ANEXO 1 - SQL

ÍNDICE DE CONTENIDO

1.1 ¿ Que es SQL ?

El SQL (Structured Query Language), Lenguaje de Consulta Estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan.

Como su nombre indica, el SQL nos permite realizar consultas a la base de datos. Pero el nombre se queda corto ya que SQL además realiza funciones de definición, control y gestión de la base de datos. Las sentencias SQL se clasifican según su finalidad dando origen a tres 'lenguajes' o mejor dicho sublenguajes:

- el DDL (Data Description Language), lenguaje de definición de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. (Es el que más varia de un DBMS a otro)
- el DCL (Data Control Language), lenguaje de control de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.
- el DML (Data Manipulation Language), lenguaje de manipulación de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

Desde el punto de vista de SQL, una base de datos es sólo un conjunto de relaciones (o tablas), y para organizarlas o distinguirlas se accede a ellas mediante su nombre. A nivel de sistema operativo, cada base de datos se guarda en un directorio diferente.

1.2 Tipos de datos SQL – JAVA

Los métodos get tienen como nombre ***get*** seguido del tipo de datos a recoger (**`getString`, `getByte`, `getBigDecimal`**,...). Se les pasa entre comillas el nombre del campo a recoger (nombre según esté puesto en la base de datos) o el número de campo según el orden en el que aparezca en la tabla de la base de datos (el primer campo es el 0). Para ello hay que conocer la equivalencia entre los tipos SQL y los tipos Java:

Tipo SQL	Tipo JAVA	Tamaño
`TINYINT`	`byte`	1 byte
`SMALLINT`	`short`	2 bytes
`INTEGER o INT`	`int`	4 bytes
`BIGINT`	`long`	8 bytes

NUMERIC(m,n) O DECIMAL (m,n)	java.math.BigDecimal	Varia según motor
FLOAT	float	4 bytes
REAL	double	8 bytes
CHAR(n)	String	n bytes, 0 <= n <= 255
VARCHAR(n)	String	L+1 bytes, donde L <= n y 0 <= n <= 255
BOOLEAN o BOOL o BIT	boolean	1 byte
DATE	java.sql.Date	3 bytes
TIME	java.sql.Time	3 bytes
TIMESTAMP	java.sql.Timestamp	4 bytes
BINARY(n)	byte[]	n bytes, 0 <= n <= 255
BLOB	java.sql.Blob	L+2 bytes, donde L < 2^16
CLOB	java.sql.Clob	L+4 bytes, donde L < 2^32
ARRAY	java.sql.Array	varia

1.3 Sentencias SQL

Una sentencia SQL es como una frase (escrita en inglés) con la que decimos lo que queremos obtener y de donde obtenerlo. Estas sentencias se ingresan desde una linea de comandos o embebidas en lenguajes como PHP o JAVA para comunicarnos con el servidor. Cada sentencia debe acabar con un punto y coma (;).

La sensibilidad a mayúsculas y minúsculas, depende del sistema operativo, MS Windows no es keysensitive pero GNU Linux si lo es. Por tal se recomiendo siempre usar minúsculas y no utilizar caracteres que pueden variar en diferentes codificaciones como acentos, la ñ, etc..

1.3.1 Sentencias DDL

Palabra Clave SQL	Descripción
CREATE	Crea objetos en el motor como ser base de datos, tablas e indices.
ALTER	Cambia la estructura de objetos de la base de datos
DROP	Elimina objetos de la base de datos
TRUNCATE	Remueve todas las filas de una tabla, mantiene su estructura.
COMMENT	Agrega comentarios al directorio de datos
RENAME	Renombra una tabla de la base de datos

1.3.2 Sentencias DCL

Palabra Clave SQL	Descripción
CREATE USER	Crea objetos en el motor como ser base de datos, tablas e indices.
ALTER USER	Cambia la estructura de objetos de la base de datos

DROP USER	Elimina objetos de la base de datos
GRANT	Remueve todas las filas de una tabla, mantiene su estructura.
REVOKE	Agrega comentarios al directorio de datos
RENAME USER	Renombra una tabla de la base de datos
SET PASSWORD	Setea la contraseña para el usuario

1.3.3 Sentencias DML

Palabra Clave SQL	Descripción
SELECT	Obtiene datos de una o más tablas.
FROM	Las tablas involucradas en la consulta. Se requiere para cada SELECT.
WHERE	Los criterios de selección que determinan cuáles filas se van a recuperar, eliminar o actualizar. Es opcional en una consulta o instrucción de SQL.
GROUP BY	Criterios para agrupar filas. Es opcional en una consulta SELECT.
ORDER BY	Criterios para ordenar filas. Es opcional en una consulta SELECT.
INNER JOIN	Fusionar filas de varias tablas.
INSERT	Insertar filas en una tabla especificada.
UPDATE	Actualizar filas en una tabla especificada.
DELETE	Eliminar filas de una tabla especificada.

1.4 Consola de comandos

Todo RDBMS puede ser administrado desde una consola de comandos, aquí se expondrán algunos de los comandos utilizados para MySQL.

La conexión al servidor MySQL para crear, modificar o realizar cualquier otra operación sobre bases de datos se realiza mediante el programa **mysql** que se encuentra en la carpeta *bin* del software instalado, sea el motor de forma independiente o como conjunto de software (ej XAMPP).

La ejecución de este programa nos lleva al llamado monitor de MySQL que es la línea de comandos desde la que podemos ejecutar instrucciones MySQL.

Este programa posee numerosas opciones de arranque. Para saber cuáles son, basta ejecutar la instrucción `mysql --help` desde la línea de comandos.

Normalmente la entrada al monitor se hace mediante:

```
> mysql -u usuario -p contraseña
```

Hay un usuario de privilegios absolutos que se llama **`root`**, depende como haya sido instalado puede acceder con o sin contraseña. Si hemos accedido bien al monitor, en la línea de comandos aparecerá el texto `mysql>`. Que indicará que el monitor está esperando comandos para ejecutar sobre el servidor.

Para abandonar el monitor basta escribir el comando `quit`.

Sobre los comandos hay que tener en cuenta que:

- Da lo mismo escribir en mayúsculas o en minúsculas
- Todos los comandos terminan con el símbolo “;”
- El comando termina su ejecución si en la línea de comandos observamos el texto mysql>
- Se pueden realizar operaciones aritméticas (Ej: 3 * 6)
- En la misma línea se pueden colocar dos comandos (Ej: SELECT 3 * 6; SELECT SIN(PI());) siempre y cuando los puntos y comas se coloquen de forma adecuada
- Una instrucción puede abarcar más de 1 línea (para informar que no ha terminado la instrucción, el monitor coloca el símbolo “->”, en lugar del normal “mysql>”). Ej:

```
mysql> SELECT * FROM clientes
    -> WHERE apellido = "Garcia";
```

- Una instrucción se puede anular antes del punto y coma, colocando el texto “`\c`”
- Las cadenas de texto literal puede ir entre símbolos de comilla simple o símbolos de comillas doble. Si se pulsa Intro antes de cerrar la cadena, el monitor lo índica mostrando `'>` o `">` en lugar del habitual `->` o `mysql>`.

Mostrar Bases de Datos

```
mysql> SHOW DATABASES;
```
Permite visualizar las bases de datos actualmente activas y a las que tenga acceso el usuario logueado.

Usar una Base de Datos

```
mysql> USE PruebaBD;
```
Permite usar la base de datos PruebaBD.

Crear una Base de Datos

```
mysql> CREATE DATABASE EjemploBD;
```
Crea la base de datos EjemploBD.

Eliminar una Base de Datos

```
mysql> DROP DATABASE EjemploBD;
```
Elimina la base de datos EjemploBD.

Mostrar tablas de una Base de Datos

```
mysql> SHOW TABLES;
```
Muestra las tablas de la base de datos actual.

Crear tablas

```
mysql> CREATE TABLE personas (nombre varchar(30), apellido varchar(30),
    -> mail varchar(30), telefono varchar(12));
```
Crea la tabla personas con los campos nombre, apellido y mail de tipo varchar y maximo 30 caracteres y telefono de 12 caracteres.

Durante la creación de campos se pueden indicar opciones sobre los campos. Estas opciones se colocan tras el tipo de dato del campo.

```
mysql> CREATE TABLE personas (nombre varchar(30) not null,
    ->apellido varchar(30), mail varchar(30), telefono varchar(12) unique);
```

not null indica que el campo no puede estar vacío de dato. **Unique** que no admite repetición.
Se puede añadir la palabra **primary key** tras el tipo de datos del campo que se desea sea la clave de la tabla. Si la clave es más de un campo se realiza colocando la palabra **primary key** como nombre de campo, seguida de los campos que forman la clave entre paréntesis.

```
mysql> CREATE TABLE pieza (codigo1 varchar(5), codigo2 int(2), peso int,
    ->descripcion text, primary key (codigo1, codigo2);
```

Modificador SQL	Se aplica a	Uso
AUTO_INCREMENT	Enteros	El valor se va incrementando automáticamente en cada registro
BINARY	**CHAR** y **VARCHAR**	Convierte las cadenas a forma binaria en la que se distingue entre mayúsculas y minúsculas
DEFAULT	Todos menos **TEXT** y **BLOB**	Coloca un valor por defecto, el valor se coloca justo detrás de esta palabra)
NOT NULL	Todos	Impide que un campo sea nulo
PRIMARY KEY	Todos	Hace que el campo se considere clave primaria
UNIQUE	Todos	Evita la repetición de valores
UNSIGNED	Enteros	Sólo valen los valores positivos
ZEROFILL	Enteros	Rellena con ceros a la izquierda hasta llegar al ajuste

Ver la estructura de una tabla

```
mysql> DESCRIBE personas;
```

Modificar la estructura de tablas

```
mysql> ALTER TABLE personas CHANGE nombre nombre varchar(20);
```

Modifica el campo nombre para tener un tamaño de 20

```
mysql> ALTER TABLE personas CHANGE nombre identificador varchar(20);
```

Modifica el campo nombre pasa llamarse identificador

```
mysql> ALTER TABLE personas CHANGE identificador nombre varchar(40) NOT NULL -
    > DEFAULT “pepe”;
```

Modifica el campo identificador pasa llamarse nombre como varchar de 40 caracteres maximo, no admite nulos y su valor por defecto es “pepe”.

```
mysql> ALTER TABLE personas RENAME clientes;
```

Modifica el nombre de la tabla a clientes

Eliminar tablas

```
mysql> DROP TABLE personas;
```

Elimina la tabla personas.

Eliminar columnas de una tabla

```
mysql> ALTER TABLE personas DROP telefono;
```

Elimina la columna telefono de la tabla personas.

Agregar columnas a una tabla

```
mysql> ALTER TABLE personas ADD direccion varchar (120);
```

Agrega a la tabla personas la columna direccion.

Crear indices de tablas

```
mysql> ALTER TABLE personas ADD INDEX (nombre, apellido);
```

Crea un indice en la tabla personas con los campos nombre y apellido.

El indice de una base de datos 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.

El índice tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice dicho elemento para, una vez encontrado, devolver un registro que se encuentre en la posición marcada por el índice.

Los índices pueden ser creados usando una o más columnas, proporcionando la base tanto para búsquedas rápidas al azar como de un ordenado acceso a registros eficiente.

En una base de datos relacional un índice es una copia de una parte de la tabla.

Clave principal de una tabla

Una tabla suele tener una columna o una combinación de columnas cuyos valores identifican de forma única cada fila de la tabla. Estas columnas se denominan claves principales de la tabla y exigen la integridad de entidad de la tabla. Debido a que las restricciones de clave principal garantizan datos únicos, con frecuencia se definen en una columna de identidad.

Cuando especifica una restricción de clave principal en una tabla, el motor de base de datos exige la unicidad de los datos mediante la creación automática de un índice único para las columnas de clave principal. Este índice también permite un acceso rápido a los datos cuando se usa la clave principal en las consultas. Si se define una restricción de clave principal para más de una columna, puede haber valores duplicados dentro de la misma columna, pero cada combinación de valores de todas las columnas de la definición de la restricción de clave principal debe ser única.

```
mysql> ALTER TABLE personas ADD PRIMARY KEY (idpersona);
```

Indica que el campo idpersona sera la clave principal de la tabla y por ende su indice.

```
mysql> CREATE TABLE clientes (dni int(8), nombre varchar(30) not null,
    -> apellido varchar(30) not null, direccion varchar(80), tel varchar(12),
    -> PRIMARY KEY (dni), INDEX (apellido, nombre));
```

Clave externa

Una clave externa es una columna o combinación de columnas que se usa para establecer y aplicar un vínculo entre los datos de dos tablas a fin de controlar los datos que se puede almacenar una tabla de clave externa. En una referencia de clave externa, se crea un vínculo entre dos tablas cuando las columnas de una de ellas hacen referencia a las columnas de la otra que contienen el valor de clave principal. Esta columna se convierte en una clave externa para la segunda tabla.

Las claves externas se crean para campos de una tabla relacionados con campos que forman índices en otras tablas (normalmente forman claves principales, es decir son los campos que permiten relacionar tablas). El propósito de la clave externa es asegurar la integridad referencial de los datos. En otras palabras, sólo se permiten los valores que se esperan que aparezcan en la base de datos.

```
mysql> CREATE TABLE tabla (lista y propiedades de campos e indices)
    -> CONSTRAINT nombreDeClave
    -> FOREIGN KEY (camposQueFormanClave)
    -> REFERENCES tabla2 (camposClaveDeTabla2));
```

La palabra **CONSTRAINT** es opcional y permite indicar un nombre para la clave externa.

FOREIGN KEY indica los campos de esta tabla relacionados con campos de otra.

REFERENCES indica el nombre de la tabla relacionada y el nombre de los campos relacionados.

Se pueden indicar al final las palabras:

- **on delete cascade**. Que hace que al borrar registros en la tabla principal, se borren registros en la relacionada.
- **on update cascade**. Que permite que al cambiar los datos de los campos en la tabla principal, se actualicen en la relacionada.
- **on delete set null**. Hace que si se borra un registro de la tabla principal, los valores de los campos relacionados se coloque a null
- **on update set null**. Al cambiar valores principales se dejan a null los relacionados.
- **on update restrict**. No permite modificar valores en la tabla principal si hay registros relacionados en otra tabla.
- **on delete restrict**. No permite eliminar valores en la tabla principal si hay registros relacionados en otra tabla.
- **on update no action**
- **on delete no action**

Eliminar un indice o clave principal de una tabla

```
mysql> ALTER TABLE personas DROP PRIMARY KEY;
mysql> ALTER TABLE personas DROP INDEX (nombre, apellido);
```

Introducir datos a una tabla

```
mysql> INSERT INTO personas VALUES ('Matias', 'Garcia', 'mimail@gmail.com',
    -> '154545-8989');
```

Agrega un nuevo registro en la tabla personas con los datos completos para cada campo. El orden de los datos debe corresponder con el de los campos de la tabla.

```
mysql> INSERT INTO personas (nombre, apellido, telefono) VALUES (‘Angélica’,
    -> ‘Lione’, ‘154747-9897’);
```
Agrega un registro completando los campos indicados. El campo mail que no fue especificado ingresa como NULL en el registro.

Modificar datos de una tabla

```
mysql> UPDATE personas SET apellido = ‘González’ WHERE nombre = ‘Angélica’;
```
Modifica el apellido del registro cuyo campo nombre tenga el texto indicado.

```
mysql> UPDATE articulos SET precio = precio * 1,21, descuento = descuento/ 2;
```
Modifica el precio y el descuento de todos los registros de la tabla articulos.

Eliminar datos de una tabla

```
mysql> DELETE FROM personas WHERE nombre = ‘Angélica’;
```
Elimina todos los registros cuyo campo nombre tenga el texto indicado.

```
mysql> DELETE FROM articulos WHERE precio > 100;
```
Elimina todos los registros de la tabla articulos cuyo precio sea mayor a 100.

Consultas a una base de datos

La instrucción **SELECT** es la fundamental del lenguaje SQL y por tanto de MySQL. Esta instrucción permite realizar consultas sobre la base de datos. El formato básico de la instrucción es:

```
mysql> SELECT listaDeCampos FROM tablas WHERE condicion;
```

```
mysql> SELECT * FROM personas;
```
El campo especial “*” sirve para representar todos los campos de una tabla.

```
mysql> SELECT nombre, apellido FROM personas;
```
indica la lista exacta de campos que queremos que aparezcan

Condiciones

```
mysql> SELECT nombre, apellido FROM personas WHERE edad > 25;
```
El apartado **WHERE** de la instrucción **SELECT** nos permite poner una condición de modo que sólo aparezcan en la consulta los registros que cumplan la condición. Se pueden usar todos los operadores de comparación y los conectores **AND** y **OR**.

```
mysql> SELECT * FROM personas WHERE apellido LIKE 'G%';
```
Obtiene todas las personas cuyo primer apellido empiece por “G”. Es decir el símbolo “%” hace de comodín. Otras expresiones posibles para like son:

Expresión LIKE	Significado
'g%'	Que empiece por g
'%g'	Que termine con g
'%g%'	Que tenga una g
'_____'	Que tenga cinco caracteres

Esto se puede extender de forma más poderosa utilizando **REGEXP** en lugar de **LIKE**. **REGEXP** permite utilizar expresiones regulares. Algunas posibilidades son:

Expresión regular	Significado
“.”	Cualquier carácter, pero sólo uno
“[xyz]”	El carácter x, el y o el z
“[x-z]”	Igual que el anterior
“[0-9]”	Cualquier número
“x*”	Una o más equis
“.*”	Cualquier número de caracteres
“^b”	Que empiece por b
“b$”	Que termine por b
“[69].*”	Que empiece por 6 o por 9
“^[69]”	Que empiece por 6 o por 9
“^.....$”	Que tenga exactamente cinco caracteres
“^.{5}$”	Que tenga exactamente cinco caracteres

Ordenamiento

```
mysql> SELECT * FROM personas WHERE edad <= 30 ORDER BY apellido,nombre;
```

La cláusula **ORDER BY** sirve para ordenar en base a una o más columnas. Normalmente la ordenación se realiza en ascendente (de la A a la Z o de menor a mayor en el caso de los números), pero se puede colocar la palabra clave **DESC** tras el nombre del campo por el que se ordena, para indicar que ese campo irá en descendente.

Consulta de campos calculados

```
mysql> SELECT nombre_art, precio, precio * 0,21 FROM articulos;
```

Esto permite realizar cálculos con las columnas de consulta. El resultado de cada cálculo se coloca en una nueva columna de la tabla. Para los cálculos se pueden utilizar operadores aritméticos y funciones.

A la columna (o columnas) de cálculo se le puede poner nombre haciendo uso de **AS**. Ejemplo:

```
mysql> SELECT nro_art, precio, precio * 1,21 AS total FROM articulos ORDER BY
    -> nro_art;
```

Consultas agrupadas

```
mysql> SELECT provincia, COUNT(*) FROM localidades GROUP BY provincia;
```

Se pueden agrupar los resultados según uno o más campos. La consulta muestra las provincias presentes en la tabla de localidades. Si no hubiera apartado **GROUP BY** también saldrían las provincias, pero cada provincia saldría tantas veces como localidades incluya. La mayor ventaja que ofrecen estas consultas es que se pueden hacer cálculos sobre los grupos. En este caso aparece una segunda columna que contará los registros de cada grupo (es decir las localidades de cada provincia). Otros operadores son **SUM**, **MAX**, **MIN** y **AVG** (media).

Consultas utilizando varias tablas

Se trata de consultas realizadas sobre datos de varias tablas. para ello esas tablas deben estar relacionadas por al menos un campo.

```
mysql> SELECT nombre, apellido, fecha_alquiler FROM cliente, alquiler WHERE
    -> cliente.dni = alquiler.dni;
```

Consulta a la relación que existe entre la tabla cliente y la tabla alquiler.

A veces el nombre de los campos son ambiguos (porque el mismo nombre se emplea en más de una de las tablas implicadas en la consulta) y entonces se debe indicar la tabla junto al nombre del campo, separados por un punto:

```
mysql> SELECT cliente.idcliente, cliente.nombre, cliente.apellido,
    -> alquiler.fecha_alquiler FROM cliente, alquiler WHERE cliente.dni =
    -> alquiler.dni;
```

Consulta a la relación que existe entre la tabla cliente y la tabla alquiler.

Se puede poner un alias a las tablas con la palabra **AS**:

```
mysql> SELECT c.nombre, c.apellido, fecha_alquiler FROM cliente AS c, alquiler
    -> WHERE c.dni = alquiler.dni;
```

Esto muestra las fechas de cada alquiler junto con nombre y apellidos del cliente que alquiló. Para ello ambas tablas deben estar relacionadas por el DNI. Esta misma consulta se puede hacer en el formato SQL ANSI-92 (totalmente soportado por MySQL) de esta forma:

```
mysql> SELECT nombre, apellido, fecha_alquiler FROM cliente JOIN alquiler ->
    -> ON cliente.dni = alquiler.dni;
```

Es más recomendable esta segunda forma ya que permite realizar asociaciones avanzadas. De hecho es posible usar estas formas de unión en el apartado **JOIN**

- **CROSS JOIN**. Producto cruzado. Combina cada registro de la primera tabla con cada registro de la tabla relacionada.
- **INNER JOIN**. Unión normal. Muestra sólo registros de ambas tablas que estén relacionados.
- **LEFT JOIN**. Muestra todos los registros de la primera tabla y sólo los registros relacionados en la segunda.
- **RIGHT JOIN**. Muestra todos los registros de la segunda tabla y sólo los registros relacionados en la primera.

2.1 phpMyAdmin

phpMyAdmin es un software de código abierto, diseñado para manejar la administración y gestión de bases de datos MySQL a través de una interfaz gráfica de usuario basada en web. Escrito en PHP,

multiplataforma, con traducción en mas de 50 idiomas y con licencia GPL, se ha convertido en una de las más populares herramientas de gestión de MySQL.

Con esta herramienta puedes crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos.

Las especificaciones proveídas por este software incluyen:

- Interface Web
- Manejador de base de datos MySQL, MariaDB y Drizzle
- Importación de datos desde CSV y SQL
- Exporta datos a varios formatos: CSV, SQL, XML, PDF (via la biblioteca TCPDF), ISO/IEC 26300 - OpenDocument Text y Spreadsheet, Word, Excel, LaTeX y otros
- Administración de múltiples servidores
- Crea gráficos PDF del diseño de la base de datos
- Crea consultas complejas usando Query-by-Example (QBE)
- Búsqueda global en una base de datos o un subconjunto de esta
- Transforma datos almacenados a cualquier formato usando un conjunto de funciones predefinidas, tal como BLOB
- Live charts para monitorear las actividades del servidor MySQL tales como conexiones, procesos, uso de CPU/Memoria, etc.

2.2 Instalación

Al ser un software multiplataforma y con licencia GPL, su descarga es libre y gratuita y se debe optar por el archivo de instalación que corresponda a la versión de sistema operativo que se tenga. Se debe contar con un navegador web para poder ver la interfaz gráfica.

Hay básicamente dos formas de instalación:

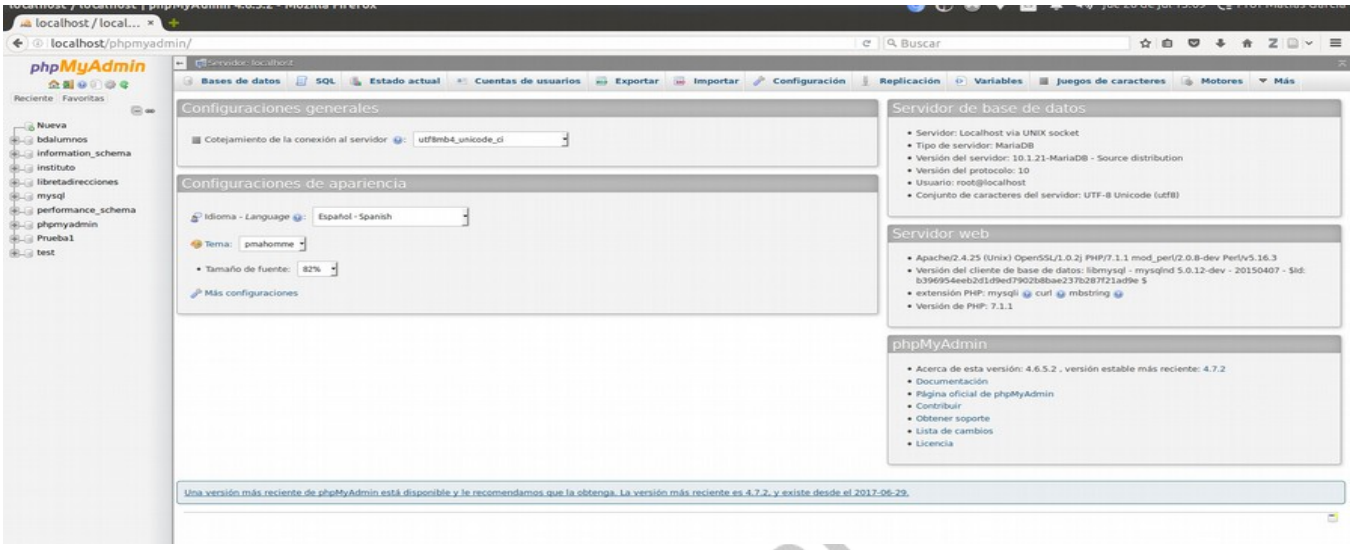
1. Paquete de Software, son instaladores que ademas de instalar phpMyAdmin, instalan otro software que puede ser de utilidad como el servidor web y el motor de base de datos. Ej. XAMPP – WAMPP – LAMPP – WAMPServer.
2. Instalar solo el software, descargándolo de la pagina oficial https://www.phpmyadmin.net/ e instalarlo. Obviamente se debe tener antes instalado el motor de base de datos, un servidor web para poder levantar la instancia y un navegador web para poder visualizar.

2.3 Acceso

PhpMyAdmin cuenta con una interfaz gráfica que se utiliza desde un navegador web. Para acceder a esta se deberá ingresar la URL que corresponda según sea un servidor Local, Remoto en una PC dentro

de un red a la cual se tenga acceso o en un Host Server en Internet.

- Local: deben estar inicializados el motor de base de datos y el servidor web local y desde un navegador web, generalmente, se accede a http://localhost/phpmyadmin/



- Remoto: en la PC donde se encuentre el servidor web y phpMyAdmin se deberá dar los permisos necesarios para acceder de forma remota. En el navegador se deberá ingresar la URL y el puerto de conexión para poder acceder. Ej 192.168.0.5/phpmyadmin
- Si el servicio se encuentra en un Host Server desde el panel de control del mismo tendremos el acceso a phpMyAdmin.

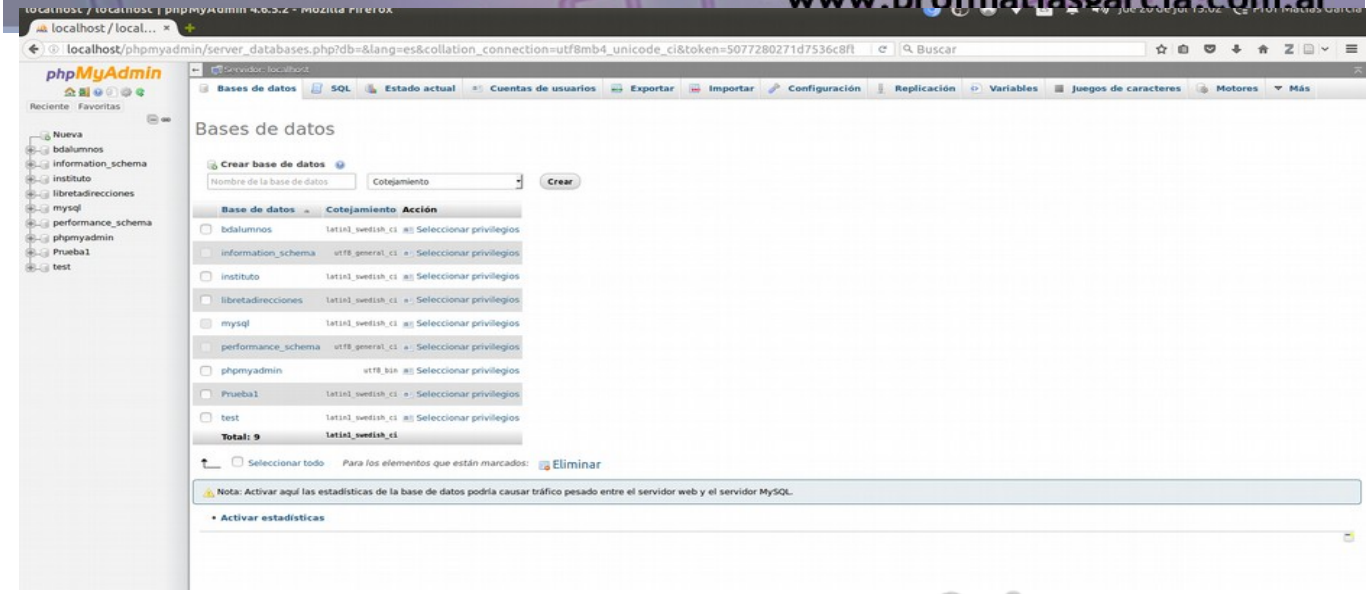
2.4 UTILIZACIÓN

Al ingresar veremos del lado izquierdo un panel donde se visualizan las bases de datos ya creadas dentro del servidor de base de datos y si desplegamos veremos parte de la información de la base, como tablas que contiene, indices, etc. En este panel también encontraremos unos iconos para acceder a la Pagina de inicio, Documentación, Configuración del panel, etc.

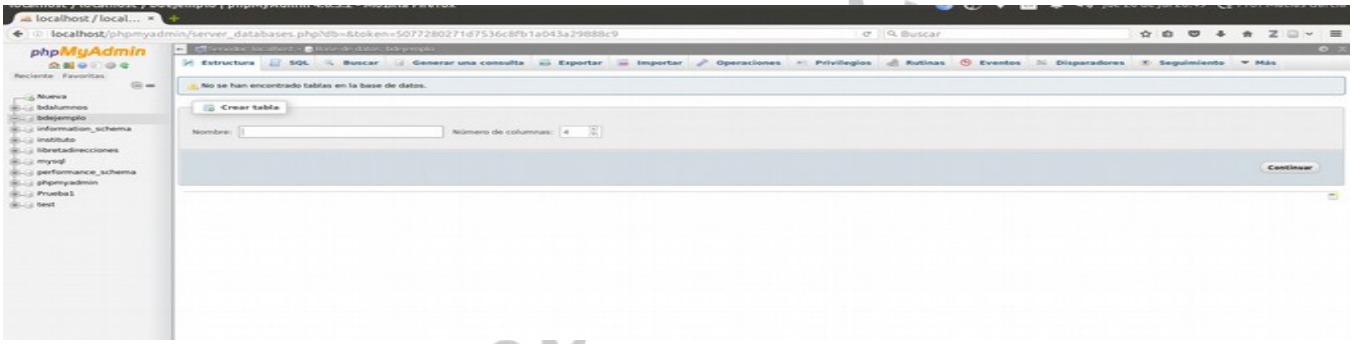
Arriba de la interfaz encontraremos los botones para acceder a diferentes pestañas de aplicaciones o información, la cual cambiará de acuerdo lo que se este realizando.

Ejemplificare el uso de phpMyAdmin creando una base de datos “bdejemplo”

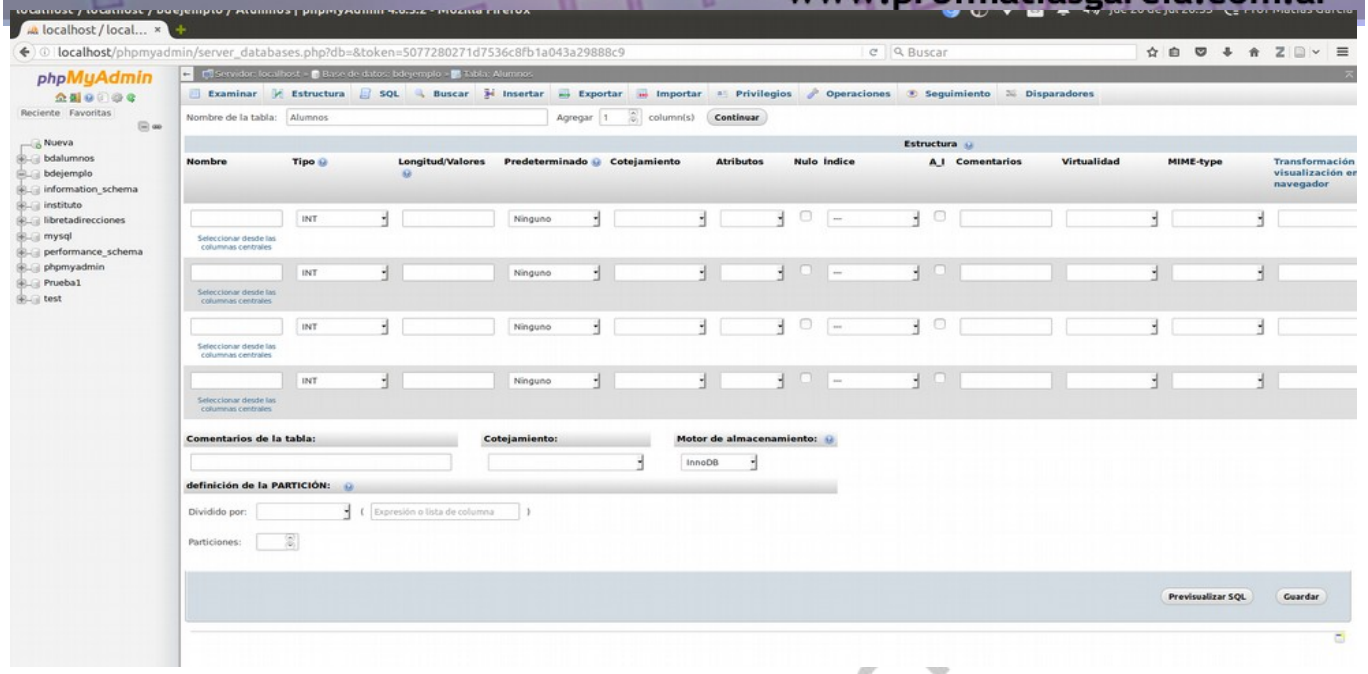
1. Al hacer click en *Bases de datos* veremos la opción para crear una y el listado de las ya creadas. Se ingresara allí el nombre de la base a crear y seleccionaremos la codificación (*cotejamiento*) a usar para los datos de la base. En Argentina generalmente se utiliza *utf8_spanish_ci* que permite ingresar valores con ñ y acentos, aunque no es aconsejable utilizar dichos caracteres.



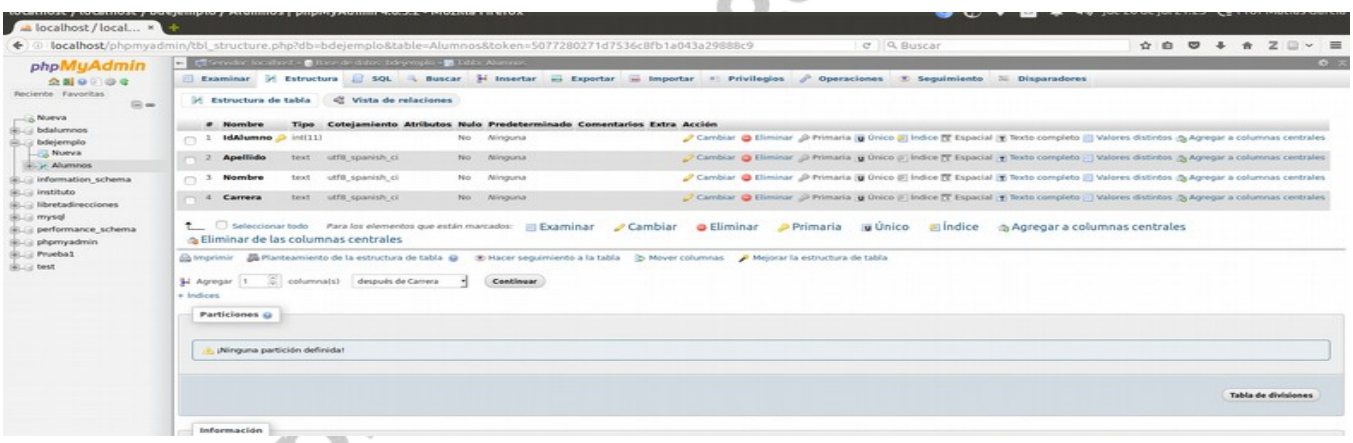
2. Luego de crear la base nos solicitará crear una tabla e indicar cuantas columnas (atributos) tendrá. Se podría realizar luego este paso igualmente para crear mas tablas.



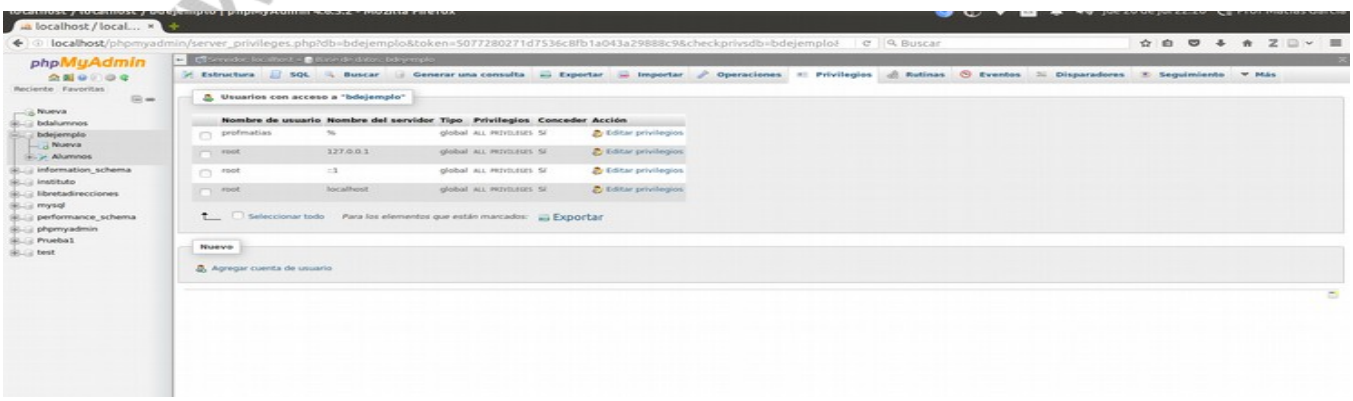
3. Definir cada uno de los campos, el nombre que tendrá, indicando el tipo de dato (tener en cuenta como se manejara desde el lenguaje que accederá al campo), se puede ingresar la longitud máxima, si ya tiene un valor predeterminado en caso de que no se ingrese nada, si puede ser nulo, el indice puede indicar si el atributo es la clave principal (PRIMARY) y comentarios.



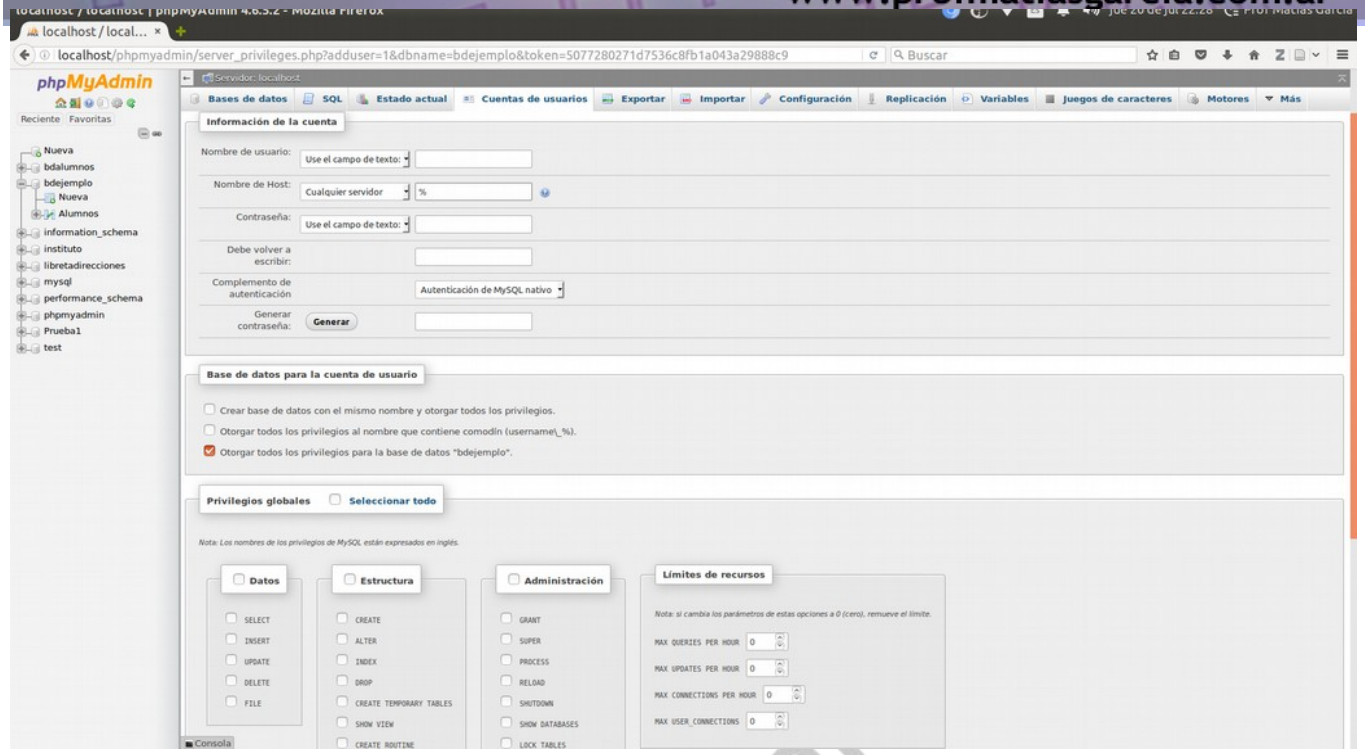
4. Seleccionando la tabla de la base de datos realizada se podrá modificar su estructura, o sea, modificar cada campo, insertar registros en la tabla, realizar consultas SQL, exportar o importar datos de la tabla, verificar los privilegios de acceso y generar triggers.



5. Seleccionando desde el panel izquierdo el nombre de la base de datos podemos ingresar a la pestaña de privilegios y crear y administrar los usuarios que tendrán acceso a la misma.



6. Para crear un nuevo usuario se deberá completar el nombre, contraseña y los privilegios que tendrá. Estos usuarios son los que deberemos configurar para tener acceso desde el software que se este desarrollando para acceder a la base de datos.



3.1 Peligros en el diseño de bases de datos relacionales

Uno de los retos en el diseño de la base de datos es el de obtener una estructura estable y lógica tal que:

- El sistema de base de datos no sufra de anomalías de almacenamiento.
- El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una base de datos implantada sobre un modelo bien diseñado tiene mayor esperanza de vida aun en un ambiente dinámico, que una base de datos con un diseño pobre. Una base de datos bien diseñada tendrá un buen desempeño aunque aumente su tamaño, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales.

Existen diversos riesgos en el diseño de las bases de datos relacionales que afecten la funcionalidad de la misma, los riesgos generalmente son la redundancia de información y la inconsistencia de datos.

La normalización es el proceso de simplificar la relación entre los campos de un registro.

Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.

- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

En términos más sencillos la normalización trata de simplificar el diseño de una base de datos, esto a través de la búsqueda de la mejor estructuración que pueda utilizarse con las entidades involucradas en ella.

Pasos de la normalización:

1. Descomponer todos los grupos de datos en registros bidimensionales.
2. Eliminar todas las relaciones en la que los datos no dependan completamente de la llave primaria del registro.
3. Eliminar todas las relaciones que contengan dependencias transitivas.

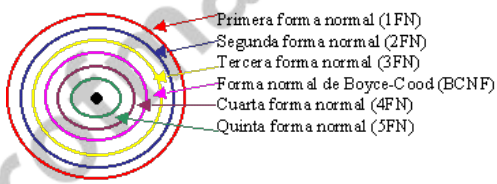
La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

3.2 Formas normales.

Son las técnicas para prevenir las anomalías en las tablas.

Dependiendo de su estructura, una tabla puede estar en primera forma normal, segunda forma normal o en cualquier otra.

Relación entre las formas normales:



3.2.1 Primera forma normal.

Definición formal:

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valores atómicos.

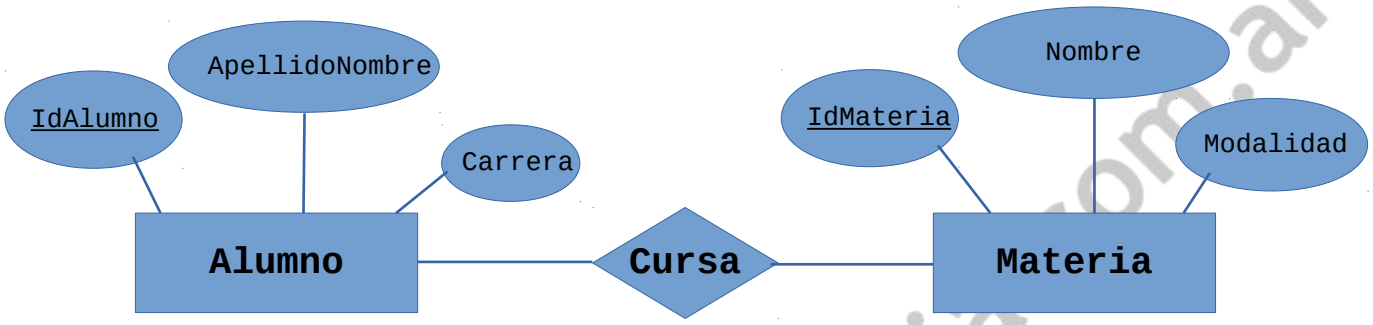
Abreviada como 1FN, se considera que una relación se encuentra en la primera forma normal cuando cumple lo siguiente:

1. Las celdas de las tablas poseen valores simples y no se permiten grupos ni arreglos repetidos como valores, es decir, contienen un solo valor por cada celda.
2. Todos los ingresos en cualquier columna(atributo) deben ser del mismo tipo.
3. Cada columna debe tener un nombre único, el orden de las columnas en la tabla no es importante.
4. Dos filas o renglones de una misma tabla no deben ser idénticas, aunque el orden de las filas no

es importante.

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal.

Para ejemplificar como se representan gráficamente las relaciones en primera forma normal consideremos la relación alumno cursa materia cuyo diagrama E-R es el siguiente:



Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN.

3.2.2 Segunda forma normal.

Para definir formalmente la segunda forma normal requerimos saber que es una dependencia funcional: Consiste en edificar que atributos dependen de otro(s) atributo(s).

Definición formal:

Una relación R está en 2FN si y solo si está en 1FN y los atributos no primos dependen funcionalmente de la llave primaria.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de la clave . De acuerdo con está definición, cada tabla que tiene un atributo único como clave, esta en segunda forma normal.

La segunda forma normal se representa por dependencias funcionales como:

Nótese que las llaves primarias están representadas con doble cuadro, las flechas nos indican que de estos atributos se puede referenciar a los otros atributos que dependen funcionalmente de la llave primaria.

3.2.3 TERCERA FORMA NORMAL.

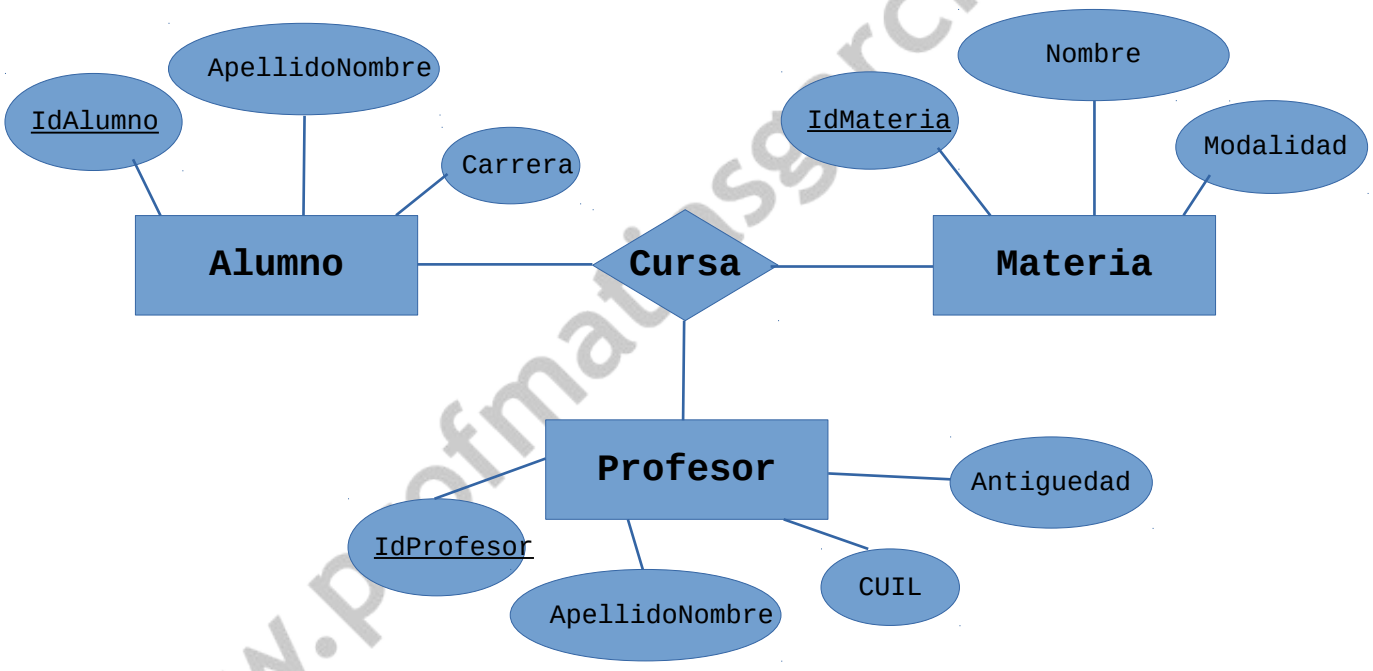
Para definir formalmente la 3FN necesitamos definir dependencia transitiva: En una afinidad (tabla bidimensional) que tiene por lo menos 3 atributos (A,B,C) en donde A determina a B, B determina a C pero no determina a A.

Definición formal:

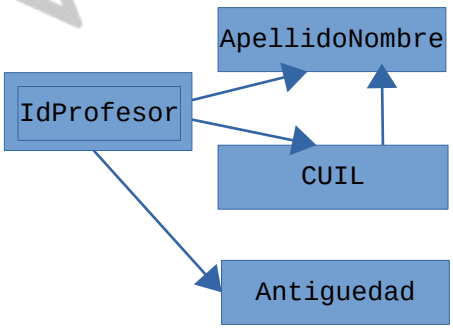
Una relación R está en 3FN si y solo si esta en 2FN y todos sus atributos no primos dependen no transitivamente de la llave primaria.

Consiste en eliminar la dependencia transitiva que queda en una segunda forma normal, en pocas palabras una relación esta en tercera forma normal si está en segunda forma normal y no existen dependencias transitivas entre los atributos, nos referimos a dependencias transitivas cuando existe más de una forma de llegar a referencias a un atributo de una relación.

Por ejemplo, consideremos el siguiente caso:

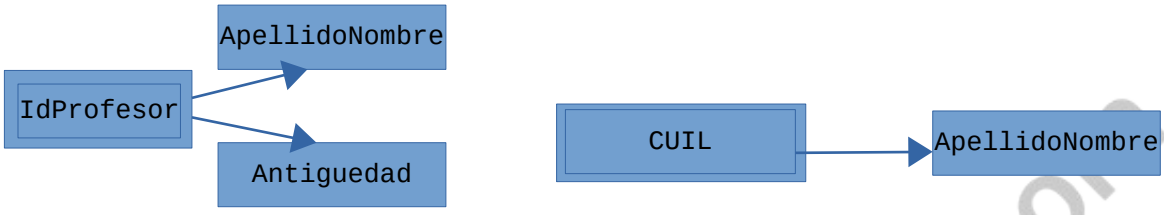


Tenemos la relación Alumno-Cursa-Materia manejada anteriormente, pero ahora consideramos al elemento Profesor, gráficamente lo podemos representar de la siguiente manera:



Podemos darnos cuenta que se encuentra graficado en segunda forma normal, es decir que todos los

atributos llave están indicados en doble cuadro indicando los atributos que dependen de dichas llaves, sin embargo en la llave IdProfesor tiene como dependientes a 3 atributos en el cual el ApellidoNombre puede ser referenciado por dos atributos: IdProfesor y CUIL (Existe dependencia transitiva), podemos solucionar esto aplicando la tercera forma normal que consiste en eliminar estas dependencias separando los atributos, entonces tenemos:



3.2.4 Forma normal de Boyce Codd.

Determinante: Uno o más atributos que, de manera funcional, determinan otro atributo o atributos. En la dependencia funcional (A,B)-->C, (A,B) son los determinantes.

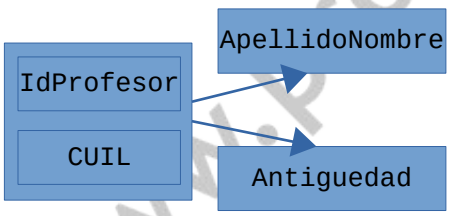
Definición formal:

Una relación R esta en FNBC si y solo si cada determinante es una llave candidato.

Denominada por sus siglas en ingles como BCNF; Una tabla se considera en esta forma si y sólo sí cada determinante o atributo es una llave candidato.

Continuando con el ejemplo anterior, si consideramos que en la entidad Profesor sus atributos IdProfesor y CUIL nos puede hacer referencia al atributo ApellidoNombre, entonces decimos que dichos atributos pueden ser llaves candidato.

Gráficamente podemos representar la forma normal de Boyce Codd de la siguiente forma:



Obsérvese que a diferencia de la tercera forma normal, agrupamos todas las llaves candidato para formar una global (representadas en el recuadro) las cuales hacen referencia a los atributo que no son llaves candidato.

3.2.5 Cuarta forma normal.

Definición formal:

Un esquema de relaciones R está en 4FN con respecto a un conjunto D de dependencias funcionales y de valores múltiples sí, para todas las dependencias de valores múltiples en D de la forma X->->Y,

donde X<=R y Y<=R, se cumple por lo menos una de estas condiciones:

- X->->Y es una dependencia de valores múltiples trivial.
- X es una superllave del esquema R.

Para entender mejor aún esto consideremos una afinidad (tabla) llamada estudiante que contiene los siguientes atributos: Clave, Especialidad, Curso tal y como se demuestra en la siguiente figura:

Clave	Especialidad	Curso
S01	Sistemas	Natacion
S01	Bioquimica	Danza
S01	Sistemas	Guitarra
B01	Bioquimica	Natacion
C03	Derecho	Danza

Suponemos que los estudiantes pueden inscribirse en varias especialidades y en diversos cursos. El estudiante con clave S01 tiene su especialidad en Sistemas y Bioquímica y toma los cursos de Natación, Danza y Guitarra, el estudiante B01 tiene la especialidad en Bioquímica y toma el curso de Natación, el estudiante con clave C03 tiene la especialidad de Derecho y toma el curso de Danza.

En esta tabla o relación no existe dependencia funcional porque los estudiantes pueden tener distintas especialidades, un valor único de clave puede poseer muchos valores de especialidades al igual que de valores de cursos. Por lo tanto existe dependencia de valores múltiples. Este tipo de dependencias produce redundancia de datos, como se puede apreciar en la tabla anterior, en donde la clave S01 tiene tres registros para mantener la serie de datos en forma independiente lo cual ocasiona que al realizarse una actualización se requiera de demasiadas operaciones para tal fin.

Existe una dependencia de valores múltiples cuando una afinidad tiene por lo menos tres atributos, dos de los cuales poseen valores múltiples y sus valores dependen solo del tercer atributo, en otras palabras en la afinidad R (A,B,C) existe una dependencia de valores múltiples si A determina valores múltiples de B, A determina valores múltiples de C, y B y C son independientes entre sí.

En la tabla anterior Clave determina valores múltiples de especialidad y clave determina valores múltiples de curso, pero especialidad y curso son independientes entre sí.

Las dependencias de valores múltiples se definen de la siguiente manera: Clave ->->Especialidad y Clave->->Curso; Esto se lee "Clave multidetermina a Especialidad, y clave multidetermina a Curso"

Para eliminar la redundancia de los datos, se deben eliminar las dependencias de valores múltiples. Esto se logra construyendo dos tablas, donde cada una almacena datos para solamente uno de los atributos de valores múltiples.

Para nuestro ejemplo, las tablas correspondientes son:

Tabla EEspecialidades

Clave	Especialidad
S01	Sistemas
B01	Bioquimica
C03	Derecho

Tabla ECursos

Clave	Curso
S01	Natacion
S01	Danza
S01	Guitarra
B01	Natacion
C03	Danza

3.2.6 Quinta forma normal.

Definición formal:

Un esquema de relaciones R está en 5FN con respecto a un conjunto D de dependencias funcionales, de valores múltiples y de producto, si para todas las dependencias de productos en D se cumple por lo menos una de estas condiciones:

- (R1, R2, R3, ... Rn) es una dependencia de producto trivial.
- Toda Ri es una superllave de R.

La quinta forma normal se refiere a dependencias que son extrañas.

Tiene que ver con tablas que pueden dividirse en subtablas, pero que no pueden reconstruirse.

BIBLIOGRAFÍA

- Carbonell Moises “phpMyAdmin Introducción a la creación de bases de datos” (UPV 2009)
- González Flora, López Tomas, Rodríguez Cesar, "Diseño de Bases de Datos relacionales" (2005)
- Oppel Andy, Sheldon Robert, “Fundamentos de SQL” 3ra. Ed. (Mc Graw Hill 2009)
- Sánchez, Jorge, “JAVA 2” (2004)
- Sánchez, Jorge, “MySQL guía rápida” (2004)

LICENCIA

Matías E. García

Prof. & Tec. en Informática Aplicada
www.profmatiasgarcia.com.ar
info@profmatiasgarcia.com.ar