Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Tutorial de Prácticas del Gestor de Bases de Datos MySQL

Trabajo de graduación previo a la obtención del título de Ingeniero de Sistemas

Autores: Diana María Coello Serrano
José Luis Izquierdo Aguirre

Director: Ing. Oswaldo Merchán Manzano

Cuenca, Ecuador
2008

DEDICATORIA

Esta monografía está dedicada a nuestras familias quienes han estado presentes incondicionalmente, animándonos y alentándonos a continuar con nuestros estudios, también se la dedicamos a nuestros amigos y a todas las personas que nos han apoyado durante todo el transcurso de nuestra vida universitaria, y han ejercido alguna influencia para ayudarnos a ser mejores día a día.

También queremos dedicar esta monografía a los estudiantes de la Escuela de Sistemas, ya que son ellos la razón principal del desarrollo de este tutorial, esperando que sea de su agrado y sirva para reforzar sus conocimientos en la materia de Bases de Datos.

AGRADECIMIENTOS

Queremos agradecer a nuestras familias por ser el pilar fundamental en nuestras vidas, que, a más de ser imprescindibles nos han brindado su cariño y apoyo a lo largo de nuestras vidas.

También queremos agradecer a nuestros amigos quienes han luchado con nosotros, brindándonos su tiempo y apoyo dentro y fuera de las aulas.

Asimismo queremos expresar nuestros agradecimientos a nuestros queridos profesores de la Universidad del Azuay y de la Universidad de Buenos Aires, especialmente a aquellos que realmente se esmeran y preocupan por formar a los profesionales que son la actual generación y futuro de este país, entre ellos: Ing. Miguel Moscoso Cobos, Ing. Patricia Ortega, Ing. Roberto Cobos, Ing. Rubén Ortega, Ing. Hernán Astudillo, Ing. Miguel Martínez Loaiza, Ing. Juan Córdova, Ing. Juana Bersoza, Ing. Boris Quinde, Ing. Fabián Carvajal, Ing. Pablo Esquivel, Ing. María Inés Acosta, Ing. Iván Andrade, Ing. Paúl Ochoa, Ing. Rómulo Terreros, Ing. Pablo Pintado, Ing. Jorge Espinoza Idrovo, Ing. Juan Pablo Carvallo, Lcdo. Carlos Pérez, Lcdo. Galo Fajardo, Ing. Marcelo Utard, Ing. Fernando Bonelli, Ing. Luis Valle, Lcdo. Enrique Marzullo, Ing. Hugo Pagola y a Nicolás Matsunaga.

De igual manera agradecemos a la Universidad del Azuay por abrirnos las puertas y darnos la formación profesional y humana gracias a la cual somos quienes somos y de manera especial a nuestro querido maestro, amigo y director de monografía, el Ing. Oswaldo Merchán, siendo él quien planteó, motivó e impulsó el desarrollo de este tutorial.

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES Y CUADROS

RESUMEN

El presente tutorial pretende ser una guía para los estudiantes y usuarios del Gestor de Bases de Datos *MySQL versión 5.0*, por lo que, recoge todos los conceptos básicos para la gestión de las bases de datos, instrucciones precisas y detalladas del lenguaje *MySQL*, ejemplos prácticos y un proyecto final que ayudará a estos usuarios a poder entender y utilizar de forma clara y precisa el mencionado gestor.

Este tutorial servirá para asentar y reforzar de forma práctica los conocimientos adquiridos en Bases de Datos, pues, una de las mejores maneras de aprender es ejercitarnos constantemente mediante la práctica, por lo que, utilizando el gestor de bases de datos, podemos desarrollar y adquirir fácilmente las destrezas que nos permitan el dominio de la estructura de las instrucciones *SQL* y su correcta manipulación, además de utilizar éste conocimiento para poder extendernos al manejo de otros gestores.

ABSTRACT

This tutorial is intended to be a guide for students and users of the 5.0 version MySQL Database Agent. For that reason it contains all the basic concepts for the management of databases as well as precise and detailed instructions of the MySQL language, practical examples, and a final project that will help these users to be able to understand and use the mentioned agent in a clear and precise way.

This tutorial will serve to reinforce, in a practical way, the knowledge acquired in databases since one of the best ways of exercising is through practice. Therefore, by using the database agent we can learn the structure of SQL instructions and its manipulation more easily, and we can use this knowledge for the handling of other agents.

El contenido que versa en las siguientes páginas, es de exclusiva responsabilidad de sus autores.

Diana María Coello Serrano

José Luis Izquierdo Aguirre

INTRODUCCIÓN

El presente documento es un tutorial ilustrado, tema escogido para facilitar al estudiante el aprendizaje del Gestor de Bases de datos de *MySQL*, por medio del desarrollo de ejercicios prácticos guiados paso a paso. En este tutorial también se han incluido algunos conceptos básicos para clarificar y reforzar el conocimiento que posee el estudiante.

El objetivo de este tutorial es aportar al uso y aplicación del Gestor de Bases de Datos *MySQL*, ya que siendo éste de uso libre (*software* libre), cada día se vuelve más popular, además es una de las tendencias más importantes de la industria informática actual gracias a las herramientas y características que presenta y su manipulación se realiza mediante el lenguaje universal de las bases de datos que es el *SQL*.

Para facilitar la comprensión de este tutorial, lo hemos dividido en 6 capítulos, desarrollados de forma progresiva, esto para que sirva como guía para el estudiante que se encuentre cursando la materia de Bases de Datos, además el presente documento aportará para la realización de trabajos prácticos y el desarrollo del proyecto final de esta materia.

Al final del tutorial, hemos adjuntado unos anexos, en los cuales explicamos como crear y manipular una base de datos y su contenido por medio de *MySQL Administrator* y *MySQL Query Browser*. También se explica como migrar una base de datos desde otro gestor utilizando *MySQL Migration Toolkit.* Esto con el fin de que el estudiante tenga conocimientos sobre las herramientas de *MySQL.*

CAPITULO 1

MYSQL V. 5.0

INTRODUCCIÓN

El presente capítulo hace referencia a la instalación del Gestor de Bases de Datos *MySQL* v.5.0 para el sistema operativo *Windows XP SP2*, con sus respectivas herramientas y una breve descripción de estas, explicando paso a paso y con la ayuda de imágenes la forma de realizarla. De igual manera se explica la configuración, conexión y desconexión de Gestor *MySQL*, al igual que la correcta utilización del lenguaje *SQL*.

1.1 Instalación de *MySQL* v. 5.0

Abrimos la carpeta *MySQL* 5.0 y procedemos a instalar *MySQL*-5.0.22-win32, luego nos aparecerá la pantalla de bienvenida, en la que damos clic en Next >.

Figura 1.1. Pantalla de bienvenida del asistente de instalación de MySQL

Se nos solicitará escoger el tipo de instalación que deseamos, en nuestro caso seleccionamos el modo *Custom* (personalizado) y damos un clic en Next >.

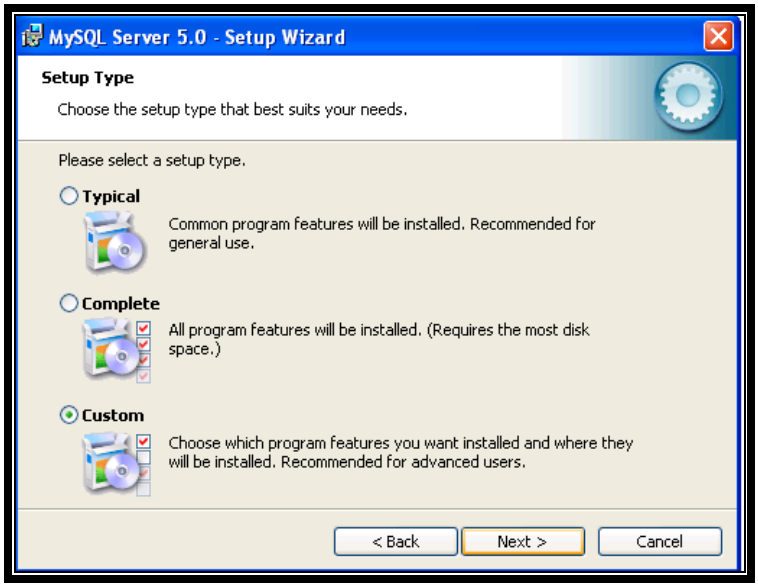


Figura 1.2. Tipo de instalación de MySQL

Verificar que en todas las opciones esté seleccionado "*This feature, and all subfeatures, will be installed on local hard drive*" y damos un clic en Next >.

Figura 1.3. Componentes de MySQL

Entonces aparece una pantalla, en la cual, se resume el tipo y directorio en el que se está realizando la instalación, luego de verificar que los datos sean correctos, damos un clic en Install y comenzará la instalación.

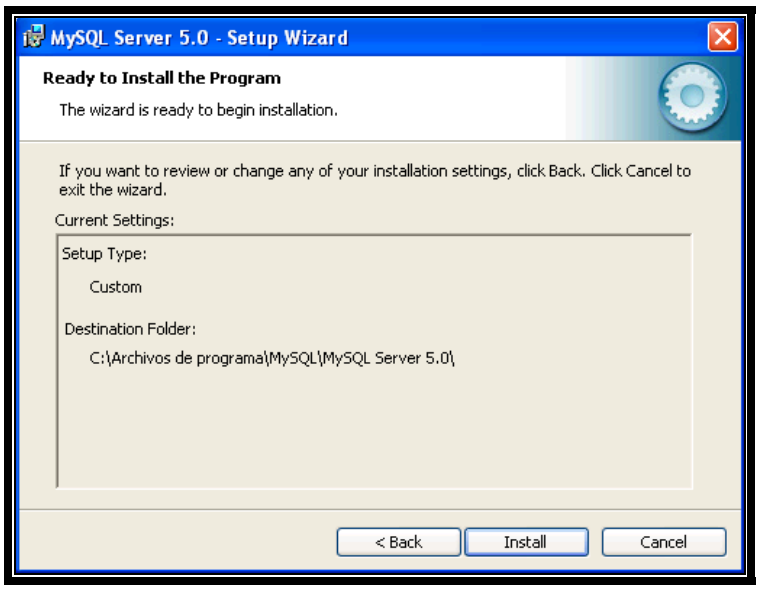


Figura 1.4. Resumen con el directorio y el tipo de instalación de MySQL

Seleccionamos la opción *Skip Sign-Up* y damos un clic en Next >. En caso de tener una cuenta en *MySQL*.com, utilizaremos la opción *Login to MySQL.com*, proporcionando el email y password respectivos. Si se desea crear una cuenta utilizaremos la opción *Create a new free MySQL.com account*. Estas últimas dos opciones se utilizan para cuentas en el sitio web de *MySQL*.

Figura 1.5. Creación o ingreso de cuenta para MySQL.com

Verificamos que la casilla de *Configure the MySQL Server now,* se encuentre marcada, damos un clic en 

y la instalación estará completa y dará paso entonces a la configuración.

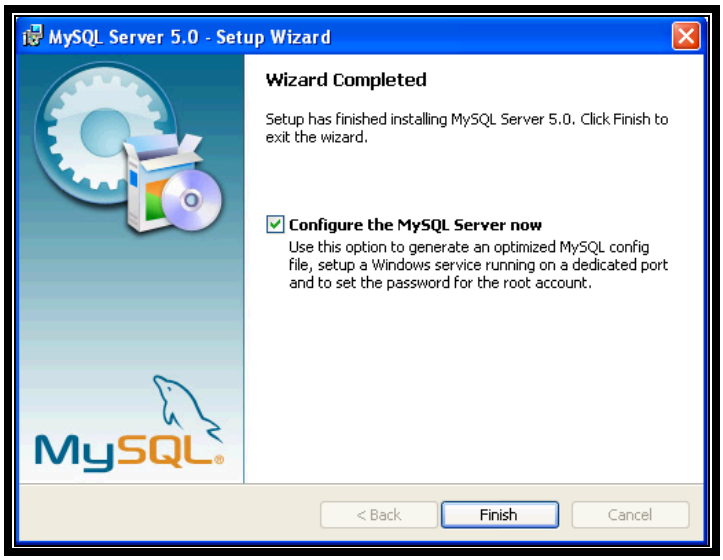


Figura 1.6. Finalización del asistente de instalación de MySQL

1.2 Configuración de MySQL

Una vez culminada la instalación se mostrará la pantalla de bienvenida del asistente de configuración de *MySQL Server*, en la cual damos clic en Next >.

Figura 1.7. Pantalla de bienvenida del asistente de configuración de MySQL

A continuación seleccionamos la configuración del servidor y elegimos *Detailed Configuration* (Configuración Detallada) y damos clic en Next >.

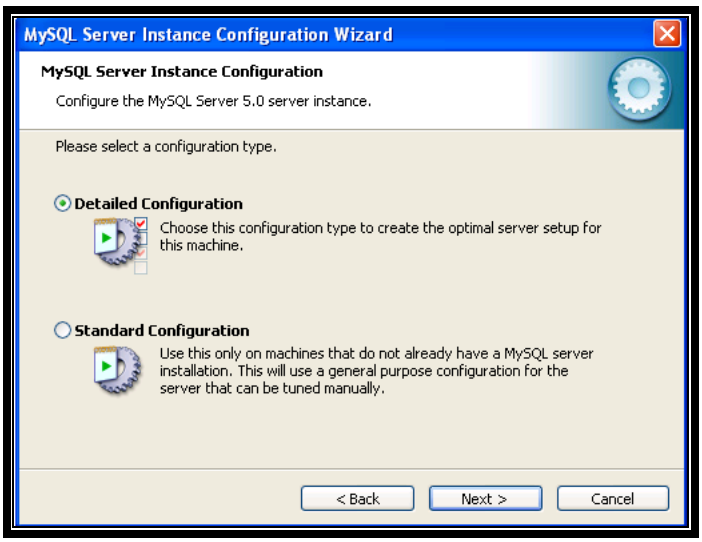


Figura 1.8. Tipo de configuración del servidor de MySQL

En nuestro caso utilizaremos este gestor de base de datos únicamente con fines didácticos, por lo tanto, escogemos la opción *Developer Machine* y damos clic en Next >. La opción *Server Machine* se utiliza para servidores de aplicaciones web y la opción *Dedicated MySQL Server Machine* se utiliza para servidores de bases de datos exclusivamente y utilizará toda la memoria disponible.

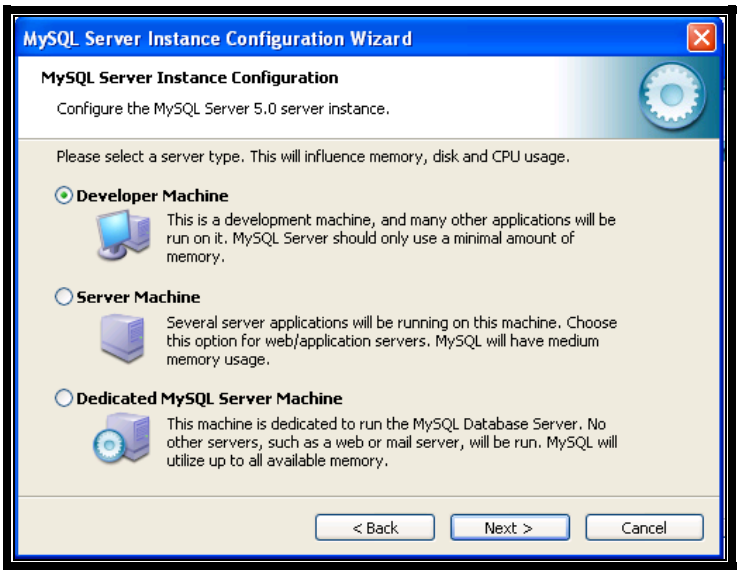


Figura 1.9. Tipo de servidor de MySQL

A continuación nos pide seleccionar el uso que se le va a dar a la base de datos, en nuestro caso seleccionamos la opción *Multifunctional Database* y damos clic en Next >.

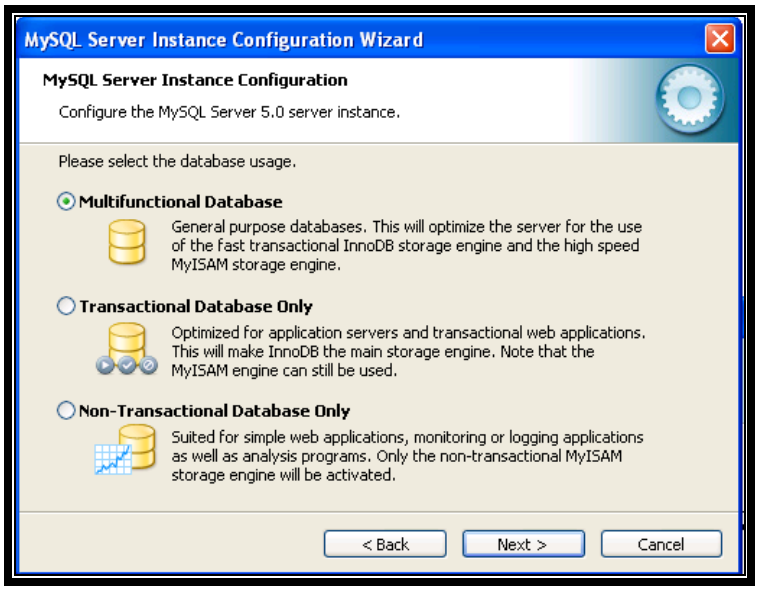


Figura 1.10. Uso de la base de datos en MySQL

Aparecerá una pantalla en la cual se solicitará la unidad y el directorio en el que se colocará la base de datos, luego escoger *MySQL Datafiles*, damos clic en Next >.

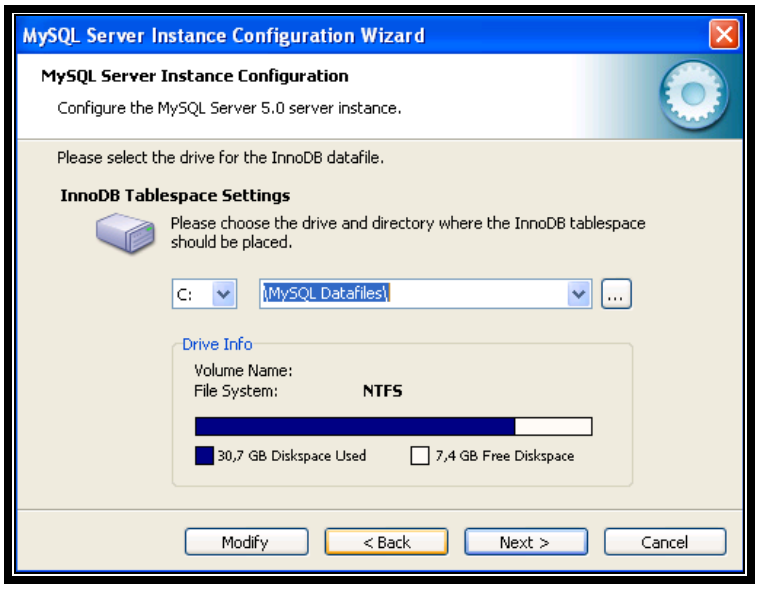


Figura 1.11. Path de la ubicación de la base de datos

No olvidar seleccionar el *path* donde se desea colocar la base de datos, de lo contrario, el servicio no iniciará.

En la siguiente pantalla se nos solicitará el número aproximado de conexiones concurrentes, seleccionamos la opción *Decisión Support* (*DSS)/OLAP* que esta marcada por default, ya que, en este caso vamos a realizar solo nuestra conexión, damos clic en Next >.

También se podría utilizar la opción *Manual Setting* y escoger el número de conexiones que deseamos.

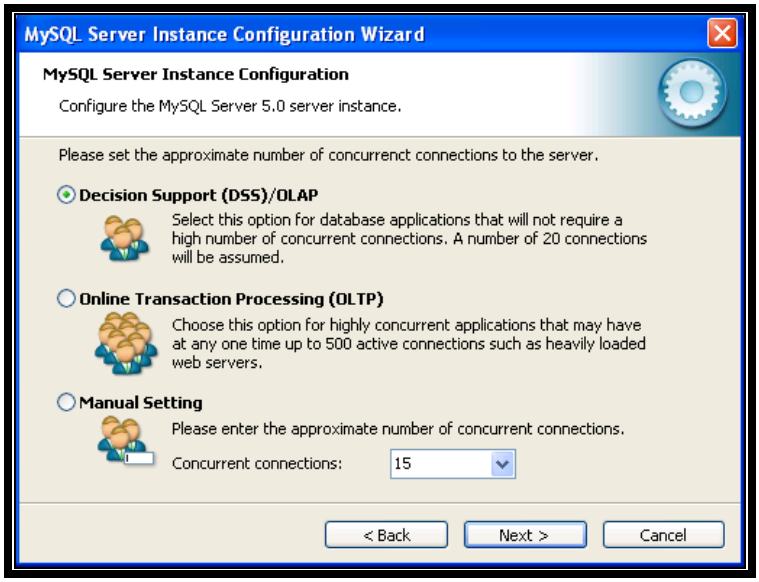


Figura 1.12. Selección del soporte para las conexiones concurrentes

En la pantalla de opciones de red, verificamos que estén marcadas las dos casillas, la primera sirve para habilitar las conexiones *TCP/IP* y escoger el puerto por el cual se realizarán; en caso de desactivarla se permitirá únicamente el acceso local, la segunda opción nos permite hacer que el servidor se asemeje más a un servidor tradicional de base de datos, damos clic en Next >.

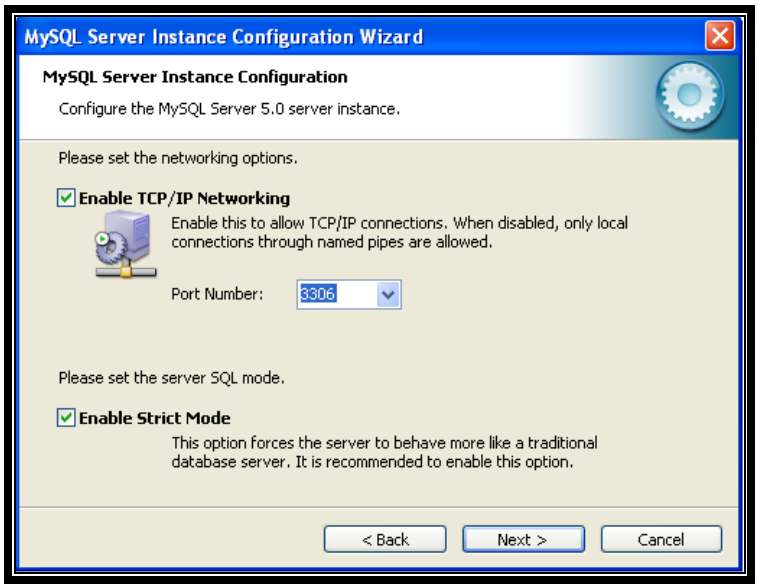


Figura 1.13. Configuración de las opciones de red del servidor

Seleccionamos el tipo de caracteres que aceptará la base de datos, para este caso seleccionamos la opción *Standard Character Set* y damos un clic en 

.

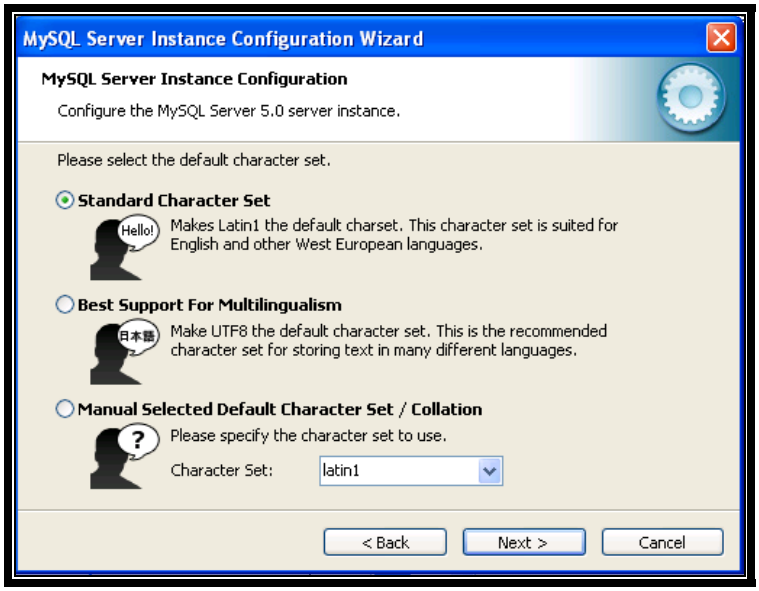


Figura 1.14. Selección de caracteres que soportará el servidor

Seleccionamos las opciones para *Windows*, verificamos que las dos casillas estén marcadas, en la primera opción seleccionamos el nombre del servicio; éste aparecerá en las herramientas administrativas de *Windows*; es recomendable que se marque la

casilla *Launch the MySQL Server automatically* para que el servicio inicie automáticamente, de lo contrario se tendrá *que* iniciar desde las herramientas administrativas de *Windows*. La segunda opción nos permitirá acceder a *MySQL* desde la línea de comando, damos clic en Next >.

Figura 1.15. Selección de opciones de Windows

A continuación nos aparecen las opciones de seguridad, la primera opción deberá estar marcada y pondremos como *password* la palabra *admin*, se puede también activar la opción para acceder como usuario *root* de manera remota.

En la segunda casilla, al marcarla creamos un usuario anónimo, esto en caso de que simplemente se conectarse a la base de datos. Al crear esta cuenta genera un problema, el cual consiste en que todas las cuentas que se creen en lo posterior ingresen simplemente tecleando el usuario, sin tener que teclear la contraseña. Damos clic en Next > para continuar.

Figura 1.16. Configuraciones de seguridad

Una vez realizadas todas las configuraciones damos clic en Execute para que estas sean aplicadas, en caso de querer hacer alguna modificación en las configuraciones podemos regresar a la pantalla anterior con la opción < Back.

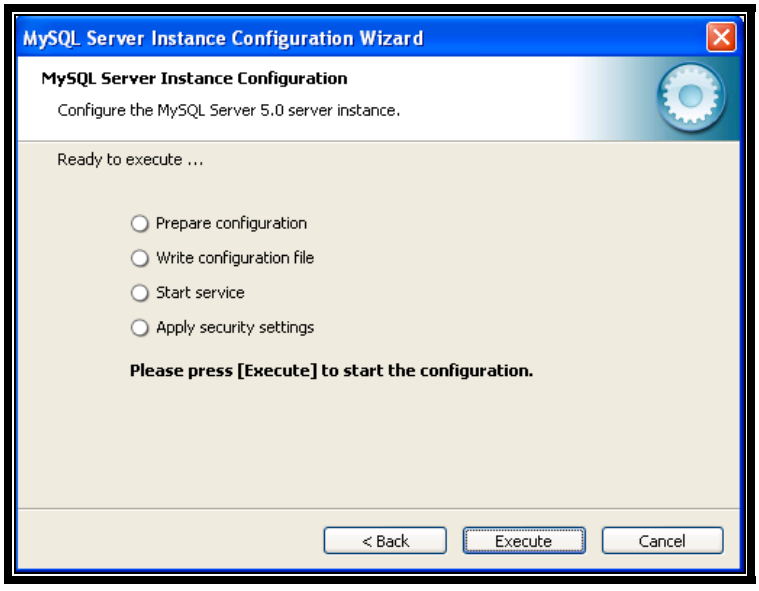


Figura 1.17. Pantalla de ejecución de la configuración

Una vez ejecutados los cambios se deberá mostrar la siguiente pantalla, verificar que al lado de cada opción aparezca un visto y luego dar clic en Finish.

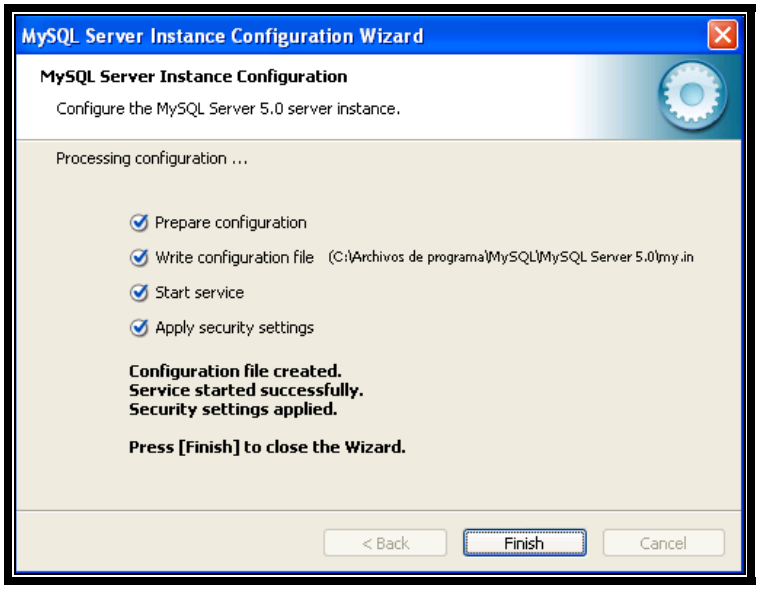


Figura 1.18. Finalización del asistente de configuración

Otra manera de acceder a las configuraciones para poder cambiarlas es desde Inicio>Todos los programas>*MySQL*>*MySQL Server 5.0*>*MySQL Server Instante Config Wizard*, aparecerá entonces la pantalla de bienvenida del asistente de configuración, damos clic en Next > y aparecerá una pantalla, en la cual, nos da la opción de reconfigurar o de eliminar, escogemos entonces la opción que ejecutaremos y damos clic en Next >.

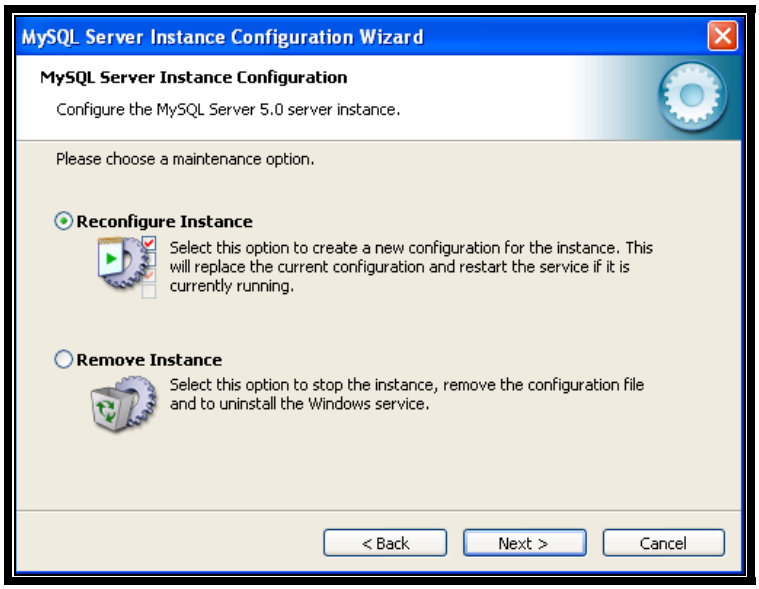


Figura 1.19. Opciones de mantenimiento de configuración

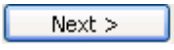
Para realizar cambios revisar el instructivo en la parte de configuraciones.

1.3 Instalación de Herramientas

Para instalar las herramientas utilizaremos el directorio *MySQL* 5.0, Tools.

1.3.1 MySQL Administrator

Para proceder a instalar el Administrador de *MYSQL* vamos a la ruta antes nombrada e instalamos *MySQL*-administrator-1.1.9-win.

Primero, aparece la pantalla de bienvenida del asistente de instalación del administrador en la cual procederemos a dar clic en 

.

Figura 1.20. Pantalla de bienvenida del asistente de instalación del administrador

Luego escogemos el directorio donde queremos instalar, se puede cambiar con Change..., luego damos en Next >.

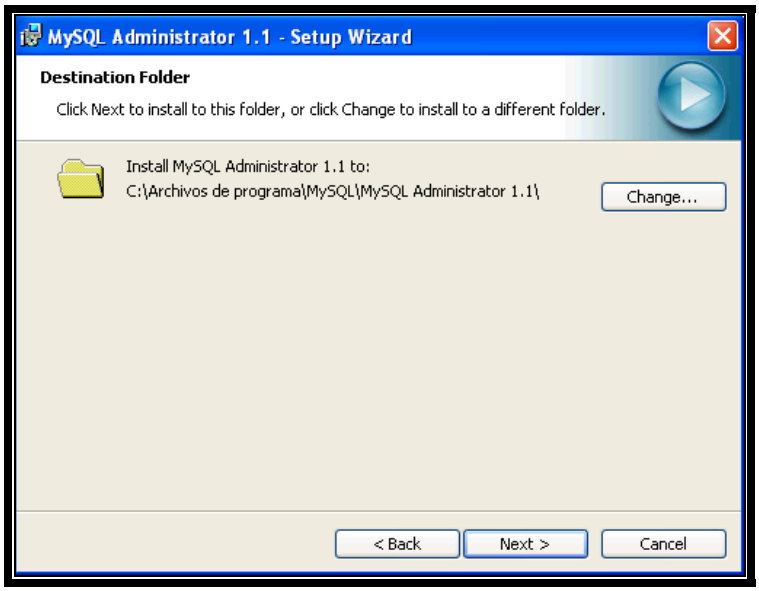
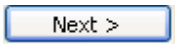


Figura 1.21. Ubicación de la carpeta de destino del administrador

En la pantalla de tipo de instalación escogemos *Custom* y luego Next >.

Figura 1.22. Tipo de instalación del administrador

Aparece entonces la pantalla con las características que se desean instalar, verificamos en las dos opciones que esté seleccionado "*This feature, and all subfeatures, will be installed on local hard drive"* y damos un clic en 

. Se puede cambiar también el directorio de instalación pero es recomendable dejarlo como está.

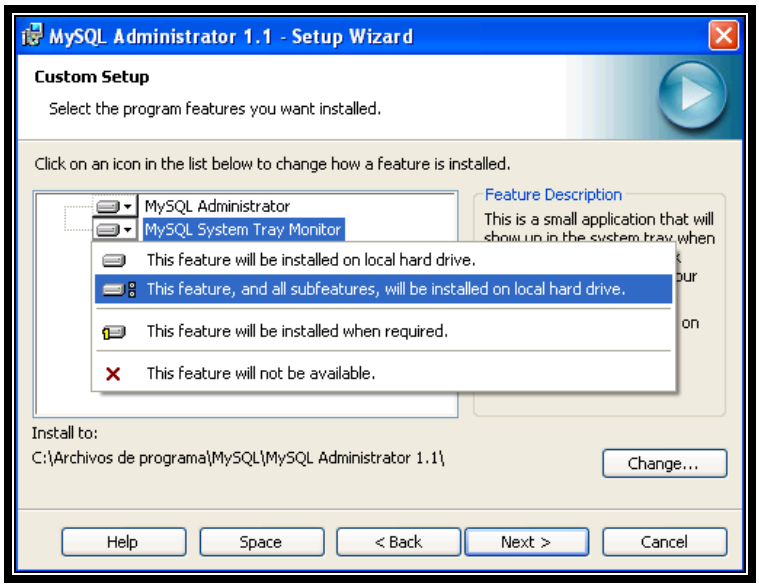


Figura 1.23. Selección de características de instalación del administrador

Aparecerá una pantalla en donde se muestra un resumen del tipo de instalación realizada y el directorio en el cual se hará. Damos clic en Install.

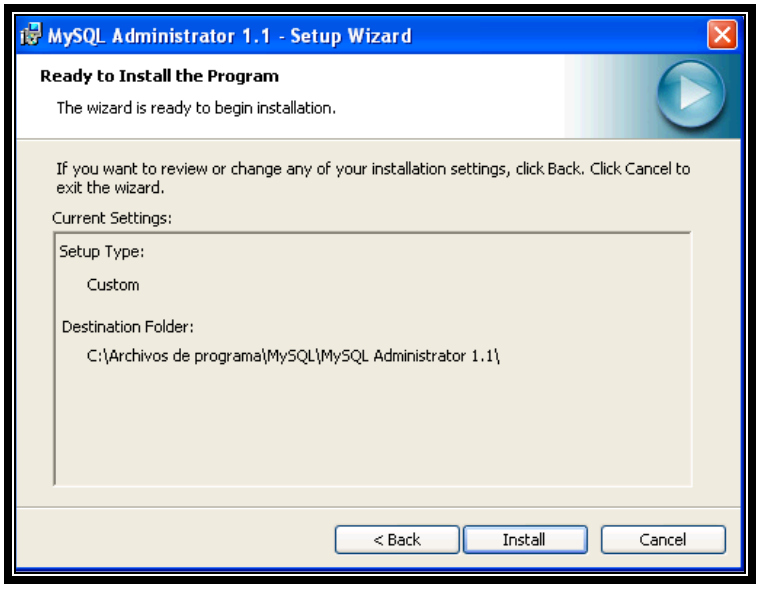


Figura 1.24. Resumen de tipo y ubicación de instalación del administrador

Una vez terminada la instalación, aparecerá una pantalla en la cual se da por terminado el asistente de instalación del administrador. Damos clic en Finish.

Figura 1.25. Pantalla de finalización del asistente de instalación del administrador

1.3.2 MySQL Migration Toolkit

Esta es una herramienta para gestionar migraciones y respaldos. Para instalar la herramienta de migraciones vamos a *MySQL* 5.0, Tools e instalamos *MySQL*-migration-toolkit-1.0.25-win32.

Primero, aparece la pantalla de bienvenida del asistente de instalación de las herramientas de migraciones y respaldos, en la cual procederemos a dar clic en Next >.

Figura 1.26. Pantalla de bienvenida de instalación de las herramientas de migración

En la siguiente pantalla seleccionamos el lugar donde deseamos instalar la herramienta de migraciones y respaldos, si deseamos cambiar el lugar damos un clic en Change..., caso contrario en Next >.

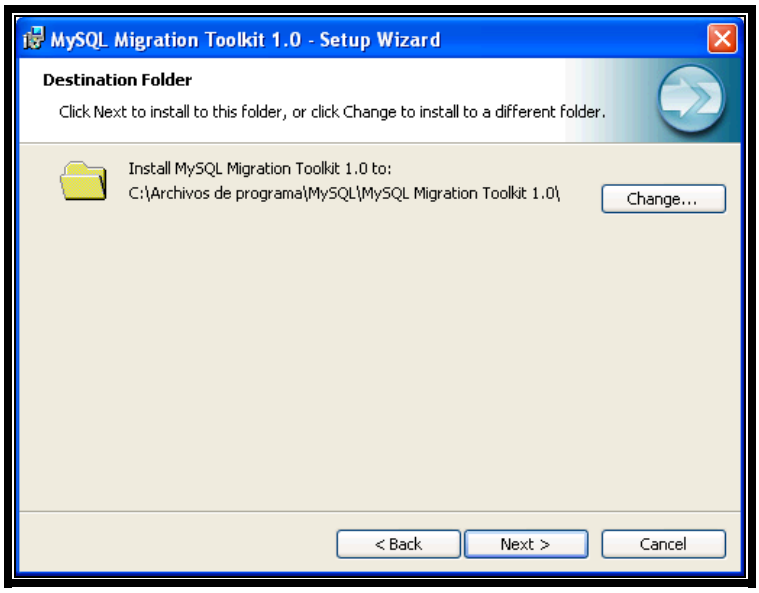
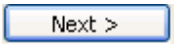


Figura 1.27. Ubicación de la carpeta de destino de la herramienta de migración

Seleccionamos el tipo de instalación que deseamos, en nuestro caso seleccionamos el tipo *Custom* y a continuación damos clic en 

.

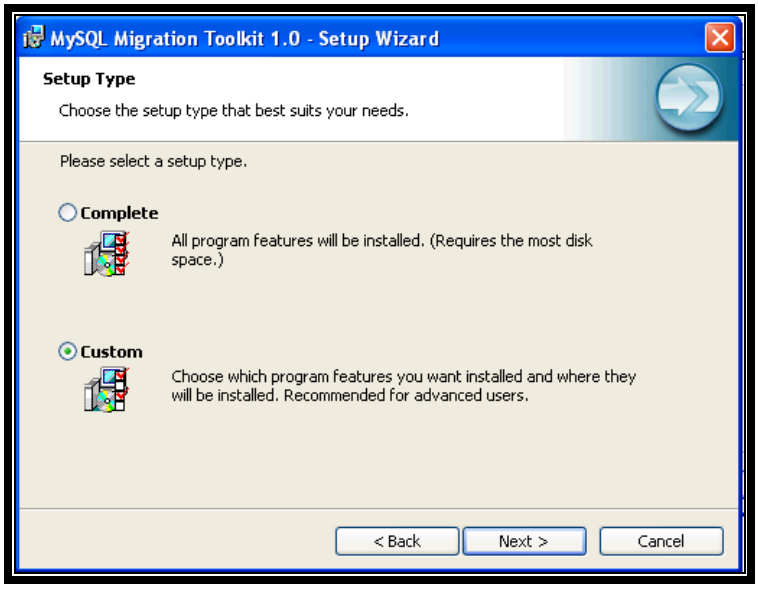


Figura 1.28. Tipo de instalación de la herramienta de migración

Aparece entonces la pantalla con las características que se desean instalar, verificamos en que esté seleccionado "*This feature, and all subfeatures, will be installed*

on local hard drive" y damos un clic en Next >. Se puede cambiar también el directorio de instalación pero es recomendable dejarlo como está.

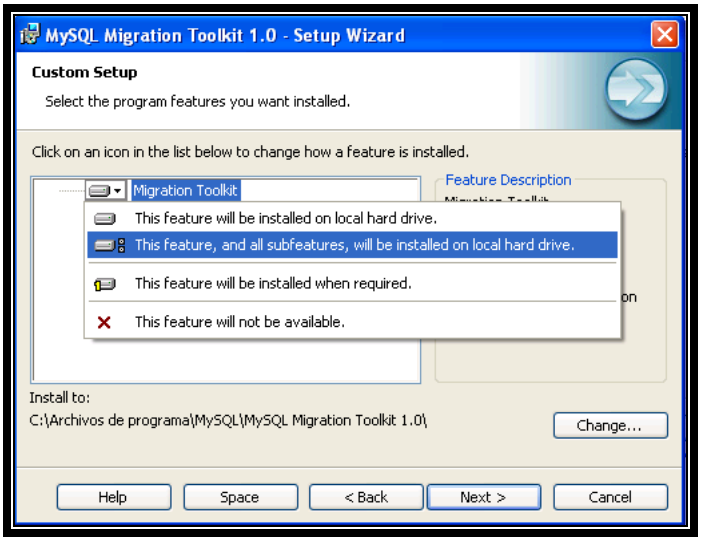


Figura 1.29. Selección de características de instalación de la herramienta de migración

Después aparecerá una pantalla en donde se muestra un resumen del tipo de instalación realizada y el directorio en el cual se hará. Damos clic en Install.

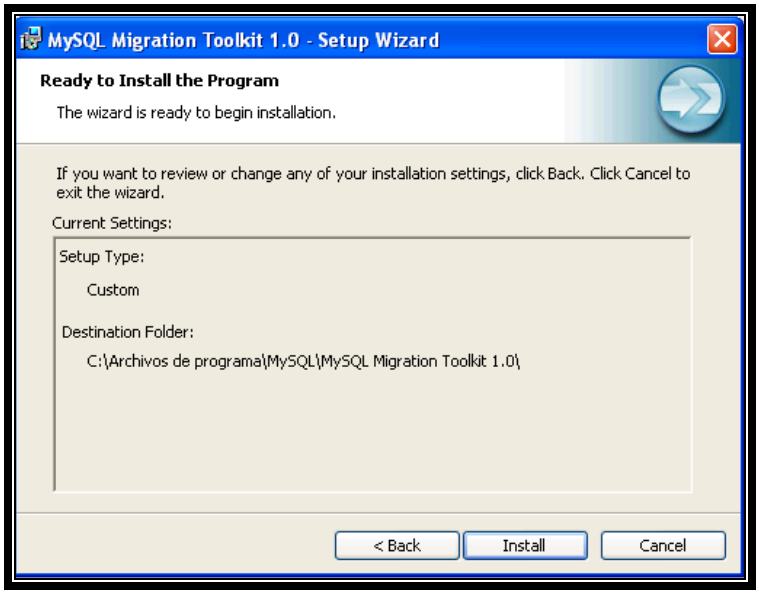


Figura 1.30. Resumen de tipo y ubicación de instalación de la herramienta de migración

Una vez terminada la instalación, aparecerá una pantalla en la cual se da por terminado el asistente de instalación de las herramientas de migración y respaldos. Damos clic en Finish.

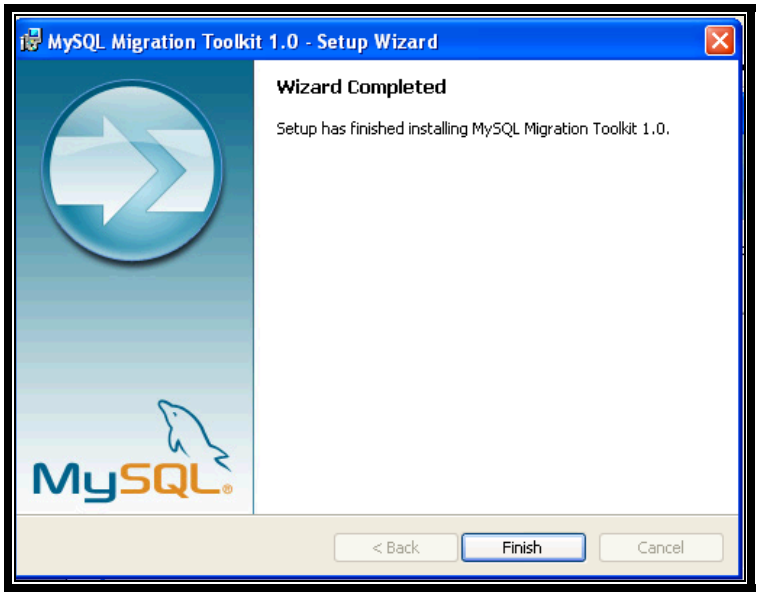


Figura 1.31. Pantalla de finalización del asistente de instalación de la herramienta de migración

1.3.3 MySQL Query Browser

Esta herramienta nos permitirá realizar consultas y manipular la base de datos desde la línea de comando desde una interfaz muy amigable. Para instalar la herramienta de consultas vamos a *MySQL* 5.0 >Tools e instalamos *MySQL*-query-browser-1.1.20-win.

Primero, aparece la pantalla de bienvenida del asistente de instalación de la herramienta de consultas, en la cual procederemos a dar clic en Next >.

Figura 1.32. Pantalla de bienvenida de instalación de las herramientas de consultas

En la siguiente pantalla seleccionamos el lugar donde deseamos instalar la herramienta de consultas, si deseamos cambiar el lugar damos un clic en Change..., caso contrario en Next >.

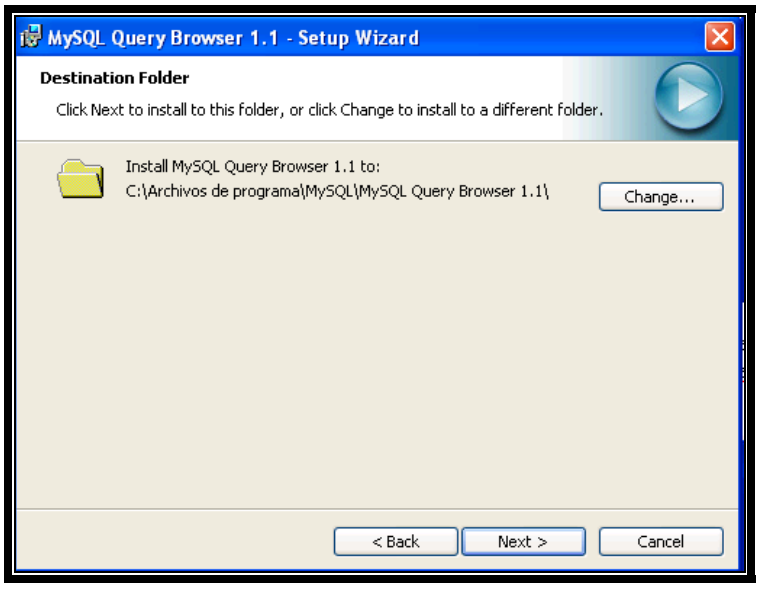


Figura 1.33. Ubicación de la carpeta de destino de la herramienta de consultas

Seleccionamos el tipo de instalación que deseamos, en nuestro caso seleccionamos el tipo *Custom* y a continuación damos clic en Next >.

Figura 1.34. Tipo de instalación de la herramienta de consultas

Aparece entonces la pantalla con las características que se desean instalar, verificamos en que esté seleccionado "*This feature, and all subfeatures, will be installed on local hard drive"* y damos un clic en Next >. Se puede cambiar también el directorio de instalación pero es recomendable dejarlo como está.

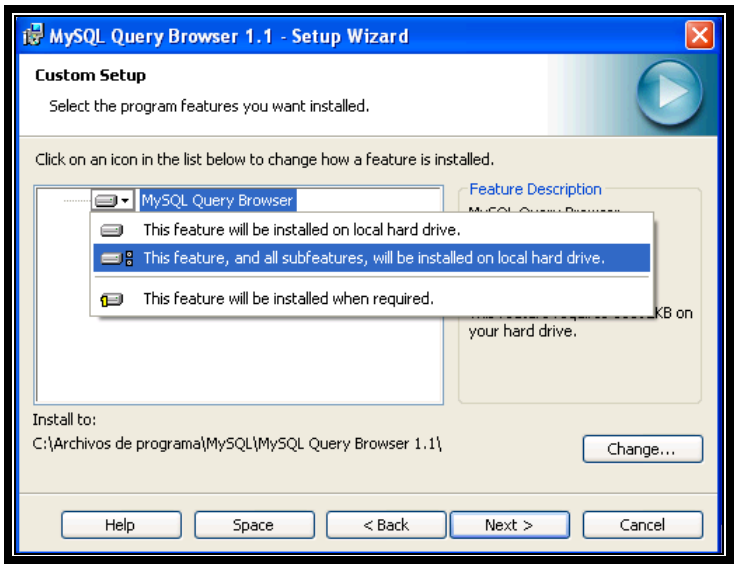


Figura 1.35. Selección de características de instalación de la herramienta de consulta

Después aparecerá una pantalla en donde se muestra un resumen del tipo de instalación realizada y el directorio en el cual se hará. Damos clic en Install.

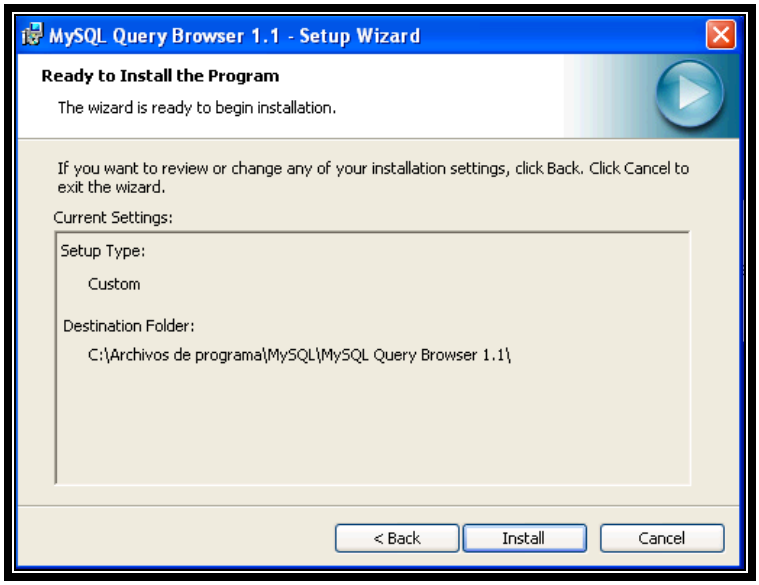


Figura 1.36. Resumen de tipo y ubicación de instalación de la herramienta de consultas

Una vez terminada la instalación, aparecerá una pantalla en la cual se da por terminado el asistente de instalación de la herramienta consultas. Damos clic en Finish.

Figura 1.37. Pantalla de finalización del asistente de instalación de la herramienta de consultas

1.3.4 MySQL System Try Monitor

Para iniciarlo Vamos a Inicio > Todos los programas > *MySQL System Tray Monitor*. Es un monitor del sistema que, como se mencionó anteriormente se muestra en el área de notificaciones**.**

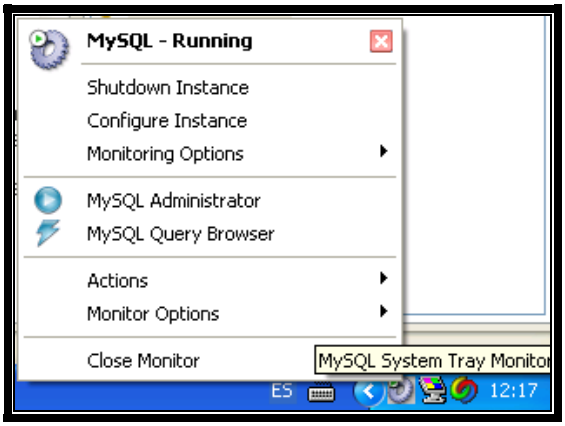


Figura 1.38. Monitor del sistema en la barra de notificación

Para ver sus opciones se da clic derecho sobre el icono, lo cual despliega el menú contextual. La primera línea muestra el estado del servidor, se puede detener el servicio, configurar una base de datos, acceder al Administrador o al Navegador de consultas y modificar otras opciones. Se puede cerrar el monitor con *Close Monitor.*

1.4 Conexiones y Desconexiones de MySQL

1.4.1 Conexión con MySQL Administrator

Como su nombre lo indica, posee un conjunto de herramientas para poder hacer la administración sobre el gestor de base de datos. Para ingresar al administrador tenemos que ir al menú Inicio > Todos los programas > *MySQL* > *MySQL Administrador*, entonces aparece una pantalla en la cual se nos solicita ingresar usuario y contraseña, usaremos entonces los que ingresamos en el momento de la configuración. En este caso el usuario será *root* y la contraseña *admin*.

Adicionalmente en *server host* se utilizará la palabra *localhost* o la dirección *IP* del mismo que en este caso es 127.0.0.1.

Se debe tener en cuenta el puerto que se utilizó en la configuración, en este caso es el 3306, luego damos clic en OK y aparecerá la pantalla del administrador.

Figura 1.39. Pantalla de conexión del administrador MySQL

En la pantalla del administrador se muestran varias herramientas con diferentes funciones de las cuales hablaremos brevemente.

Server Information Proporciona información de la conexión, servidor y cliente. Muestra si el servidor se encuentra activo o inactivo, el usuario con el que estamos conectados, el puerto y el nombre del servidor.

Service Control En éste se puede detener o iniciar el servicio manualmente, además nos permite configurar opciones sobre como arrancar el servicio automáticamente, y renombrarlo si es que se desea el path de los archivos del servidor y las características del servidor

Permite cambiar parámetros de seguridad, red, parámetros generales, rendimiento, etc.

Para crear o editar usuarios, privilegios de las bases de datos, y recursos de la cuenta del usuario anónimo.

Muestra todos los intentos de conexión y todos los que están conectados actualmente.

Muestra información de la conexión de los clientes, el tráfico, el número de consultas SQL, variables del sistema, variables del servidor y el uso de la memoria.

Muestra los errores de registros.

Muestra información de la replicación del servidor en caso de tenerla.

Permite realizar respaldos, seleccionando la información que se va a guardar y programar para realizar automáticamente.

Permite restaurar los respaldos que se hayan realizado.

Contiene todas las bases de datos, de las cuales muestra las tablas, índices, crear, editar, o eliminar vistas y procedimientos.

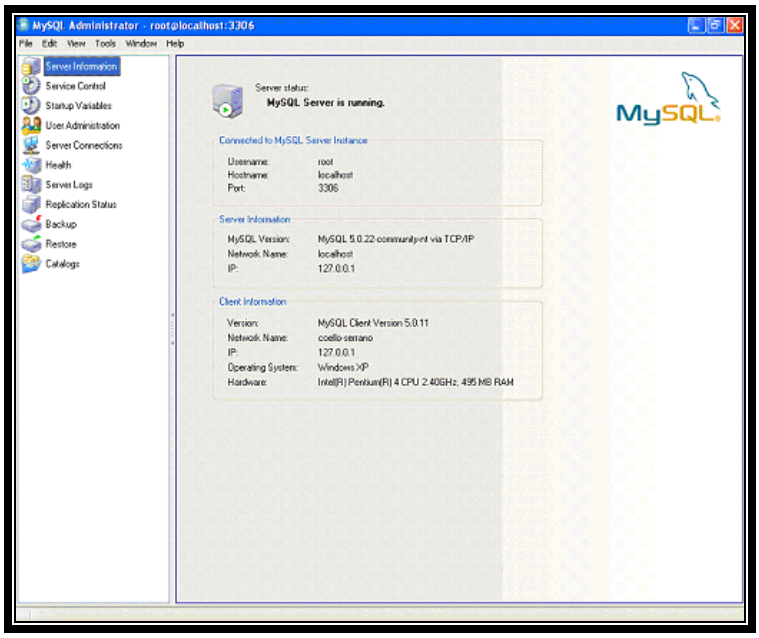
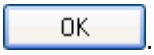


Figura 1.40. Pantalla principal del administrador MySQL

También desde la opción *Tools* de la barra de herramientas se puede acceder al *Query Browser* (Navegador de consultas), *Command Line Client* (Línea de comando en DOS) *System Tray Monitor* que se mostrará en el área de notificaciones de la barra de tareas.

1.4.2 Conexión con MySQL Query Browser

Para iniciarlo Vamos a Inicio > Todos los programas > *MySQL Query Browser*, entonces aparecerá la pantalla de petición de conexión, en esta pondremos en *Server Host* 127.0.0.1 o *localhost*, *Username root, Password admin, Default Schema* compania que es en este caso el que estaremos utilizando para nuestras prácticas y luego 

.

Si es que se pone en *Default Schema* un nombre que no exista éste se creará automáticamente.

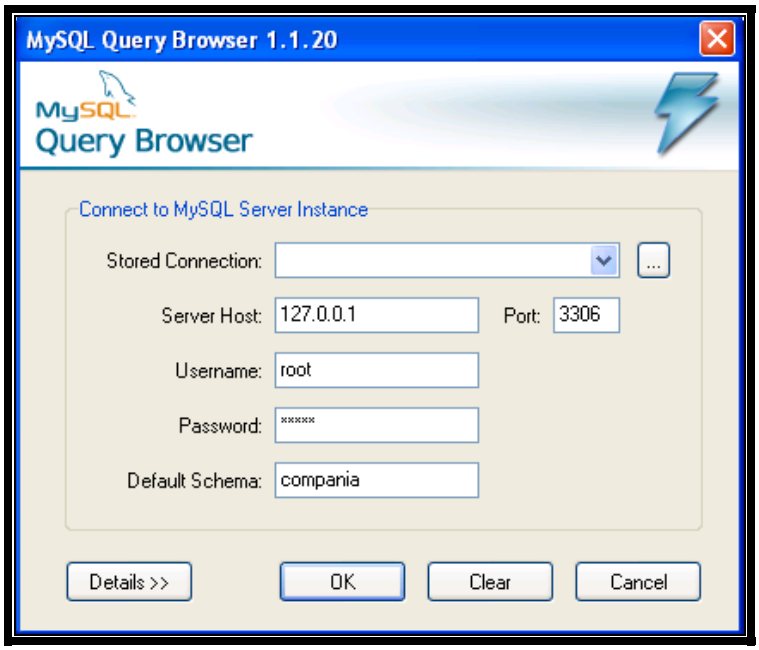


Figura 1.41. Pantalla de conexión de la herramienta de consultas

También se puede dejar en blanco *Default Schema*, en este caso se mostrará una ventana en la cual pide especificar una base por defecto si es que no se quiere especificar se seleccionará la opción 

y entonces aparece la ventana del *Query Browser*.

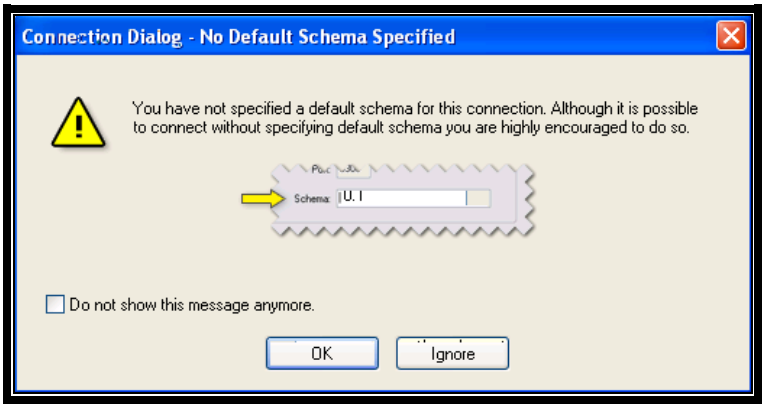


Figura 1.42. Pantalla de error de esquema no definido por defecto

El botón 

muestra la consulta anterior, 

la siguiente consulta realizada, 

actualiza lo que se esta mostrando, hay un campo en el que se escribe la consulta a realizar o se muestra la consulta que se va a realizar de acuerdo a lo que se vaya señalando en el campo *Scehmata*. Las palabras reservadas como *SELECT, FROM*, etc., se mostrarán con un color azul.

Para efectuar una consulta se procederá con el botón 

y se podrá detener con *Stop*

Figura 1.43. Campo de manipulación de MySQL

Hay un campo en el cual se muestran los resultados en distintas pestañas.

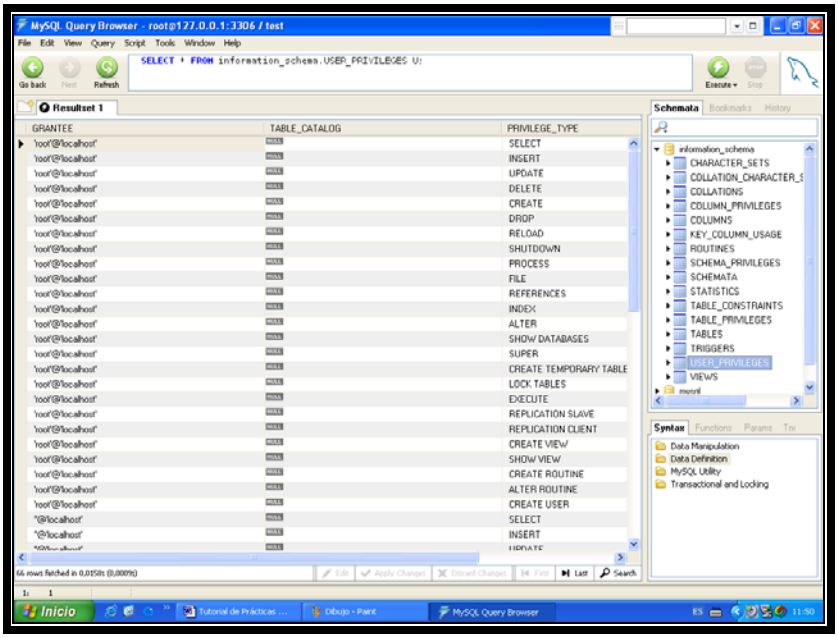


Figura 1.44. Campo de manipulación de MySQL y resultados

A la derecha de *Schemdata y de Bookmarks* se encuentra *History*. En esta se muestra el historial de consultas realizadas.

En la parte inferior se pueden encontrar pestañas de *Syntax, Functions y Params* con ayuda y su uso.
La ventana del *Query Browser* tiene otras utilidades las cuales serán revisadas posteriormente.

1.5 Lenguaje SQL

El lenguaje *SQL* es un lenguaje orientado a la manipulación de las bases de datos, brindando una interface sencilla y amigable para el usuario, para poder utilizarlo se debe tener conocimientos sobre la base de datos a manipular y sobre las instrucciones propias del lenguaje. Este está compuesto por comandos, cláusulas, operadores y funciones.

1.5.1 Comandos para la Manipulación de Bases de Datos

Create: Sirve para crear tablas o un nuevo esquema de base de datos.

Drop: Sirve para eliminar columnas, tablas o bases de datos.

Alter: Se utiliza para modificar la estructura de una tabla, éste puede ser combinado con otros comandos como: *ADD, DROP, ADD PRIMARY KEY(), ADD FOREIGN KEY*(), entre otros.

Describe: Muestra la estructura de las tablas de la base de datos.

Show Tables: Muestra las tablas que pertenecen a la base de datos.

1.5.2 Comandos para la Manipulación de Información

***Select*:** Es utilizado para realizar consultas a la base de datos que cumplan con alguna condición.

***Insert*:** Es utilizado para la inserción de información en la base de datos.

Update: Se utiliza para la modificación o actualización de información en la base de datos.

Delete: Elimina registros de la base de datos.

1.5.3 Comandos para el control de Acceso

Grant: Asigna privilegios de acceso y manipulación a los usuarios.

Revoke: Quita privilegios de acceso o manipulación a los usuarios.

1.5.4 Cláusulas

From: Se utiliza para especificar la tabla o tablas de las cuales se va a seleccionar los registros.

Where: Es utilizado para determinar las condiciones que debe cumplir la información seleccionada.

Group By: Sirve para agrupar los registros seleccionados en grupos específicos.

Having: Expresa la condición que debe satisfacer cada grupo.

Order By: Sirve para ordenar registros de acuerdo a algún campo específico. Esta cláusula puede realizarse de forma ascendente o descendente.

1.5.5 Operadores Lógicos

And: Es el equivalente a “y”, devuelve valor de verdad solo si todas las condiciones son verdaderas.

Or: Es el equivalente a “o”, devuelve valor de verdad si al menos una de las condiciones es verdadera.

Not: Es el equivalente a negación, devuelve valor verdadero cuando la condición es falsa y viceversa.

1.5.6 Operadores de Comparación

<: Menor que.

>: Mayor que.

<>: Diferente.

<=: Menor o igual que.

>=: Mayor o igual que.

=: Igual que.

Between: Se utiliza para especificar un rango de valores.

Like: Se utiliza para la comparación de un modelo

In: Se utiliza para especificar registros de una base de datos.

1.5.7 Funciones de Columna

Avg: Devuelve como resultado el promedio de un campo determinado.

Count: Devuelve como resultado el número de registros que cumplen una condición.

Sum: Devuelve como resultado la suma de los valores de un campo determinado.

Max: Devuelve el valor más alto de un campo determinado.

Min: Devuelve el valor más bajo de un campo determinado.

1.6 Conclusiones

Siguiendo todos los pasos desarrollados anteriormente, se podrá lograr la correcta instalación del Gestor de Bases de Datos *MySQL*, con lo que prácticamente se garantiza el correcto funcionamiento de este gestor. Adicionalmente en este capítulo se explica detalladamente la configuración y las herramientas que proporciona el mencionado gestor.

Un aporte adicional de este capítulo es la explicación de las principales cláusulas y comandos que se utilizan en el lenguaje *SQL*, los cuales a nuestro parecer son fundamentales en el desarrollo de los siguientes capítulos de este tutorial.

CAPITULO 2

ADMINISTRACIÓN DE USUARIOS Y BASES DE DATOS

INTRODUCCIÓN

En este capítulo trataremos sobre la administración de bases de datos y los usuarios involucrados en la gestión de base de datos, todo esto realizado paso a paso por medio de la ventana de comandos (*DOS*), utilizando instrucciones *SQL*, se desarrollarán temas como: creación de usuarios, privilegios que pueden ser asignados a los mismos, creación y manipulación de bases de datos, tablas y registros. También como punto final de este capítulo se propondrá el desarrollo de un ejercicio práctico, en el cual, se deberá aplicar todo lo aprendido.

2.1 Cuentas de Usuario

2.1.1 Creación de Usuarios

Para crear cuentas de usuario en *MySQL* se necesita tener permisos de creación de usuarios. Una vez que se tienen los permisos procedemos a crear usuarios por medio del comando *CREATE USER*. A cada cuenta se le puede asignar una contraseña por medio de la cláusula *IDENTIFIED BY,* si se desea que la clave se grabe en texto plano, no se utilizará la palabra *PASSWORD,* caso contrario la clave se grabará cifrada con el valor *hash* que es devuelto por la función *PASSWORD*( ).

Por ejemplo vamos a crear el usuario alumno con la contraseña alumnouda, para lo cual utilizamos la siguiente instrucción:

```
CREATE USER alumno
IDENTIFIED BY ‘alumnouda’;
```

El usuario alumno ha sido creado, sin embargo puede conectarse al servidor, pero hace falta asignarle los diferentes privilegios, para que pueda realizar cualquier tipo de tarea.

2.1.2 Borrar Usuarios

De igual forma que para crear usuarios, necesitamos tener los permisos pertinentes para poder borrarlos, luego procedemos a borrar el usuario deseado por medio del comando *DROP USER.*

Para borrar el usuario que creamos anteriormente usaríamos la siguiente sentencia:

```
DROP USER alumno;
```

2.1.3 Privilegios de los Usuarios

Para ver los privilegios asignados a una cuenta utilizamos el comando *SHOW GRANTS.* Para consultar los privilegios que tiene en este momento el usuario alumno usamos la siguiente sentencia:

```
SHOW GRANTS FOR alumno;
```

Aparecerá una tabla en la cual se mostrará el usuario alumno, el *hash* del *password* y la lista de privilegios que posee, podemos observar que solo tiene la cuenta activada, pero no tiene ningún privilegio asignado aún.

Para asignar privilegios a una cuenta usamos el comando *GRANT*, el cual permite asignar a una cuenta diferentes niveles de permisos, siendo los siguientes:

- **Globales:** Otorga los privilegios a un usuario sobre todo el servidor, esto se realiza por medio del comando *GRANT* privilegios *ON* *.* *TO* nombre_usuario;, o para borrarlos el comando *REVOKE* privilegios *ON* *.* *FROM* nombre_usuario;.

- **Base de Datos:** Otorga los privilegios a un usuario de manera que solo pueda manipular una o varias bases de datos, por medio del comando *GRANT* privilegios *ON* nombre_base_de_datos.* *TO* nombre_usuario;, o para borrarlos el comando *REVOKE* privilegios *ON* nombre_base_de_datos.* *FROM* nombre_usuario;.

- **Tabla:** Otorga privilegios a todas las columnas de una tabla específica por medio del comando *GRANT* privilegios *ON* base_datos.nombre_tabla *TO* nombre_usuario y para borrarlos *REVOKE* privilegios *ON* base_datos.nombre_tabla *FROM* nombre_usuario.

- **Columna:** Otorga privilegios a columnas específicas de una tabla dada, por medio del comando GRANT **privilegios** (columna) ON base_datos.tabla TO usuario y para borrarlos *REVOKE* **privilegios** (columna) ON base_datos.tabla FROM usuario.

Algunos de los privilegios que pueden ser utilizados son:

- ***SELECT:*** Para realizar selecciones.

- ***INSERT:*** Para ingresar datos.

- ***UPDATE:*** Para actualizar o cambiar datos.

- ***DELETE:*** Para borrar datos de las tablas.

- ***INDEX:*** Para crear o borrar índices.

- ***ALTER:*** Para modificar las tablas.

- ***CREATE:*** Para la creación de bases de datos, tablas e índices.

- ***DROP:*** Para borrar bases de datos y tablas.

- ***GRANT:*** Para habilitar a un usuario el manejo de privilegios de otros usuarios, además de poder crear y borrar usuarios.

- ***REFERENCES:*** Para bases de datos y tablas, para el uso de referencias.

- ***RELOAD:*** Para recargar y vaciar los parámetros del servidor.

- ***SHUTDOWN:*** Para parar el servidor***.***

- ***PROCESS:*** Para habilitar procesos del servidor o matarlos.

- ***FILE:*** Para trabajar con archivos, por ejemplo sacando información en archivos de salida.

- ***ALL ON *.*:*** Otorga todos los privilegios en la base de datos.

Para quitar todos los permisos otorgados a un usuario, en lugar de quitarlos uno por uno, utilizamos el comando *REVOKE* privilegios *PRIVILEGES, GRANT OPTION FROM* nombre_usuario;.

2.1.4 Renombrar Usuarios.

Si se desea renombrar cuentas de usuario se necesita tener permisos pertinentes, para cambiar el nombre de un usuario utilizamos el comando *RENAME USER* nombre_actual_usuario *TO* nuevo_nombre;.

2.1.5 Contraseñas

Para cambiar contraseñas o asignar contraseñas se necesita tener los permisos necesarios, una vez que se los tiene usamos el comando *SET PASSWORD FOR* nombre_usuario = *PASSWORD*(‘nueva_contraseña’);.

2.2 Creación de la Base de Datos

Para el desarrollo de este tutorial nos vamos a basar en el modelo entidad relación del libro la página nro. 55 del Libro "*Fundamentals of Database Systems*", utilizando como nombre de la base de datos la palabra compania en lugar de compañía, debido a que el gestor de bases de datos presenta problemas con caracteres especiales como la ñ y tildes en la línea de comandos.

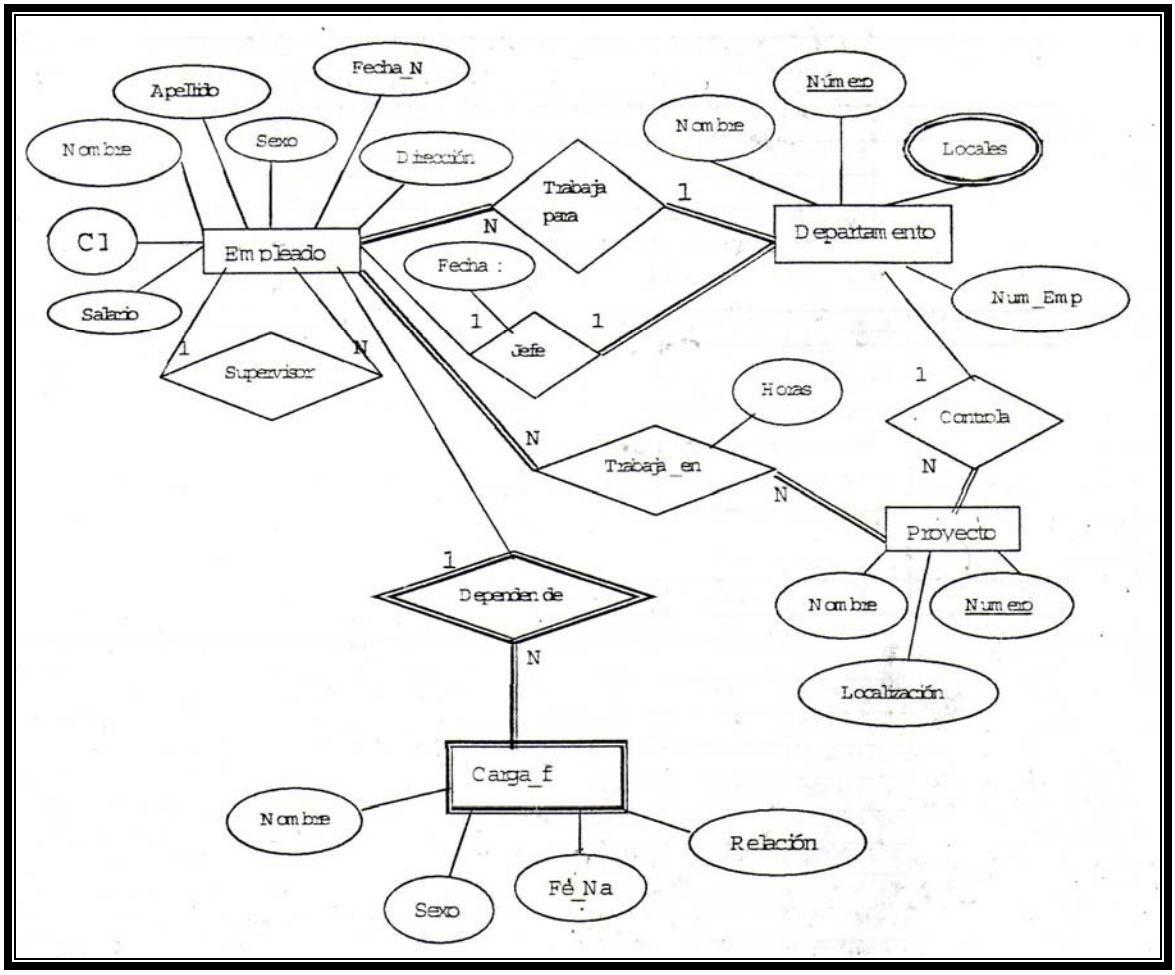


Figura 2.1. Modelo Entidad-Relación de la base de datos compañía
(del Libro "*Fundamentals of Database Systems*)

La creación de la base de datos se puede hacer de diferentes formas, una de ellas es por medio de la línea de comando en *DOS*, en el menú inicio > todos los programas >

accesorios > símbolo del sistema, ejecutamos la siguiente línea de comando para realizar la conexión a *MySQL*:

MySQL –u root –p

Y a continuación se nos solicitará una contraseña, que en nuestro caso es *admin*.

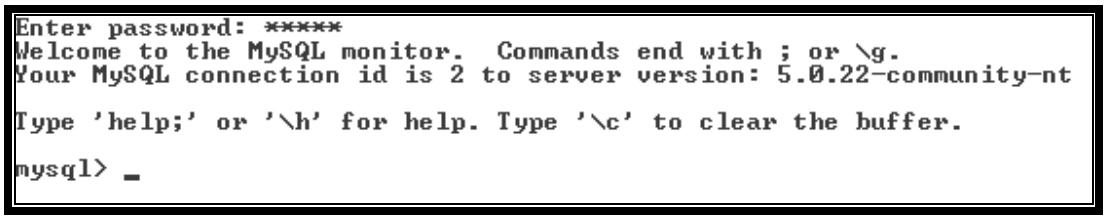


Figura 2.2. Solicitud de contraseña en línea de comando

Para crear la base de datos, ejecutamos la siguiente línea de comando:

CREATE DATABASE nombre_base_de_datos;

En nuestro caso el nombre de la base de datos será compania:

Figura 2.3. Creación de la base de datos compania en línea de comando

Una vez creada la base de datos, debemos ingresar a la misma para poder manipularla, ya sea creando tablas, ingresando datos, etc.

Para ingresar a la base de datos utilizamos la siguiente instrucción:

USE compania;

Si la base de datos fue cambiada, se mostrará el mensaje *Database changed*, caso contrario nos aparecerá un mensaje de error de que la base de datos no existe. Ahora podemos crear tablas dentro de esta base de datos.

La base de datos se crea solo una vez pero cada vez que iniciemos una sesión en *MySQL* es necesario seleccionarla para hacer cambios dentro de la misma con la instrucción anterior.

2.3 Relaciones entre tablas

Existen tres tipos de relaciones entre tablas, a continuación hablaremos de ellos:

2.3.1 Relación N:M

Cuando existe una relación de este tipo (varios a varios), se debe crear una nueva tabla, la cual estará compuesta por los atributos de la relación en caso de tenerlos, además de las llaves primarias de las entidades que participan en la relación.

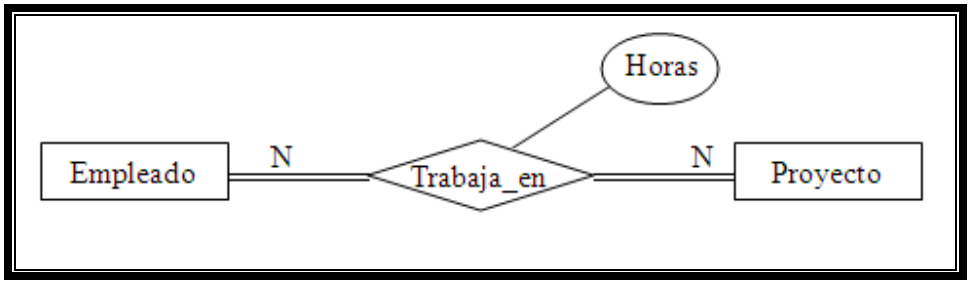


Figura 2.4. Ejemplo de relación varios a varios de la base de datos compania

En este caso se creará una nueva tabla llamada Trabaja_en, cuyas llaves serán la cédula del empleado y el número del proyecto, además del atributo horas que pertenece a la relación.

EMPLEADO (nombre, apellido, ci, fecha_n, dirección, sexo, salario, superci, dno)
PROYECTO (pnombre, pnumero, plocal, dnum)
TRABAJA_EN (eci, pno, horas)

2.3.2 Relación 1:N

En el caso de que exista este tipo de relación (uno a varios), se realiza una propagación de la llave de la tabla N hacia la tabla 1.
Existen casos en los que es recomendable no propagar la llave sino transformar a la relación en una tabla nueva, como si se tratara de una relación N:M, como por ejemplo:

- Cuando el número de ocurrencias es muy pequeño, esto es para evitar valores nulos.

- Cuando se considera que a futuro la relación se convertirá en una relación N:M.

- Cuando existen atributos propios de la relación.

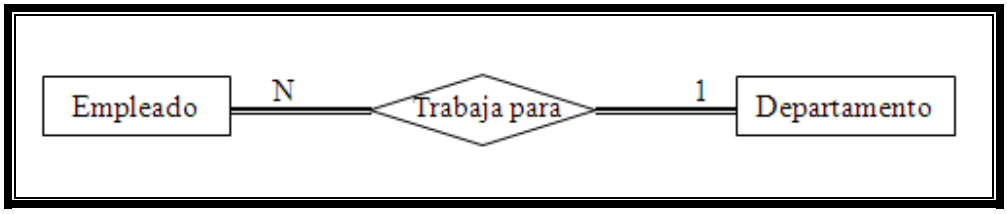


Figura 2.5. Ejemplo de relación uno a varios de la base de datos compania

2.3.3 Relación 1:1

En este tipo de relación (uno a uno), se realiza una propagación de llave, esto se puede hacer de forma bidireccional, es recomendable propagar la llave de la entidad con participación parcial a la entidad con participación total.

En el siguiente ejemplo consideramos que todos los departamentos deberán tener a un empleado como jefe del mismo, pero no todos los empleados van a ser jefes de algún departamento.

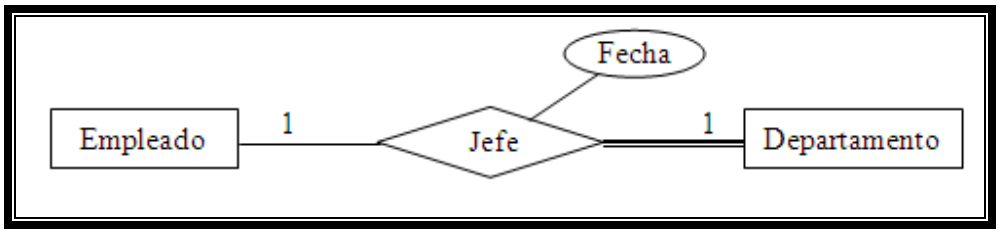


Figura 2.6. Ejemplo de relación uno a uno de la base de datos compania, con participación parcial-total

EMPLEADO (nombre, apellido, ci, fecha_n, dirección, sexo, salario, superci, dno)
DEPARTAMENTO (dnombre, dnumero, jefeci, fecha)

En caso de que las dos entidades tengan participación parcial, es recomendable crear otra tabla con la relación, en el siguiente ejemplo se considera que pueden haber departamentos que no tengan jefe, en el cual, es recomendable la creación de una nueva tabla, debido a la participación parcial de las dos entidades.

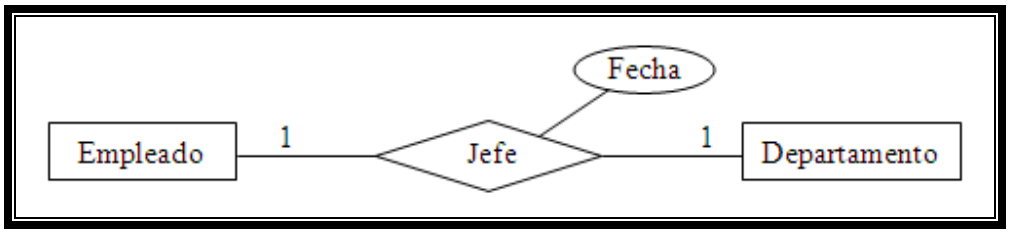


Figura 2.7. Ejemplo de relación uno a uno de la base de datos compania, con participación parcial-parcial

EMPLEADO (nombre, apellido, ci, fecha_n, dirección, sexo, salario, superci, dno)
DEPARTAMENTO (dnombre, dnumero)
JEFE (ci,dnumero,fecha)

2.4 Creación de Tablas

En este momento la base de datos se encuentra vacía, esto se lo puede comprobar utilizando la siguiente instrucción:

```
SHOW TABLES;
```

Como resultado obtendremos el mensaje "*Empty set*" que quiere decir que la base de datos no tiene tablas creadas aun.

Para comenzar a crear las tablas de la base de datos usamos el comando *CREATE* de la siguiente forma:

CREATE TABLE nombre_tabla (columnas –tipo- -tamaño- NOT NULL);

NOT NULL impide la introducción de valores nulos, mostrando un mensaje de error al tratar de ingresar un valor nulo. A continuación crearemos las siguientes tablas:

EMPLEADO

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SEXO	SALARIO	SUPERCI	DNO
Juan	Polo	123456789	3-mar-59	Sucre 7-12	M	3000	333445555	5
Humberto	Pons	333445555	25-dic-60	Bolívar 5-67	M	4000	888665555	5
Irma	Vega	999887777	13-nov-50	P Córdova 3-45	F	2500	987654321	4
Elena	Tapia	987654321	03-may-61	Ordóñez 7-29	F	4300	888665555	4
Pablo	Castro	666884444	15-sep-55	Bolívar 1-50	M	3800	333445555	5
Marcia	Mora	453453453	29-mar-60	Colombia 4-23	F	2500	333445555	5
Manuel	Bonilla	987987987	16-jul-58	B. Malo 1-10	M	2500	987654321	4
Jaime	Pérez	888665555	5-abr-57	Sangurima 8-34	M	5500	NULL	1

Tabla 2.1. Empleado

DEPARTAMENTO

DNOMBRE	DNUMERO	JEFECI	JEFE_FI
Investigación	5	333445555	12-may-80
Administrativo	4	987654321	05-dic-82
Compras	1	888665555	06-jun-78

Tabla 2.2. Departamento

LOCALIZACION

DNUMERO	DEP_LOCA
1	Cuenca
4	Guayaquil
5	Quito
5	Manta
5	Cuenca

Tabla 2.3 Localizacion

TRABAJA_EN

ECI	PNO	HORAS
123456789	1	12,5
123456789	2	15,6
666884444	3	14,7
453453453	1	10
453453453	2	10
333445555	2	20
333445555	3	10
333445555	10	10
333445555	20	10
999887777	30	30
999887777	10	5
987987987	10	15
987987987	30	17
987654321	30	10
987654321	20	12
888665555	20	NULL

Tabla 2.4 Trabaja_en

PROYECTO

PNOMBRE	PNUMERO	PLOCAL	DNUM
ProductoX	1	Quito	5
ProductoY	2	Manta	5
ProductoZ	3	Cuenca	5
Computadora	10	Guayaquil	4
Reorganizar	20	Cuenca	1
Beneficios	30	Guayaquil	4

Tabla 2.5. Proyecto

CARGA_F

ECI	DEP_NOM	SEXO	FECHAN_N	RELACION
333445555	María	F	2/02/86	Hija
333445555	Teodoro	M	10/10/90	Hijo
333445555	Ana	F	15/09/65	Cónyuge
987654321	Alberto	M	6/07/67	Cónyuge
123456789	Miguel	M	5/11/84	Hijo
123456789	María	F	9/01/87	Hija
123456789	Elizabeth	F	12/12/60	Cónyuge

Tabla 2.6. Carga_f

Para crear las tablas anteriores, utilizaremos las siguientes instrucciones:

```
CREATE TABLE empleado    (nombre     VARCHAR (30),
                          apellido   VARCHAR (30),
                          ci         VARCHAR (10)     NOT NULL,
                          fecha_n    DATE,
                          direccion  VARCHAR (30),
                          sexo       CHAR (1),
```

```
                                salario       INTEGER (4),
                                superci       VARCHAR (10),
                                dno           INTEGER (1));

CREATE TABLE departamento   (dnombre      VARCHAR (30),
                             dnumero      INTEGER (1)          NOT NULL,
                             jefeci       VARCHAR (10),
                             jefe_fi      DATE);

CREATE TABLE localizacion    (dnumero     INTEGER (1)          NOT NULL,
                              dep_loca    VARCHAR (30));

CREATE TABLE trabaja_en      (eci         VARCHAR (10)         NOT NULL,
                              pno         INTEGER (2)          NOT NULL,
                              horas       DOUBLE (4,2));

CREATE TABLE proyecto        (pnombre     VARCHAR (30),
                              pnumero     INTEGER (2)          NOT NULL,
                              plocal      VARCHAR (30),
                              dnum        INTEGER (1));

CREATE TABLE carga_f         (eci         VARCHAR (10)          NOT NULL,
                              dep_nom     VARCHAR (30),
                              sexo        VARCHAR (1),
                              fechan_n    DATE,
                              relacion    VARCHAR (10));
```

Para verificar que las tablas fueron creadas correctamente utilizamos la función *SHOW TABLES*, esta nos mostrará el nombre de las tablas existentes en nuestra base de datos.

Figura 2.8. Función para mostrar tablas en una base de datos

Si queremos ver a detalle los campos que contiene la tabla, utilizaremos la función *DESCRIBE* [nombre de la tabla], se mostrará el contenido de la tabla seleccionada.

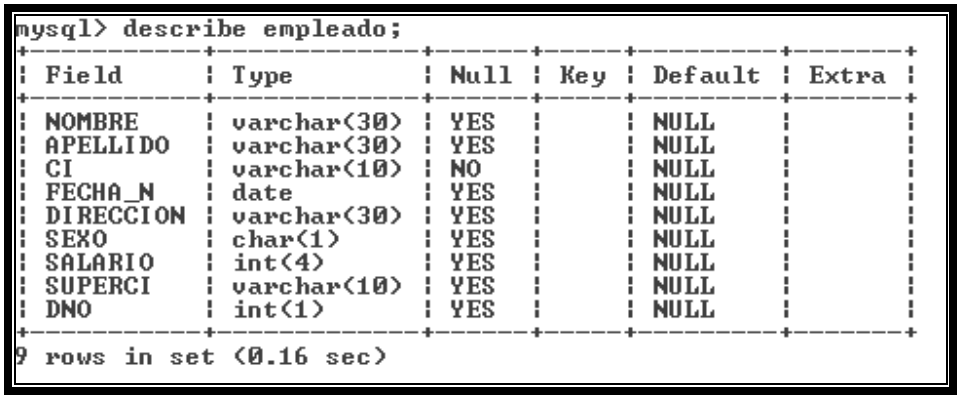


Figura 2.9. Función para mostrar los campos de una tabla

2.5 Creación de Llaves Primarias y Foráneas

2.5.1 Llaves Primarias

Se conoce como llave primaria al campo de la tabla que es único, es decir, no puede repetirse y por el cual se podrá identificar y relacionar claramente toda la información de uno o varios registros de una o varias tablas. La llave primaria no puede ser un valor nulo o *NULL* y una tabla solo puede tener una llave primaria.

Para crear llaves primarias en una tabla utilizamos la siguiente instrucción:

```
ALTER TABLE nombre_tabla
ADD PRIMARY KEY ([nombre_columna]);
```

También las llaves primarias pueden ser definidas al mismo tiempo que se crea la tabla, por ejemplo vamos a crear la tabla proyecto definiendo cual será su llave primaria:

```
CREATE TABLE PROYECTO (pnombre     VARCHAR (30),
                       pnumero     INTEGER (2)  NOT NULL
                       AUTO_INCREMENT,
                       plocal      VARCHAR (30),
                       dnum        INTEGER (1),
                       PRIMARY KEY(pnumero));
```

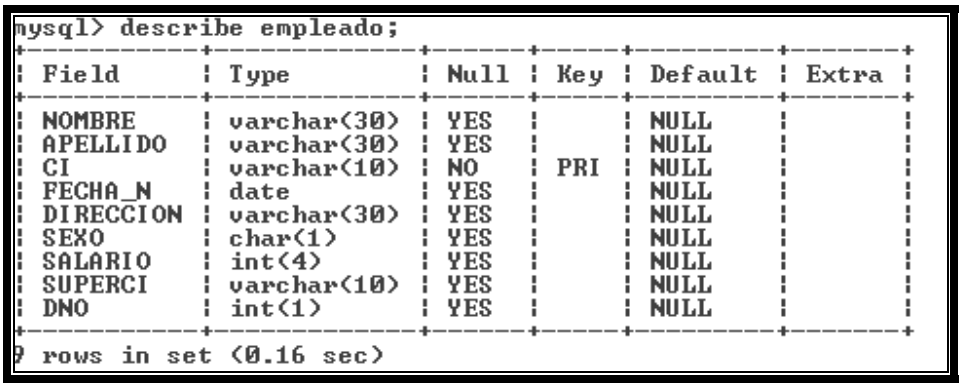
En el ejemplo anterior podemos observar la palabra *AUTO_INCREMENT,* esta permite que la llave primaria se incremente de forma automática, ya no es necesario ingresar el número del proyecto, para usar la instrucción *AUTO_INCREMENT*, es necesario que el campo sea numérico.

A continuación crearemos todas las llaves primarias de las tablas antes establecidas que se encuentran con notación subrayada, basándonos en el modelo entidad-relación de la página 39.

Tabla Empleado:

```
ALTER TABLE empleado
ADD PRIMARY KEY (ci);
```

Al aplicar el comando *DESCRIBE* de la tabla se mostrará el cambio realizado. Aparecerá en la columna *Key* del campo CI la palabra PRI.



```
mysql> describe empleado;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| NOMBRE    | varchar(30) | YES  |     | NULL    |       |
| APELLIDO  | varchar(30) | YES  |     | NULL    |       |
| CI        | varchar(10) | NO   | PRI | NULL    |       |
| FECHA_N   | date        | YES  |     | NULL    |       |
| DIRECCION | varchar(30) | YES  |     | NULL    |       |
| SEXO      | char(1)     | YES  |     | NULL    |       |
| SALARIO   | int(4)      | YES  |     | NULL    |       |
| SUPERCI   | varchar(10) | YES  |     | NULL    |       |
| DNO       | int(1)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
9 rows in set (0.16 sec)
```

Figura 2.10. Descripción de la tabla empleado con su llave primaria

Tabla Departamento:

```
ALTER TABLE departamento
ADD PRIMARY KEY (dnumero);
```

Tabla Proyecto:

```
ALTER TABLE proyecto
ADD PRIMARY KEY (pnumero);
```

2.5.2 Llaves Foráneas

Se conoce como llave foránea a los campos que vienen de las relaciones entre tablas en el modelo entidad relación. Al igual que las llaves primarias, las foráneas tampoco pueden tener valor nulo y una tabla puede tener más de una llave foránea.

Para crear llaves foráneas debemos indicar como se manejará la base de datos en caso de que un registro de la tabla principal sea borrado o actualizado, utilizamos la siguiente instrucción:

```
ALTER TABLE nombre_tabla
ADD FOREIGN KEY (nombre_columna)
REFERENCES  nombre_ tabla_referencia
(nombre_columna_de_tabla_referencia])
ON DELETE [CASCADE, SET NULL, RESTRICT]
ON UPDATE [CASCADE, SET NULL, RESTRICT];
```

A continuación crearemos todas las llaves foráneas de las tablas antes establecidas, basándonos en el modelo entidad-relación de la página 39.

Tabla Empleado:

```
ALTER TABLE empleado
ADD FOREIGN KEY (dno)
REFERENCES departamento (dnumero)
ON DELETE SET NULL
ON UPDATE CASCADE,
ADD FOREIGN KEY (superci)
REFERENCES empleado (ci)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

Al aplicar el comando *DESCRIBE* de la tabla se mostrará el cambio realizado. Aparecerá en la columna *Key* del campo SUPERCI y DNO la palabra MUL.

```
mysql> describe empleado;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| NOMBRE    | varchar(30) | YES  |     | NULL    |       |
| APELLIDO  | varchar(30) | YES  |     | NULL    |       |
| CI        | varchar(10) | NO   | PRI | NULL    |       |
| FECHA_N   | date        | YES  |     | NULL    |       |
| DIRECCION | varchar(30) | YES  |     | NULL    |       |
| SEXO      | char(1)     | YES  |     | NULL    |       |
| SALARIO   | int(4)      | YES  |     | NULL    |       |
| SUPERCI   | varchar(10) | YES  | MUL | NULL    |       |
| DNO       | int(1)      | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
9 rows in set (0.16 sec)
```

Figura 2.11. Descripción de la tabla empleado con su llave primaria y llaves foráneas

Tabla Departamento:

```
ALTER TABLE departamento
ADD FOREIGN KEY (jefeci)
REFERENCES EMPLEADO (CI)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

Tabla Localizacion:

```
ALTER TABLE localizacion
ADD FOREIGN KEY (dnumero)
REFERENCES departamento (dnumero)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Tabla Trabaja_en:

```
ALTER TABLE trabaja_en
ADD FOREIGN KEY (eci)
REFERENCES empleado (ci)
ON DELETE CASCADE
ON UPDATE CASCADE,
ADD FOREIGN KEY (pno)
REFERENCES proyecto (pnumero)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

Tabla Proyecto:

```
ALTER TABLE proyecto
ADD FOREIGN KEY (dnum)
REFERENCES departamento (dnumero)
ON DELETE SET NULL
ON UPDATE CASCADE;
```

Tabla Carga_f:

```
ALTER TABLE carga_f
ADD FOREIGN KEY (eci)
REFERENCES empleado (ci)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

2.6 Restricción UNIQUE

Cuando se desea que la información contenida en una columna no se repita y esta no es una llave primaria, usamos la restricción *UNIQUE.*

Por ejemplo en la tabla departamento de nuestro ejercicio, la llave principal es el número del departamento, pero esto no impide que el nombre del departamento se repita, por lo cual aplicaremos la restricción *UNIQUE* a la columna dnombre.

```
ALTER TABLE departamento
ADD UNIQUE (dnombre);
```

2.7 Edición de Bases de Datos

2.7.1 Borrar Bases de Datos

Para borrar una base de datos se utiliza el comando *DROP* con la siguiente sentencia:

```
DROP DATABASE nombre_de _la_base_de_datos;
```

2.7.2 Renombrar Tablas en una Base de Datos

Para cambiar el nombre a una tabla usamos el comando *RENAME* con el siguiente formato:

```
RENAME TABLE nombre_tabla TO nuevo_nombre_tabla;
```

2.7.3 Borrar Tablas de una Base de Datos

Para borrar tablas de una base de datos utilizamos el comando *DROP* con el siguiente formato:

```
DROP TABLE nombre_de_la_tabla;
```

2.7.4 Borrar Columnas de una Tabla de la Base de Datos

Podemos borrar columnas específicas de una tabla por medio del comando *ALTER TABLE,* acompañado del comando *DROP* con el siguiente formato:

```
ALTER TABLE nombre_de_tabla
DROP nombre_de_columna;
```

2.7.5 Añadir Columnas en una Tabla de la Base de Datos

Para añadir columnas en una tabla usamos el comando *ALTER TABLE*, acompañado de la opción *ADD COLUMN* con el siguiente formato para crear la columna al final de las existentes:

```
ALTER TABLE nombre_de_tabla
ADD COLUMN nombre_de_columna;
```

En caso de que la columna se quiera insertar en una ubicación específica, se utiliza la opción *AFTER* con el siguiente formato:

```
ALTER TABLE nombre_de_tabla
ADD COLUMN nombre_de_columna        tipo_columna
AFTER columna_de_referencia;
```

Para insertar una columna en la primera posición, utilizamos la opción *FIRST* con el siguiente formato:

```
ALTER TABLE nombre_de_tabla
ADD COLUMN nombre_de_columna        tipo_columna
FIRST;
```

2.7.6 Cambiar el Nombre a las Columnas de las Tablas

Para cambiar el nombre a una columna de cualquier tabla usamos la siguiente instrucción:

```
ALTER TABLE nombre_tabla
CHANGE COLUMN 'nombre_columna'  'nuevo_nombre'  tipo_de_campo;
```

2.7.7 Ingreso de Registros en las Tablas de la Base de Datos

Para ingresar datos en una tabla, utilizamos el comando *INSERT*, por medio de la opción *VALUES* con el siguiente formato:

```
INSERT INTO nombre de la tabla (nombres_columnas)
VALUES (‘datos’);
```

Para el ingreso de datos también se puede realizar copias de datos de otras tablas ya existentes, esto lo hacemos por medio de la siguiente instrucción:

```
INSERT INTO nombre_tabla
SELECT tabla_de_origen.*
FROM tabla_de_origen
WHERE condición;
```

La copia de datos se realiza solamente cuando la tabla donde vamos a insertar los datos ya está creada.

Por ejemplo si deseamos guardar en una nueva tabla los empleados que trabajan en el departamento de Compras realizaríamos las siguientes instrucciones:

```
CREATE TABLE compras(ci VARCHAR(10),nombre VARCHAR(30),apellido
VARCHAR (30));
INSERT INTO compras
SELECT ci, nombre, apellido
```

```
FROM empleado,departamento
WHERE dnombre=’Administrativo’ AND
        Dno=dnumero;
```

A continuación procedemos a insertar todos los datos pertenecientes al modelo relacional compania:

```
INSERT INTO empleado (`NOMBRE`, `APELLIDO`, `CI`, `FECHA_N`,
`DIRECCION`, `SEXO`, `SALARIO`)
VALUES ('Juan', 'Polo', '123456789', '1959-03-03', 'Sucre 7-12', 'M', 3000),
        ('Humberto', 'Pons', '333445555', '1960-12-25', 'Bolivar 5-67', 'M', 4000),
        ('Marcia', 'Mora', '453453453', '1960-03-29', 'Colombia 4-23', 'F', 2500),
        ('Pablo', 'Castro', '666884444', '1955-09-15', 'Bolivar 1-50', 'M', 3800),
        ('Jaime', 'Perez', '888665555', '1957-04-05', 'Sangurima 8-34', 'M', 5500),
        ('Elena', 'Tapia', '987654321', '1961-05-03', 'Ordonez 7-29', 'F', 4300),
        ('Manuel', 'Bonilla', '987987987', '1958-07-16', 'B. Malo 1-10', 'M', 2500),
        ('Irma', 'Vega', '999887777', '1950-11-13', 'P. Cordova 3-45', 'F', 2500);
```

Como se observa en la sentencia anterior, podemos ver que los campos DNO y SUPERCI no han sido ingresados, esto se debe a que la integridad referencial se ve afectada al momento de ingresar los registros, puesto que no se nos permite el ingreso de un departamento ni un supervisor que todavía no existe.

Ahora procedemos a insertar información en las demás tablas de la base de datos, para luego realizar una actualización en la tabla empleado por medio del comando *UPDATE*.

```
INSERT INTO departamento (`DNOMBRE`,`DNUMERO`,`JEFECI`,`JEFE_FI`)
VALUES ('Compras', 1, '333445555', '1978-06-06'),
        ('Administrativo', 4, '987654321', '1982-12-05'),
        ('Investigacion', 5, '888665555', '1980-12-05');

INSERT INTO localizacion (`DNUMERO`,`DEP_LOCA`)
VALUES (4, 'Guayaquil'),
```

```
(5, 'Quito'),
(5, 'Manta'),
(5, 'Cuenca'),
(1, 'Cuenca');

INSERT INTO proyecto (`PNOMBRE`, `PNUMERO`, `PLOCAL`, `DNUM`)
VALUES ('ProductoX', 1, 'Quito', 5),
('ProductoY', 2, 'Manta', 5),
('ProductoZ',3,'Cuenca', 5),
('Computadora', 10, 'Guayaquil', 4),
('Reorganizar', 20, 'Cuenca', 1),
('Beneficios', 30, 'Guayaquil', 4);

INSERT INTO `trabaja_en` (`ECI`, `PNO`, `HORAS`)
VALUES ('123456789', 1, 12.5),
('123456789', 2, 15.6),
('666884444', 3, 14.7),
('453453453', 1, 10),
('453453453', 2, 10),
('333445555', 2, 20),
('333445555', 3, 10),
('333445555', 10, 10),
('333445555', 20, 10),
('999887777', 30, 30),
('999887777', 10, 5),
('987987987', 10, 15),
('987987987', 30, 17),
('987654321', 30, 10),
('987654321', 20, 12),
('888665555', 20, NULL);

INSERT INTO carga_f (`ECI`, `DEP_NOM`, `SEXO`, `FECHAN_N`,
`RELACION`)
VALUES('333445555', 'Maria','F', '1986-02-02', 'Hija'),
```

```
('333445555', 'Teodoro', 'M', '1990-10-10', 'Hijo'),
('333445555', 'Ana', 'F', '1965-09-15', 'Conyuge'),
('987654321', 'Alberto', 'M', '1967-07-06', 'Conyuge'),
('123456789', 'Miguel', 'M', '1984-11-05', 'Hijo'),
('123456789', 'Maria', 'F', '1987-01-09', 'Hija'),
('123456789', 'Elizabeth', 'F', '1960-12-12', 'Conyuge');
```

2.7.8 Actualización de Registros de las Tablas de la Base de Datos

Para actualizar datos en una tabla, utilizamos el comando *UPDATE*, por medio de la opción *SET* con el siguiente formato:

```
UPDATE nombre_tabla
SET nombre_columna = ‘nuevos_datos’
WHERE nombre_ columna=’referencia _del_registro_a_actualizar’;
```

A continuación procedemos a actualizar los datos pertenecientes a las columnas del supervisor y número del departamento de la tabla empleado:

```
UPDATE empleado
SET superci=’333445555’, dno=’5’
WHERE ci=’123456789’;
UPDATE empleado
SET superci=’888665555’, dno=’5’
WHERE ci=’333445555’;

UPDATE empleado
SET superci=’987654321’, dno=’4’
WHERE ci=’999887777’;

UPDATE empleado
SET superci=’888665555’, dno=’4’
WHERE ci=’987654321’;
```

```
UPDATE empleado
SET superci='333445555', dno='5'
WHERE ci='666884444';

UPDATE empleado
SET superci='333445555', dno='5'
WHERE ci='453453453';

UPDATE empleado
SET superci='987654321', dno='4'
WHERE ci='987987987';

UPDATE empleado
SET dno='1'
WHERE ci='888665555';
```

2.7.9 Borrar Registros de las Tablas de la Base de Datos

Para borrar registros de una tabla, utilizamos el comando *DELETE*, con el siguiente formato:

```
DELETE nombre_columnas
FROM nombre_tabla
WHERE nombre_columna='referencia_del_registro_a_borrar';
```

En caso de que se desee borrar todos los registros de la tabla usamos la siguiente instrucción:

```
DELETE *
FROM nombre_tabla;
```

2.8 Ejercicio Propuesto

Implementar el siguiente modelo entidad relación en *MySQL*, poniendo en práctica todo lo aprendido en este capítulo. La base de datos deberá llamarse Libreria.

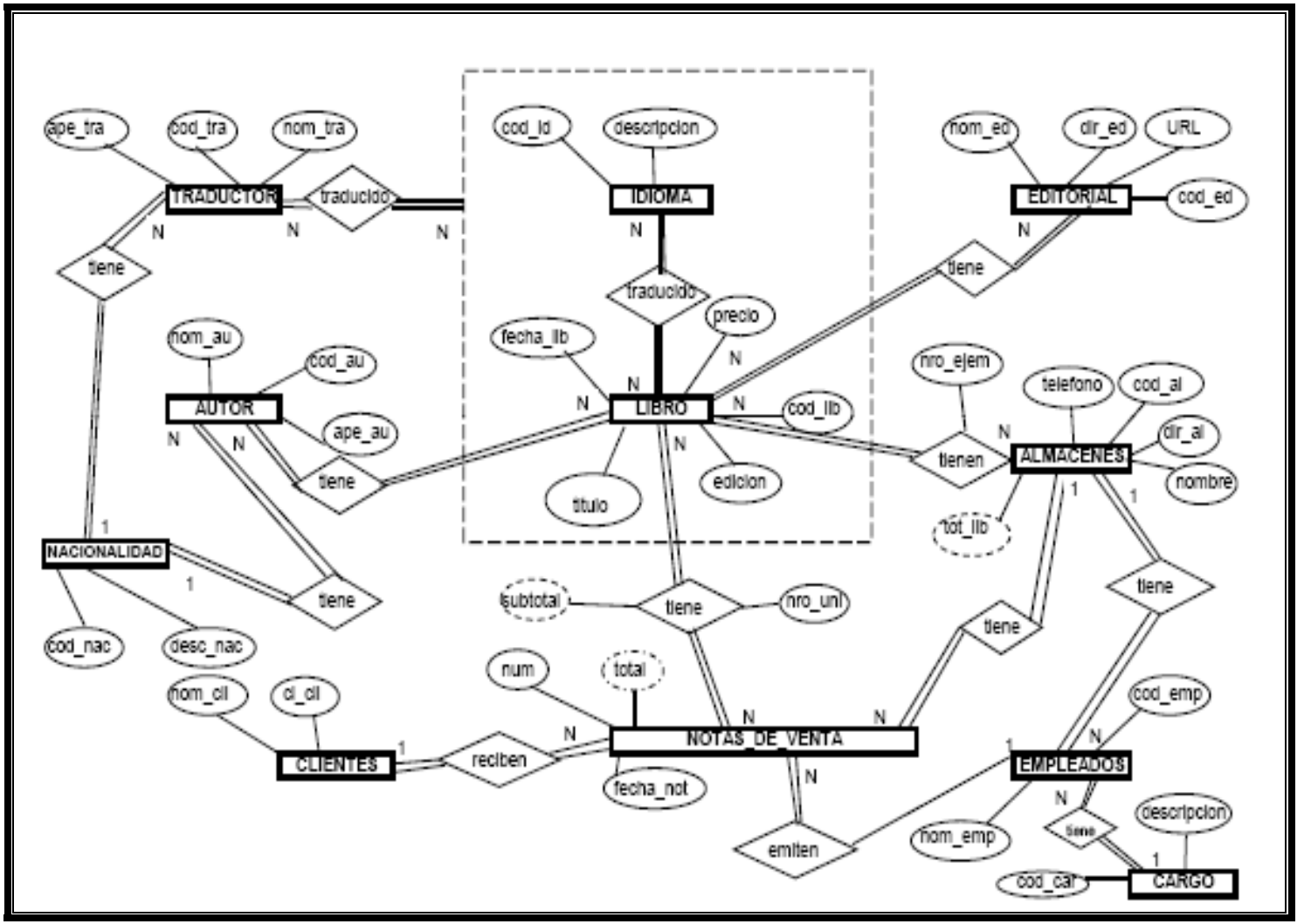


Figura 2.12. Modelo Entidad-Relación de la base de datos libreria

TRADUCTOR

COD_TRA	NOM_TRA	APE_TRA	COD_NAC
T001	Hugo	Ponce	N001
T002	Paco	López	N001
T003	Luis	Serrano	N003

Tabla 2.7. traductor

CLIENTES

CI_CLI	NOM_CLI	APE_CLI
123456789	Paula	Gómez
111222333	Juan	Piedra
121212121	Martha	Vásquez

Tabla 2.8. clientes

EDITORIAL

COD_ED	NOM_ED	DIR_ED	URL
ED01	Océano	Sucre 12-21	www.oceano.com
ED02	Salvat	Tarqui 14-74	www.salvat.com
ED03	Norma	Bolívar 11-10	www.norma.com

Tabla 2.9. editorial

NACIONALIDAD

COD_NAC	DESC_NAC
N001	Chilena
N002	Ecuatoriana
N003	Peruana

Tabla 2.10. nacionalidad

CARGO

COD_CAR	DESC_CAR
C001	Vendedor
C002	Administrador
C003	Gerente

Tabla 2.11. cargo

IDIOMA

COD_ID	DESC_ID
I001	Español
I002	Inglés
I003	Francés

Tabla 2.12. idioma

AUTOR

COD_AU	NOM_AU	APE_AU	COD_NAC
A001	Fernando	de Rojas	N003
A002	Virginia	Woolf	N002
A003	Franz	Kafka	N001

Tabla 2.13. autor

ALMACENES

COD_AL	NOM_AL	DIR_AL	TELEFONO
AL01	Monsalve	Sucre 12-10	2825487
AL02	Juan Marcet	M. Lamar 12-12	2804513
AL03	Papelesa	Av. De las Américas 21-12	2821355

Tabla 2.14. almacenes

LIBRO

COD_LIB	EDICION	FECHA_LIB	TITULO	PRECIO
L001	ED03	20/05/2000	La Celestina	11
L002	ED05	24/03/1999	Al Faro	10,5
L003	ED01	18/11/2004	La Metamorfosis	20,3

Tabla 2.15. libro

EMPLEADOS

COD_EMP	NOM_EMP	APE_EMP	COD_CAR	COD_AL
E001	Miguel	Cardoso	C001	AL01
E002	Pablo	Guamán	C001	AL02
E003	Iván	Tacuri	C001	AL01
E004	Daniel	Palacios	C003	AL03
E005	Pablo	Fernández	C003	AL01
E006	Catalina	Guerrero	C003	AL02
E007	Diego	Moscoso	C002	AL02
E008	Ximena	Aguirre	C002	AL03

Tabla 2.16. empleados

CABECERA_NOTA_VENTA

NUM	FECHA_NOT	COD_AL	COD_EMP	CI_CLIENTE	TOTAL
1	20/06/2004	AL01	E002	121212121	52,5
2	14/03/2005	AL02	E002	123456789	11
3	10/05/2005	AL01	E001	121212121	40,6

Tabla 2.17. cabecera_nota_venta

AUTOR_LIBRO

COD_AU	COD_LIB
A001	L001
A002	L002
A003	L003

Tabla 2.18. autor_libro

EDITORIAL_LIBRO

COD_ED	COD_LIB
ED02	L001
ED01	L001
ED03	L003

Tabla 2.19. editorial_libro

LIBRO_ALMACEN

COD_LIB	COD_AL	NRO_EJEM
L001	AL01	50
L002	AL02	24
L003	AL01	36
L001	AL02	10

Tabla 2.20. libro_almacen

LIBRO_IDIOMA_TRADUCTOR

COD_LIB	COD_ID	COD_TRA
L001	I002	T002
L002	I002	T001
L003	I002	T003

Tabla 2.21 libro_idioma_traductor

DETALLE_NOTA_VENTA

NUM	COD_LIB	NRO_UNI	SUB_T
1	L002	5	52,5
2	L001	1	11
3	L003	2	40,6

Tabla 2.22. detalle_nota_venta

LIBRO_IDIOMA

COD_LIB	COD_ID
L001	I002
L002	I003
L003	I002

Tabla 2.23. libro_idioma

```
CREATE DATABASE Libreria;

USE Libreria;

CREATE TABLE clientes (ci_cli       VARCHAR (10)    NOT NULL,
                       nom_cli    VARCHAR (30),
                       ape_cli    VARCHAR(30),
                       PRIMARY KEY (ci_cli));

CREATE TABLE editorial (cod_ed   VARCHAR(4)   NOT NULL,
                        nom_ed  VARCHAR(30)         UNIQUE,
                        dir_ed     VARCHAR(30),
                        URL        VARCHAR(30),
                        PRIMARY KEY (cod_ed));

CREATE TABLE nacionalidad(cod_nac   VARCHAR(4) NOT NULL,
                          desc_nac  VARCHAR (30)  UNIQUE,
                          PRIMARY KEY (cod_nac));

CREATE TABLE  traductor (cod_tra VARCHAR(4)  NOT NULL,
                         nom_tra VARCHAR(30),
                         ape_tra  VARCHAR(30),
                         cod_nac VARCHAR (4),
                         PRIMARY KEY(cod_tra),
                         FOREIGN KEY (cod_nac)
                         REFERENCES nacionalidad  (cod_nac)
                         ON DELETE SET NULL
                         ON UPDATE CASCADE);

CREATE TABLE cargo (cod_car      VARCHAR(4)          NOT NULL,
                    desc_car     VARCHAR(30)         UNIQUE,
                    PRIMARY KEY(cod_car));

CREATE TABLE idioma (cod_id    VARCHAR(4)     NOT NULL,
```

```
                    desc_id  VARCHAR(30)    UNIQUE,
                    PRIMARY KEY (cod_id));

CREATE TABLE autor (cod_au   VARCHAR(4)      NOT NULL,
                    nom_au VARCHAR (30),
                    ape_au  VARCHAR (30),
                    cod_nac VARCHAR(4),
                    PRIMARY KEY(cod_au),
                    FOREIGN KEY(cod_nac)
                    REFERENCES nacionalidad (cod_nac)
                    ON DELETE SET NULL
                    ON UPDATE CASCADE);

CREATE TABLE almacenes (cod_al  VARCHAR(4)          NOT NULL,
                        nom_al VARCHAR (30)       UNIQUE,
                        dir_al     VARCHAR (30),
                        telefono INTEGER (11),
                        PRIMARY KEY (cod_al));

CREATE TABLE libro (cod_lib   VARCHAR(4)   NOT NULL,
                    edicion  INTEGER (2),
                    fecha_lib  DATE,
                    titulo      VARCHAR (30),
                    precio    DOUBLE(4,2),
                    PRIMARY KEY (cod_lib));

CREATE TABLE empleados(cod_emp  VARCHAR(4)   NOT NULL,
                        nom_emp VARCHAR(30),
                        ape_emp  VARCHAR(30),
                        cod_car    VARCHAR(4),
                        cod_al       VARCHAR(4),
                        PRIMARY KEY(cod_emp),
                        FOREIGN KEY(cod_car)
                        REFERENCES cargo (cod_car)
```

```
                    ON DELETE SET NULL
                    ON UPDATE CASCADE,
                    FOREIGN KEY(cod_al)
                    REFERENCES almacenes (cod_al)
                    ON DELETE SET NULL
                    ON UPDATE CASCADE);

CREATE TABLE cabecera_nota_venta (num INTEGER(4)   NOT NULL
                        AUTO_INCREMENT,
                        fecha_not    DATE,
                        cod_al       VARCHAR(4),
                        cod_emp      VARCHAR(4),
                        ci_cliente   VARCHAR(10),
                        total        FLOAT (4,2),
                        PRIMARY KEY(num),
                        FOREIGN KEY(cod_al)
                        REFERENCES almacenes (cod_al)
                        ON DELETE SET NULL
                        ON UPDATE CASCADE,
                        FOREIGN KEY(cod_emp)
                        REFERENCES empleados (cod_emp)
                        ON DELETE NO ACTION
                        ON UPDATE CASCADE,
                         FOREIGN KEY(ci_cliente)
                         REFERENCES clientes(ci_cli)
                         ON DELETE NO ACTION
                         ON UPDATE CASCADE);

CREATE TABLE autor_libro (cod_au   VARCHAR(4),
                    cod_lib   VARCHAR(4),
                    FOREIGN KEY (cod_au)
                    REFERENCES autor (cod_au)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE,
```

```
                                FOREIGN KEY(cod_lib)
                                REFERENCES libro(cod_lib)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE);

CREATE TABLE editorial_libro(cod_ed   VARCHAR(4),
                                cod_lib   VARCHAR(4),
                                FOREIGN KEY (cod_ed)
                                REFERENCES editorial (cod_ed)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE,
                                FOREIGN KEY(cod_lib)
                                REFERENCES libro(cod_lib)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE);

CREATE TABLE libro_almacen (cod_lib   VARCHAR(4),
                                cod_al   VARCHAR(4),
                                nro_ejem INTEGER(4),
                                FOREIGN KEY (cod_lib)
                                REFERENCES libro (cod_lib)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE,
                                FOREIGN KEY(cod_al)
                                REFERENCES almacenes(cod_al)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE);

CREATE TABLE libro_idioma_traductor     (cod_lib   VARCHAR(4),
                                          cod_id   VARCHAR(4),
                                          cod_tra  VARCHAR(4),
                                          FOREIGN KEY (cod_lib)
                                          REFERENCES libro (cod_lib)
                                          ON DELETE CASCADE
```

```
                                        ON UPDATE CASCADE,
                                         FOREIGN KEY(cod_id)
                                         REFERENCES idioma(cod_id)
                                         ON DELETE CASCADE
                                        ON UPDATE CASCADE,
                                        FOREIGN KEY(cod_tra)
                                         REFERENCES traductor(cod_tra)
                                         ON DELETE CASCADE
                                        ON UPDATE CASCADE);

CREATE TABLE detalle_nota_venta (num   INTEGER(4),
                                 cod_lib  VARCHAR(4),
                                  nro_uni  INTEGER(4),
                                  sub_t      DOUBLE(4,2),
                                  FOREIGN KEY (cod_lib)
                                  REFERENCES libro (cod_lib)
                                  ON DELETE RESTRICT
                                  ON UPDATE RESTRICT,
                                  FOREIGN KEY (num)
                                  REFERENCES cabecera_nota_venta(num)
                                  ON DELETE CASCADE
                                  ON UPDATE CASCADE);

CREATE TABLE libro_idioma (cod_lib    VARCHAR(4),
                           cod_id    VARCHAR(4),
                           FOREIGN KEY (cod_lib)
                           REFERENCES libro (cod_lib)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
                           FOREIGN KEY (cod_id)
                           REFERENCES idioma (cod_id)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE);
```

```
INSERT INTO clientes (`ci_cli`, `nom_cli`, `ape_cli`)
VALUES ('123456789', 'Paula', 'Gomez'),
        ('111222333', 'Juan', 'Piedra'),
        ('121212121', 'Martha', 'Vasquez');

INSERT INTO editorial (`cod_ed`,`nom_ed`, `dir_ed`, `URL`)
VALUES ('ED01','Oceano', 'Sucre 12-21', 'www.oceano.com'),
        ('ED02','Salvat', 'Tarqui 14-74', 'www.salvat.com'),
        ('ED03','Norma', 'Bolivar 11-10', 'www.norma.com');

INSERT INTO nacionalidad (`cod_nac`,`desc_nac`)
VALUES ('N001','chilena'),
        ('N002','ecuatoriana'),
        ('N003','peruana');

INSERT INTO traductor (`cod_tra`,`nom_tra`, `ape_tra`,  `cod_nac`)
VALUES ('T001','Hugo', 'Ponce', 'N001'),
        ('T002','Paco', 'Lopez', 'N001'),
        ('T003','Luis', 'Serrano', 'N003');

INSERT INTO cargo (`cod_car`,`desc_car`)
VALUES ('C001','vendedor'),
        ('C002','administrador'),
        ('C003','gerente');

INSERT INTO idioma (`cod_id`,`desc_id`)
VALUES ('I001','espanol'),
        ('I002','ingles'),
        ('I003','frances');

INSERT INTO autor (`cod_au`,`nom_au`, `ape_au`,  `cod_nac`)
VALUES ('A001','Fernando', 'de Rojas',  'N003'),
        ('A002','Virginia', 'Woolf',  'N002'),
        ('A003','Franz', 'Kafka',  'N001');
```

```
INSERT INTO almacenes (`cod_al`,`nom_al`, `dir_al`, `telefono`)
VALUES ('AL01','Monsalve', 'Sucre 12-10', 2888888),
        ('AL02','Juan Marcet', 'M. Lamar 12-12', 2804513),
        ('AL03','Papelesa', 'Av. De las Américas 21-12', 2821355);

INSERT INTO libro (`cod_lib`,`edicion`, `fecha_lib`, `titulo`, `precio`)
VALUES ('L001','5', '00-05-20', 'La Celestina', '11'),
        ('L002','4', '99-03-24', 'Al Faro', '10.50'),
        ('L003','1', '04-11-18', 'La Metamorfosis', '20.3');

INSERT INTO empleados (`cod_emp`,`nom_emp`, `ape_emp`,  `cod_car`, `cod_al`)
VALUES ('E001','Miguel', 'Cardoso',  'C001', 'AL01'),
        ('E002','Pablo', 'Guaman',  'C001', 'AL02'),
        ('E003','Ivan', 'Tacuri',  'C001', 'AL01'),
        ('E004','Daniel', 'Palacios',  'C003', 'AL03'),
        ('E005','Pablo', 'Fernandez',  'C003', 'AL01'),
        ('E006','Catalina', 'Guerrero',  'C003', 'AL02'),
        ('E007','Diego', 'Moscoso',  'C002', 'AL02'),
        ('E008','Ximena', 'Aguirre',  'C002', 'AL03');

INSERT INTO cabecera_nota_venta (`fecha_not`, `cod_al`, `cod_emp`, `ci_cliente`,
`total`)
VALUES ('04-06-20', 'AL01', 'E002', '121212121', '52.5'),
        ('05-03-14', 'AL02', 'E002', '123456789', '11'),
        ('05-05-10', 'AL01', 'E001', '121212121', '40.6');

INSERT INTO autor_libro (`cod_au`, `cod_lib`)
VALUES ('A001', 'L001'),
        ('A002', 'L002'),
        ('A003', 'L003');

INSERT INTO editorial_libro (`cod_ed`, `cod_lib`)
VALUES ('ED02', 'L001'),
        ('ED01', 'L001'),
        ('ED03', 'L003');
```

```
INSERT INTO libro_almacen (`cod_lib`, `cod_al`, `nro_ejem`)
VALUES ('L001', 'AL01', '50'),
        ('L002', 'AL02', '24'),
        ('L003', 'AL01', '36'),
        ('L001', 'AL02', '10');

INSERT INTO libro_idioma_traductor (`cod_lib`, `cod_id`, `cod_tra`)
VALUES ('L001', 'I002', 'T002'),
        ('L002', 'I002', 'T001'),
        ('L003', 'I002', 'T003');

INSERT INTO detalle_nota_venta (`num`, `cod_lib`, `nro_uni`, `sub_t`)
VALUES ('1', 'L002', '5', '5'),
        ('2', 'L001', '1', '11'),
        ('3', 'L003', '2', '40.6');

INSERT INTO libro_idioma (`cod_lib`, `cod_id`)
VALUES ('L001', 'I002'),
        ('L002', 'I003'),
        ('L003', 'I002');
```

2.9 Conclusiones

El tener conocimientos sobre el uso del lenguaje *SQL* hace posible el manejo del Gestor MySQL y nos facilita el uso de otros gestores de bases de datos, ya que, el *SQL* es el lenguaje universal más aceptado para el manejo de bases de datos. Además, si dominamos la estructura de este lenguaje, podremos realizar diferentes tareas como consultas o manipulación de la base de datos, y la información que estas contienen; temas que son de fundamental importancia y de uso cotidiano en la gestión de bases de datos.

CAPITULO 3

CONSULTAS SIMPLES

INTRODUCCIÓN

En este capítulo trataremos sobre como obtener información de una determinada base de datos, por medio de la cláusula *SELECT* y sus diferentes combinaciones con otras cláusulas para obtener el resultado deseado.

3.1 Sentencia Select

Para realizar una consulta en la base de datos se utilizará el comando *SELECT*, acompañado de la cláusula *FROM* para indicar el lugar del que queremos obtener la información.

El formato de esta instrucción es el siguiente:

```
SELECT nombre_columnas
FROM nombre de tablas;
```

Para obtener toda la información de una tabla se remplazará el nombre de las columnas por *, esto nos dará como resultado todo el contenido de la tabla.

En el siguiente ejemplo vamos a realizar una consulta de toda la información de la tabla proyectos del modelo que se está desarrollando en este tutorial.

```
SELECT *
FROM proyecto;
```

El resultado que se mostrará al realizar esta consulta será el siguiente:

```
mysql> SELECT * FROM proyecto;
+-------------+---------+-----------+------+
| PNOMBRE     | PNUMERO | PLOCAL    | DNUM |
+-------------+---------+-----------+------+
| ProductoX   |       1 | Quito     |    5 |
| ProductoY   |       2 | Manta     |    5 |
| ProductoZ   |       3 | Cuenca    |    5 |
| Computadora |      10 | Guayaquil |    4 |
| Reorganizar |      20 | Cuenca    |    1 |
| Beneficios  |      30 | Guayaquil |    4 |
+-------------+---------+-----------+------+
6 rows in set (0.09 sec)
```

Figura 3.1. Resultado de la selección de toda la información de la tabla proyecto

En caso de que se desee ver únicamente los nombres y apellidos de los empleados de la tabla empleado, se realizará la siguiente consulta:

```
SELECT nombre, apellido
FROM empleado;
```

El resultado será el siguiente:

```
+----------+----------+
| nombre   | apellido |
+----------+----------+
| Juan     | Polo     |
| Humberto | Pons     |
| Marcia   | Mora     |
| Pablo    | Castro   |
| Jaime    | Perez    |
| Elena    | Tapia    |
| Manuel   | Bonilla  |
| Irma     | Vega     |
+----------+----------+
8 rows in set (0.00 sec)
```

Figura 3.2. Resultado de la selección de la tabla empleado

3.2 Concatenación de Datos y Uso de la Cláusula As

Cuando en el resultado de una consulta se desea mostrar la información concatenada utilizamos la palabra *CONCAT*, y para crear una columna temporal en la cual se mostrarán los resultados concatenados usamos la cláusula *AS*, la cual nos permite definir un alias para almacenar los datos temporales.

La cláusula *AS* puede ser usada en las cláusulas GROUP *BY, ORDER BY o HAVING* y puede aparecer tanto en *SELECT* como en *FROM*.

El formato es el siguiente:

```
SELECT CONCAT(columna1,'separador',columna2) AS columna_temporal
FROM nombre_tabla;
```

Como separador entre las columnas concatenadas se puede usar la coma (,), punto (.) o espacio ( ), éste debe estar siempre entre comilla simple.

Por ejemplo deseamos listar los nombres de los empleados de la compañía, sin utilizar *CONCAT* ni *AS*, el resultado sería el siguiente:

```
SELECT nombre, apellido
FROM empleado;
```

```
+----------+----------+
| nombre   | apellido |
+----------+----------+
| Juan     | Polo     |
| Humberto | Pons     |
| Marcia   | Mora     |
| Pablo    | Castro   |
| Jaime    | Perez    |
| Elena    | Tapia    |
| Manuel   | Bonilla  |
| Irma     | Vega     |
+----------+----------+
8 rows in set (0.00 sec)
```

Figura 3.3. Resultado de la selección sin concatenación

Realizando la consulta concatenando datos, obtendremos el siguiente resultado:

```
SELECT CONCAT(nombre, ' ', apellido) AS empleados
FROM empleado;
```

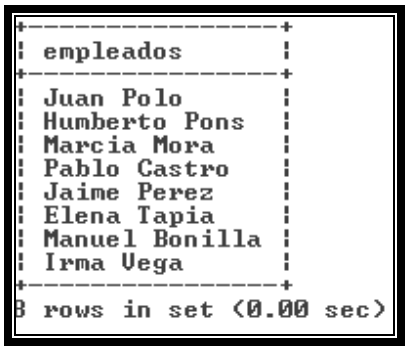


Figura 3.4. Resultado de la selección concatenado

3.3 Selección de Registros con Condiciones Específicas

Cuando se quiere obtener un solo registro como resultado, es necesario utilizar la cláusula *WHERE* a continuación de la instrucción antes mencionada, la estructura es la siguiente:

```
SELECT nombre_columnas
FROM nombre_tablas
WHERE condición;
```

En este caso la condición hará referencia a un campo conocido por el cual realizaremos la consulta, a continuación deseamos consultar toda la información del empleado con el número de cédula 123456789.

```
SELECT *
FROM empleado
WHERE ci='123456789';
```

El resultado será el siguiente:

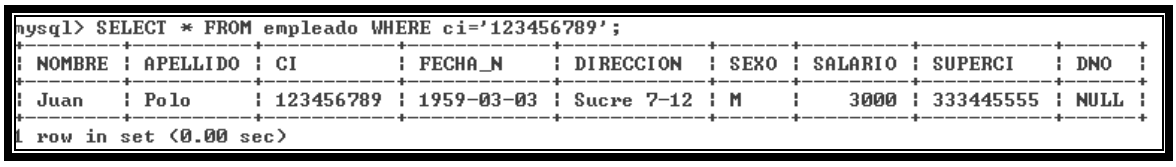


Figura 3.5. Resultado de la selección del registro con ci nro. 123456789

Ahora vamos a consultar el nombre y apellido de los empleados de sexo femenino.

```
SELECT nombre, apellido
FROM empleado
WHERE sexo='F';
```

```
+--------+----------+
| nombre | apellido |
+--------+----------+
| Marcia | Mora     |
| Elena  | Tapia    |
| Irma   | Vega     |
+--------+----------+
3 rows in set (0.00 sec)
```

Figura 3.6. Resultado de los empleados de sexo femenino

Algunas consultas pueden ser expresadas de distintas formas, por ejemplo la consulta anterior de listar el nombre y apellido de todos los empleados de sexo femenino puede ser expresada también de la siguiente forma y el resultado obtenido será el mismo.

```
SELECT nombre, apellido
FROM empleado
WHERE sexo != 'M';
```

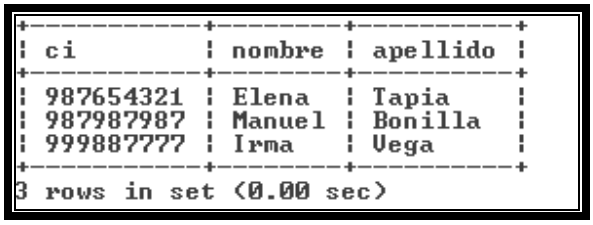
```
+--------+----------+
| nombre | apellido |
+--------+----------+
| Marcia | Mora     |
| Elena  | Tapia    |
| Irma   | Vega     |
+--------+----------+
3 rows in set (0.00 sec)
```

Figura 3.7. Resultado de los empleados que no son de sexo masculino

La cláusula *WHERE* también es utilizada para realizar consultas en las que se utilizará más de una tabla, siempre y cuando exista algún tipo de relación entre ellas.

A continuación listaremos la cédula, nombre y apellido de los empleados que trabajan en el departamento Administrativo.

```
SELECT ci, nombre, apellido
FROM empleado, departamento
WHERE dno=dnumero AND
        dnombre='Administrativo';
```



```
+-----------+--------+----------+
| ci        | nombre | apellido |
+-----------+--------+----------+
| 987654321 | Elena  | Tapia    |
| 987987987 | Manuel | Bonilla  |
| 999887777 | Irma   | Vega     |
+-----------+--------+----------+
3 rows in set (0.00 sec)
```

Figura 3.8. Empleados del departamento administrativo

3.4 Eliminación de Filas Duplicadas

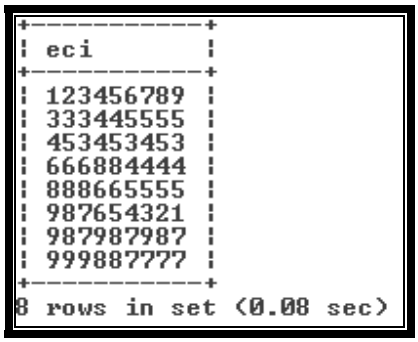
La eliminación de filas duplicadas se realiza por medio de la cláusula *DISTINCT, e*sta cláusula es utilizada cuando se repite varias veces un mismo registro y se quiere eliminar los valores duplicados y se aplica solamente a una columna.

La cláusula *DISTINCT* puede ser utilizada en combinación con las funciones *SUM( ), AVG( ) y COUNT( ),* no es aplicable para las funciones *MAX*( ) ni *MIN*( ), ya que no afectaría el resultado.

```
SELECT DISTINCT nombre_columna
FROM  nombre_tabla;
```

A continuación vamos a listar todas las cédulas de los empleados que trabajan en algún proyecto.

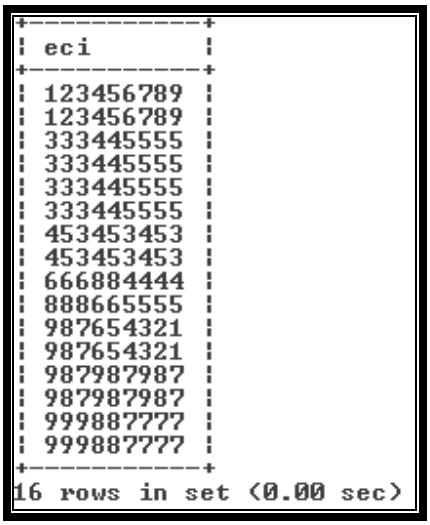
```
SELECT DISTINCT eci
FROM trabaja_en;
```



```
+-----------+
| eci       |
+-----------+
| 123456789 |
| 333445555 |
| 453453453 |
| 666884444 |
| 888665555 |
| 987654321 |
| 987987987 |
| 999887777 |
+-----------+
8 rows in set (0.08 sec)
```

Figura 3.9. Selección de cédulas sin repetición

Si no hubiésemos escrito la palabra *DISTINCT* el resultado hubiese sido el siguiente:



```
+-----------+
| eci       |
+-----------+
| 123456789 |
| 123456789 |
| 333445555 |
| 333445555 |
| 333445555 |
| 333445555 |
| 453453453 |
| 453453453 |
| 666884444 |
| 888665555 |
| 987654321 |
| 987654321 |
| 987987987 |
| 987987987 |
| 999887777 |
| 999887777 |
+-----------+
16 rows in set (0.00 sec)
```

Figura 3.10. Selección de todas las cédulas de los empleados

3.5 Consultas con Valores Nulos

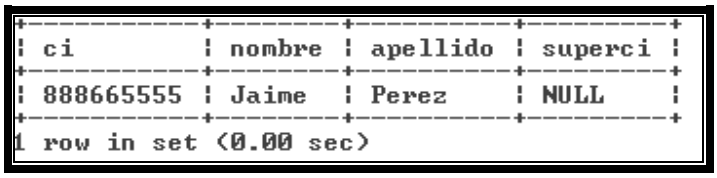
Los valores nulos se permiten en ciertos campos de las tablas, esto sirve generalmente para el manejo de las llaves foráneas al momento de actualizar o borrar las llaves a las que hacen referencia, también se refiere a los campos que no obligadamente se tienen que llenar.

El formato para seleccionar valores nulos es la siguiente:

```
SELECT nombre_columnas
FROM nomre_tablas
WHERE nombre_columna IS NULL;
```

A continuación vamos a listar la cédula, nombre y apellido de los empleados que no tienen supervisor.

```
SELECT ci, nombre, apellido, superci
FROM empleado
WHERE superci IS NULL;
```



```
+-----------+--------+----------+---------+
| ci        | nombre | apellido | superci |
+-----------+--------+----------+---------+
| 888665555 | Jaime  | Perez    | NULL    |
+-----------+--------+----------+---------+
1 row in set (0.00 sec)
```

Figura 3.11. Empleados sin supervisor

El formato para seleccionar valores no nulos es el siguiente:

```
SELECT nombre_columnas
FROM nomre_tablas
WHERE nombre_columna IS NOT NULL;
```

A continuación vamos a listar la cédula, nombre y apellido de los empleados que tienen supervisor.

```
SELECT ci, nombre, apellido, superci
FROM empleado
WHERE superci IS NOT NULL;
```

ci	nombre	apellido	superci
123456789	Juan	Polo	333445555
333445555	Humberto	Pons	888665555
453453453	Marcia	Mora	333445555
666884444	Pablo	Castro	333445555
987654321	Elena	Tapia	888665555
987987987	Manuel	Bonilla	987654321
999887777	Irma	Vega	987654321

7 rows in set (0.00 sec)

Figura 3.12. Empleados con supervisor

3.6 Test de Correspondencia con Patrón

Se utiliza cuando se desea realizar una consulta con respecto a un patrón conocido, que se puede aplicar a las cadenas de caracteres en los campos de una tabla. Esta consulta devolverá cualquier información seleccionada que corresponda con el patrón ingresado para la búsqueda.

Para realizar estas consultas existen dos caracteres comodines, los cuales son:

- % Para realizar búsquedas reemplazando cualquier secuencia de caracteres.

- _ Para realizar búsquedas reemplazando una o varias posiciones específicas de la cadena.

A continuación vamos a listar los empleados cuyo nombre tenga 6 caracteres y su apellido contenga ‘a’ en cualquier parte del apellido.

```
SELECT nombre, apellido
FROM empleado
WHERE nombre LIKE '_____' AND
        apellido LIKE '%a%';
```

nombre	apellido
Marcia	Mora
Manuel	Bonilla

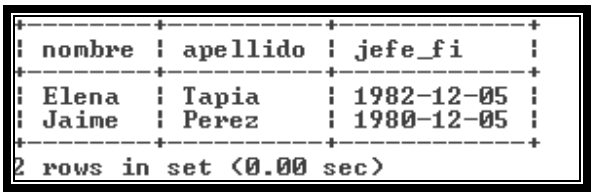
2 rows in set (0.01 sec)

Figura 3.13. Nombres que contienen seis caracteres y al menos una 'a'

3.7 Consultas con Rangos de Fechas

EL *SQL* permite hacer cálculos con fechas, usando meses, años y días unidos o separados. Por ejemplo vamos a listar todos los jefes que han ingresado desde el año 1980 hasta el año 1983.

```
SELECT nombre, apellido, jefe_fi
FROM empleado, departamento
WHERE jefeci=ci AND
        YEAR(jefe_fi) BETWEEN 1979 AND 1983;
```



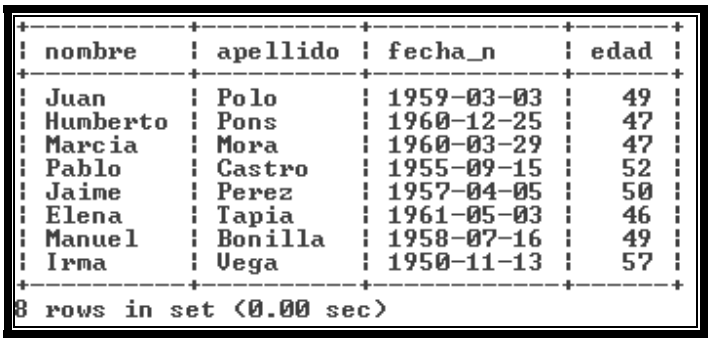
nombre	apellido	jefe_fi
Elena	Tapia	1982-12-05
Jaime	Perez	1980-12-05

2 rows in set (0.00 sec)

Figura 3.14. Empleados nacidos entre 1979 y 1983

Si tenemos una fecha de nacimiento en la base de datos y queremos calcular la edad de una persona, la consulta se realiza de la siguiente forma, por ejemplo vamos a calcular la edad de todos los empleados.

```
SELECT nombre, apellido, fecha_n, (YEAR(CURRENT_DATE) - YEAR(fecha_n))
–(RIGHT(CURRENT_DATE,5) < RIGHT(fecha_n,5))AS edad FROM empleado;
```



nombre	apellido	fecha_n	edad
Juan	Polo	1959-03-03	49
Humberto	Pons	1960-12-25	47
Marcia	Mora	1960-03-29	47
Pablo	Castro	1955-09-15	52
Jaime	Perez	1957-04-05	50
Elena	Tapia	1961-05-03	46
Manuel	Bonilla	1958-07-16	49
Irma	Vega	1950-11-13	57

8 rows in set (0.00 sec)

Figura 3.15. Cálculo de la edad de los empleados

El resultado no afecta a la tabla empleado, puesto que las consultas realizadas se muestran en tablas de resultados temporales.

Para el cálculo necesitamos la fecha actual del sistema, que la obtenemos de esta manera: *YEAR(CURRENT_DATE),* luego extraemos el año del campo fecha_n de la tabla empleado con la siguiente instrucción: *YEAR(fecha_n).*

Se puede observar que se realiza la resta de la fecha actual y la de nacimiento, lo que nos daría la edad basada solamente en los años. Para poder restar los meses y días y obtener la edad exacta respecto a la fecha actual extraemos los 5 últimos dígitos de la fecha actual y de la fecha de nacimiento, los comparamos y si es que la fecha de nacimiento en meses y días es mayor a la fecha actual, esta se restará.

Para poder mostrar el resultado se define y muestra la variable edad de la siguiente manera: *AS* edad y luego se especifica que queremos sacar la información de la tabla empleado, dando como resultado la consulta antes presentada.

3.8 Consultas usando alias

Esto se utiliza para distinguir dos campos con el mismo nombre pero que se encuentran en diferentes tablas. Para realizar este tipo de consultas se utiliza el siguiente formato:

```
SELECT tabla1.nombre_columna, tabla2.nombre_columna
FROM tabla1, tabla2
WHERE tabla1.nombre_columna = tabla2.nombre_columna;
```

Por ejemplo vamos a listar el código y nombre de los departamentos localizados en Cuenca.

```
SELECT departamento.dnumero, dnombre
FROM departamento, localizacion
WHERE  departamento.dnumero = localizacion.dnumero AND
        dep_loca =’Cuenca’;
```

3.9 Consultas Renombrando Tablas

Para consultar información con campos de una misma tabla es necesario renombrar las tablas. También se puede aplicar lo mismo para cualquier tabla, para no tener que escribir todo el nombre de la tabla. Para hacer la consulta se utilizará el siguiente formato:

```
SELECT nuevo_nombre_tabla.nombre_columna
FROM nombre_tabla    nuevo_nombre_tabla
WHERE condición;
```

Por ejemplo vamos a listar el nombre y apellido de todos los empleados con su respectivo supervisor.

```
SELECT e.ci, e.nombre, e.apellido, s.ci, s.nombre, s.apellido
FROM empleado e, empleado s
WHERE e.superci=s.ci;
```

Aquí podemos apreciar que la tabla empleado se renombra dos veces, la una con la letra e y la otra con la letra s.

3.10 Conclusiones

El lenguaje *SQL* nos permite obtener de forma clara y precisa la información requerida o necesaria en la administración de bases de datos, por medio de la realización de consultas, subconsultas o consultas anidadas, usando de manera correcta la estructura del lenguaje.

SQL es práctico y adaptable al momento de realizar consultas, manipular la información y realizar gestiones en la administración de la base de datos, ya que, nos permite estructurar las instrucciones de manera clara con el fin de poder evitar o corregir errores en estos procedimientos.

CAPITULO 4

ATRIBUTOS DE COLUMNA

INTRODUCCIÓN

En este capítulo se tratarán temas que nos permitirán manipular los resultados obtenidos en las consultas, aplicando funciones a las columnas, ordenando y agrupando los resultados obtenidos, también citaremos ciertas condiciones que se pueden aplicar a los grupos de datos. Al final de este capítulo se incluyen una serie de ejercicios de consultas que se aplicarán a las tablas generadas en el capitulo II de este tutorial. Además al terminar la parte instructiva de este capítulo se presenta un ejercicio práctico para consolidar los conocimientos adquiridos durante la realización del mismo.

4.1 Funciones de Columna

En el lenguaje *SQL* se pueden realizar algunas funciones sobre las columnas de una tabla, dependiendo del tipo de campo con el que se hayan definido.

SQL nos permite calcular el promedio de una columna, contar filas, sumar, mostrar el valor mínimo o máximo de una columna. A continuación vamos a realizar algunos ejemplos de funciones de columnas.

Contar el número de empleados que trabajan en la empresa.

```
SELECT COUNT(ci)
FROM empleado;
```

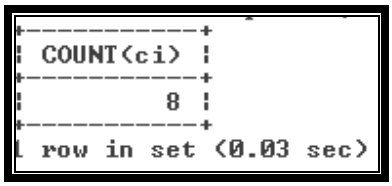


Figura 4.1. Función COUNT( )

Calcular el salario promedio de los empleados de la empresa.

```
SELECT AVG(salario)
FROM empleado;
```

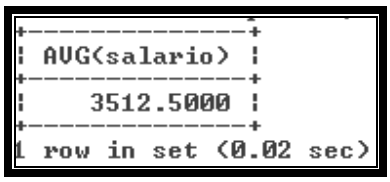


Figura 4.2. Función AVG( )

Sumar el número de horas que ha trabajado el empleado Humberto Pons en los diferentes proyectos.

```
SELECT SUM(horas)
FROM empleado, trabaja_en
WHERE eci=ci AND
        nombre='Humberto' AND
        apellido='Pons';
```

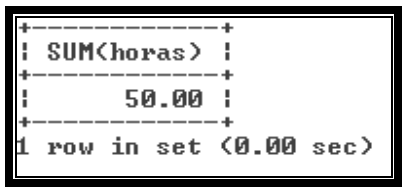


Figura 4.3 Función SUM( )

Calcular el salario más alto y el más bajo de los empleados de la empresa.

```
SELECT MAX(salario), MIN(salario)
FROM empleado;
```

```
+--------------+--------------+
| MAX(salario) | MIN(salario) |
+--------------+--------------+
|         5500 |         2500 |
+--------------+--------------+
1 row in set (0.00 sec)
```

Figura 4.4. Funciones MAX( ), MIN( )

4.2 Ordenamiento de los Resultados de Consulta (Order by)

En *SQL* es posible obtener una consulta ordenada por medio de la cláusula *ORDER BY* y especificando el nombre de la columna por la cual se desea obtener el orden; éste puede realizarse de forma ascendente o descendente.

Ahora vamos a listar todos los empleados de la empresa por orden alfabético de sus apellidos.

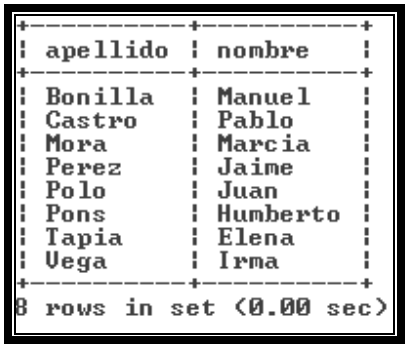
```
SELECT apellido, nombre
FROM empleado
ORDER BY apellido desc;
```

```
+----------+----------+
| apellido | nombre   |
+----------+----------+
| Vega     | Irma     |
| Tapia    | Elena    |
| Pons     | Humberto |
| Polo     | Juan     |
| Perez    | Jaime    |
| Mora     | Marcia   |
| Castro   | Pablo    |
| Bonilla  | Manuel   |
+----------+----------+
8 rows in set (0.00 sec)
```

Figura 4.5. Lista descendente de los empleados

Si en la consulta no especificamos el orden en el que queremos que salga la lista, *SQL* lo hará por defecto de forma ascendente.

```
SELECT apellido, nombre
FROM empleado
ORDER BY apellido;
```



apellido	nombre
Bonilla	Manuel
Castro	Pablo
Mora	Marcia
Perez	Jaime
Polo	Juan
Pons	Humberto
Tapia	Elena
Vega	Irma

8 rows in set (0.00 sec)

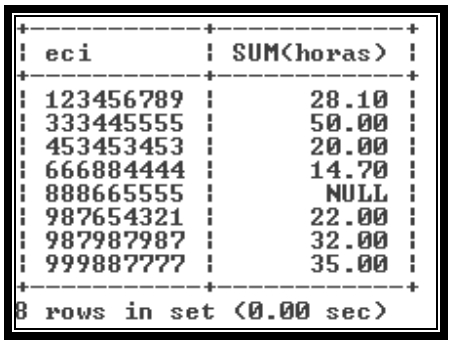
Figura 4.6. Lista de empleados

4.3 Consultas Agrupadas (Group by)

Para agrupar los resultados por un campo determinado utilizamos la cláusula *GROUP BY*, esta clausula es usada en combinación con las funciones de columna, cuando se desea mostrar además del resultado de la función aplicada otros datos de la tabla.

Por ejemplo deseamos consultar el total de horas trabajadas por cada uno de los empleados.

```
SELECT eci, nombre, apellido, SUM(horas)
FROM trabaja_en, empleado
WHERE ci=eci
GROUP BY eci;
```



eci	SUM(horas)
123456789	28.10
333445555	50.00
453453453	20.00
666884444	14.70
888665555	NULL
987654321	22.00
987987987	32.00
999887777	35.00

8 rows in set (0.00 sec)

Figura 4.7. Total de horas trabajadas por cada empleado

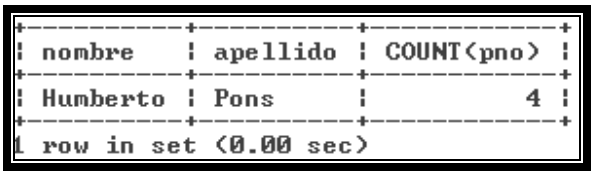
4.4 Condiciones de Búsqueda en Grupos (Having)

Cuando se quiere aplicar condiciones de búsqueda a un grupo se utiliza la cláusula *HAVING*. Las condiciones de búsqueda se especificarán luego de la cláusula ya mencionada.

En el caso de usar la cláusula *HAVING*, las relaciones entre tablas se harán en la cláusula *WHERE*. No olvidar que *HAVING* solo puede utilizarse luego de la cláusula *GROUP BY.*

Por ejemplo vamos a listar el nombre de los empleados que han trabajado en más de 3 proyectos.

```
SELECT nombre, apellido, COUNT(pno)
FROM empleado, trabaja_en
WHERE eci=ci
GROUP BY ci
HAVING count(pno)>3;
```



nombre	apellido	COUNT(pno)
Humberto	Pons	4

1 row in set (0.00 sec)

Figura 4.8. Empleados que han trabajado en más de 3 proyectos

4.5 Ejercicios de Consultas Simples de la Base de Datos Compania

1. Listar el nombre de todos los departamentos.

```
SELECT dnombre
FROM departamento;
```

2. Listar la cédula, nombre, apellido y salario de los empleados que trabajan en el departamento de Compras.

```
SELECT ci, nombre, apellido, salario
FROM empleado, departamento
WHERE dnumero=dno AND
        dnombre= 'Compras';
```

3. Listar nombre, apellido y salario de los empleados que ganan más de 3000 dólares.

```
SELECT nombre, apellido, salario
FROM empleado
WHERE salario>3000;
```

4. Listar los empleados que trabajan en el proyecto ProductoX;

```
SELECT nombre, apellido
FROM proyecto, empleado, trabaja_en
WHERE pnumero=pno AND
        eci=ci AND
        pnombre= 'productoX';
```

5. Contar las cargas familiares del empleado Humberto Pons.

```
SELECT nombre, apellido, COUNT(eci)
FROM carga_f, empleado
WHERE eci=ci AND
```

```
        nombre= ‘Humberto’ AND
        apellido= ‘Pons’
GROUP BY eci;
```

6. Listar todos los empleados que tengan un sueldo entre 2000 y 3000 dólares.

```
SELECT nombre, apellido
FROM empleado
WHERE  salario BETWEEN  2500 AND 3000;
```

7. Listar la cédula, nombre, apellido y total de horas trabajadas de todos los empleados.

```
SELECT eci, nombre, apellido, SUM(horas)
FROM trabaja_en, empleado
WHERE ci=eci
GROUP BY eci;
```

8. Calcular el promedio de horas trabajadas por Humberto Pons.

```
SELECT nombre, apellido, AVG(horas)
FROM empleado, trabaja_en
WHERE ci=eci AND
        nombre=’Humberto’ AND
        apellido=’Pons’
GROUP BY eci;
```

9. Calcular la edad de las cargas familiares del empleado Juan Polo.

```
SELECT nombre, apellido,  dep_nom, relacion,
(YEAR(CURRENT_DATE) - YEAR(fechan_n)) –
(RIGHT(CURRENT_DATE,5) < RIGHT(fechan_n,5)) AS edad
FROM carga_f, empleado
WHERE eci=ci AND
```

```
nombre='Juan' AND
apellido ='Polo';
```

10. Listar todos los empleados nacidos en el mes de marzo.

```
SELECT ci, nombre, apellido, fecha_n
FROM empleado
WHERE MONTH(fecha_n)=3;
```

11. Listar el nombre de los empleados que tienen más de 1 carga familiar de sexo femenino.

```
SELECT  nombre, apellido, COUNT(eci)
FROM empleado, carga_f
WHERE eci=ci  AND
        carga_f.sexo='f'
GROUP BY eci
HAVING COUNT(eci)>1;
```

12. Listar todos los proyectos localizados en Cuenca.

```
SELECT pnombre, plocal
FROM proyecto
WHERE plocal='Cuenca';
```

13. Listar todos los empleados que tengan cargas familiares cuyo nombre termine con la letra A.

```
SELECT nombre, dep_nom
FROM empleado, carga_f
WHERE ci=eci AND
        dep_nom LIKE '%a';
```

14. Listar todos los proyectos que pertenecen al departamento ubicado en Cuenca.

```
SELECT pnombre, dep_loca
FROM proyecto, localizacion
WHERE dnumero=dnum AND
        dep_loca='Cuenca';
```

15. Listar todos los empleados que hayan nacido entre el año 1959 y 1961.

```
SELECT nombre, apellido, fecha_n
FROM empleado
WHERE YEAR(fecha_n) BETWEEN 1959 AND 1960;
```

16. Listar los empleados que tengan como sueldo 2500,3000 o 4000 dólares.

```
SELECT nombre, apellido, salario
FROM empleado
WHERE salario IN (2500,3000,4000);
```

17. Liste todos los empleados y sus respectivas cargas siempre y cuando hayan nacido en el mismo mes.

```
SELECT nombre, fecha_n, dep_nom, fechan_n
FROM empleado, carga_f
WHERE ci=eci AND
        MONTH(fecha_n) = MONTH(fechan_n);
```

18. Listar las cargas familiares ordenadas por sexo y por nombre.

```
SELECT *
FROM carga_f
ORDER BY sexo, dep_nom;
```

19. Listar todos los empleados cuyo nombre comience con la letra M.

```
SELECT nombre, apellido
FROM empleado
WHERE nombre LIKE 'M%';
```

20. Listar el nombre del empleado, departamento y el nombre del proyecto en el cual trabaja el empleado con la cédula 999887777.

```
SELECT nombre, apellido, dnombre, pnombre
FROM empleado, departamento, proyecto
WHERE dno=dnumero AND
      dnumero=dnum AND
      ci=888665555;
```

21. Mostrar las cargas familiares de los empleados cuyo salario es mayor o igual a 4000 dólares o cuyo salario es igual a 3800 dólares.

```
SELECT ci, nombre, dep_nom, relacion, salario
FROM empleado, carga_f
WHERE  ci=eci AND
       (salario>=4000 OR
       salario = 3800);
```

22. Listar los departamentos en donde los empleados ganan más de 4000 dólares.

```
SELECT dnombre, nombre, salario
FROM departamento, empleado
WHERE dno=dnumero AND
      Salario >4000;
```

23. Calcular la suma de todos los salarios, el salario promedio, salario máximo y salario mínimo de los empleados.

```
SELECT SUM(salario),  AVG(salario), MAX(salario), MIN(salario)
FROM empleado;
```

24. Listar el nombre del departamento que tiene más de 3 empleados.

```
SELECT dnombre, COUNT(ci)
FROM departamento, empleado
WHERE dno=dnumero
GROUP BY dno
HAVING COUNT(ci)>3;
```

25. Calcular el salario máximo y mínimo del departamento de Compras.

```
SELECT dnombre, MAX(salario), MIN(salario)
FROM empleado, departamento
WHERE dno=dnumero AND
        dnombre= 'Investigacion'
GROUP BY (dnombre);
```

26. Calcular el salario promedio de todos los departamentos.

```
SELECT dnombre, AVG(salario)
FROM departamento, empleado
WHERE dnumero=dno
GROUP BY dnombre;
```

27. Contar los salarios diferentes.

```
SELECT COUNT(DISTINCT salario)
FROM empleado;
```

28. Calcular el total de horas trabajadas en cada proyecto.

```
SELECT pnombre,SUM(horas)
```

```
FROM proyecto, trabaja_en
WHERE pno=pnumero
GROUP BY pnumero;
```

29. Listar los proyectos cuyo total supere las 25 horas.

```
SELECT pnombre, SUM(horas)
FROM trabaja_en, proyecto
WHERE pno=pnumero
GROUP BY pno
HAVING SUM(horas)>25;
```

30. Listar el nombre y apellido de todos los empleados con su respectivo supervisor.

```
SELECT e.ci, e.nombre, e.apellido, s.ci, s.nombre, s.apellido
FROM empleado e, empleado s
WHERE e.superci=s.ci;
```

31. Listar el nombre de los empleados y del proyecto en el que han trabajado menos de 20 horas.

```
SELECT nombre, apellido, SUM(horas), pnombre
FROM empleado, trabaja_en, proyecto
WHERE eci=ci AND
        Pno=pnumero
GROUP BY eci
HAVING SUM(horas)<20;
```

32. Listar los empleados cuyos apellidos estén entre la letra M y la letra Q.

```
SELECT nombre, apellido
FROM empleado
WHERE apellido BETWEEN ‘M’ AND ‘Q’;
```

33. Listar los supervisores que tienen cargas familiares.

```
SELECT distinct superci, nombre, apellido
FROM empleado, carga_f
WHERE  eci=superci;
```

34. Contar cuantos empleados existen en cada departamento.

```
SELECT dnombre, COUNT(ci)
FROM empleado, departamento
WHERE dno=dnumero
GROUP BY dno;
```

35. Listar la cédula y nombre de los supervisores.

```
SELECT  distinct s.ci, s.nombre, s.apellido
FROM empleado e, empleado s
WHERE s.ci=e.superci;
```

36. Listar todas las cargas familiares del empleado Juan Polo que tengan 6 caracteres.

```
SELECT dep_nom
FROM empleado, carga_f
WHERE ci=eci AND
        dep_nom LIKE '______';
```

4.6 Ejercicios de Consultas Simples de la Base de Datos Libreria

1. Listar el total y número de la factura del cliente con la cédula 123456789.

```
SELECT ci_cli, nom_cli, ape_cli,  num, total
FROM clientes, cabecera_nota_venta
WHERE ci_cli =  '123456789' AND
          ci_cli=ci_cliente;
```

2. Listar la cantidad de libros con el título “Al Faro” que hay en el almacén Juan Marcet.

```
SELECT nro_ejem
FROM almacenes a, libro_almacen la, libro l
WHERE a.cod_al = la.cod_al AND
        l.cod_lib = la.cod_lib AND
        titulo = ‘Al Faro‘ AND
        nom_al = ‘Juan Marcet‘;
```

3. Listar la nacionalidad del autor Fernando de Rojas.

```
SELECT nom_au, ape_au,  desc_nac
FROM autor a, nacionalidad n
WHERE a.cod_nac = n.cod_nac AND
         nom_au = ‘Fernando‘ AND
         ape_au = ‘de Rojas‘;
```

4. Mostrar el idioma y traductor del libro La Celestina.

```
SELECT desc_id, nom_tra, ape_tra, titulo
FROM idioma i, traductor t, libro l, libro_idioma_traductor lit
WHERE  lit.cod_lib = l.cod_lib AND
         i.cod_id = lit.cod_id AND
         t.cod_tra = lit.cod_tra AND
         titulo=‘La Celestina‘;
```

5. Calcular el total a pagar si comprara 4 libros “”La Metamorfosis”.

```
SELECT titulo, 4*PRECIO AS pagar
FROM libro
WHERE titulo=‘La Metamorfosis‘;
```

6. Listar la cantidad y título de los libros que tiene el almacén Monsalve.

```
SELECT titulo, nro_ejem
FROM libro, almacenes, libro_almacen
WHERE libro.cod_lib = libro_almacen.cod_lib AND
        Almacenes.cod_al = libro_almacen.cod_al AND
        Nom_al = 'Monsalve';
```

7. Listar el nombre y apellido de los empleados y el almacén al que pertenecen.

```
SELECT nom_emp, ape_emp, nom_al
FROM empleados, almacenes
WHERE empleados.cod_al = almacenes.cod_al;
```

8. Listar los almacenes cuyo total de libros supere los 30.

```
SELECT nom_al, SUM(nro_ejem)
FROM almacenes, libro_almacen
WHERE almacenes.cod_al = libro_almacen.cod_al
GROUP BY almacenes.cod_al
HAVING SUM(nro_ejem)>30;
```

9. Listar el total de libros vendidos en todas las facturas.

```
SELECT SUM(nro_uni)
FROM detalle_nota_venta;
```

10. Listar las fechas en las que fueron realizadas las notas de venta.

```
SELECT fecha_not
FROM cabecera_nota_venta;
```

11. Calcular el promedio, el valor máximo y mínimo de todas las ventas de las librerías.

```
SELECT AVG(total), MAX(total), MIN(total)
```

```
FROM cabecera_nota_venta;
```

12. Listar los números de las notas de venta realizadas en el mes de junio y el almacén en donde fueron emitidas.

```
SELECT num, nom_al
FROM cabecera_nota_venta, almacenes
WHERE MONTH(fecha_not)=6 AND
        almacenes.cod_al = cabecera_nota_venta.cod_al;
```

13. Listar el nombre de los empleados y el número de notas de venta que ha emitido.

```
SELECT nom_emp, ape_emp, COUNT(cabecera_nota_venta.cod_emp)
AS numero
FROM empleados, cabecera_nota_venta
WHERE empleados.cod_emp = cabecera_nota_venta.cod_emp
GROUP BY cabecera_nota_venta.cod_emp;
```

14. Listar el nombre de todos los autores con sus respectivos libros.

```
SELECT nom_au, ape_au, titulo
FROM libro, autor, autor_libro
WHERE libro.cod_lib = autor_libro.cod_lib AND
        autor.cod_au = autor_libro.cod_au;
```

15. Listar los clientes que comiencen con la letra P.

```
SELECT nom_cli, ape_cli
FROM clientes
WHERE nom_cli LIKE 'P%';
```

16. Listar todos los libros cuyo costo sea de 11 o 23 dólares.

```
SELECT titulo, precio
```

```
FROM libro
WHERE precio IN (11,23);
```

17. Liste los nombres de los libros ordenados de forma descendente.

```
SELECT titulo
FROM libro
ORDER BY titulo desc;
```

18. Listar el nombre del empleado cuyo total de las notas de venta emitidas es mayor o igual a 45 dólares o menor o igual a 12 dólares.

```
SELECT nom_emp, ape_emp, total
FROM empleados, cabecera_nota_venta
WHERE empleados.cod_emp = cabecera_nota_venta.cod_emp AND
        (total >=45 OR
        total <=12);
```

19. Listar el nombre del almacén que tenga más de 1 empleado.

```
SELECT nom_al, COUNT(empleados.cod_al)
FROM almacenes, empleados
WHERE almacenes.cod_al =empleados.cod_al
GROUP BY empleados.cod_al
HAVING COUNT(empleados.cod_al)>1;
```

20. Listar todos los tipos de cargos.

```
SELECT desc_car
FROM cargos;
```

21. Listar los empleados de las librerías con sus respectivos cargos y el almacén al que pertenecen.

```
SELECT nom_emp, ape_emp, desc_car, nom_al
```

```
FROM empleados, almacenes, cargo
WHERE empleados.cod_al = almacenes.cod_al AND
        Empleados.cod_car = cargo.cod_car;
```

22. Contar cuantos empleados tiene cada almacén.

```
SELECT nom_al, COUNT(empleados.cod_al)
FROM empleados, almacenes
WHERE empleados.cod_al = almacenes.cod_al
GROUP BY empleados.cod_al;
```

23. Listar el nombre de los gerentes de cada almacén.

```
SELECT nom_emp, ape_emp, nom_al
FROM empleados, almacenes, cargo
WHERE empleados.cod_al = almacenes.cod_al AND
        empleados.cod_car = cargo.cod_car AND
        cargo.desc_car = 'Gerente';
```

24. Listar los empleados cuyos apellidos empiecen con las letras M, P o C.

```
SELECT nom_emp, ape_emp
FROM empleados
WHERE ape_emp LIKE 'M%' OR
        ape_emp LIKE 'P%' OR
        ape_emp LIKE 'C%';
```

25. Listar los nombres de los clientes cuyo nombre tenga 4 caracteres.

```
SELECT nom_cli, ape_cli
FROM clientes
WHERE nom_cli LIKE '____';
```

26. Listar los nombres de los empleados cuyo apellido tenga 7 caracteres y se encuentre entre las letras A y M.

```
SELECT nom_emp, ape_emp
FROM empleados
WHERE ape_emp LIKE '_______' AND
ape_emp BETWEEN 'A' AND 'M';
```

4.7 Conclusiones

En este capítulo aprendimos usos complementarios para la sentencia *SELECT*, por medio de los cuales el estudiante se encuentra en capacidad de realizar diferentes tipos de consultas que aunque no sean de mayor complejidad, permitirán procesar y manipular de mejor manera la información. También se pueden realizar cálculos con fechas, rangos, agrupar la información en base a alguna condición, ordenar la información y aplicar funciones de tipo carácter o numéricas según se requiera en las columnas, para facilitar la obtención de los resultados deseados.

CAPITULO 5

SUBCONSULTAS Y SUBCONSULTAS ANIDADAS

INTRODUCCIÓN

En este capítulo se demostrará que las consultas pueden ser mucho más complejas, de acuerdo a la información que se desee obtener, para lo cual, es necesaria la utilización de subconsultas, de manera que, haciendo consultas sencillas una dentro de otra (anidando consultas), nos llevarán a la obtención de los resultados esperados o requeridos. Al final de este capítulo se propone una serie de subconsultas de las tablas generadas en el capítulo II, las cuales nos permitirán poner en práctica lo aprendido en este capítulo. Además de proporcionar una visión mucho más amplia de las potencialidades del lenguaje *SQL.*

5.1 Subconsultas

Una subconsulta consiste en una instrucción *SELECT* anidada dentro de otra instrucción, las que pueden ser *HAVING* o *WHERE*, estas retornan un valor único. Las Subconsultas pueden devolver varios tipos de resultados tales como:

- Una fila-una columna
- Varias filas-una única columna
- Varias columnas-una única o varias filas

La estructura de una subconsulta es la siguiente:

```
SELECT lista_de_columnas
FROM lista_de_tablas
WHERE <expresión> <condición> (SELECT nombre_columna
                               FROM lista_de_tablas
                               WHERE condición);
```

Por ejemplo vamos a realizar la consulta de los nombres de las cargas familiares del empleado Humberto Pons.

```
SELECT dep_nom
FROM carga_f
WHERE eci = (SELECT ci
             FROM empleado
             WHERE nombre='Humberto' AND
                   apellido='Pons');
```

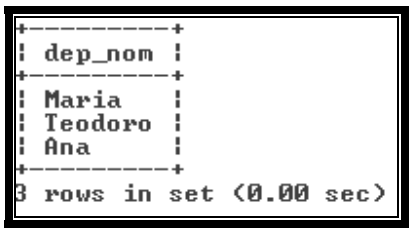


Figura 5.1. Resultado de la subconsulta de las cargas familiares del empleado Humberto Pons

Como podemos observar las subconsultas permiten realizar la unión de dos o más tablas, la anterior subconsulta puede realizarse también por medio de una consulta sencilla, la cual se realizaría de la siguiente forma:

```
SELECT dep_nom
FROM carga_f, empleado
WHERE eci=ci AND
      nombre='Humberto' AND
      apellido='Pons';
```

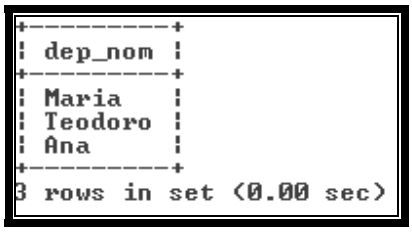


Figura 5.2. Resultado de la consulta de las cargas familiares del empleado Humberto Pons

Podemos observar que el resultado es exactamente el mismo, pero en otros casos es necesario realizar solamente subconsultas para obtener el resultado requerido.

5.2 Condiciones de Búsqueda en las Subconsultas

5.2.1 Test de Comparación (=, <>, <, >, <=,>=)

Compara la expresión principal con el resultado de la subconsulta y devuelve *TRUE* si la comparación es verdadera, si la subconsulta no devuelve ningún valor, el valor a devolver será *NULL*.

Ahora vamos a listar la cédula, nombre, apellido de todos los empleados que trabajan en el departamento de Investigación.

```
SELECT ci, nombre, apellido
FROM empleado
WHERE dno = (SELECT dnumero
             FROM departamento
             WHERE dnombre = 'Investigacion');
```

```
+-----------+----------+----------+
| ci        | nombre   | apellido |
+-----------+----------+----------+
| 123456789 | Juan     | Polo     |
| 333445555 | Humberto | Pons     |
| 453453453 | Marcia   | Mora     |
| 666884444 | Pablo    | Castro   |
+-----------+----------+----------+
4 rows in set (0.00 sec)
```

Figura 5.3. Empleados que trabajan en el departamento de investigación

5.2.2 Test de Conjunto (*IN*)

Este tipo de test recupera únicamente los registros de la consulta principal para los que algunos registros de la subconsulta contienen un valor igual.

Vamos a listar todos los empleados cuyas cargas familiares hayan nacido entre los años 1980 y 1990.

```
SELECT ci, nombre, apellido
FROM empleado
WHERE  ci IN (SELECT eci
              FROM carga_f
              WHERE YEAR(fechan_n) BETWEEN '1980' AND '1990');
```

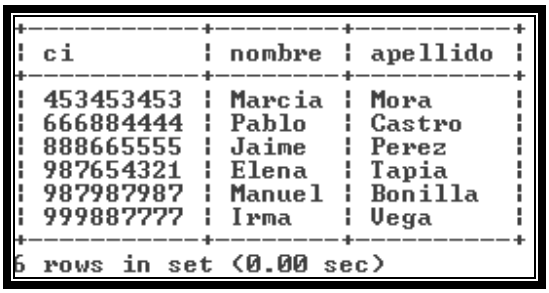
```
+-----------+----------+----------+
| ci        | nombre   | apellido |
+-----------+----------+----------+
| 123456789 | Juan     | Polo     |
| 333445555 | Humberto | Pons     |
+-----------+----------+----------+
2 rows in set (0.00 sec)
```

Figura 5.4. Empleados con cargas familiares nacidas entre 1980 y 1990

Por el contrario, también podemos utilizar *NOT IN*, que realiza la función inversa a *IN*, es decir, recupera únicamente los registros de la consulta principal para los que los registros de la subconsulta no contengan un valor igual.

Como ejemplo vamos a listar los empleados cuyas cargas familiares no hayan nacido entre los años 1980 y 1990.

```
SELECT ci, nombre, apellido
FROM empleado
WHERE  ci NOT IN (SELECT eci
                  FROM carga_f
                  WHERE YEAR(fechan_n) BETWEEN '1980'
                  AND '1990');
```



ci	nombre	apellido
453453453	Marcia	Mora
666884444	Pablo	Castro
888665555	Jaime	Perez
987654321	Elena	Tapia
987987987	Manuel	Bonilla
999887777	Irma	Vega

6 rows in set (0.00 sec)

Figura 5.5. Empleados cuyas cargas no hayan nacido entre 1980 y 1990

5.2.3 Test de Existencia (*EXISTS*)

Este comando de existencia es utilizado para comparaciones en las cuales el resultado va a ser verdadero o falso, dependiendo de si la subconsulta devuelve o no un registro respectivamente. Este tipo de test es utilizado únicamente en subconsultas.

A continuación vamos a listar los empleados que trabajan en el departamento nro. 4.

```
SELECT nombre, apellido, dno
FROM empleado
WHERE EXISTS (SELECT *
                FROM departamento
               WHERE dnumero = '4' AND
               dno=dnumero);
```

nombre	apellido	dno
Elena	Tapia	4
Manuel	Bonilla	4
Irma	Vega	4

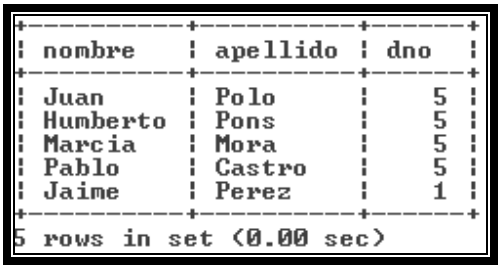
3 rows in set (0.00 sec)

Figura 5.6. Empleados que trabajan en el departamento 4

De la misma forma que el test de conjunto *IN*, el test de existencia tiene su forma inversa, la cual es *NOT EXISTS*.

Ahora vamos a listar los empleados que no trabajen en el departamento número 4.

```
SELECT nombre, apellido, dno
FROM empleado
WHERE NOT EXISTS (SELECT *
                  FROM departamento
                  WHERE dnumero = '4' AND
                        dno=dnumero);
```



nombre	apellido	dno
Juan	Polo	5
Humberto	Pons	5
Marcia	Mora	5
Pablo	Castro	5
Jaime	Perez	1

5 rows in set (0.00 sec)

Figura 5.7. Empleados que no trabajan en el departamento 4

5.2.4 Test Cuantificados

5.2.4.1 Test *ANY*

El test *ANY* recupera registros de la consulta principal, los cuales cumplan con la comparación de alguno de los registros de la subconsulta. Este tipo de *test* también puede realizarse con la palabra *SOME*.

Como ejemplo vamos a listar todos los departamentos en los cuales al menos un empleado gane 2500 dólares.

```
SELECT dnombre
FROM departamento
WHERE dnumero = ANY (SELECT dno
                     FROM empleado
                     WHERE salario = '2500');
```

Figura 5.8. Departamentos con al menos un empleado que gana 2500

5.2.4.2 Test *ALL*

El *test ALL* recupera solamente los registros de la consulta principal que cumplan con la comparación de todos los registros de la subconsulta.

Obtener una lista del nombre y apellido de todos los empleados cuyo salario sea mayor al de todos los supervisores.

```
SELECT e.nombre, e.apellido
        FROM empleado e
        WHERE   e.salario > ALL (SELECT s.salario
                                 FROM empleado s, empleado e
                                 WHERE e.superci=s.ci AND
                                       e.superci is not null);
```

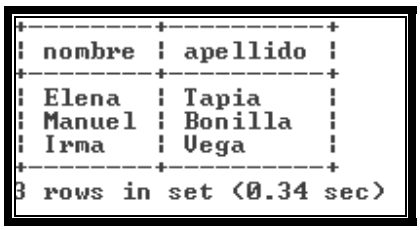
Figura 5.9. Empleados con salario mayor al de todos los supervisores

5.3 Subconsultas Anidadas

Son conocidas como Subconsultas Anidadas aquellas que tienen más de dos niveles, es decir, es una expresión dentro de otra subconsulta y esta puede estar a su vez dentro de otra y así sucesivamente sin límites.

Por ejemplo, vamos a listar los empleados que trabajan en el proyecto Beneficios.

```
SELECT nombre, apellido
FROM empleado
WHERE ci IN (SELECT eci
             FROM trabaja_en
             WHERE pno  IN (SELECT pnumero
                            FROM proyecto
                            WHERE pnombre='Beneficios'));
```



```
+--------+----------+
| nombre | apellido |
+--------+----------+
| Elena  | Tapia    |
| Manuel | Bonilla  |
| Irma   | Vega     |
+--------+----------+
3 rows in set (0.34 sec)
```

Figura 5.10. Empleados que trabajan en el proyecto beneficios

5.4 Ejercicios de Subconsultas de la Base de Datos Compania

1. Listar las cargas familiares de la empleada Elena Tapia.

```
SELECT dep_nom
FROM carga_f
WHERE eci = (SELECT ci
             FROM empleado
             WHERE nombre='Elena' AND
                   apellido='Tapia');
```

2. Listar los empleados que trabajan en el departamento Administrativo.

```
SELECT nombre, apellido
FROM empleado
WHERE dno IN (SELECT dnumero
              FROM departamento
              WHERE dnombre='Investigacion');
```

3. Listar los proyectos en los que los empleados tengan cargas familiares de sexo femenino.

```
SELECT pnumero, pnombre
FROM proyecto
WHERE pnumero IN (SELECT pno
                  FROM trabaja_en
                  WHERE eci in (SELECT eci
                                FROM carga_f
                                WHERE sexo='F'));
```

4. Listar todos los departamentos en los cuales todos sus empleados ganan entre 2500 y 4300 dólares.

```
SELECT dnombre
FROM departamento
WHERE dnumero IN (SELECT dno
                  FROM empleado
                  WHERE salario BETWEEN 2500 AND 4300);
```

5. Listar todos los empleados que no tengan cargas familiares.

```
SELECT ci, nombre, apellido
FROM empleado
WHERE NOT EXISTS (SELECT eci
                  FROM carga_f
                  WHERE ci=eci);
```

6. Listar los proyectos que pertenecen al departamento de Compras.

```
SELECT pnombre
FROM proyecto
WHERE dnum = (SELECT dnumero
              FROM departamento
              WHERE  dnombre='Compras');
```

7. Listar los empleados que trabajan en el proyecto ProductoZ.

```
SELECT nombre, apellido
FROM empleado
WHERE ci IN (SELECT eci
             FROM trabaja_en
             WHERE pno IN (SELECT pnumero
                           FROM proyecto
                           WHERE pnombre =  'ProductoZ'));
```

8. Listar el nombre de los empleados que pertenezcan a más de un proyecto.

```
SELECT nombre, apellido
FROM empleado
WHERE ci IN (SELECT eci
             FROM trabaja_en
             GROUP BY eci
             HAVING count(eci)>1);
```

9. Listar todos los proyectos que pertenecen al departamento ubicado en Quito.

```
SELECT pnumero, pnombre
FROM proyecto
WHERE dnum = (SELECT dnumero
              FROM localizacion
              WHERE dep_loca= 'Quito');
```

10. Liste todos los empleados y sus respectivas cargas siempre y cuando hayan nacido en el mismo mes.

```
SELECT nombre, apellido
FROM empleado
WHERE  ci = (SELECT eci
             FROM carga_f
             WHERE MONTH(fecha_n) =MONTH(fechan_n));
```

11. Listar el nombre del departamento que tiene más de 3 empleados.

```
SELECT dnombre
FROM departamento
WHERE  EXISTS (SELECT count(ci)
                    FROM empleado
                    WHERE dno=dnumero
                    GROUP BY dno
                    HAVING count(ci)>3);
```

12. Listar el departamento con el mayor salario.

```
SELECT dnombre
FROM departamento, empleado
WHERE dno=dnumero
GROUP BY dno
HAVING AVG(salario)>=ALL(SELECT AVG(salario)
                                FROM empleado
                                GROUP BY dno);
```

13. Listar el nombre de los empleados que por cada proyecto han trabajado más horas que las trabajadas por cada empleado en el proyecto número 30.

```
SELECT nombre, apellido
FROM empleado, trabaja_en
WHERE ci=eci
GROUP BY eci
HAVING SUM(horas)>ALL(SELECT horas
                             FROM trabaja_en
                             WHERE pno='30');
```

5.5 Ejercicios de Subconsultas de la Base de Datos Libreria

1. Listar la cantidad de libros “La Celestina” que existen en cada uno de los almacenes.

```
SELECT nom_al, nro_ejem
FROM almacenes a, libro_almacen la
WHERE a.cod_al = la.cod_al AND
        la.cod_lib IN (SELECT cod_lib
                       FROM libro
                       WHERE titulo = ‘La Celestina’);
```

2. Listar la nacionalidad del autor Franz Kafka.

```
SELECT desc_nac
FROM nacionalidad
WHERE cod_nac = (SELECT cod_nac
                 FROM autor
                 WHERE nom_au = ‘Franz’ AND
                       ape_au = ‘Kafka’);
```

3. Mostrar el idioma y traductor del libro “Al Faro”.

```
SELECT desc_id, nom_tra, ape_tra
FROM idioma i, traductor t, libro_idioma_traductor lit
WHERE i.cod_id = lit.cod_id AND
      t.cod_tra = lit.cod_tra AND
      lit.cod_lib = (SELECT cod_lib
                     FROM libro
                     WHERE titulo =’Al Faro’);
```

4. Listar la cantidad de libros “La Metamorfosis” que tiene el almacén Monsalve.

```
SELECT nro_ejem
FROM libro_almacen
WHERE cod_lib IN (SELECT cod_lib
                  FROM libro
                  WHERE titulo = 'La Metamorfosis' AND
                        cod_lib IN (SELECT cod_lib
                                    FROM libro_almacen la, almacenes a
                                    WHERE la.cod_al = a.cod_al AND
                                          a.nom_al = 'Monsalve'));
```

5. Listar el nombre, apellido y cargo de los empleados que pertenecen al almacén Juan Marcet.

```
SELECT nom_emp, ape_emp, desc_car
FROM empleados, cargo
WHERE empleados.cod_car = cargo.cod_car AND
      empleados.cod_al IN (SELECT cod_al
                           FROM almacenes
                           WHERE nom_al = 'Juan Marcet');
```

6. Listar el almacén con el mayor número de libros.

```
SELECT nom_al, SUM(nro_ejem)
FROM almacenes, libro_almacen
WHERE almacenes.cod_al = libro_almacen.cod_al
GROUP BY almacenes.cod_al
HAVING SUM(nro_ejem)>=ALL(SELECT SUM(nro_ejem)
                          FROM libro_almacen
                          GROUP BY cod_al);
```

7. Listar los almacenes donde no exista administrador.

```
SELECT nom_al
FROM almacenes
WHERE NOT EXISTS (SELECT c.cod_car
```

```
                                    FROM cargo c, empleados e
                                    WHERE desc_car = ‘administrador’ AND
                                          e.cod_car = c.cod_car AND
                                          e.cod_al =almacenes.cod_al);
```

8. Listar el nombre del empleado cuyo total de notas de venta emitidas sea mayor a 50 dólares.

```
SELECT nom_emp, ape_emp
FROM empleados
WHERE cod_emp IN (SELECT cod_emp
                  FROM cabecera_nota_venta
                  WHERE total >50);
```

9. Listar el nombre y apellido del gerente del almacén Papelesa.

```
SELECT nom_emp, ape_emp
FROM empleados
WHERE cod_car =(SELECT cod_car
                FROM cargo
                WHERE desc_car =’gerente’ AND
                      Cod_al =(SELECT cod_al
                               FROM almacenes
                               WHERE nom_al=’Papelesa’));
```

10. Listar la editorial del libro “La Metamorfosis”.

```
SELECT nom_ed
FROM editorial, editorial_libro
WHERE editorial.cod_ed = editorial_libro.cod_ed AND
      editorial_libro.cod_lib IN(SELECT cod_lib
                                 FROM libro
                                 WHERE titulo=’La Metamorfosis’);
```

5.6 Conclusiones

Muchas veces se puede obtener los mismos resultados por medio de las consultas simples o subconsultas, pero en algunos casos es necesario utilizar estas últimas para llegar a los resultados deseados. También por medio de las subconsultas se puede plantear la obtención de un resultado de manera más fácil y organizada que beneficie a la productividad y eficiencia en la gestión de bases de datos.

Otro punto importante dentro de este capítulo a más de la utilización de las subconsultas, es el poder categorizar o clasificar la información que queremos obtener mediante la utilización de parámetros de comparación, de inclusión, de existencia y otros mencionados en este capítulo. Mediante la realización de la práctica existente en este capítulo, el estudiante ya se encuentra en capacidad de manejar las subconsultas y parámetros relacionados a estas, con lo que, podrá gestionar de mejor manera las consultas a las bases de datos.

CAPITULO 6

UTILIZACION DE SCRIPTS Y TRANSACCIONES

INTRODUCCIÓN

En este capítulo se tratarán brevemente aspectos fundamentales como el uso de *scripts* para evitar tener que escribir todo el código cada vez que tengamos que hacer una tarea repetitiva. Aprenderemos también la utilización de los comandos *ROLLBACK* y *COMMIT*, los cuales nos ayudarán a mantener la integridad de la base de datos y la información contenida en esta.

6.1 Procesamiento por Lotes o Modo Batch

El procesamiento por lotes o modo *batch* consiste en ejecutar un programa o un *script* (procedimiento) sin necesidad de que el usuario interactúe, controle o realice la operación de forma directa, es decir sin necesidad de escribir una y otra vez las sentencias que requiere utilizar.

Este tipo de procesamientos son recomendables cuando se posee gran información, ya que, al utilizar un *script* nos ahorraríamos tiempo en escribir nuevamente todas las instrucciones que se encuentran incluidas en el *script*.

A continuación vamos a proceder a borrar la base de datos compania para crearla nuevamente con un script que hemos realizado y guardado con el nombre compania.sql, en el cual, ya están las instrucciones para la creación de la base de datos, tablas, llaves primarias y foráneas y la inserción de datos en las tablas, que ya realizamos en el transcurso del desarrollo de este tutorial.

Para borrar la base de datos ejecutamos la siguiente instrucción:

DROP DATABASE compañía;

Para poder ejecutar los *Scripts*, es preferible que estos se encuentren ubicados en la siguiente dirección:

C:\Archivos de programa*MySQL**MySQL* Server 5.0\bin

Ahora procedemos a ejecutar el script compania.sql, para lo cual usamos la siguiente sentencia:

\. C:\Archivos de programa*MySQL**MySQL* Server 5.0\bin\compania.sql

Podemos observar que todos los datos han sido nuevamente creados.

El contenido del Script es el siguiente:

```
/*************************************BASE DE DATOS*****************************************/
create database compania;
use compania;
/*************************************CREACION DE TABLAS*******************************/
CREATE TABLE empleado       (nombre     VARCHAR (30),
                             apellido   VARCHAR (30),
                             ci         VARCHAR (10)       NOT NULL,
                             fecha_n    DATE,
                             direccion  VARCHAR (30),
                             sexo       CHAR (1),
                             salario    INTEGER (4),
                             superci    VARCHAR (10),
                             dno        INTEGER (1),
                             PRIMARY KEY(ci));

CREATE TABLE departamento   (dnombre    VARCHAR (30)       UNIQUE,
```

```
                                dnumero     INTEGER (1)          NOT NULL,
                                jefeci      VARCHAR (10),
                                jefe_fi     DATE,
                                PRIMARY KEY(dnumero),
                                FOREIGN KEY(jefeci)
                                REFERENCES empleado(ci)
                                ON DELETE SET NULL
                                ON UPDATE CASCADE);

CREATE TABLE localizacion     (dnumero     INTEGER (1)          NOT NULL,
                               dep_loca    VARCHAR (30),
                               FOREIGN KEY(dnumero)
                               REFERENCES departamento(dnumero)
                               ON DELETE CASCADE
                               ON UPDATE CASCADE);

CREATE TABLE proyecto      (pnombre      VARCHAR (30)         UNIQUE,
                            pnumero      INTEGER (2)          NOT NULL,
                            plocal       VARCHAR (30),
                            dnum         INTEGER (1),
                            PRIMARY KEY(pnumero),
                            FOREIGN KEY(dnum)
                            REFERENCES departamento(dnumero)
                            ON DELETE SET NULL
                            ON UPDATE CASCADE);

CREATE TABLE trabaja_en    (eci          VARCHAR (10)         NOT NULL,
                            pno          INTEGER (2)          NOT NULL,
                            horas        DOUBLE (4,2),
                            FOREIGN KEY(eci)
                            REFERENCES empleado (ci)
                            ON DELETE CASCADE
                            ON UPDATE CASCADE,
                            FOREIGN KEY (pno)
```

```
                        REFERENCES proyecto (pnumero)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE);

CREATE TABLE carga_f    (eci        VARCHAR (10)    NOT NULL,
                        dep_nom     VARCHAR (30),
                        sexo        VARCHAR (1),
                        fechan_n    DATE,
                        relacion    VARCHAR (10),
                        FOREIGN KEY(eci)
                        REFERENCES empleado(ci)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE);
/***************************************LLAVES FORANEAS*******************************/
ALTER TABLE empleado
ADD FOREIGN KEY (dno)
REFERENCES departamento (dnumero)
ON DELETE SET NULL
ON UPDATE CASCADE,
ADD FOREIGN KEY (superci)
REFERENCES empleado (ci)
ON DELETE SET NULL
ON UPDATE CASCADE;
/***********************************INGRESO DE DATOS*********************************/
INSERT INTO `empleado` (`NOMBRE`, `APELLIDO`, `CI`, `FECHA_N`, `DIRECCION`,
`SEXO`, `SALARIO`)
VALUES ('Juan','Polo','123456789','1959-03-03','Sucre 7-12','M',3000),
        ('Humberto','Pons','333445555','1960-12-25','Bolivar 5-67','M',4000),
        ('Marcia','Mora','453453453','1960-03-29','Colombia 4-23','F',2500),
        ('Pablo','Castro','666884444','1955-09-15','Bolivar 1-50','M',3800),
        ('Jaime','Perez','888665555','1957-04-05','Sangurima 8-34','M',5500),
        ('Elena','Tapia','987654321','1961-05-03','Ordo¤ez 7-29','F',4300),
        ('Manuel','Bonilla','987987987','1958-07-16','B. Malo 1-10','M',2500),
        ('Irma','Vega','999887777','1950-11-13','P. Cordova 3-45','F',2500);
```

```
INSERT INTO departamento (`DNOMBRE`,`DNUMERO`,`JEFECI`,`JEFE_FI`)
VALUES ('Compras', 1, '333445555', '1978-06-06'),
        ('Administrativo', 4, '987654321', '1982-12-05'),
        ('Investigacion', 5, '888665555', '1980-12-05');

INSERT INTO localizacion (`DNUMERO`,`DEP_LOCA`)
VALUES (4, 'Guayaquil'),
        (5, 'Quito'),
        (5, 'Manta'),
        (5, 'Cuenca'),
        (1, 'Cuenca');

INSERT INTO proyecto (`PNOMBRE`, `PNUMERO`, `PLOCAL`, `DNUM`)
VALUES ('ProductoX', 1, 'Quito', 5),
        ('ProductoY', 2, 'Manta', 5),
        ('ProductoZ',3,'Cuenca', 5),
        ('Computadora', 10, 'Guayaquil', 4),
        ('Reorganizar', 20, 'Cuenca', 1),
        ('Beneficios', 30, 'Guayaquil', 4);

INSERT INTO `trabaja_en` (`ECI`, `PNO`, `HORAS`)
VALUES ('123456789', 1, 12.5),
        ('123456789', 2, 15.6),
        ('666884444', 3, 14.7),
        ('453453453', 1, 10),
        ('453453453', 2, 10),
        ('333445555', 2, 20),
        ('333445555', 3, 10),
        ('333445555', 10, 10),
        ('333445555', 20, 10),
        ('999887777', 30, 30),
        ('999887777', 10, 5),
        ('987987987', 10, 15),
```

```
('987987987', 30, 17),
('987654321', 30, 10),
('987654321', 20, 12),
('888665555', 20, NULL);

INSERT INTO carga_f (`ECI`, `DEP_NOM`, `SEXO`, `FECHAN_N`, `RELACION`)
VALUES ('333445555', 'Maria','F', '1986-02-02', 'Hija'),
('333445555', 'Teodoro', 'M', '1990-10-10', 'Hijo'),
('333445555', 'Ana', 'F', '1965-09-15', 'Conyuge'),
('987654321', 'Alberto', 'M', '1967-07-06', 'Conyuge'),
('123456789', 'Miguel', 'M', '1984-11-05', 'Hijo'),
('123456789', 'Maria', 'F', '1987-01-09', 'Hija'),
('123456789', 'Elizabeth', 'F', '1960-12-12', 'Conyuge');

UPDATE empleado SET superci='333445555', dno='5' where ci='123456789';
UPDATE empleado SET superci='888665555', dno='5' where ci='333445555';
UPDATE empleado SET superci='987654321', dno='4' where ci='999887777';
UPDATE empleado SET superci='888665555', dno='4 ' where ci='987654321';
UPDATE empleado SET superci='333445555', dno='5 ' where ci='666884444';
UPDATE empleado SET superci='333445555', dno='5 ' where ci='453453453';
UPDATE empleado SET superci='987654321', dno='4 ' where ci='987987987';
UPDATE empleado SET dno='1' where ci='888665555';
```

6.2 Transacciones

Las transacciones facilitan el procesamiento de la información, sin una interacción directa con el usuario. Las transacciones empiezan con la instrucción *START TRANSACTION* y nos ayudan a mantener la integridad de los datos por medio de las instrucciones *COMMIT* y *ROLLBACK.*

6.2.1 Comando COMMIT

La instrucción *COMMIT* se utiliza para finalizar una transacción exitosa, esta guardará y comprometerá todos los datos manipulados en la transacción. El formato es el siguiente:

```
START TRANSACTION;
…
COMMIT;
```

Por ejemplo vamos a realizar una transacción que cree una tabla auxiliar y que se guarde en esta toda la información de la tabla departamento.

```
START TRANSACTION;
CREATE TABLE auxiliar (dnombre     VARCHAR (30),
                       dnumero     INTEGER (1)          NOT NULL,
                       jefeci      VARCHAR (10),
                       jefe_fi     DATE);
INSERT INTO auxiliar (`DNOMBRE`,`DNUMERO`,`JEFECI`,`JEFE_FI`)
VALUES ('Compras', 1, '333445555', '1978-06-06'),
       ('Administrativo', 4, '987654321', '1982-12-05'),
       ('Investigacion', 5, '888665555', '1980-12-05');
COMMIT;
```

6.2.2 Comando ROLLBACK

La instrucción *ROLLBACK* se utiliza para finalizar una transacción, pero dejando la base de datos tal cual como se encontraba antes de que empiece la transacción. El formato es el siguiente:

```
START TRANSACTION;
…
ROLLBACK;
```

Por ejemplo vamos a insertar nuevos datos en la tabla auxiliar para luego aplicar el comando *ROLLBACK*.

```
START TRANSACTION;
INSERT INTO auxiliar (`DNOMBRE`,`DNUMERO`,`JEFECI`,`JEFE_FI`)
VALUES ('Bodegas', 4, '333445555', '1978-06-06'),
        ('Ventas', 5, '333445555', '1978-06-06'),
        ('Auditoria', 6, '333445555', '1978-06-06');
ROLLBACK;
```

Como podemos observar, al realizar un select * de la tabla auxiliar, los datos no fueron insertados, demostrando así el uso del comando antes mencionado.

6.3 Conclusiones

Mediante el uso de los *scripts* podemos ahorrar tiempo, debido a que no es necesario volver a escribir una y otra vez el código en tareas repetitivas, también mediante esta técnica podemos tener un respaldo con todas las operaciones realizadas en la base de datos en caso de que esta sufra algún daño.

Las transacciones se pueden utilizar para realizar procedimientos en los cuales se tiene que acceder a un gran volumen de información, facilitando de esta manera la manipulación de la misma, además de garantizar la integridad de los datos por medio del comando *ROLLBACK,* que nos puede facilitar la recuperación de los datos en caso de producirse un error, manteniendo así la integridad de la base de datos y de la información que esta contiene. De igual manera, utilizando el comando *COMMIT*, se compromete la transacción, con lo que los datos quedan consolidados en nuestra base de datos, brindando de esta manera seguridad y garantizando la integridad referencial.

CAPITULO 7

CONCLUSIONES

7.1 Conclusiones Teóricas

Esta monografía propende a la correcta instalación, configuración y manipulación del Gestor de Bases de Datos *MySQL, haciendo* todo esto con un enfoque paso a paso, con lo que el alumno podrá instalar el mencionado gestor de una forma fácil y segura.

Se puede seguir la instalación y configuración del gestor de una forma fácil y sencilla mediante los gráficos incluidos en el capítulo I, por lo que no da margen para un error durante la instalación. En los capítulos consecuentes se explica de manera pormenorizada la utilización del lenguaje *SQL*, aplicada a la gestión de bases de datos y aplicada a la gestión de datos, objetivo primordial de esta monografía.

7.2 Conclusiones Metodológicas

Para la realización de esta monografía, se utilizó una metodología que consistió en la recopilación de los conocimientos adquiridos durante nuestra vida universitaria, refiriéndonos principalmente a la materia de Bases de Datos, además de investigar utilizando diferentes medios como son internet, manuales, revistas y medios relacionados a las bases de datos, debido a la gran cantidad de información existente se tomo la información de las fuentes más relevantes y reconocidas a nuestra disposición.

Una vez recopilada toda esta información, se procedió a clasificarla y ordenarla, para después acoplarla a un formato de tutorial para que el mismo pueda ser aplicado en el aprendizaje de bases de datos.

7.3 Conclusiones Pragmáticas

Uno de los objetivos primordiales de esta monografía fue el de crear un tutorial práctico para la enseñanza de bases de datos, que presenta un amplio marco teórico y conceptual, con prácticas puntualizadas a explicar la aplicación de uno u otro comando y una práctica general, con la cual se puedan aplicar todos los conocimientos adquiridos durante la realización de este manual.

Una aplicación potencial a futuro, es la inclusión de este tutorial de manejo de bases de datos al pensum actual de la carrera de Ingeniería de Sistemas de la Universidad del Azuay, precisamente a la materia de Bases de Datos, este tutorial, puede ser utilizado como material de consulta teórico-práctica o como material de seguimiento curricular para la mencionada materia.

REFERENCIAS

Capítulo 2

- Elmasri, Navathe (2003). *Fundamentals of Database Systems*. Benjamin Cumming Publishing.

BIBLIOGRAFÍA

- Korth, Silberchatz (2006). *Fundamentos de Bases de Datos.* 5ta Edición. McGraw-Hill.
- Connolly, Begg (2005), *Sistemas de bases de datos*, 4ta Edicion, Pearson
- Elmasri, Navathe (2003). *Fundamentals of Database Systems*. Benjamin Cumming Publishing.
- [Groff y Weinberg 2002] James R. Groff y Paul N. Weinberg, SQL Tutorial de Referencia*.* McGraw- Hill.
- Kroenke (2003). *Procesamiento de Bases de Datos, Fundamentos Diseño e Instrumentación.* 8ta Edición. Prentice may
- Adad, Medina, Careaga (1993). *Fundamentos de las Estructuras de Datos Relacionales.* Megabyte.
- [Moreno y otros 2002] Pilar Moreno, Iñigo Molina, Santiago Ormeño, *Curso de Fundamentos de sistemas de información geográfico,* Universidad Politécnica de Madrid, Cepade.
- [Piattini y otros 1996] Mario Piattini, José Antonio Calvo, Joaquín Cervera, Luis Fernando Sanz, *Análisis y diseño detallado de aplicaciones informáticas de gestión.* Ra-Ma.
- [Ramos 2002] Humberto Ramos, *Curso de Gestión de Bases de Datos,* Universidad Politécnica de Madrid, Cepade.
- Texto Guía “Fundamentos de Bases de Datos”
- http://dev.MySQL.com/doc/refman/5.0/es/tutorial.html

  Citado [20/03/2008 11h50]

- http://dev.MySQL.com/doc/refman/5.0/es/scalar-subqueries.html
  Citado [20/03/2008 12h40]

- http://MySQL.conclase.net/curso/index.php?sen=SELECT
  Citado [20/03/2008 15h00]

- http://MySQL.netvisao.pt/doc/refman/5.0/es/subquery-restrictions.html
  Citado [01/04/2008 14h45]

- http://www.programacion.net/bbdd/tutorial/sql/7/
  Citado [01/04/2008 15h10]

- http://www.ayc.unavarra.es/vmohedano/gbd/documentos/0506SQL_SUBCONSULTAS.pdf
  Citado [05/04/2008 10h30]

- http://www.xtec.es/~acastan/textos/Administracion%20de%20MySQL.html
  Citado [06/04/2008 17h00]

- http://www.programatium.commysql.htm
  Citado [11/04/2008 18h15]

ANEXO I

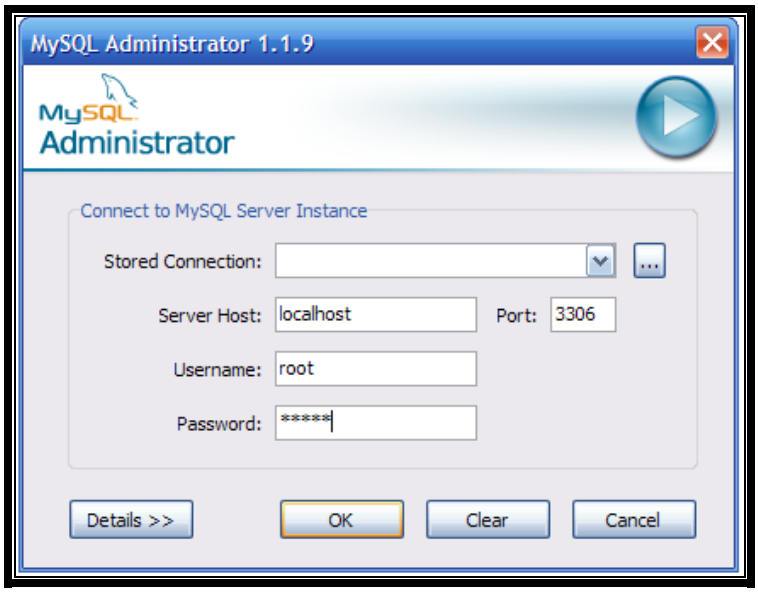
CREAR UNA BASE DE DATOS USANDO *MYSQL* ADMINISTRATOR Y *MYSQL* QUERY BROWSER.

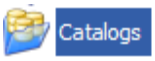
CREACIÓN DE TABLAS

Primero nos conectamos al servidor usando *MySQL* Administrator, Inicio > Todos los Programas > *MySQL* > *MySQL* Administrator

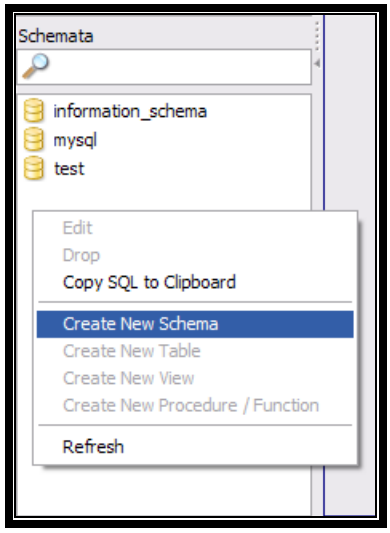
Luego aparece una ventana para realizar la conexión, en *Server Host* escribimos *localhost* o 127.0.0.1, en *Port*: dejamos el que está por defecto o en caso de haber configurado otro ponemos el número de puerto correspondiente. En *User*: *root*, y en *Password: admin* y damos clic en OK.

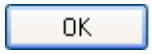
Se debe tener precaución al momento de ingresar el usuario y la contraseña ya que estos campos son sensibles a la mayúsculas y minúsculas.



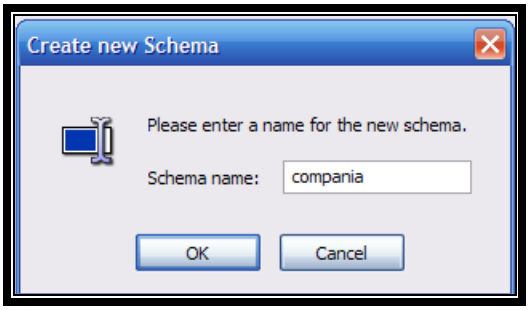
Luego vamos a 

, en la parte inferior damos clic derecho y en el menú contextual seleccionamos *Create New Schema.*

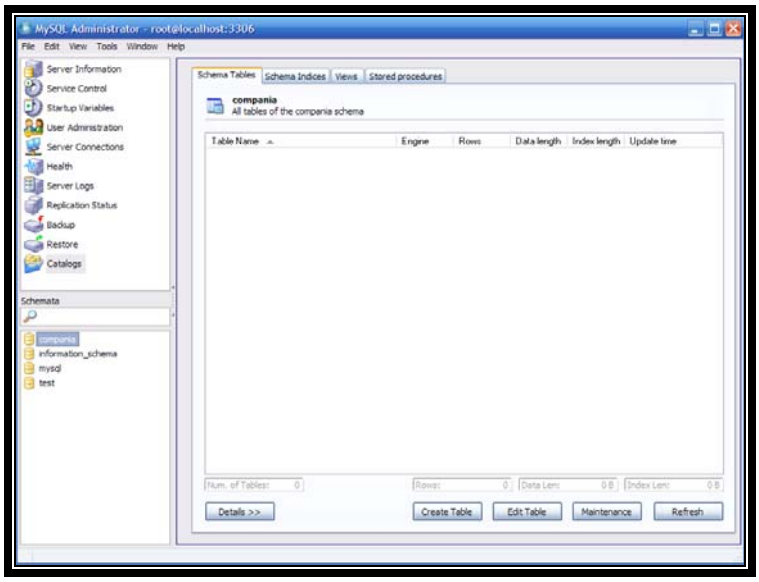


Aparece entonces una ventana en la cual nos pide ingresar el nombre de la base de datos a crear, en este caso será compania y damos clic en 

.



Luego en la parte donde se encuentran todas las bases de datos aparecerá compania, damos un clic y se mostrarán en la ventana todas las opciones disponibles.

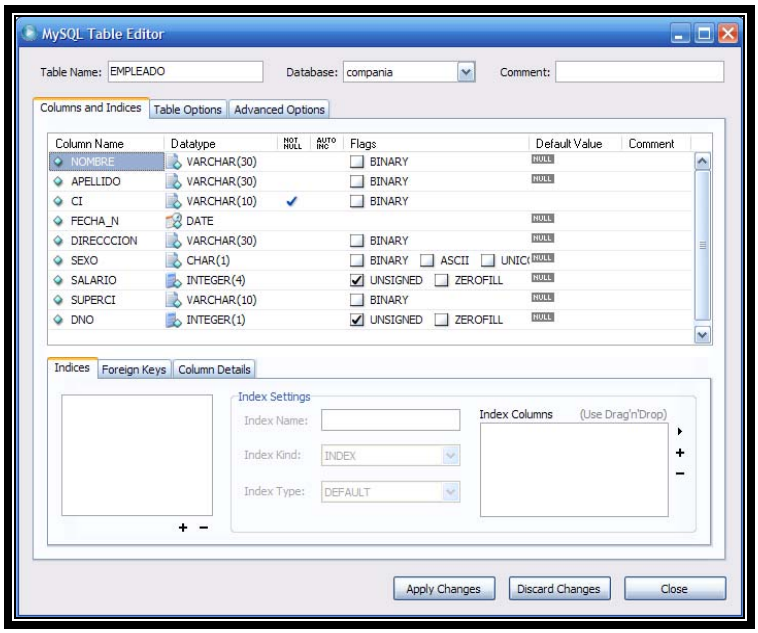


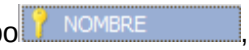
En la pestaña *Schema Tables* podemos apreciar que el campo donde se muestran las tablas está vacío, por lo tanto procederemos a la creación de las mismas. Damos entonces clic en el botón 

. Aparece entonces la ventana *Table Editor* en donde crearemos las columnas de la tabla.

TABLA EMPLEADO

El campo *Database:* debe tener compania, en el campo *Columns and Indices* procederemos a ingresar una por una el nombre, tipo y longitud de cada una de las columnas que conformarán la tabla en cuestión, como se muestra en el gráfico. Para poder ingresar un campo se debe dar doble clic debajo de *Column Name*.



Al ingresar la primera columna; en este caso NOMBRE, el gestor puede asumir que es la llave primaria, para distinguirla se mostrará una llave amarilla al lado del campo 

, para volverla una columna normal daremos un clic sobre la llave y se volverá un rombo 

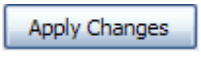
.

Si se desea que no se permita en una columna el valor *NULL* marcaremos con un clic en la columna 

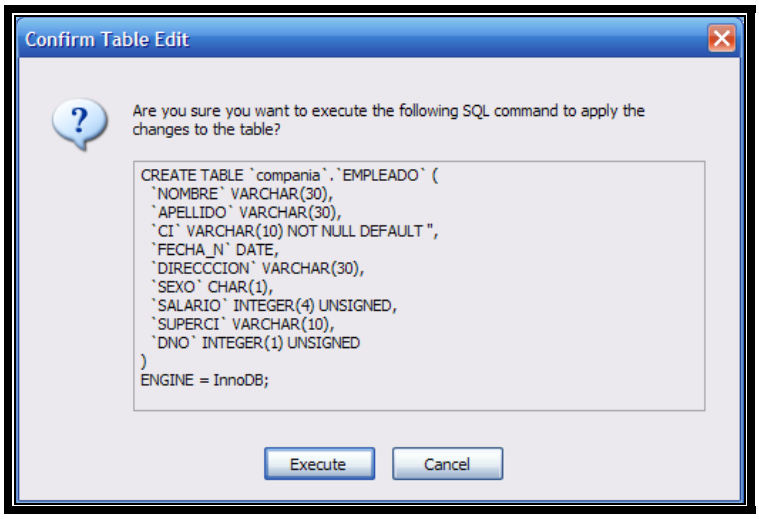
junto al campo que deseamos restringir, caso contrario no marcamos la casilla.

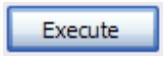
En caso de querer que un campo sea auto incremental En la columna 

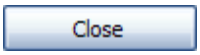
marcaremos en campo que se desea. Esto se utiliza generalmente en llaves.

Una vez ingresado todo lo requerido damos clic en 

. Se mostrará entonces una ventana en donde se muestra la sentencia *SQL* de la creación de la tabla en cuestión.

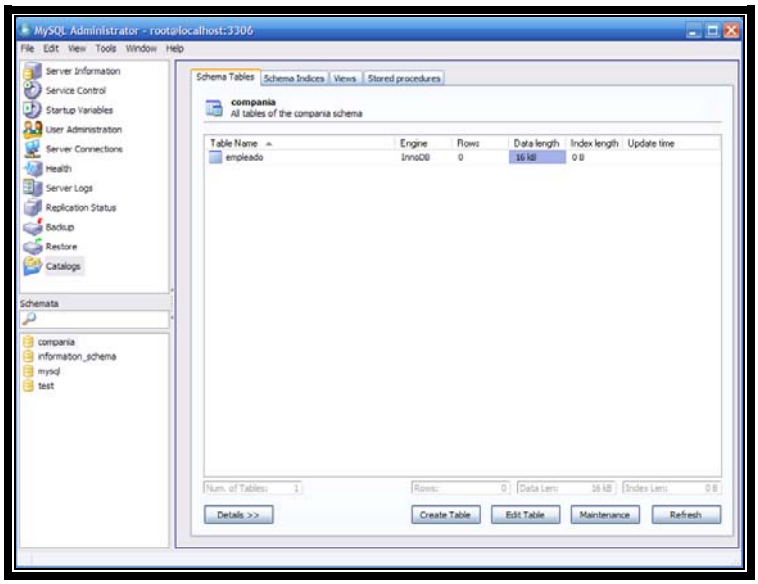


Entonces damos clic en 

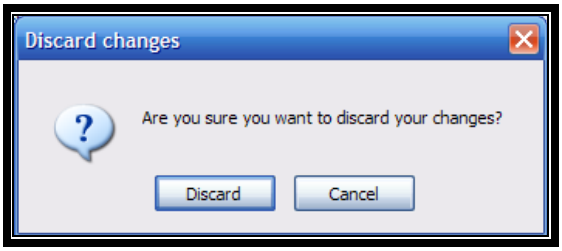
y se procederá con la creación de la tabla, volverá entonces a la ventana *Table Editor* donde ingresamos las columnas, aplicaremos entonces 

y la tabla quedará creada.

Se mostrará entonces nuevamente la ventana principal del administrador, en donde en la pestaña *Schema Tables* del esquema compania se mostrará ya la tabla creada.

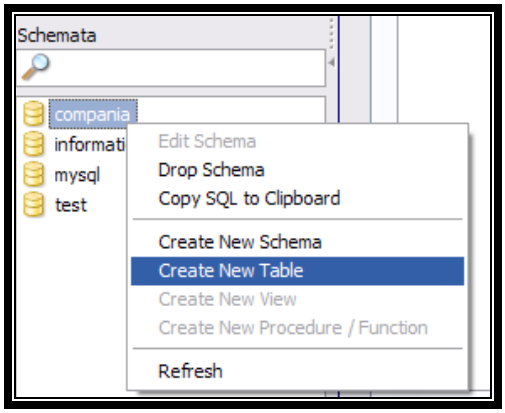


En caso de no querer crear la tabla procederemos con Discard Changes en *Table Editor,* o en su defecto si es que está en la ventana donde se muestra la instrucción *SQL* primero usaremos Cancel. Al dar clic en Discard Changes aparecerá una ventana de confirmación en la cual daremos en Discard.

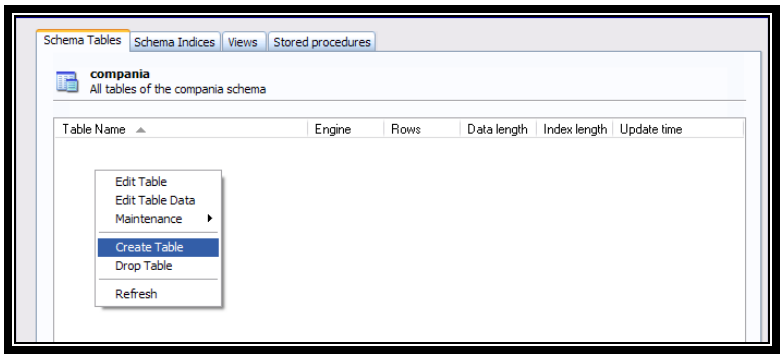


Si se quiere ver la estructura de la tabla damos doble clic sobre empleado se mostrara la ventana *Table Editor* con todas las columnas, es la misma que en la que ingresamos las columnas al crear la tabla. Para cerrarla damos clic en Close.

Otra manera es con clic derecho sobre compania, en el menú contextual damos clic en *Create New Table* y seguimos los mismos pasos anteriores de creación de tablas.



También se puede proceder dando clic derecho en el área donde se muestran las tablas y en el menú contextual dar clic en Create Table.



Procederemos a crear de igual manera cada una de las tablas del modelo en cuestión.

Tabla DEPARTAMENTO

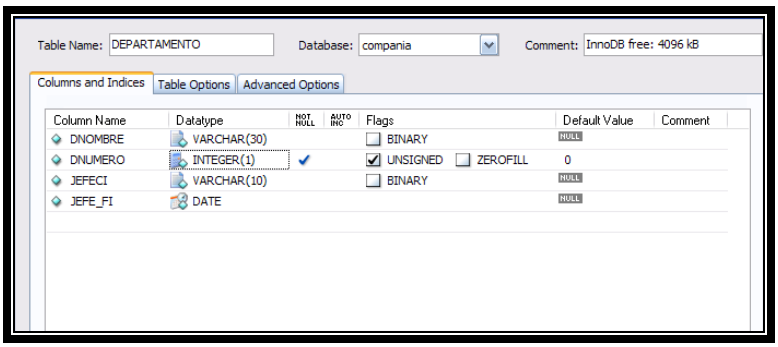


Tabla LOCALIZACION

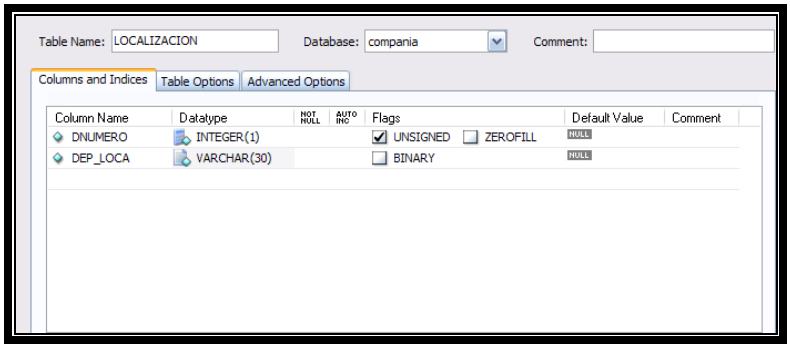
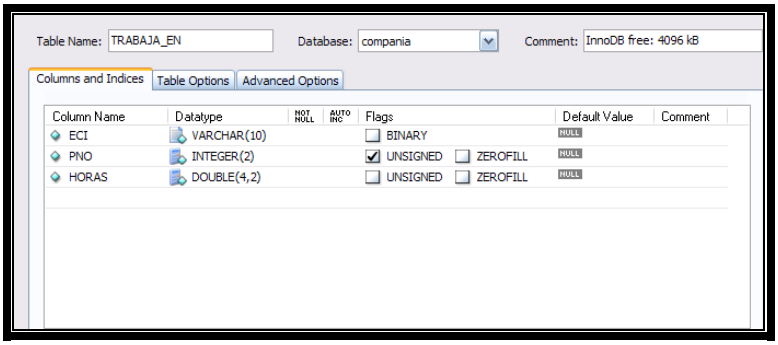
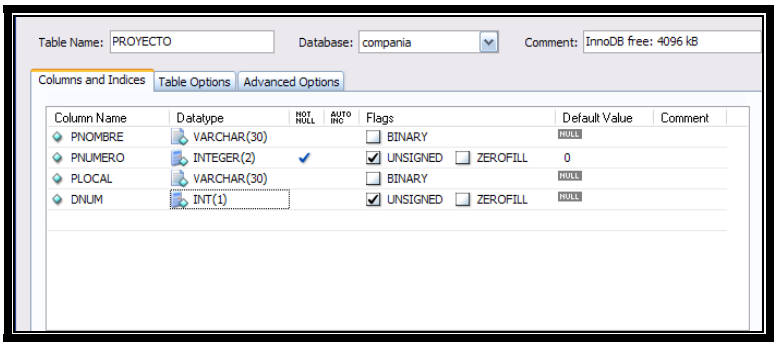


Tabla TRABAJA_EN

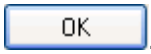


Se debe tener en cuenta el punto decimal al definir campos *DOUBLE.*

TABLA PROYECTO

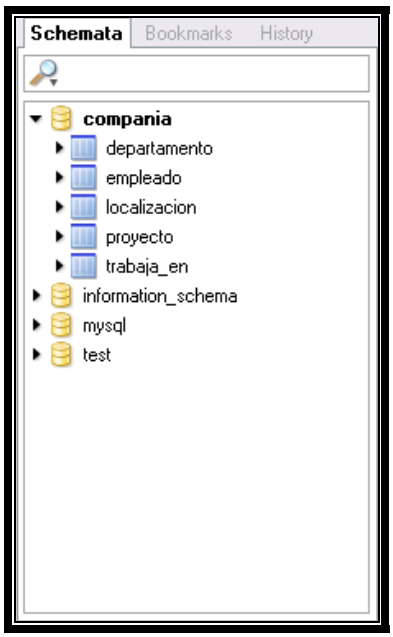


También podemos crear tablas desde el *Query Browser*. Ingresamos por inicio, Todos los Programas *MySQL*, *MySQL* Query *Browser*, Luego nos conectamos a la base de datos.

En *Server Host: localhost* o 127.0.0.1, en *Port*: 3306 en *Username: root*, en *Password*: admin, en *Default Schema*: compania y por último 

.

Una vez conectados a la base de datos, en la Pestaña *Schemata* damos doble clic en compania, esta se marcara en letras negritas.

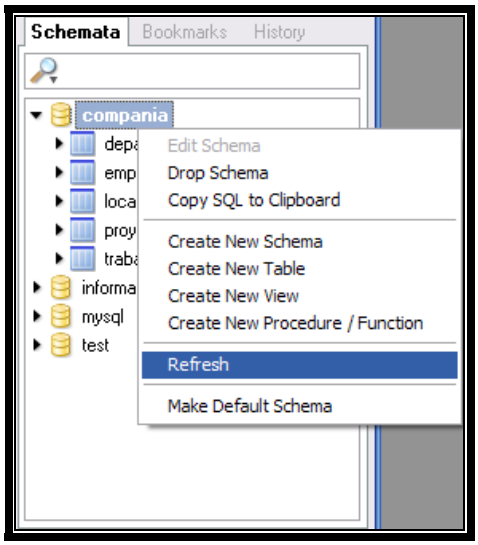


Vamos entonces a crear la tabla CARGA_F, en el campo de línea de comando escribiremos la sintaxis para crear una tabla:

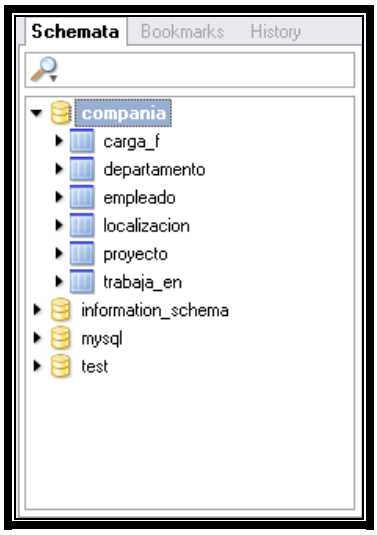
```
CREATE TABLE CARGA_F (`ECI` VARCHAR(10) NOT NULL,
                      `DEPNOM` VARCHAR(30),
                       `SEXO` VARCHAR(1),
                       `FECHAN_N` DATE,
                      `RELACION` VARCHAR(10))
```

Luego daremos clic en  y en la parte inferior del campo donde se muestran los resultados de las consultas se mostrará lo siguiente:

Luego en la pestaña *Shemata* marcamos con un clic compania, luego damos un clic derecho y en el menú contextual, escogemos la opción *Refresh*.



Luego se mostrará la tabla CARGA_F en la lista de tablas.



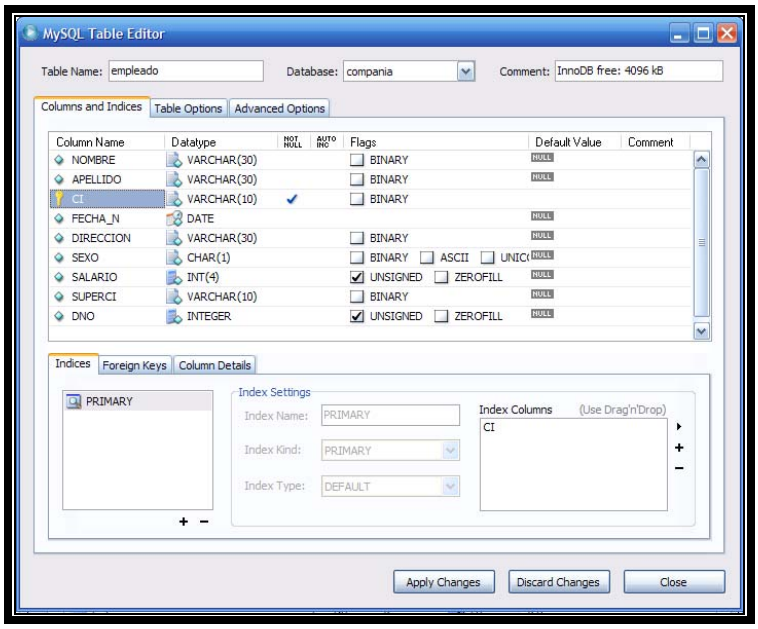
LLAVES PRIMARIAS.

Una vez creadas las tablas procederemos definir las llaves primarias. Para proceder con las llaves primarias, en caso de haberlas; en la pestaña *Schema Tables* damos doble clic sobre la tabla de la cual se quieren definir las llaves. A continuación vamos a definir las llaves de las tablas creadas

TABLA EMPLEADO.

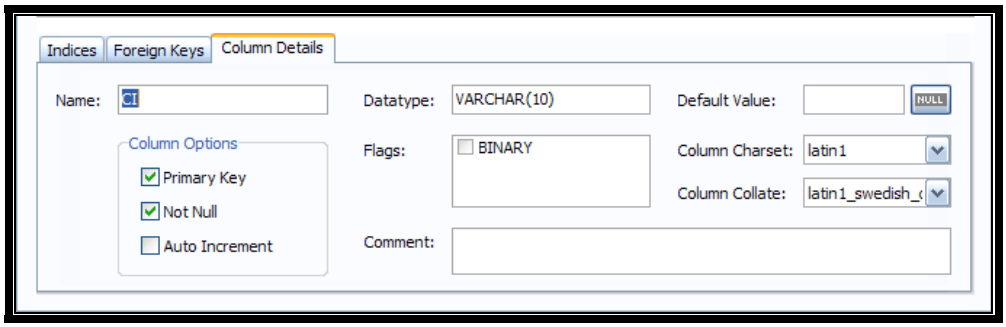
Damos doble clic sobre la tabla EMPLEADO, al abrirse la ventana del *Table Editor* con todas las columnas, procedemos a dar un clic sobre la columna que se desea convertir en llave primaria, en este caso es el campo CI. Como se indico anteriormente al definir la llave primaria aparecerá una llave amarilla al lado del campo seleccionado.

En la parte inferior en la pestaña de índices aparecerá *PRIMARY* y en el campo *Index Columns* aparecerá CI. Al definir una llave primaria automáticamente el campo se marcará como *NOT NULL*.

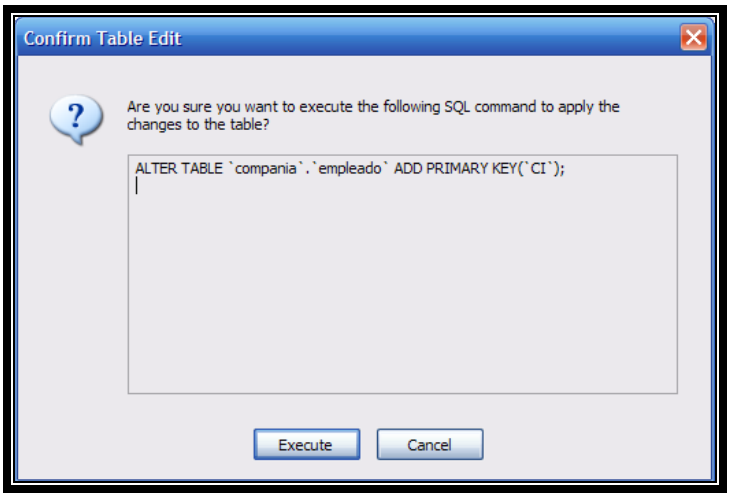


En la pestaña *Column Details* aparecerán detalles y opciones modificables de cada columna seleccionada. Podemos ver en el campo *Name*: aparecerá la el nombre de la columna que haya sido seleccionado.

En este caso se puede ver que la columna CI está seleccionada, en *Column Options* deberá estar marcado *Primary Key* y *Not Null*, en *Datatype*: *Varchar*(10), Se puede hacer el campo *Auto Incremental* en caso de necesitarlo marcando la opción *Auto Increment*, cambiar los caracteres soportados por la columna, poner un valor inicializado. *Default Value.*

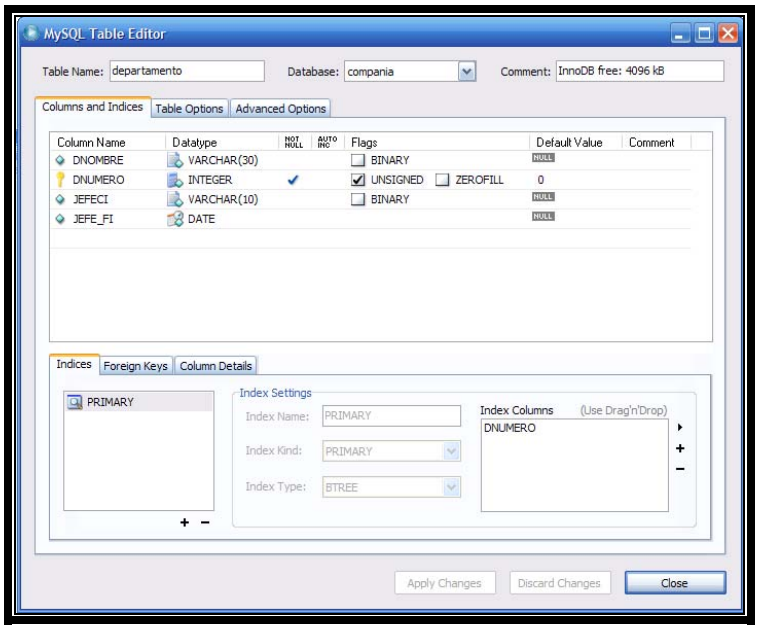


Luego de configurar todo lo que necesitamos daremos clic en Apply Changes. Aparecerá entonces una ventana mostrándonos el *SQL* de las configuraciones realizadas. Daremos clic en Execute.



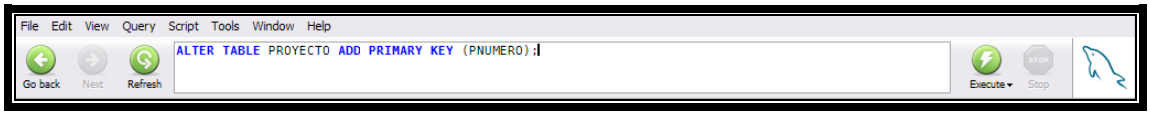
Luego volverá al *Table Editor*, daremos clic en Close.

TABLA DEPARTAMENTO



También se puede crear llaves desde el *Query Browser*, nos conectamos en a la base de datos y en el campo de línea ce comandos escribiremos la sentencia para llaves primarias. En este caso para la tabla PROYECTO se definirá de la siguiente manera:

ALTER TABLE PROYECTO ADD PRIMARY KEY (PNUMERO);



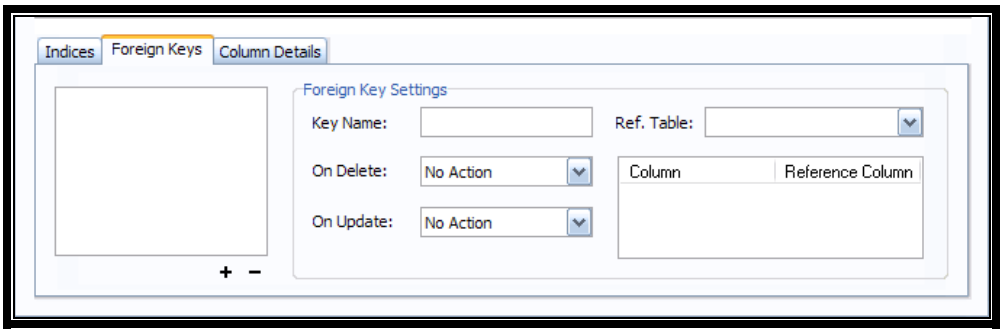
Luego damos clic en 

. Se mostrará en la parte inferior del campo de las consultas lo siguiente:

LLAVES FORANEAS

Para definir las llaves foráneas procedemos de igual manera que en el caso de las primarias, en el Table Editor.

Primero accedemos a *Schema*, compania dentro del *MySQL Administrator*, luego damos doble clic en la tabla en la cual vamos a definir la llave foránea, se abre entonces el *Table Editor*, en la pestaña *Foreign Keys* de la parte inferior damos un clic en el botón +.



Aparecerá entonces una ventana en la cual nos pide ingresar el nombre de la llave foránea; cambiaremos entonces el texto FK_tabla_1, por el nombre de la llave que necesitemos definir

En el campo *Foreign key* Settings vamos a configurar las características de las llaves foráneas que tomarán acciones sobre la tabla en la cual está la llave foránea al manipular los registros de la tabla en las que está la llave a la cual se hace referencia.

Set Null: Colocará el valor *NULL* en el campo de llave foránea si es que el registro que tiene la llave principal de la tabla a la que hace referencia es borrado.

Restrict: hace que no se pueda borrar el registro de la llave principal sin antes haberlo borrado de la tabla en cuestión.

Cascade: borrará de la tabla en cuestión si es que el registro es borrado de la tabla principal.

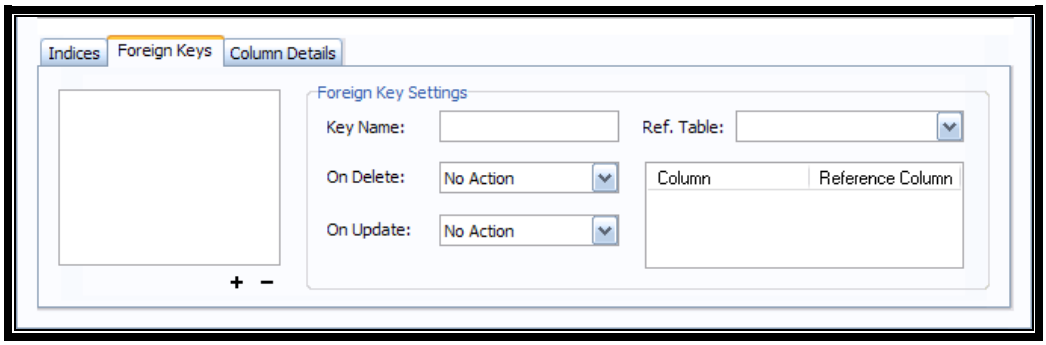
No Action: no tomará acción alguna en la tabla en cuestión si es que el registro de la tabla principal es borrado.

En el campo *Ref. Table* seleccionaremos la tabla a la cual queremos hacer referencia con la llave foránea. En la parte inferior, en la columna *Reference Column* seleccionaremos la columna a la cual queremos hacer referencia, generalmente aparece la definida como llave principal. En *Column* seleccionaremos la columna de la tabla que es la llave foránea.

Se pueden escoger los campos de las tablas dando doble clic debajo de *Reference Column* o de *Column* según se necesite respectivamente.

TABLA EMPLEADO,

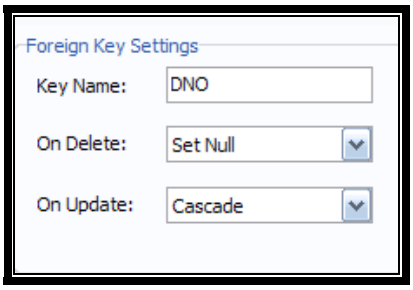
Abrimos entonces el *Table Editor*. En la pestaña *Foreign Keys* de la parte inferior damos un clic en el botón +.



Aparecerá entonces una ventana en la cual nos pide ingresar el nombre de la llave foránea; cambiaremos entonces el texto FK_empleado_1 por DNO y damos en OK.

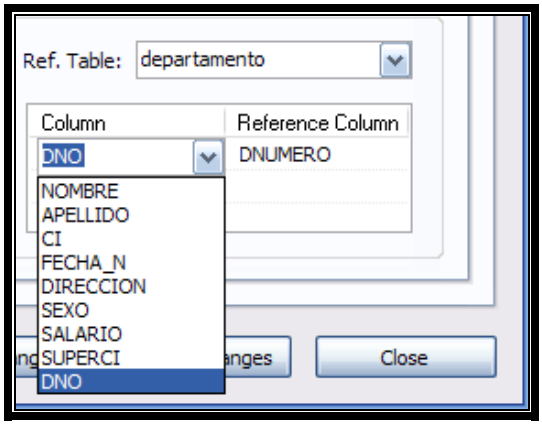
Se mostrará entonces DNO en el campo *Foreign Keys.*

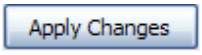
En este caso para DNO de la tabla empleado en *Key Name*: DNO, *On Delete*: *Set Null, On Update Cascade*.



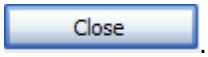
En este caso se hará referencia a la columna DNUMERO de la tabla DEPARTAMENTO, entonces en Ref. *Table,* escogeremos la tabla departamento.

Debería aparecer DNUMERO debajo de *Reference Column,* ya que es la llave principal de departamento y escogeremos DNO en el campo *Column*.

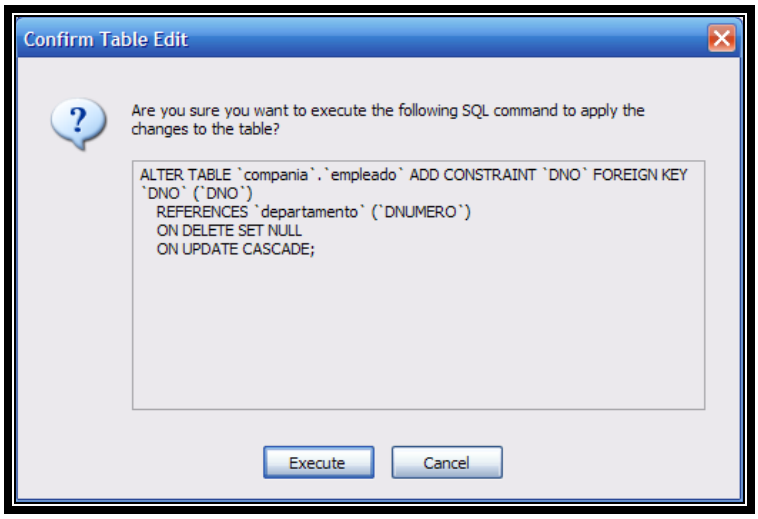


Luego procederemos con un clic en 

, se mostrará entonces una ventana en la que se muestra un *SQL* un las modificaciones realizadas, daremos clic en 

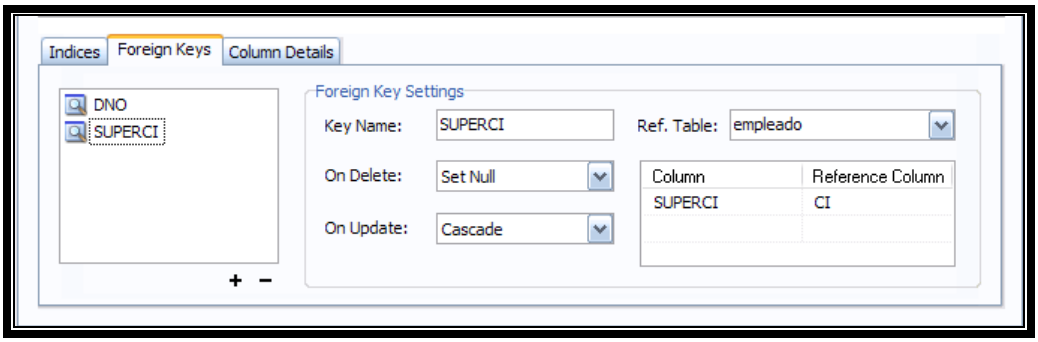
. Volverá el control a *Table Editor*, daremos clic en 

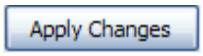
.



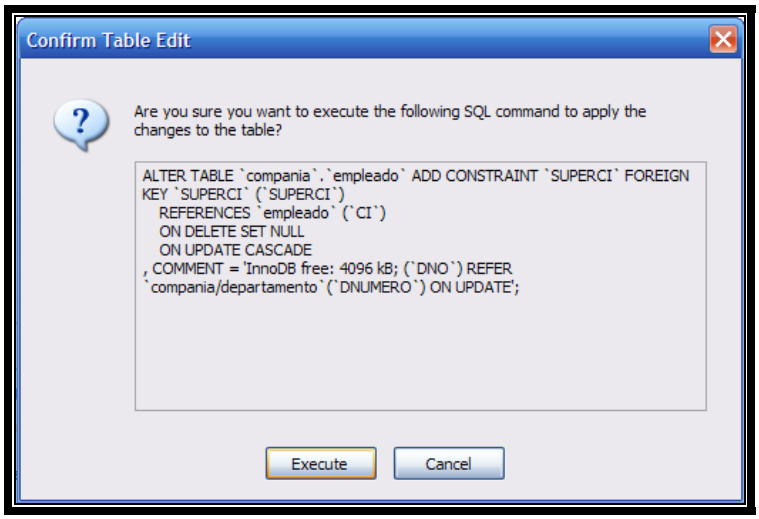
Ahora vamos a poner la llave SUPERCI de la tabla EMPLEADO. Esta es una llave recursiva, es foránea haciendo referencia a la llave principal de su misma tabla.

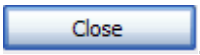
Procedemos de igual manera, abrimos el *Table Editor* dando doble clic sobre la tabla empelado. Luego en la pestaña *Foreign Keys* de la parte inferior damos un clic en el botón **+** y lo llenaremos de la siguiente manera:



Luego, daremos clic en 

, lo que dará paso a la ventana donde se muestran los cambios en instrucciones *SQL*.



Aquí daremos clic en  , volverá entonces a la pantalla del *Table Editor* y finalmente daremos clic en  . Para verificar abriremos el *Table Editor*, en la parte inferior en Indices se mostrarán los campos creados o definidos con alguna propiedad de *Index*, llaves primarias y foráneas.

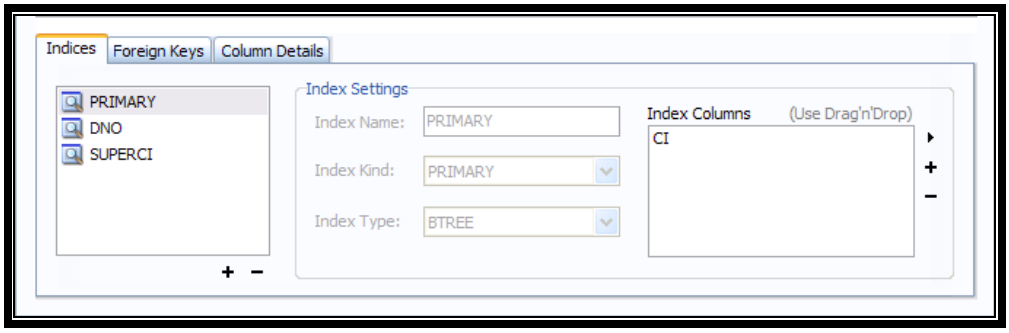


TABLA DEPARTAMENTO

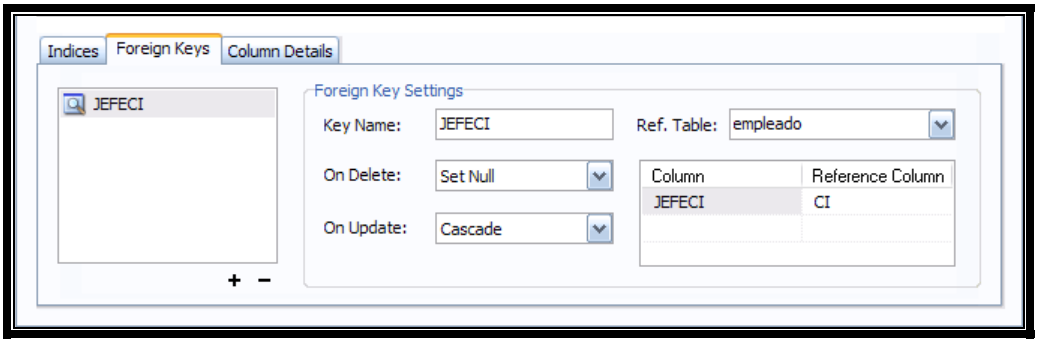


TABLA LOCALIZACION

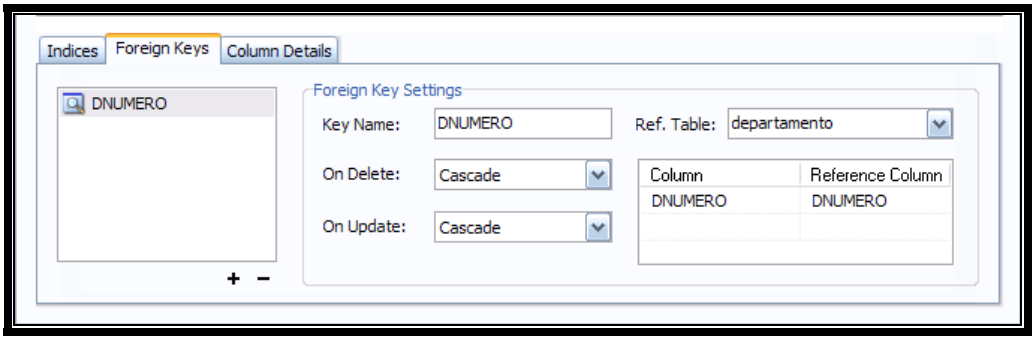
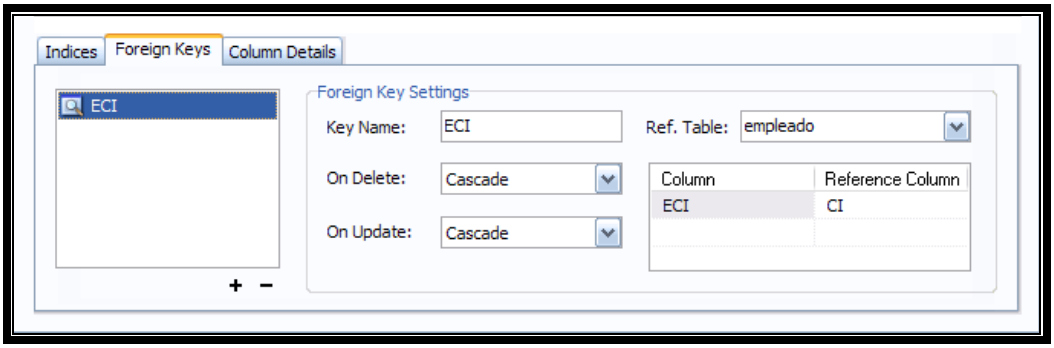
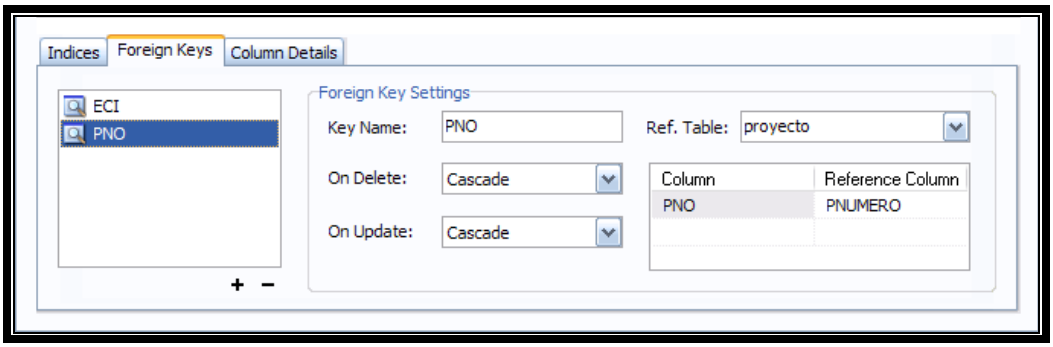
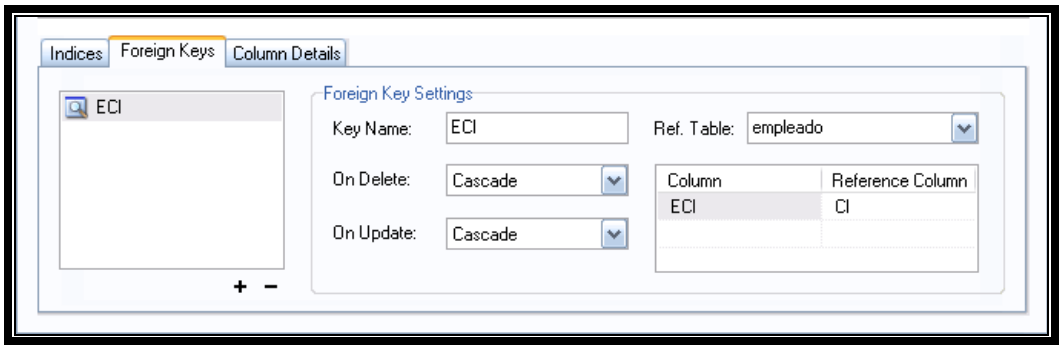


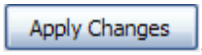
TABLA TRABAJA_EN



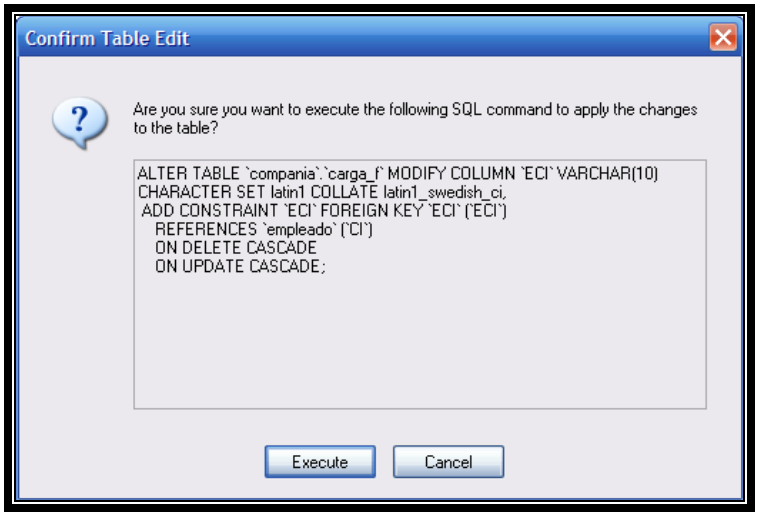


Para poder agregar la llave foránea de la tabla CARGA_F respecto a la tabla EMPLEADO debemos tener en cuenta que la columna que va a ser designada como llave es ECI el cual coincide con el campo ECI de la tabla TRABAJA_EN.

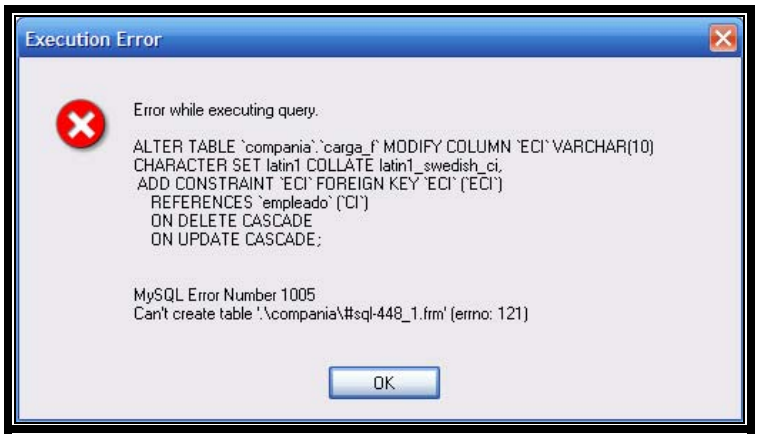


Se activará la opción 

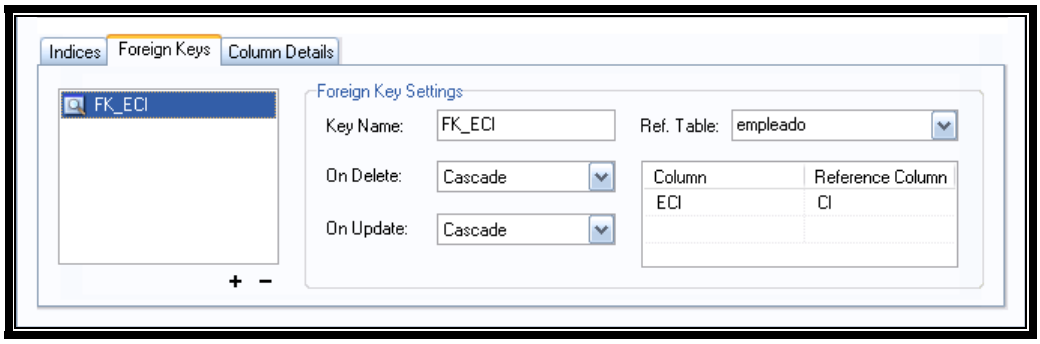
, se mostrará el *SQL* Habilitando 



En este caso se definió ya en la base de datos el nombre ECI referenciando al campo CI de EMPLEADO, este nombre no se puede repetir por lo tanto al intentar definir la llave con el mismo nombre se mostrará el siguiente error:



Vamos a definir las llaves foráneas entonces con otro nombre, esto no quiere decir que se cambie el nombre de la columna ni afectará a la integridad referencial; simplemente afecta al nombre de la llave.



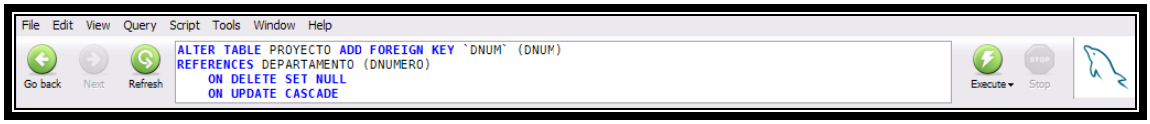
- Si por algún motivo se presenta un error al momento de crear una llave foránea se debe cambiar el nombre ya que puede estar en uso. Esto no afectará la integridad referencial.

- De igual manera se debe revisar la coherencia entre lo que estamos planteando, por ejemplo si se quiere poner en una referencia *ON DELETE SET NULL* se debe verificar que el campo en cuestión admita el valor *NULL*.

- Se debe tener en cuenta que para borrar una *PRIMARY KEY* se debe borrar las foráneas y revisar as relaciones entre las tablas, de lo contrario no se borrarán por las referencias.

Se puede también utilizar el *MySQL Query Browser* para realizar las llaves foráneas. Vamos a definir la llave foránea de la tabla PROYECTO

```
ALTER TABLE PROYECTO ADD FOREIGN KEY `DNUM` (DNUM)
REFERENCES DEPARTAMENTO (DNUMERO)
    ON DELETE SET NULL
    ON UPDATE CASCADE
```

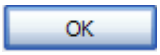


Damos clic en 

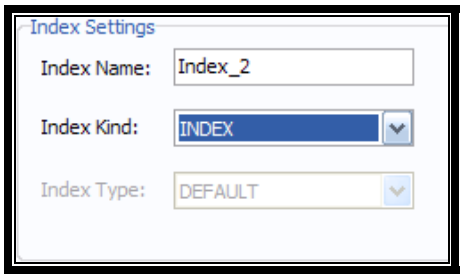
y en la parte inferior del campo donde aparecen los resultados de las consultas se mostrará lo siguiente:

INDICES

También se pueden añadir más índices sin necesidad de que sean definidos como llaves. Para agregar un índice damos un clic en el + debajo del campo en el que se muestra *PRIMARY*. Aparecerá una ventana en la cual nos pide el nombre del índice

Escribimos el nombre que deseamos y damos clic en 

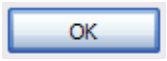
. Se habilitarán entonces las opciones de *Index Settings*. Se puede cambiar el nombre del índice en *Index Name:,* El tipo de índice (Primario, Unico, entre otros) se podrán configurar aquí.



Para definir como índice una columna existente se arrastrará la columna en cuestión hacia *Index Columns (Use Drag'n Drop).* Esto la marcara como llave primaria.

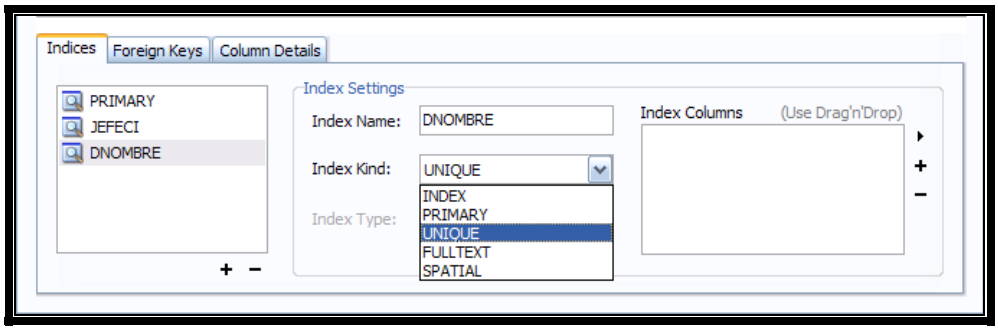
Vamos a definir índices para hacer *UNIQUE* los campos que necesitemos en nuestras tablas.

TABLA DEPARTAMENTO

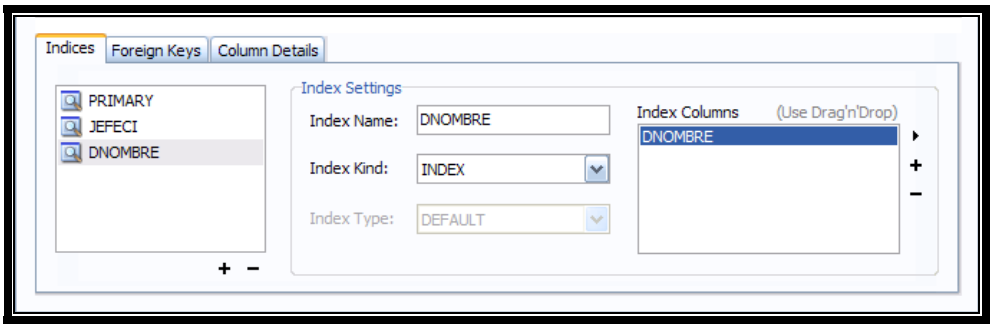
Primero abrimos la tabla con el *Table Editor*, daremos clic en la pestaña Indices y luego en +. Entonces aparece la ventana en la cual nos pide ingresar el nombre del índice, le pondremos el mismo nombre de la columna que queremos hacer *unique*, en este caso DNOMBRE, damos clic en 

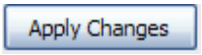
.

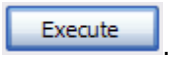
Luego en *Index Settings*, deberá estar en *Index Name*: DNOMBRE, y escogemos entonces *UNIQUE* en *Index Kind.*



Luego seleccionamos en *Index Columns* + o arrastramos la columna que deseamos.



Entonces damos clic en 

, se mostrará entonces la ventana con el *SQL* en la cual daremos clic en 

.

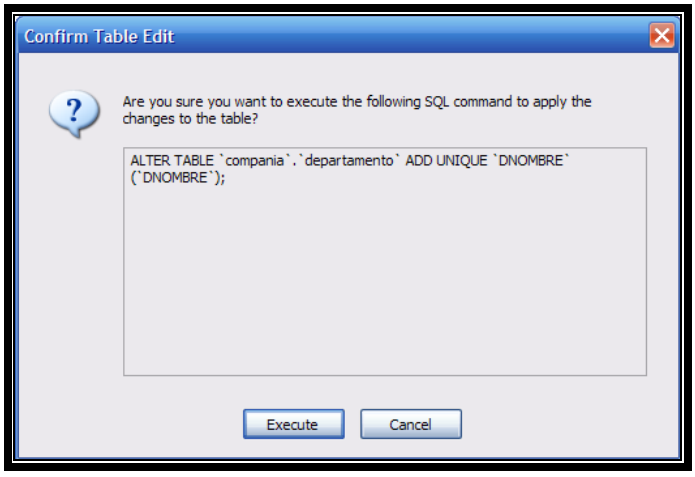
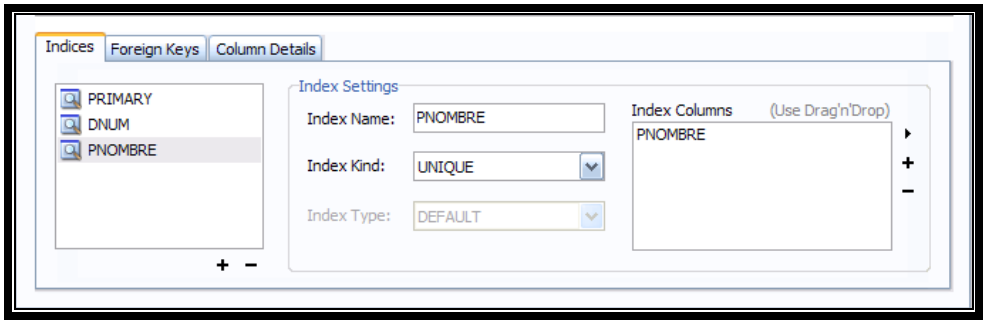
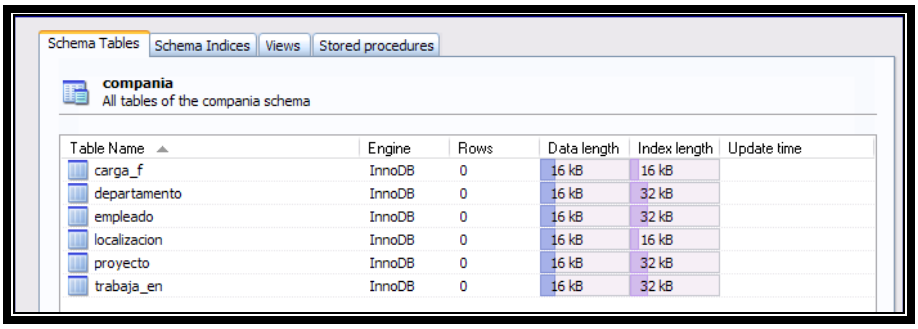


TABLA PROYECTO

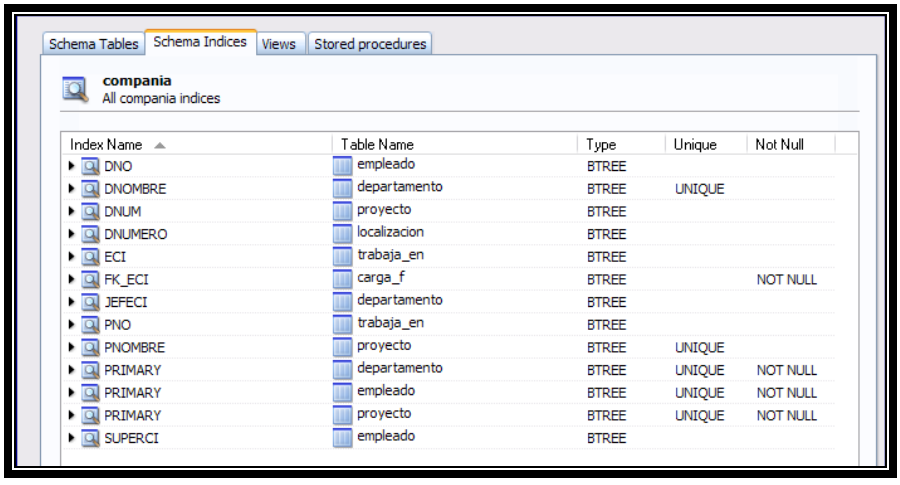


Al terminar de crear todas las tablas tendremos como resultado lo siguiente en *MySQL Administrator* en compania.

En las tablas:



En los Indices:



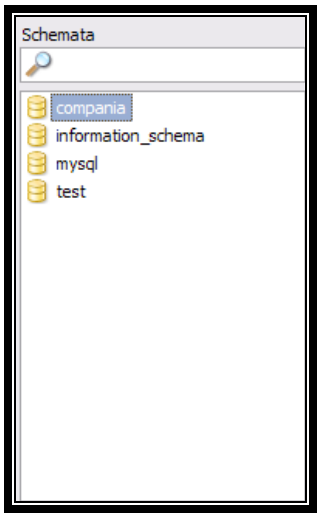
INGRESO DE INFORMACION

Una vez terminada la creación de la base de datos, tablas, llaves primarias, foráneas y todo lo que se necesite procederemos a llenar las tablas con la información correspondiente.

Para ingresar los datos nos conectamos a la base de datos desde *MySQL Administrator*. Seleccionamos la base en la cual queremos ingresar la información en *Schemata*.

Luego escogemos en la pestaña *Schema Tables* la tabla en la cual se desea ingresar la información. Luego damos clic derecho y escogemos en el menú contextual *Edit Table Data.*

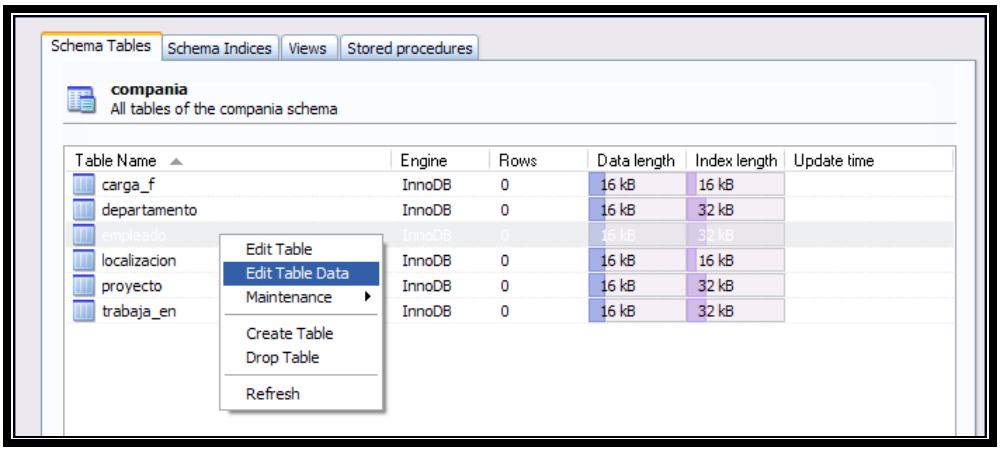
Vamos a escoger en *Schemata* compania



Luego procederemos a ingresar la información en las tablas.

TABLA EMPLEADO

Damos clic derecho y escogemos en el menú contextual *Edit Table Data.*



Se abrirá *MySQL* Query Browser, entonces en la pestaña 

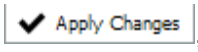
aparecerán los campos de la tabla a llenar con los nombres en los encabezados.

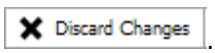
Damos un clic en la parte inferior en el botón 

y se mostrará de la siguiente manera 

como si estuviese presionado, esto nos indicará que está en modo de edición de información y nos permitirá ingresar la información en las tablas.

Damos doble clic en el campo que queremos ingresar la información en este caso será nombre, se desmarcará de azul y aparecerá el cursor, entonces podremos ingresar la información, para pasar de un campo a otro se usa la tecla *TAB* o doble clic, cuando está marcada en azul se pueden usar las flechas.

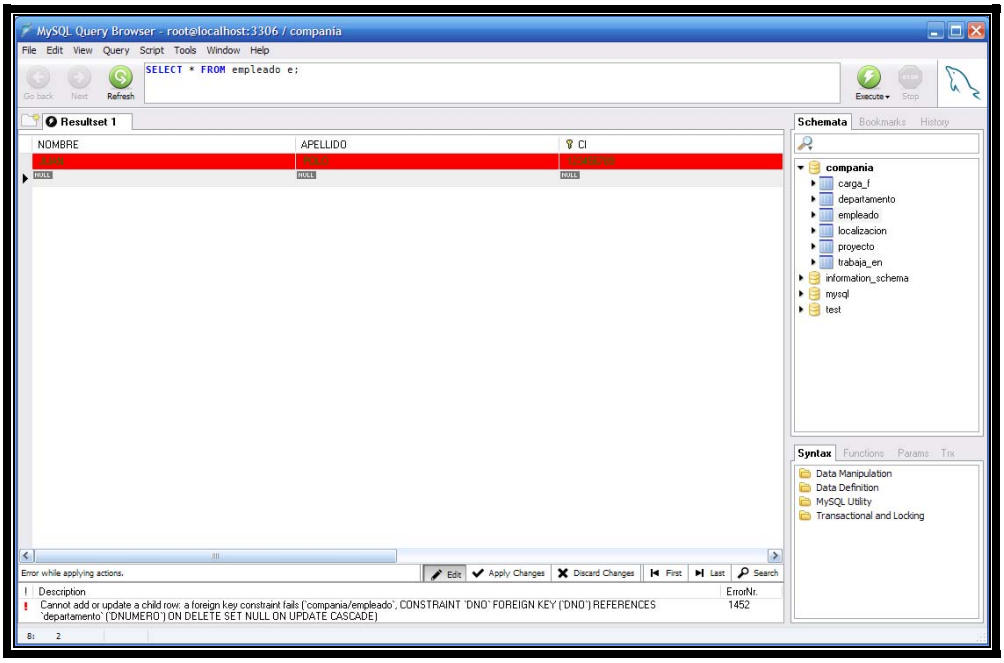
Una vez ingresada la información damos un clic en el botón 

, de lo contrario se procederá a dar clic en 

.

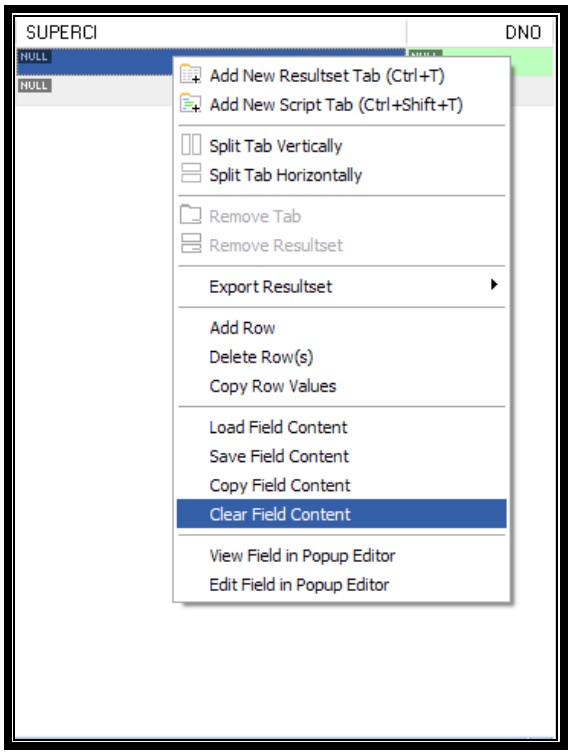
Vamos entonces a ingresar información en la tabla EMPLEADO.

En NOMBRE: JUAN, APELLIDO: POLO, CI:123456789, FECHA_N: 1959-03-03, DIRECCION: SUCRE 7-12, SEXO:M Y SALARIO:3000.



Si intentamos ingresar los registros SUPERCI y DNO nos generará un error ya que los campos son llaves foráneas y el gestor no encuentra registros en las tablas de referencia para la integridad referencial; por lo tanto, por el momento vamos a llenar el resto de campos en la tabla.

Si es que no aparece null en los campos SUPERCI y DNO daremos un clic derecho y en el menú contextual escogemos *Clear Field Content.*



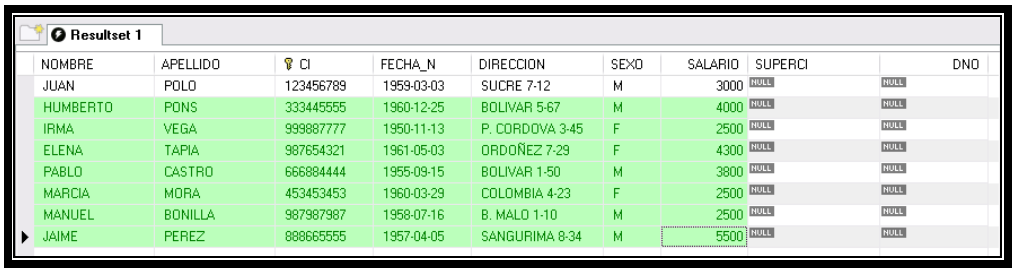
De manera que los campos queden de la siguiente manera:

De no ser así se seguirá el error y no se podrán guardar los datos, luego daremos en 

, los datos se guardarán.

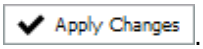
TABLA EMPLEADO

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SEXO	SALARIO
Juan	Polo	123456789	3-mar-59	Sucre 7-12	M	3000
Humberto	Pons	333445555	25-dic-60	Bolívar 5-67	M	4000
Irma	Vega	999887777	13-nov-50	P Córdova 3-45	F	2500
Elena	Tapia	987654321	03-may-61	Ordóñez 7-29	F	4300
Pablo	Castro	666884444	15-sep-55	Bolívar 1-50	M	3800
Marcia	Mora	453453453	29-mar-60	Colombia 4-23	F	2500
Manuel	Bonilla	987987987	16-jul-58	B. Malo 1-10	M	2500
Jaime	Pérez	888665555	5-abr-57	Sangurima 8-34	M	5500



Resultset 1

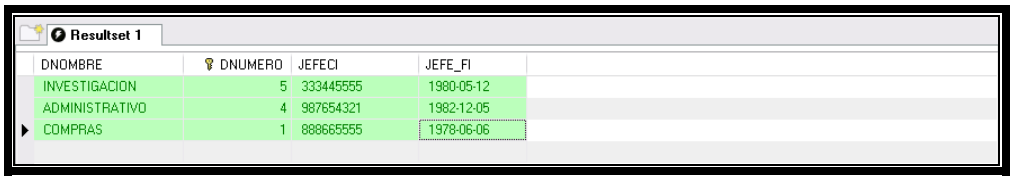
NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SEXO	SALARIO	SUPERCI	DNO
JUAN	POLO	123456789	1959-03-03	SUCRE 7-12	M	3000	NULL	NULL
HUMBERTO	PONS	333445555	1960-12-25	BOLIVAR 5-67	M	4000	NULL	NULL
IRMA	VEGA	999887777	1950-11-13	P. CORDOVA 3-45	F	2500	NULL	NULL
ELENA	TAPIA	987654321	1961-05-03	ORDOÑEZ 7-29	F	4300	NULL	NULL
PABLO	CASTRO	666884444	1955-09-15	BOLIVAR 1-50	M	3800	NULL	NULL
MARCIA	MORA	453453453	1960-03-29	COLOMBIA 4-23	F	2500	NULL	NULL
MANUEL	BONILLA	987987987	1958-07-16	B. MALO 1-10	M	2500	NULL	NULL
JAIME	PEREZ	888665555	1957-04-05	SANGURIMA 8-34	M	5500	NULL	NULL

Luego damos clic en 

. Los datos se guardarán.

TABLA DEPARTAMENTO

DNOMBRE	**DNUMERO**	**JEFECI**	**JEFE_FI**
Investigación	5	333445555	12-may-80
Administrativo	4	987654321	05-dic-82
Compras	1	888665555	06-jun-78



Resultset 1

DNOMBRE	DNUMERO	JEFECI	JEFE_FI
INVESTIGACION	5	333445555	1980-05-12
ADMINISTRATIVO	4	987654321	1982-12-05
COMPRAS	1	888665555	1978-06-06

Ahora ya podemos llenar los 2 campos restantes de la tabla EMPLEADO: SUPERCI, DNO ya que ya tenemos llenos los campos de las tablas a las que se hacen referencia.

SUPERCI	**DNO**
333445555	5
888665555	5
987654321	4
888665555	4
333445555	5
333445555	5
987654321	4
NULL	1

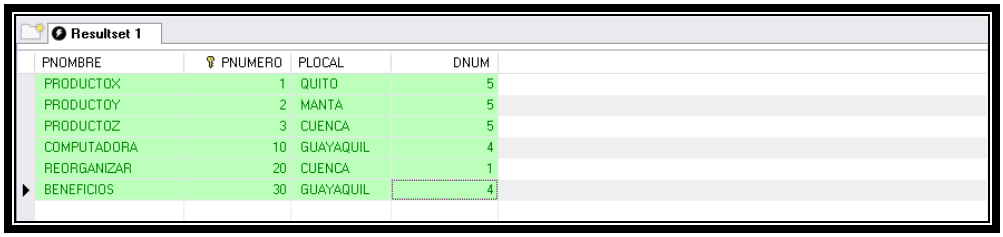
Resultset 1

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SEXO	SALARIO	SUPERCI	DNO
JUAN	POLO	123456789	1959-03-03	SUCRE 7-12	M	3000	333445555	5
HUMBERTO	PONS	333445555	1960-12-25	BOLIVAR 5-67	M	4000	888665555	5
MARCIA	MORA	453453453	1960-03-29	COLOMBIA 4-23	F	2500	333445555	5
PABLO	CASTRO	666884444	1955-09-15	BOLIVAR 1-50	M	3800	333445555	5
JAIME	PEREZ	888665555	1957-04-05	SANGURIMA 8-34	M	5500	NULL	1
ELENA	TAPIA	987654321	1961-05-03	ORDOÑEZ 7-29	F	4300	888665555	4
MANUEL	BONILLA	987987987	1958-07-16	B. MALO 1-10	M	2500	987654321	4
IRMA	VEGA	999887777	1950-11-13	P. CORDOVA 3-45	F	2500	987654321	4

Se puede apreciar que al volver a abrir la tabla los datos se encuentran ordenados por la llave principal que es el campo CI, así que tenemos que tener cuidado y fijarnos bien en donde introducimos los datos.

TABLA PROYECTO

PNOMBRE	**PNUMERO**	**PLOCAL**	**DNUM**
ProductoX	1	Quito	5
ProductoY	2	Manta	5
ProductoZ	3	Cuenca	5
Computadora	10	Guayaquil	4
Reorganizar	20	Cuenca	1
Beneficios	30	Guayaquil	4



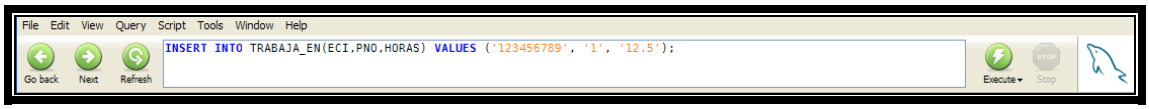
Resultset 1

PNOMBRE	PNUMERO	PLOCAL	DNUM
PRODUCTOX	1	QUITO	5
PRODUCTOY	2	MANTA	5
PRODUCTOZ	3	CUENCA	5
COMPUTADORA	10	GUAYAQUIL	4
REORGANIZAR	20	CUENCA	1
BENEFICIOS	30	GUAYAQUIL	4

Para poder llenar las tablas que carecen de llaves principales, por ejemplo las tablas resultantes de relaciones, es necesario hacerlo desde la línea de comando en *DOS* o de igual manera desde el *MySQL* Query *Browser*.

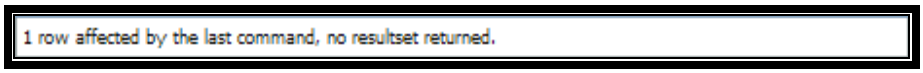
Primero nos conectamos al *MySQL Query Browser,* seleccionamos la base de datos compania, En la línea de comandos realizamos el ingreso de la información. A continuación vamos a realizar el ingreso de los datos de la tabla TRABAJA_EN

INSERT INTO TRABAJA_EN (ECI, PNO, HORAS) VALUES ('123456789', 1, 12.5);



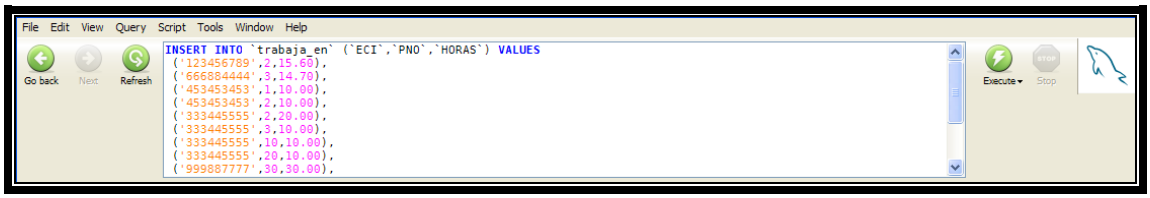
Damos clic en 

.y Se mostrará entonces el siguiente mensaje en la parte inferior:



Ahora vamos a ingresar el resto de la información de la tabla de la siguiente manera:

```
INSERT INTO `trabaja_en` (ECI,PNO,HORAS)
VALUES ('123456789',2,15.60),
       ('666884444',3,14.70),
       ('453453453',1,10.00),
       ('453453453',2,10.00),
       ('333445555',2,20.00),
       ('333445555',3,10.00),
       ('333445555',10,10.00),
       ('333445555',20,10.00),
       ('999887777',30,30.00),
       ('999887777',10,5.00),
       ('987987987',10,15.00),
       ('987987987',30,17.00),
       ('987654321',30,10.00),
       ('987654321',20,12.00),
       ('888665555',20,NULL);
```

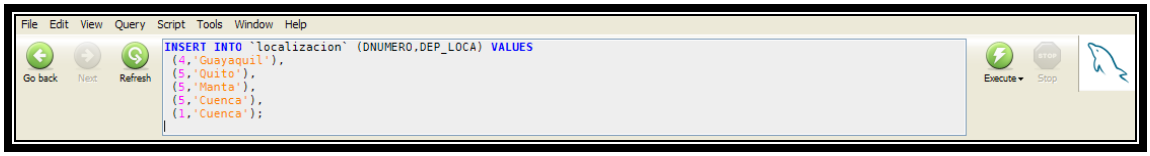


Damos clic en 

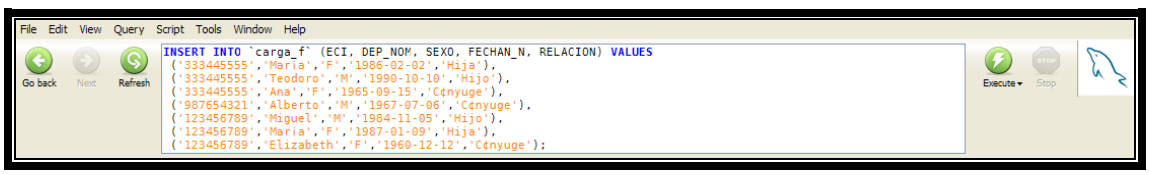
para guardar los datos y en la parte inferior nos mostrará el siguiente mensaje:

TABLA LOCALIZACION

```
INSERT INTO `localizacion` (DNUMERO, DEP_LOCA)
VALUES  (4,'Guayaquil'),
        (5,'Quito'),
        (5,'Manta'),
        (5,'Cuenca'),
        (1,'Cuenca');
```




```
INSERT INTO `carga_f` (ECI, DEP_NOM, SEXO, FECHAN_N, RELACION)
VALUES  ('333445555','Maria','F','1986-02-02','Hija'),
        ('333445555','Teodoro','M','1990-10-10','Hijo'),
        ('333445555','Ana','F','1965-09-15','Conyuge'),
        ('987654321','Alberto','M','1967-07-06','Conyuge'),
        ('123456789','Miguel','M','1984-11-05','Hijo'),
        ('123456789','Maria','F','1987-01-09','Hija'),
        ('123456789','Elizabeth','F','1960-12-12','Conyuge');
```



Vamos a Introducir en la tabla empleado el siguiente registro en la tabla EMPLEADO:

OSWALDO	MERCHAN	222222222	1950-11-13	1RO DE MAYO	M	3000	NULL	4

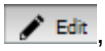
Y en la tabla TRABAJA_EN:

INSERT INTO TRABAJA_EN (ECI, PNO, HORAS) VALUES ('222222222', 1, 20);.

Para hacer editar los datos en las tablas en las cuales hay llaves primarias se hará directamente desde la opción 

, se cambiarán los datos que se necesiten.

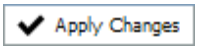
Vamos a cambiar por 5000 el salario del registro que acabamos de ingresar.

Damos clic en 

, damos clic en el campo salario, cambiamos el valor por 5000, este y los campos cambiados se marcarán de un color lila con letras azules.

Resultset 1

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SE...	SAL...	SUPERCI	DNO
JUAN	POLO	123456789	1959-03-03	SUCRE 7-12	M	3000	333445555	5
OSWALDO	MERCHAN	222222222	1950-11-13	1RO DE MAYO	M	5000	NULL	4
HUMBERTO	PONS	333445555	1960-12-25	BOLIVAR 5-67	M	4000	888665555	5
MARCIA	MORA	453453453	1960-03-29	COLOMBIA 4-23	F	2500	333445555	5
PABLO	CASTRO	666884444	1955-09-15	BOLIVAR 1-50	M	3800	333445555	5
JAIME	PEREZ	888665555	1957-04-05	SANGURIMA 8-34	M	5500	NULL	1
ELENA	TAPIA	987654321	1961-05-03	ORDOÑEZ 7-29	F	4300	888665555	4
MANUEL	BONILLA	987987987	1958-07-16	B. MALO 1-10	M	2500	987654321	4
IRMA	VEGA	999887777	1950-11-13	P. CORDOVA 3-45	F	2500	987654321	4

Entonces se da clic en 

Para editar información en las tablas que no tienen llaves primarias tenemos que hacerlo por medio de *SQL*.

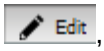
Vamos a cambiar el valor horas de 20 a 30 en el registro que ingresamos en la tabla TABAJA_EN.

UPDATE TRABAJA_EN SET HORAS=30 WHERE ECI='222222222';

Para eliminar información de una tabla que no tiene llave principal usamos *SQL*.

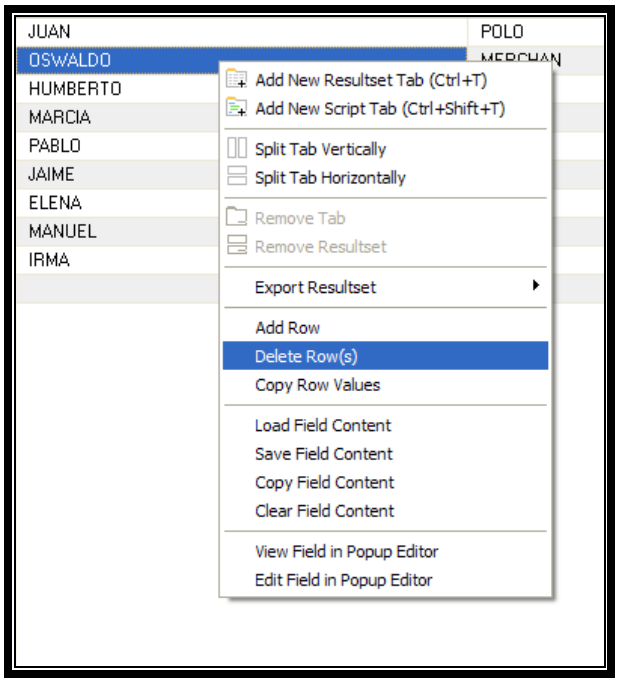
Vamos a eliminar el último registro que ingresamos:

DELETE FROM TRABAJA_EN WHERE ECI='222222222';

Para eliminar información de una tabla la cual contenga llaves primarias se lo puede hacer directamente dando un clic en 

, luego clic derecho sobre el registro que se desea borrar y seleccionando la opción *Delete Row(s)* del menú contextual.

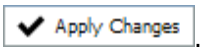
Vamos a borrar el último registro que ingresamos en la tabla EMPLEADO.



Entonces el registro se pintará de rosado.

Resultset 1 | Inline Help

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SE...	SALARIO	SUPERCI	DNO
JUAN	POLO	123456789	1959-03-03	SUCRE 7-12	M	3000	333445555	5
OSWALDO	MERCHAN	222222222	1950-11-13	1RO DE MAYO	M	5000	NULL	4
HUMBERTO	PONS	333445555	1960-12-25	BOLIVAR 5-67	M	4000	888665555	5
MARCIA	MORA	453453453	1960-03-29	COLOMBIA 4-23	F	2500	333445555	5
PABLO	CASTRO	666884444	1955-09-15	BOLIVAR 1-50	M	3800	333445555	5
JAIME	PEREZ	888665555	1957-04-05	SANGURIMA 8-34	M	5500	NULL	1
ELENA	TAPIA	987654321	1961-05-03	ORDOÑEZ 7-29	F	4300	888665555	4
MANUEL	BONILLA	987987987	1958-07-16	B. MALO 1-10	M	2500	987654321	4
IRMA	VEGA	999887777	1950-11-13	P. CORDOVA 3-45	F	2500	987654321	4

Damos clic en 

. Las consultas se las realiza de igual manera que en la línea de comando.

```
SELECT CI, NOMBRE, APELLIDO, SALARIO
FROM EMPLEADO
WHERE SALARIO > 3500;
```

```
SELECT CI, NOMBRE, APELLIDO, SALARIO
FROM EMPLEADO
WHERE SALARIO > 3500
```

Los resultados se muestran en la parte inferior de la línea de comando al dar clic en

Resultset 1

CI	NOMBRE	APELLIDO	SALARIO
333445555	HUMBERTO	PONS	4000
666884444	PABLO	CASTRO	3800
888665555	JAIME	PEREZ	5500
987654321	ELENA	TAPIA	4300

ANEXO II

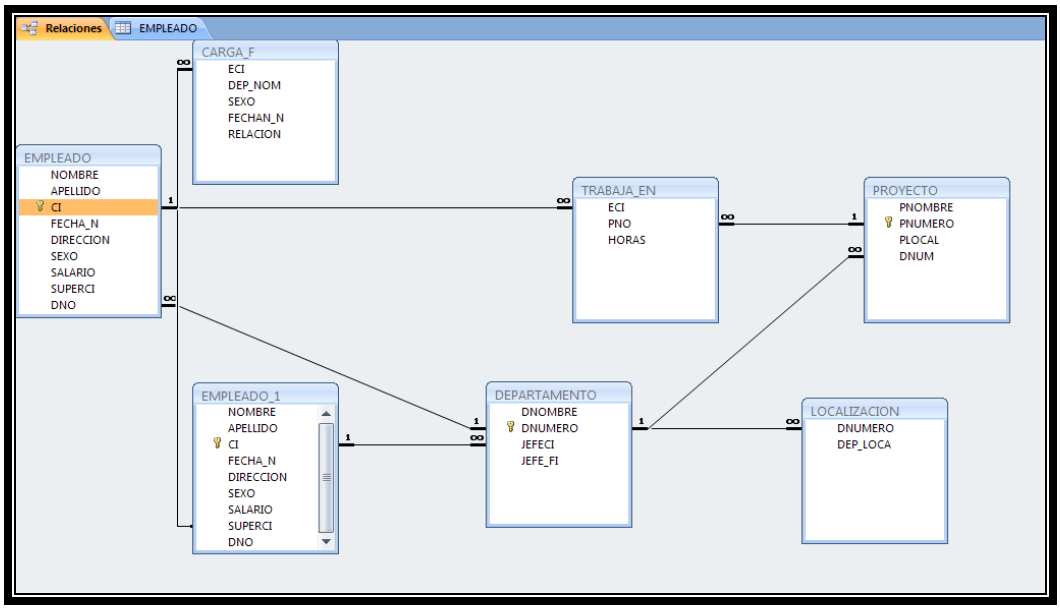
MIGRAR UNA BASE DE DATOS

Para migrar una base de datos que esté en otro gestor a *MySQL*, utilizaremos *MySQL Migration Toolkit.*

Esta herramienta puede migrar bases de datos integras desde los gestores de bases *MS ACCESS, MS SQLSERVER, MySQL SERVER, ORACLE.*

Se intentó en un principio migrar una base de datos *MS ACCESS 2007* pero fracasó en la segunda etapa, por lo cual se recomienda grabar el archivo de base de datos de *MS ACCESS* con compatibilidad 2002-2003.

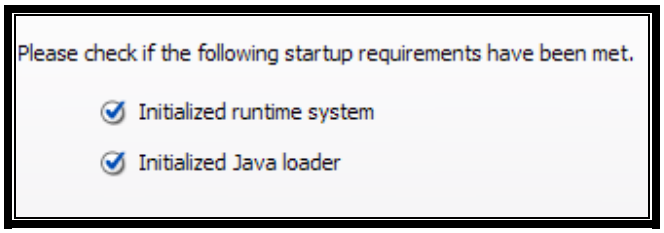
Vamos a migrar la base de datos micompania que esta creada *en MS ACCESS, l*a cual está de la siguiente manera:

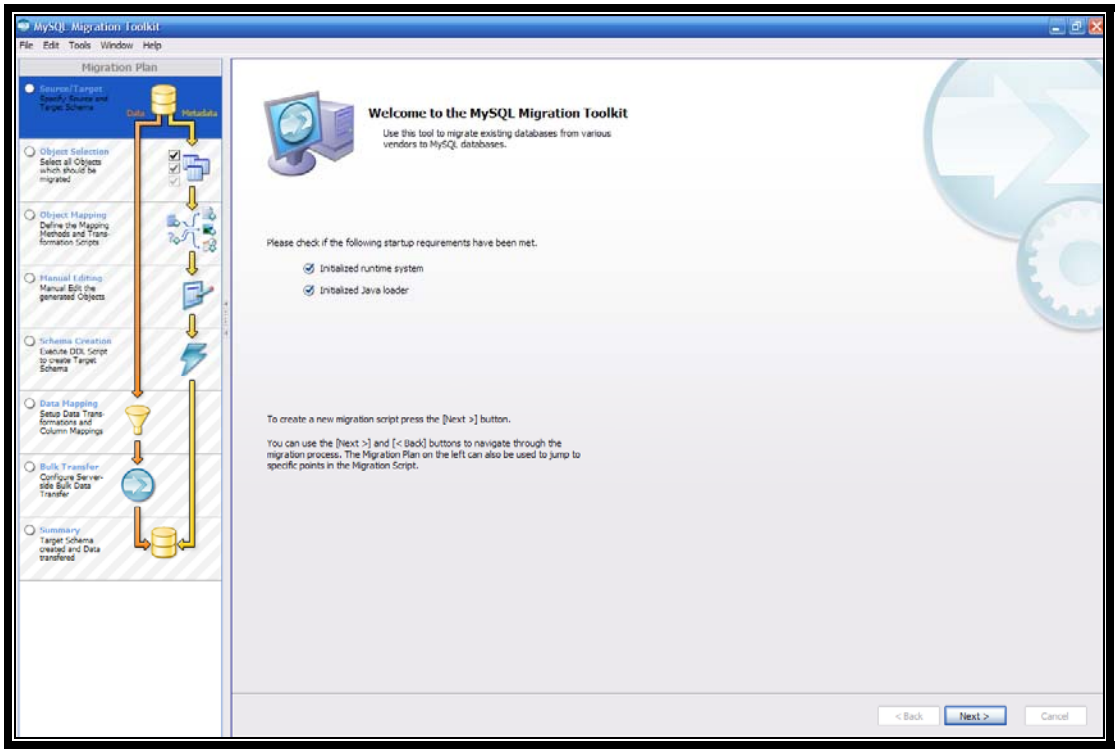


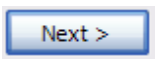
Tenemos también todos los datos ingresados en las respectivas tablas. Ingresamos entonces a Inicio, Todos los programas, *MySQL*, *MySQL Migration Toolkit*.

Para poder realizar una migración de una base de datos completa debemos seguir una serie de procesos e instrucciones.

La primera ventana en aparecer es la de bienvenida al asistente de migración de datos, En esta es fundamental observar que esté con un visto en azul las 2 casillas que se muestran a continuación, de no ser así se debe revisar la instalación de java y de la herramienta.



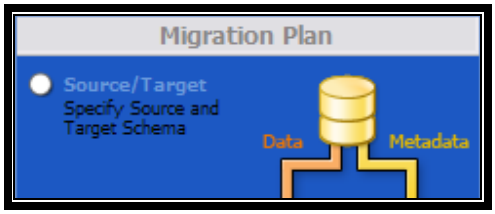


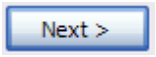
En caso de que todo esté en orden se habilitará el botón 

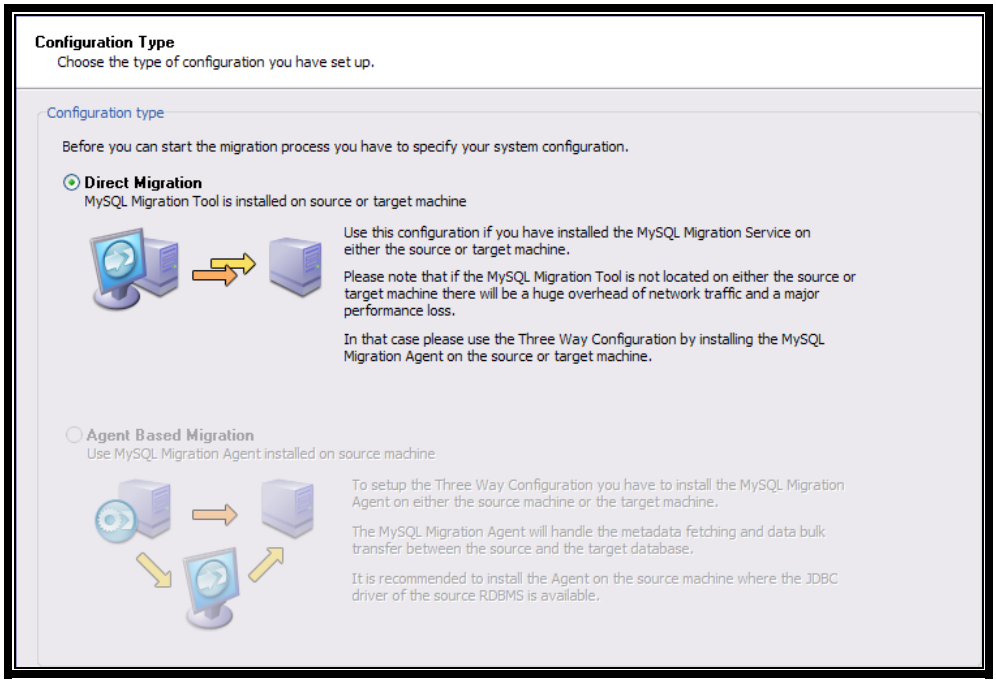
en el cual daremos clic.

En la parte izquierda se puede observar en que parte o proceso nos encontramos dentro de la migración, al ir terminando cada uno de estos procesos se irá colocando un visto azul en los círculos blancos, se marcará de azul completo el proceso en el cual estaremos actualmente trabajando.

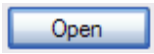
El primer proceso del plan de migraciones es seleccionar el origen y destino, se representará con el siguiente gráfico:

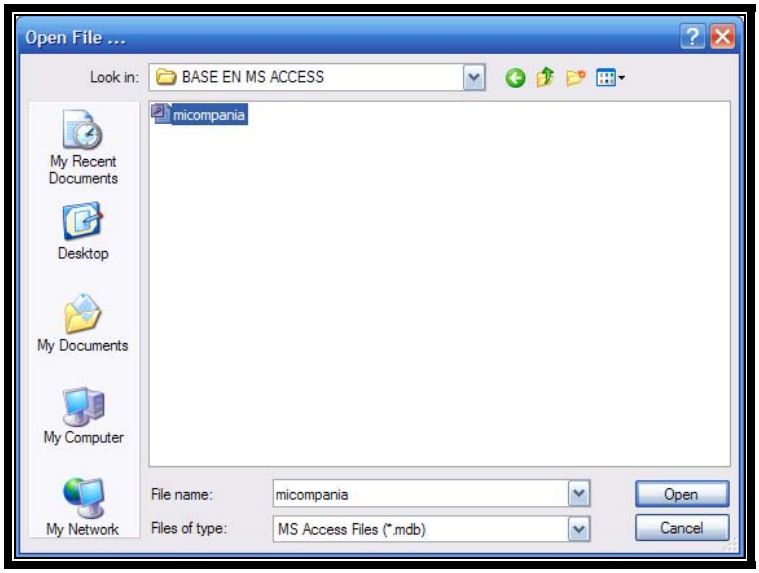


En la siguiente ventana nos pide escoger el tipo de configuración que tenemos instalado, por defecto estará señalado  , la otra opción esta deshabilitada, esta se habilitará en caso de tener una herramienta de migraciones basada en agentes en nuestro equipo. Daremos clic en 

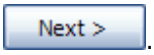


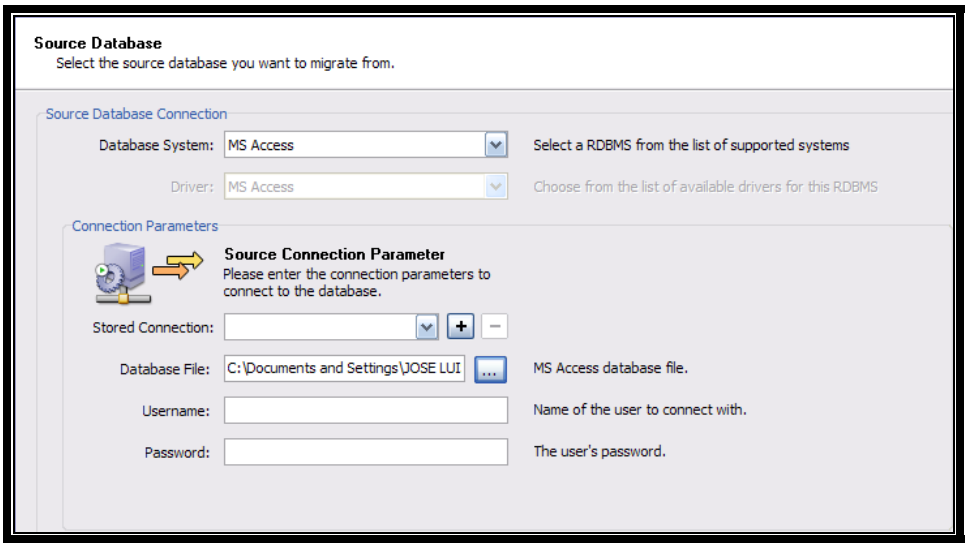
En la siguiente pantalla se nos pide seleccionar el gestor origen de la base de datos que deseamos migrar, en este caso seleccionaremos MS ACCESS en el campo Database System.

En el campo *Database File* seleccionaremos el botón  para abrir el explorador de *Windows* y así seleccionar la ubicación y escoger la base de datos micompania que está en la carpeta BASE EN *MS ACCES y* damos clic en  .



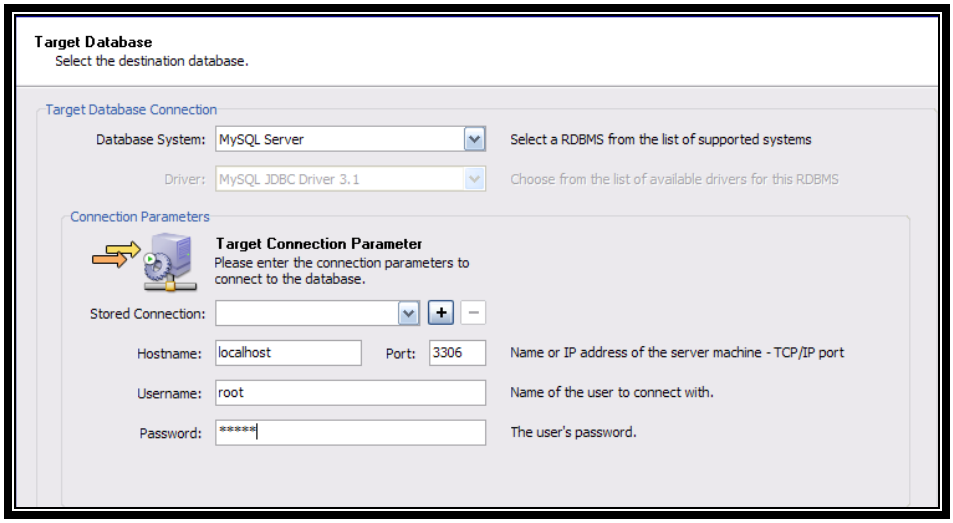
Entonces en el campo *Database File* aparecerá la ubicación de la base de datos seleccionada.

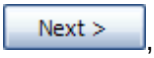
Se debe tener cuidado en verificar si la base de datos fue creada con algún Usuario y Contraseña para colocarlos en sus respectivos campos. En éste caso no se utilizó ninguno de los dos campos antes mencionados por lo que procederemos directamente a dar clic en  .



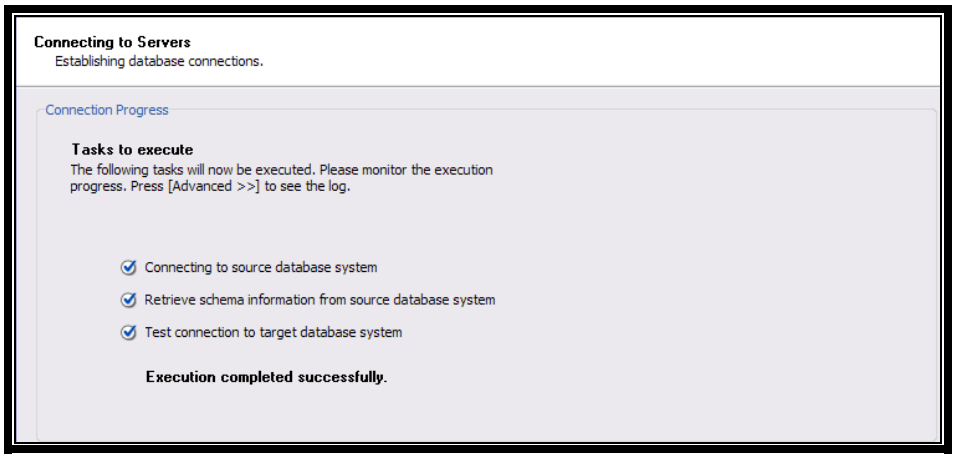
En la siguiente ventana aparecerá entonces el destino al cual queremos migrar la base de datos y el gestor al cual queremos migrar, en nuestro caso aparecerá por defecto *MySQL* Server en el campo *Database System.*

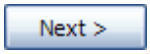
En *Conection Parameters* pondremos en *Hostname*: *localhost* o 127.0.0.1 que es la dirección que asignamos al momento de instalar el servidor, en *Port*: 3306 que también es el que pusimos al instalar el gestor, *Username: root, Password: admin*, se recuerda verificar el bloqueo de mayúsculas ya que el gestor es sensible a estas.



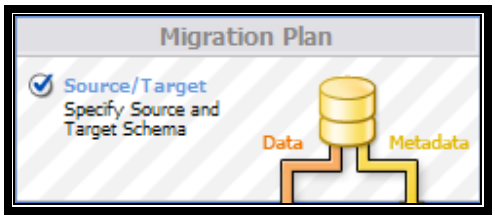
Una vez llenados los campos daremos clic en 

, se realizará entonces la conexión a los gestores de base de datos y se mostrará la siguiente ventana:

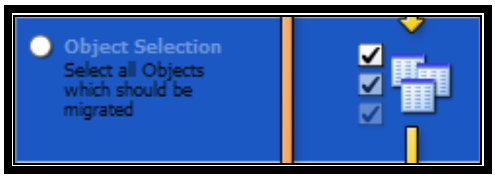


Daremos clic en 

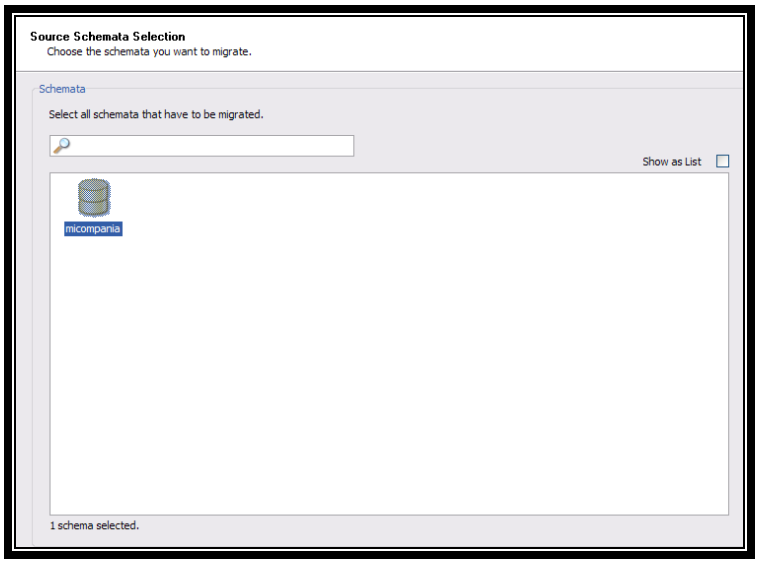
y entonces nos mostrará que culminó la primera etapa de la migración.



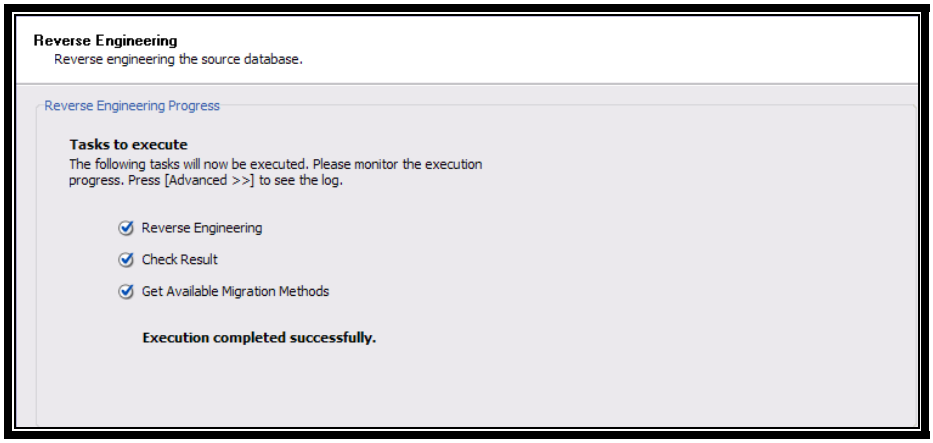
La Segunda etapa es seleccionar los objetos que se desean migrar, se mostrará el siguiente gráfico:



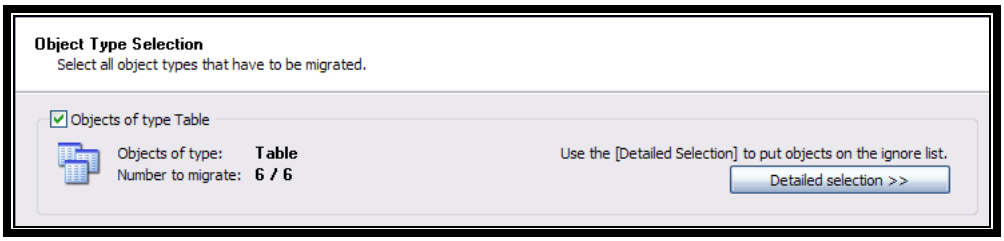
Seleccionaremos micompania en Schemata, en la parte inferior se mostrará 1 schema selected.

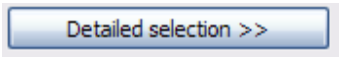


Damos clic en Next > para continuar, se mostrará la siguiente pantalla en la cual indicará que los procesos de la segunda etapa se han realizado exitosamente.



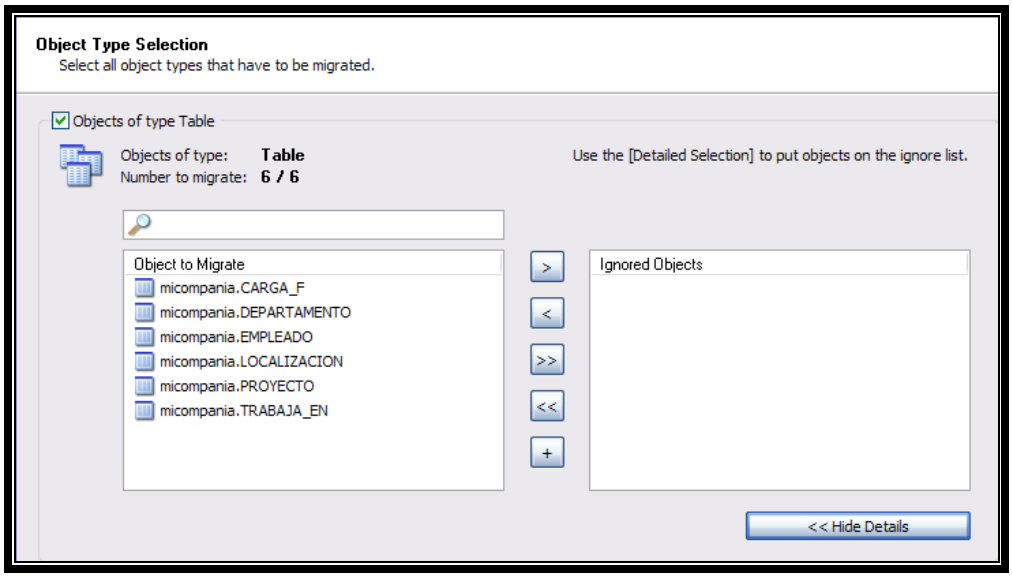
Damos clic en Next > para continuar, el asistente de migraciones nos pedirá entonces que escojamos todos aquellos objetos que queremos migrar, por defecto se seleccionarán todos y se mostrarán en la ventana los tipos de objetos disponibles.

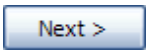


Si se quiere seleccionar de manera específica lo que queremos migrar damos clic en 

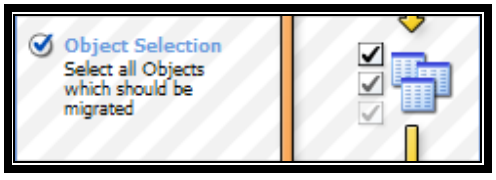
, en la ventana en la parte izquierda se muestra una lista de todas las tablas que están disponibles para la migración. En caso de que no se quiera migrar un objeto, se lo seleccionará en la lista izquierda y utilizando el botón 

se lo trasladará a la derecha donde están los objetos que serán ignorados en el proceso de migración.

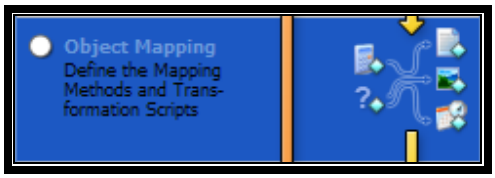


A continuación damos clic en 

para continuar, habrá culminado entonces la segunda etapa del proceso de migración de la base de datos.

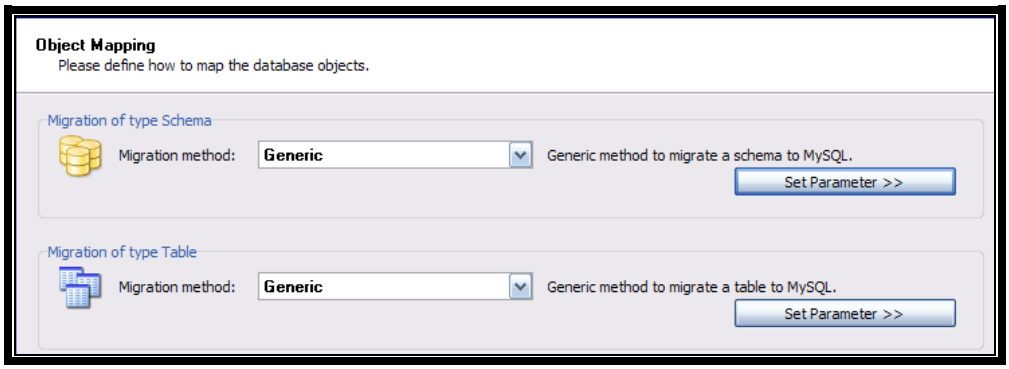


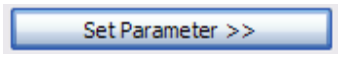
Ahora el asistente nos pide hacer mapeo de los objetos, es la tercera etapa del proceso.



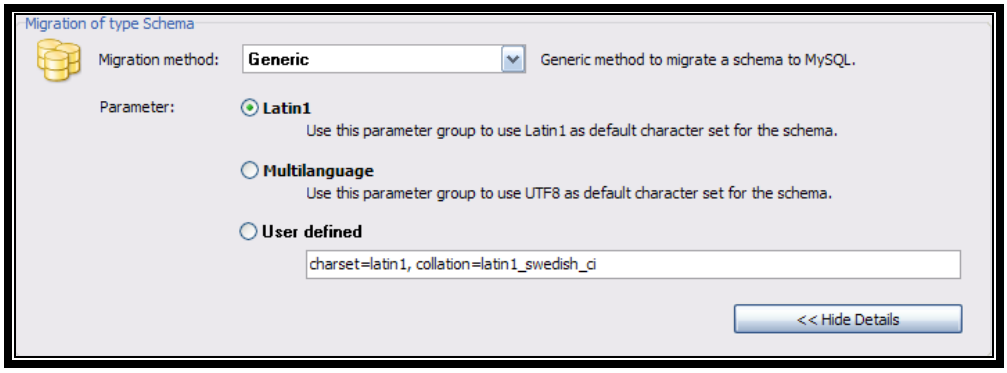
Se pide escoger el tipo de método de migración que se seguirá para la base de datos y sus tablas, haremos una migración genérica en ambos casos

.

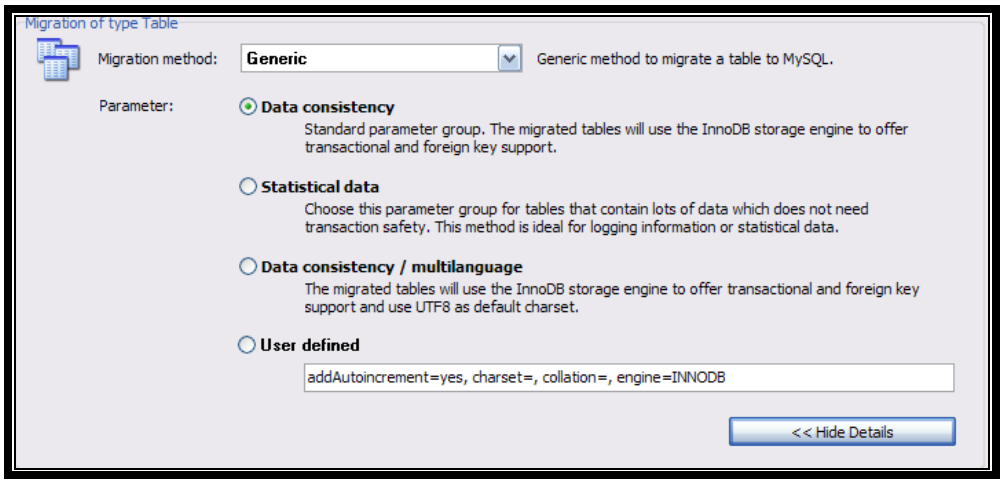


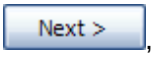
Se pueden desplegar más opciones con el botón  en la opción de *Migration of type Schema* como en *Migration of type Table.*

Al desplegar las opciones de *Migration of type Schema* podemos escoger otro tipo de caracteres soportado por la base de datos según sea necesario.

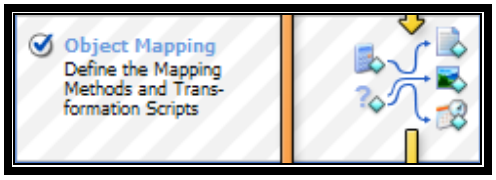


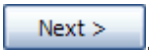
Al desplegar las opciones de *Migration of type Table* podemos escoger el tipo de trato que se les dará a las tablas, relaciones y sus datos.



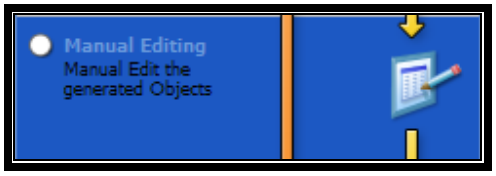
Para continuar daremos clic en 

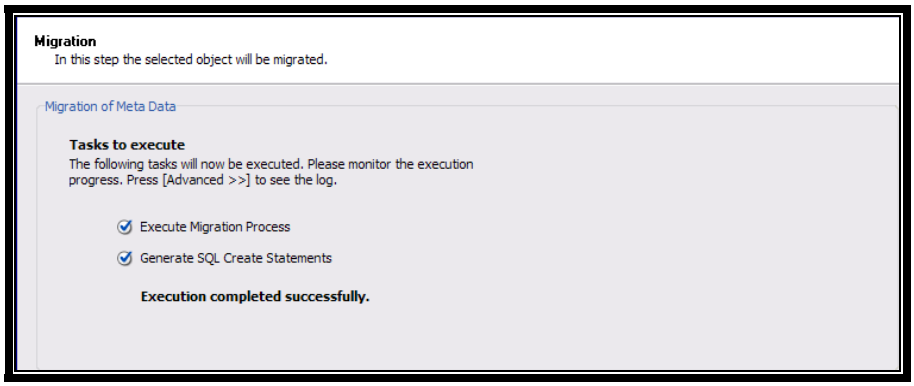
, entonces habrá culminado la tercera etapa de la migración de la base de datos



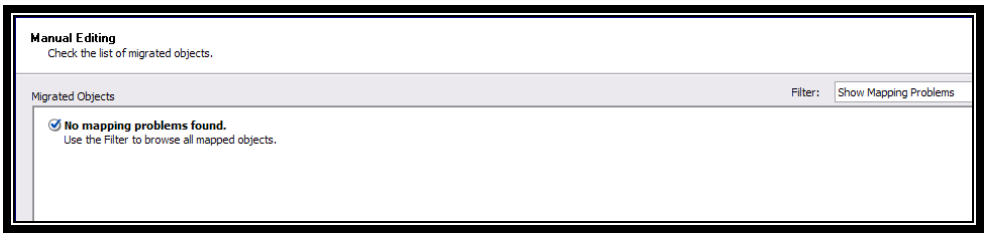
Llegamos a la cuarta etapa de la migración de la base en la cual nos muestra problemas de la base de migración, de los objetos como tablas, columnas e índices, también rutinas, procedimientos, etc. si es que existiesen. Para continuar simplemente daremos clic en 

.



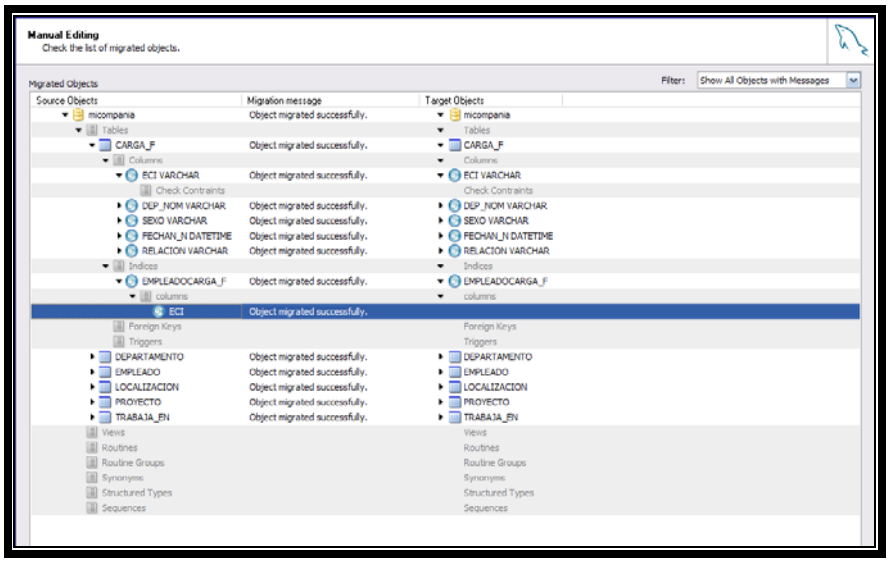


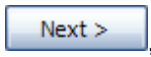
Después se mostrará la siguiente ventana en la cual en el campo *Filter*: escogeremos la opción *Show All Objects with Messages.*



Se mostrarán entonces todos los objetos y se mostrará si es que hubo algún problema al momento de realizar la migración. Se puede obtener mas detalles desplegando la lista dando clic sobre el objeto que se desea consultar.

En este caso se ha desplegado toda la información respecto a la tabla CARGA_F.

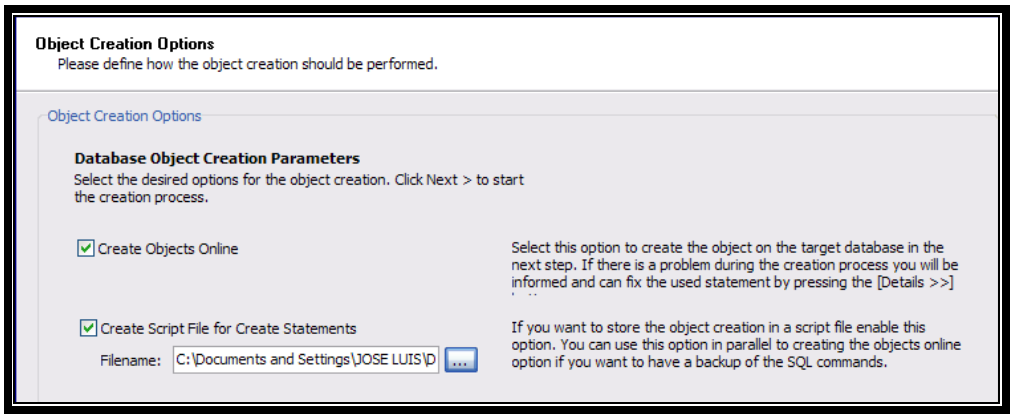


Para continuar daremos clic en 

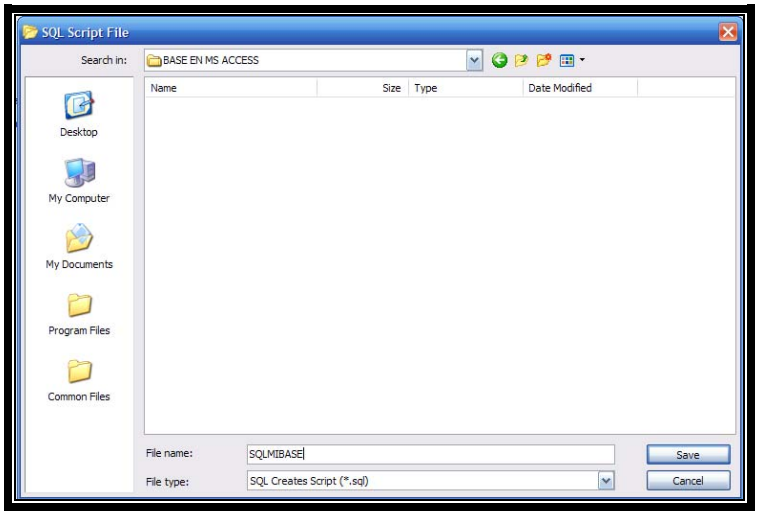
, culminará entonces la cuarta etapa de la migración.

Empezará entonces la quinta etapa en la cual ya se crea la base de datos. Se puede escoger que a más de la base de datos también cree un *script* en un archivo *SQL* en el que estará la creación integra de la base de datos, tablas, llaves, índices, etc., al seleccionar la opción *Create Script File for Create Statements*, el destino del archivo se seleccionará con el botón 

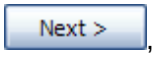
.



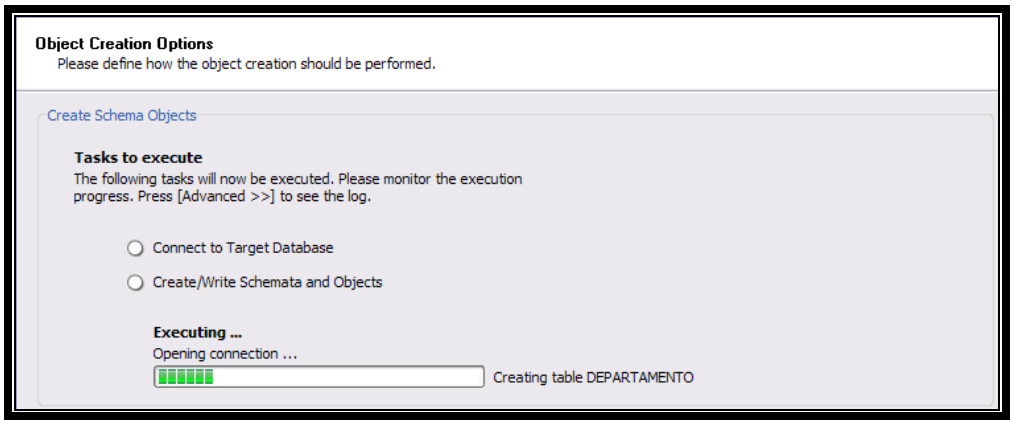
El archivo lo crearemos en la carpeta BASE EN MS ACCESS con el nombre SQLMIBASE.



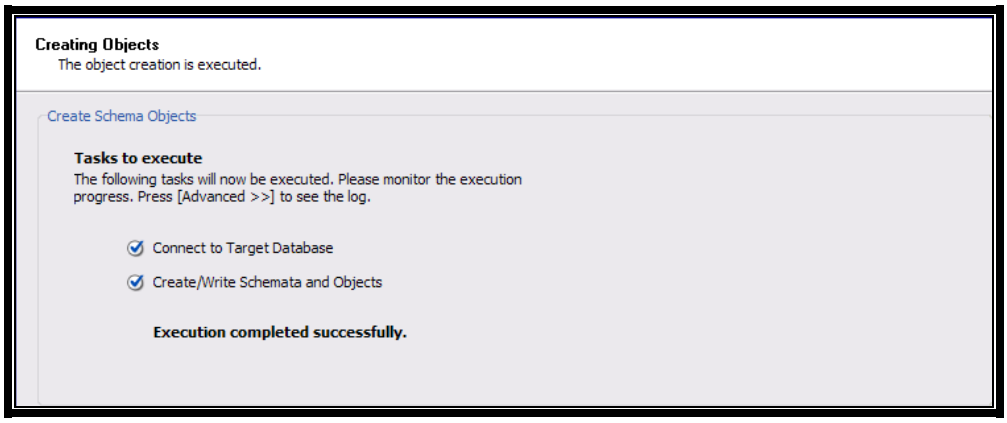
Damos clic en 

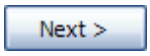
y luego daremos clic en 

, hará entonces el asistente los procesos pertinentes.

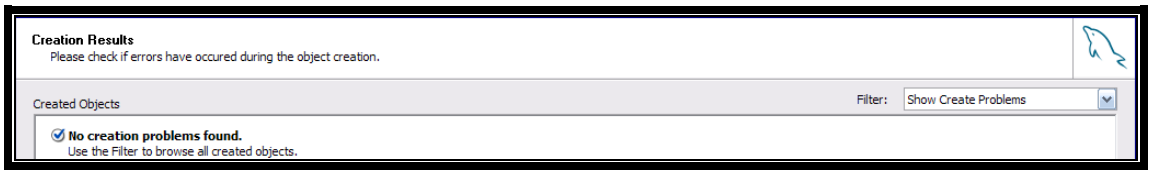


Una vez culminados estos procesos exitosamente la pantalla anterior se mostrará de la siguiente manera:



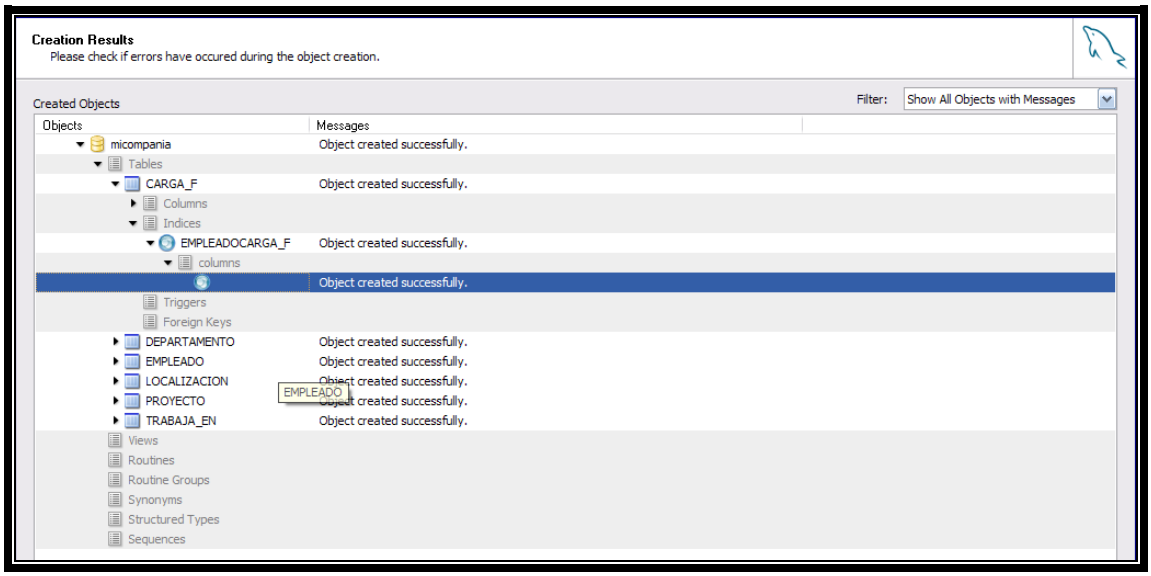
Luego daremos clic en 

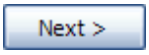
para continuar. Se mostrará la siguiente pantalla en la cual también escogeremos en el campo *Filter: Show All Objects with Messages.*

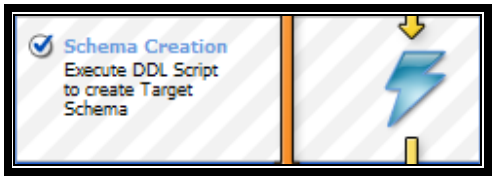


.

Se mostrarán entonces una lista de todos los objetos y si es que hubo algún problema al momento de la creación de los mismos en la base de datos. De igual manera para obtener una visión más de tallada se da clic sobre el objeto que se desea consultar y se irán desplegando sus componentes.

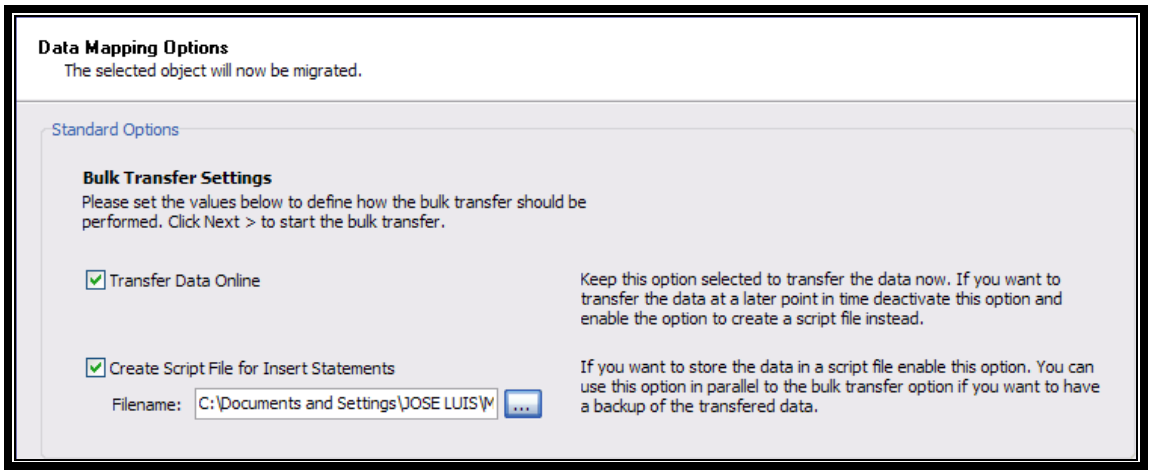


Daremos clic en  para continuar, dará fin entonces la quinta etapa del proceso.

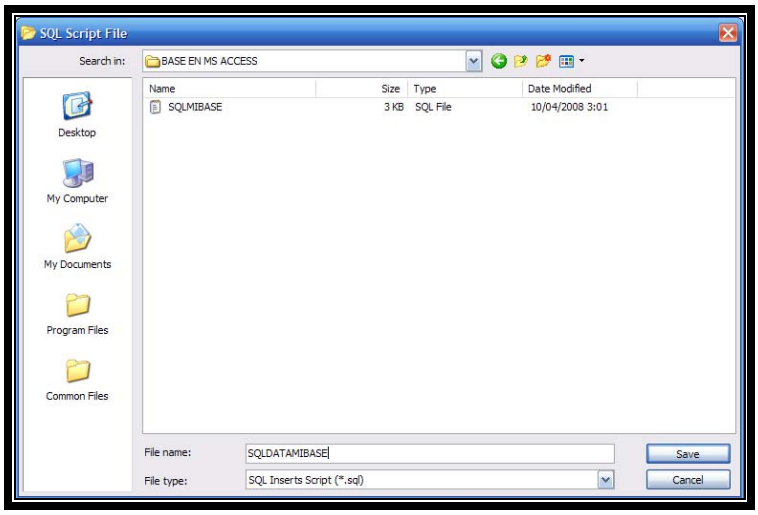


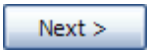
En la sexta etapa se realizará el mapeo de datos.

De igual manera al seleccionar *Create Script File for Insert Statements* vamos a generar un archivo *SQL* con toda la información de los datos contenidos en las tablas.



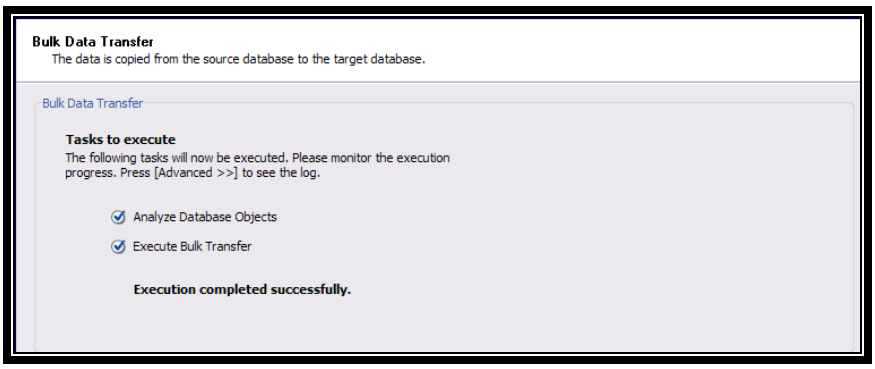
Entonces usaremos el botón  , guardaremos el archivo en la carpeta BASE EN *MS ACCESS* con el nombre SQLDATAMIBASE.

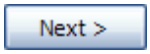


Daremos clic en  , luego daremos clic en  para continuar, habrá culminado entonces la sexta etapa.

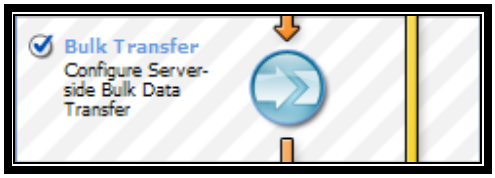
Ahora empieza la séptima etapa, en la cual se copian los datos del origen al destino.

De igual manera, una vez terminados los procesos pertinentes se mostrará la siguiente pantalla:

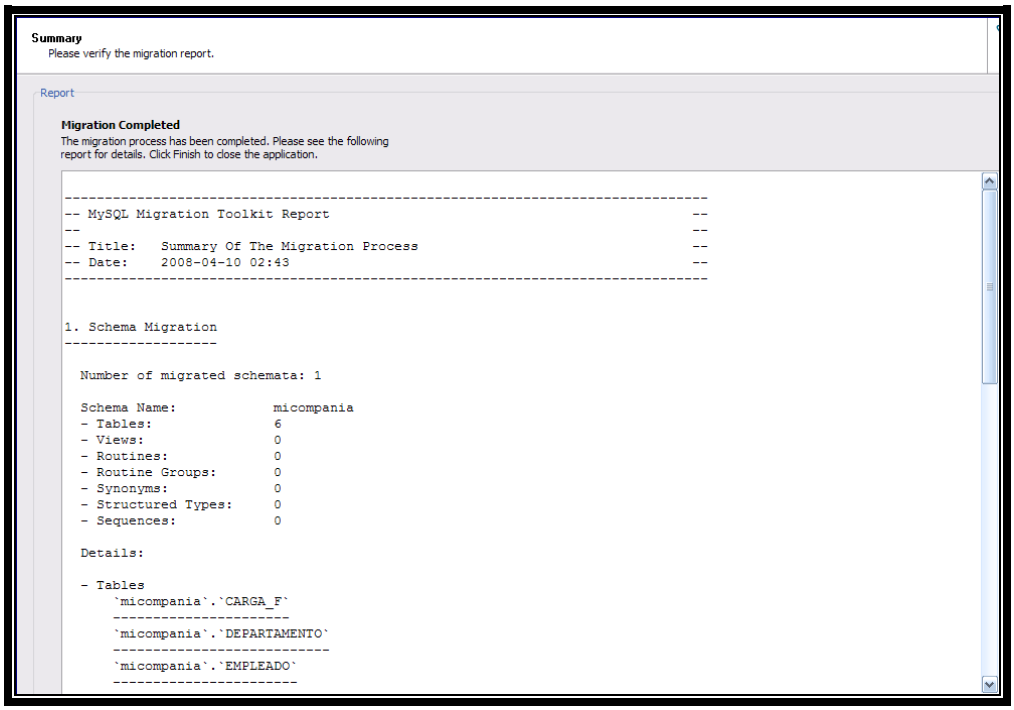


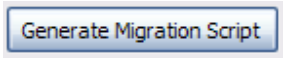
Luego daremos clic en 

para continuar, habrá terminado entonces la séptima etapa.

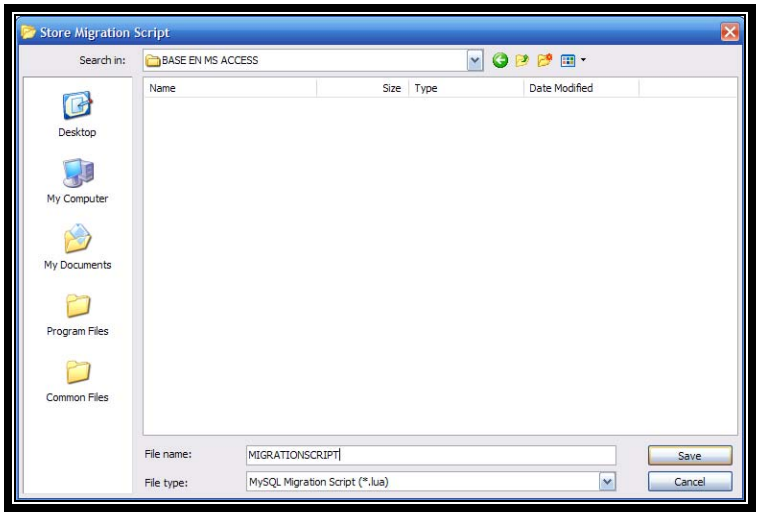


Ahora la base de datos está completamente migrada a *MySQL*. En la octava etapa simplemente se muestra un sumario con toda la información referente a la migración resultante.

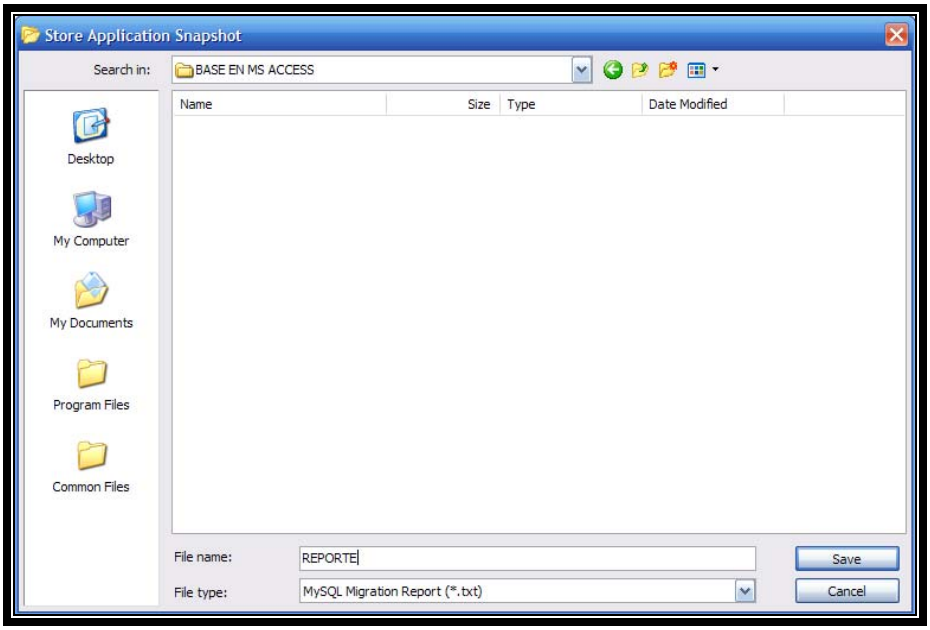


Daremos clic en 

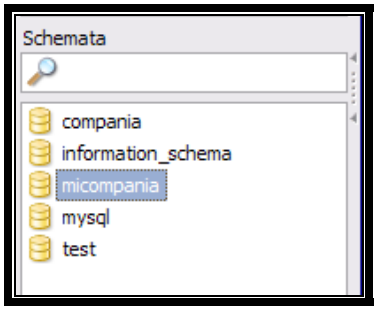
el cual nos permitirá guardar un script con la migración integra, el cual guardaremos en la carpeta BASE EN *MS ACCESS* con el nombre *MIGRATIONSCRIPT*



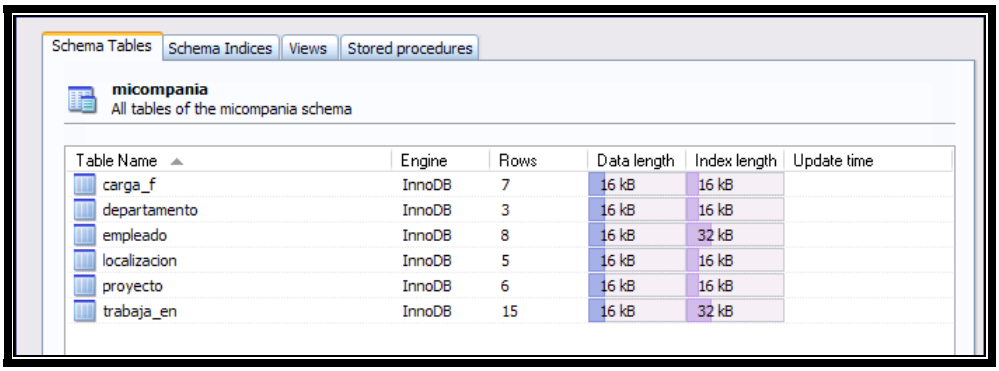
Daremos clic en Save, también guardaremos el reporte de la migración con el botón Save Report to Disk y escogeremos la carpeta BASE EN *MS ACCESS* con el nombre REPORTE.



Daremos clic en Save y por último daremos clic en Finish. Una vez terminado el proceso verificaremos entonces la base de datos. Nos conectamos a *MySQL Administrator*, en *Catalogs*, *Schemata* deberá aparecer entonces la base de datos micompania.

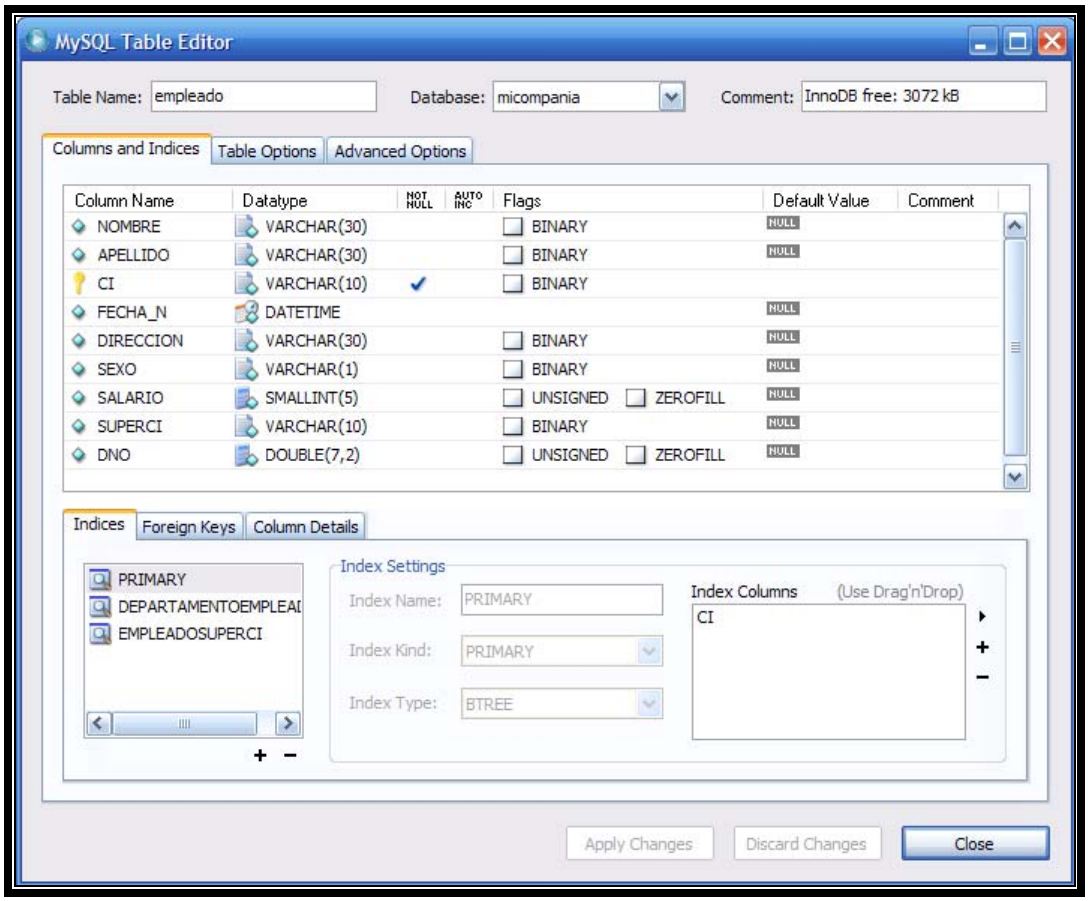


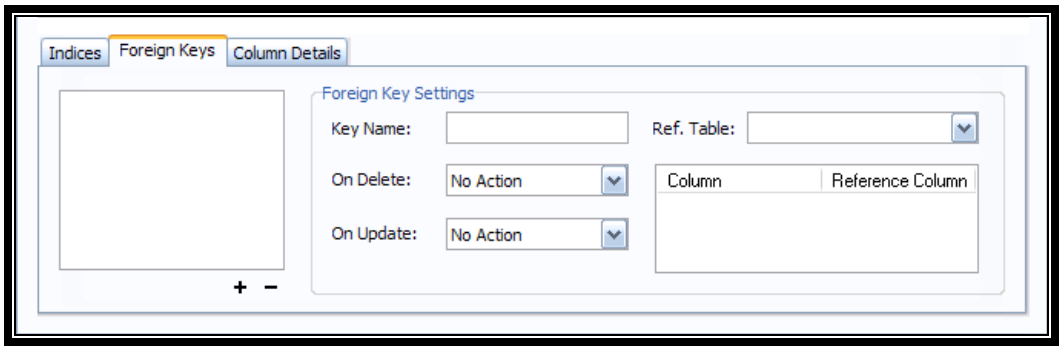
En la pestaña *Schema* tables deben estar todas las tablas como en compania.

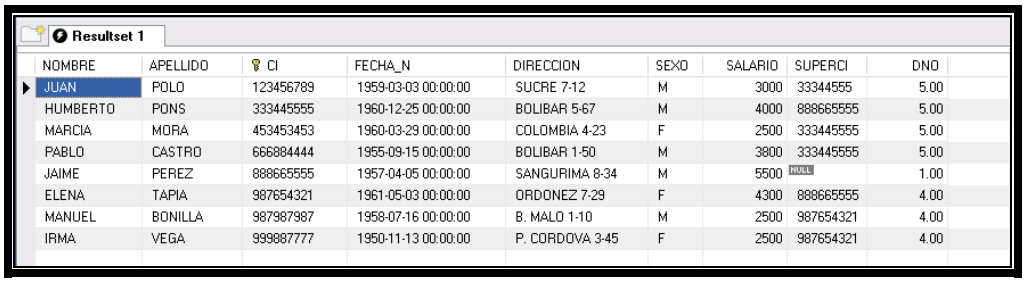


Vamos entonces a verificarla estructura de cada una de las tablas.

TABLA EMPLEADO







Resultset 1

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCION	SEXO	SALARIO	SUPERCI	DNO
JUAN	POLO	123456789	1959-03-03 00:00:00	SUCRE 7-12	M	3000	33344555	5.00
HUMBERTO	PONS	333445555	1960-12-25 00:00:00	BOLIBAR 5-67	M	4000	888665555	5.00
MARCIA	MORA	453453453	1960-03-29 00:00:00	COLOMBIA 4-23	F	2500	333445555	5.00
PABLO	CASTRO	666884444	1955-09-15 00:00:00	BOLIBAR 1-50	M	3800	333445555	5.00
JAIME	PEREZ	888665555	1957-04-05 00:00:00	SANGURIMA 8-34	M	5500	NULL	1.00
ELENA	TAPIA	987654321	1961-05-03 00:00:00	ORDONEZ 7-29	F	4300	888665555	4.00
MANUEL	BONILLA	987987987	1958-07-16 00:00:00	B. MALO 1-10	M	2500	987654321	4.00
IRMA	VEGA	999887777	1950-11-13 00:00:00	P. CORDOVA 3-45	F	2500	987654321	4.00

Como se puede ver existen pequeños cambios en la estructura de las columnas, las llaves primarias están intactas, pero las llaves foráneas se muestran únicamente como índices.