuando se cumple una condición establecida al realizar una operación. (Tomado de: Wikipedia - http://es.wikipedia.org/wiki/Trigger_%28base_de_datos%29)

7 BIBLIOGRAFÍA

e-ducativa.catedu. (2010). Clasificación de los SGBD. Febreo 5 de 2016, de e-ducativa.catedu Sitio web: http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/1000/1080/html/31_clasificacin_de_los_sgbd.html

Microsoft. (2010). Conceptos sobre el motor de base de datos. 5 de Febereo de 2016, de Microsoft Sitio web: https://technet.microsoft.com/es-es/library/ms187079(v=sql.105).aspx

Genbeta. (2008). Fundamento de las bases de datos: Modelo entidad-relación. 2 de Febrero de 2016, de Genbeta Sitio web:
http://www.vayatele.com/?utm_source=genbetadev&utm_medium=network&utm_campaign=favicons

Claudio Gutiérrez, Gonzalo Navarro. (2011). Modelo Entidad/Relación y conversión a Modelo Relacional. 10 de Enero de 2016, de Universidad de Chile Sitio web: http://users.dcc.uchile.cl/~mnmonsal/BD/guias/g-modeloER.pdf

Universidad de Chile. (2010). Modelo de Clases. Febrero de 1 de 2016, de Universidad de Chile Sitio web: http://users.dcc.uchile.cl/~psalinas/uml/modelo.html

MySQL Hispano. (2003). Normalización de bases de datos. Enero 20 de 2016, de Mysql-hispano.org Sitio web: http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf

Ma, Zongmin. (2007). Intelligent Databases : Technologies and Applications. London: Zongmin Ma, editor.

Ramez Elmasri, Shumkant Navathe. (20010). Desarrollo de Bses de Datos. Mexico: Pearson.

Nelson Jovanny Isaza Morales. (2003). Modulo para la asignatura Bases de datos I. Medellin: Corporación Universitaria Remington.