Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Tutorial de Prácticas del Gestor de Base de Datos SQL Server

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

Autor: Juan Orlando Portilla Peña

Director: Ing. Oswaldo Merchán Manzano

Cuenca, Ecuador

2010

INDICE DE CONTENIDOS

RESUMEN

La presente monografía es un tutorial cuyo propósito es el de facilitar el aprendizaje a los estudiantes del manejo de este Gestor de Base de Datos “SQL Server 2008”, también reforzar los conocimientos adquiridos en diseño y administración de base de datos y una revisión del lenguaje SQL.

El tutorial está organizado de una manera secuencial didáctica partiendo desde lo más esencial hasta lo más avanzado de manera que los estudiantes que utilicen el tutorial encuentren un apoyo.

Su contenido parte desde la correcta instalación y configuración del software, administración de base de datos, seguridades, definiciones del lenguaje SQL y ejemplos prácticos.

Al final del tutorial se ha añadido dos anexos los cuales tratan puntos de migración y creación de base de datos.

ABSTRACT

This monograph is a tutorial whose purpose is to facilitate students´ learning of the handling of Database Manager “SQL Server 2008”, as well as also reinforcing the knowledge of database design and administration and a revision of SQL language.

The tutorial is a organized in a sequential manner, dictated starting form the most essential to the most advanced in a way that students who use the tutorial find support.

Its contents begin with the correct installation and configuration of the software, database administration, securities, SQL language definitions, and practical examples.

At the end of the tutorial, two annexes have been added which deal with points about database migration and creation.

INTRODUCCIÓN

SQL Server es un Sistema de Gestión de Bases de Datos, desarrollado por Microsoft, que permite, diseñar y administrar Base de Datos Relacionales. Su interfaz sencilla con el usuario ha hecho que este gestor sea uno de los más utilizados en desarrollo de software.

El tutorial que está organizado en nueve capítulos con sus correspondientes ejercicios y conclusiones.

En el primer capítulo se indica como instalar y configurar del software, se explica cómo conectarse al servidor.

El capítulo dos se estudia la administración de Base de Datos, se enseña cómo crear, eliminar base de datos; creación, modificación y eliminación de tablas. Al final de capítulo se propone un ejercicio.

La seguridad, es el tema a tratar en el capítulo tres. Se enseña a crear usuarios, logins y asignar de privilegios.

Consultas simples, concatenación de datos, consultas con condiciones específicas, consultas con rango de fechas; son los puntos a tratar en el capítulo cuatro.

El capítulo quinto trata temas de manipulación de la información, indica cómo realizar cálculos, ordenar y agrupar los datos bajo ciertos criterios. Al final del capítulo se propone un ejercicio.

Las subconsultas, condiciones de búsqueda en las subconsultas, subconsultas anidadas son los puntos tratados en el capítulo sexto, al final del capítulo se plantea un ejercicio.

La creación, modificación y eliminación; de procedimientos almacenados y triggers son los puntos tratados en el capítulo séptimo.

En el capítulo octavo consta una comparación entre los gestores de base de datos SQL Server y MySQL, se plantea un ejercicio final que abarque todos los conocimientos expuestos.

Al final se ha incluido dos anexos, el primero indica cómo crear una Base de Datos utilizando el entorno gráfico de SQL Server Management, el segundo anexo enseña a migrar una base de datos desde otro gestor a SQL Server.

CAPITULO 1

SQL SERVER DEVELOPER 2008

INTRODUCCION

El presente capítulo hace referencia a la instalación del gestor de base de datos SQL Server 2008 Developer Edición en el sistema operativo Windows XP SP2, una descripción de los componentes de SQL Server, la manera de conectarse al gestor de base de datos y una breve definición del lenguaje SQL.

1.1Instalación de SQL Server 2008

Requisitos Software:

- S.O. Windows XP SP2
- Net Framework 3.5 SP1
- Windows Instaler 4
- Internet Explorer 6.0 o superior

Requisitos Hardware

- Memoria 512 MB como mínimo.
- Tarjeta de Red.

- Procesador como mínimo Pentium III de 1 GHz o superior.
- Disco 1 Gb de espacio, depende de las características a instalar.

Ingresamos a la carpeta SQL Server y ejecutamos el archivo setup.exe y nos aparecerá la pantalla de bienvenida, seleccionamos la opción **Installation**

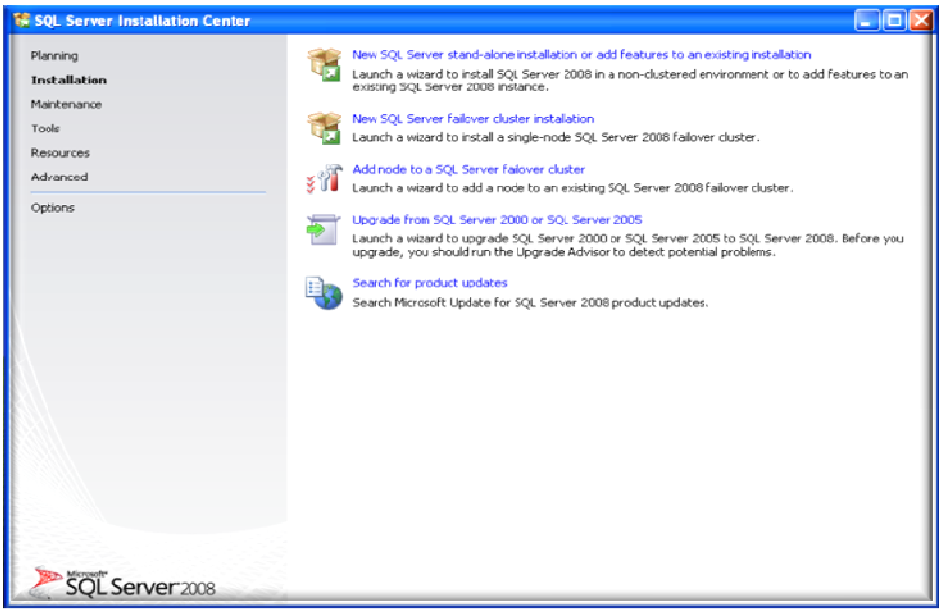


Figura 1.1. Centro de Instalación

Ingresamos la llave del producto y pulsamos el botón **Next >**

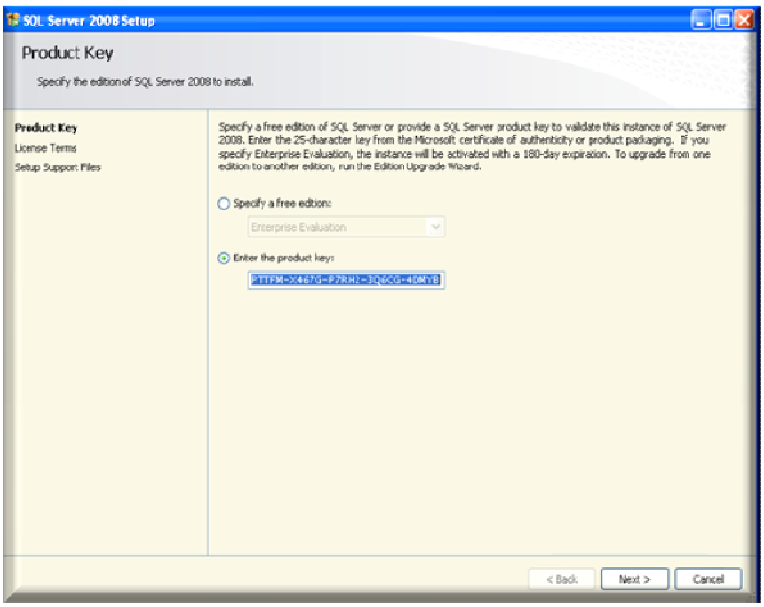


Figura 1.2. Ingreso de la llave del producto

Aceptamos los términos de la licencia, seleccionamos el casillero

y Presionamos el botón

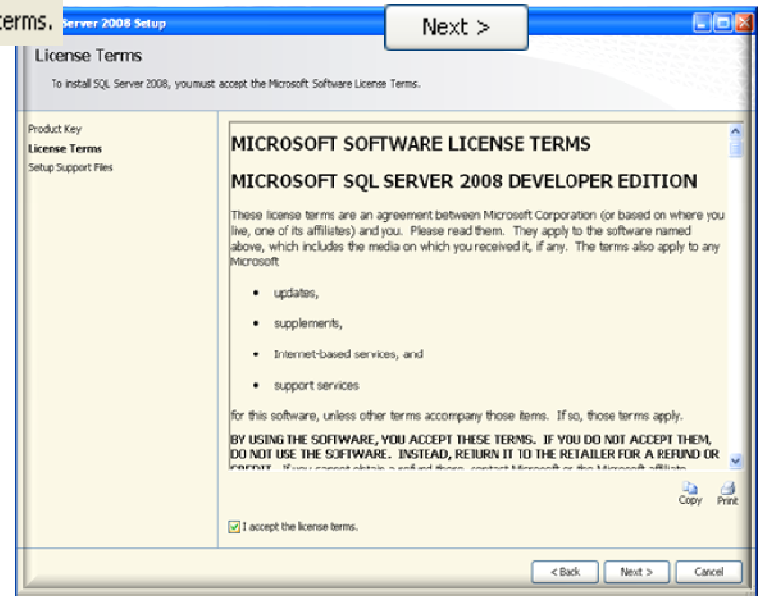


Figura 1.3. Términos de la licencia

La siguiente pantalla indica si hay problemas, si el quipo soporta los archivos de instalación, si todo esta correcto pulsamos el botón

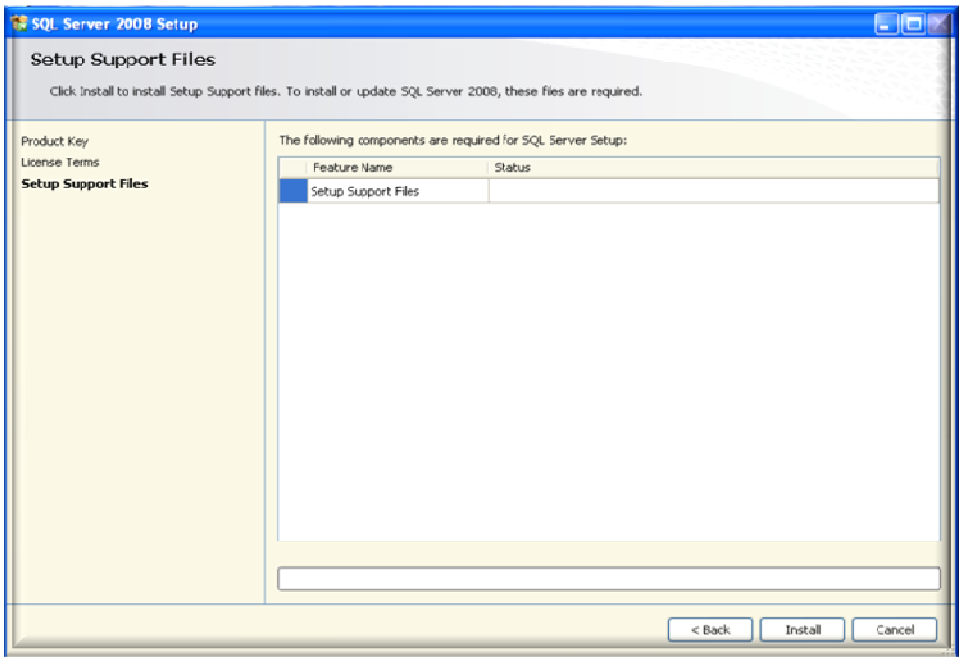
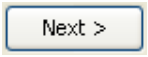


Figura 1.4. Archivos de Soporte de Instalación

1.2 Configuración de SQL Server 2008

En la instalación seleccionamos los siguientes componentes:

- Database Engine Services: Es el núcleo principal de SQL Server e instala el motor, archivos de datos, para que ejecute SQL Server.
- Reporting Services: Este componente permite generar informes.
- Client Tools: Estas herramientas proporcionan la interfaz gráfica para la administración de SQL Server.
- SQL Server Books Online: Este es el sistema de ayuda. Si se necesita más información, aclaración, o detalles adicionales en cualquier de SQL Server, entonces este es el área a recurrir.

Una vez seleccionado los componentes a instalar presionamos el botón 

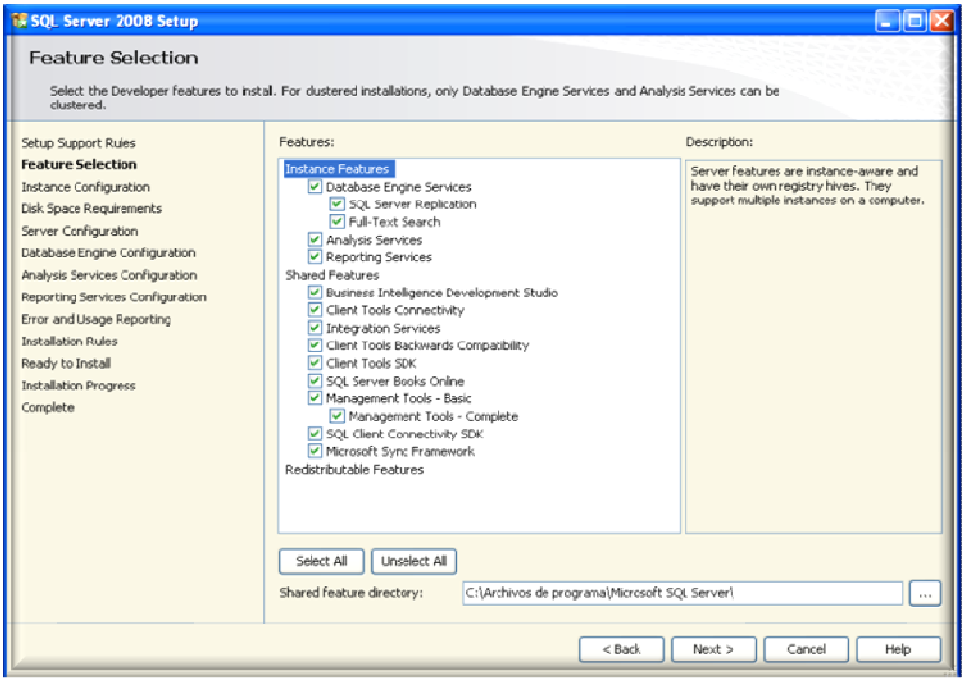
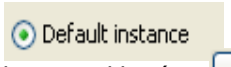
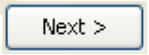


Figura 1.5. Selección de Componentes a instalar

Como nombre de instancia (motor) seleccionamos la opción 

para que el instalador nos proporcione un nombre por defecto y pulsamos el botón 

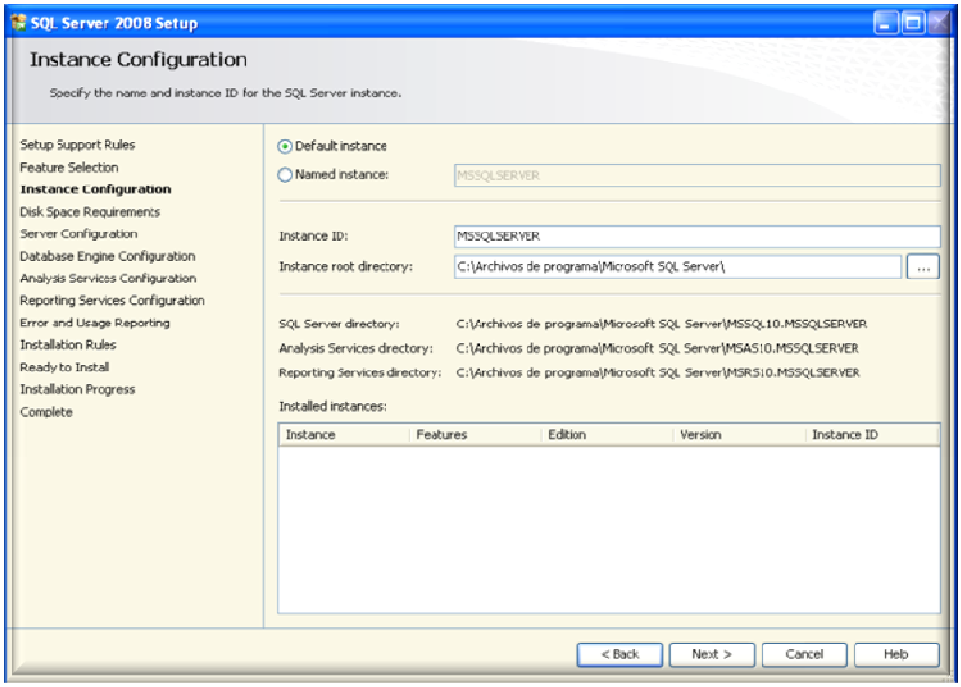


Figura 1.6. Configuración de la Instancia

En la siguiente pantalla indica si cumple con los requisitos de espacio de disco, si todo esta correcto pulsamos el botón 

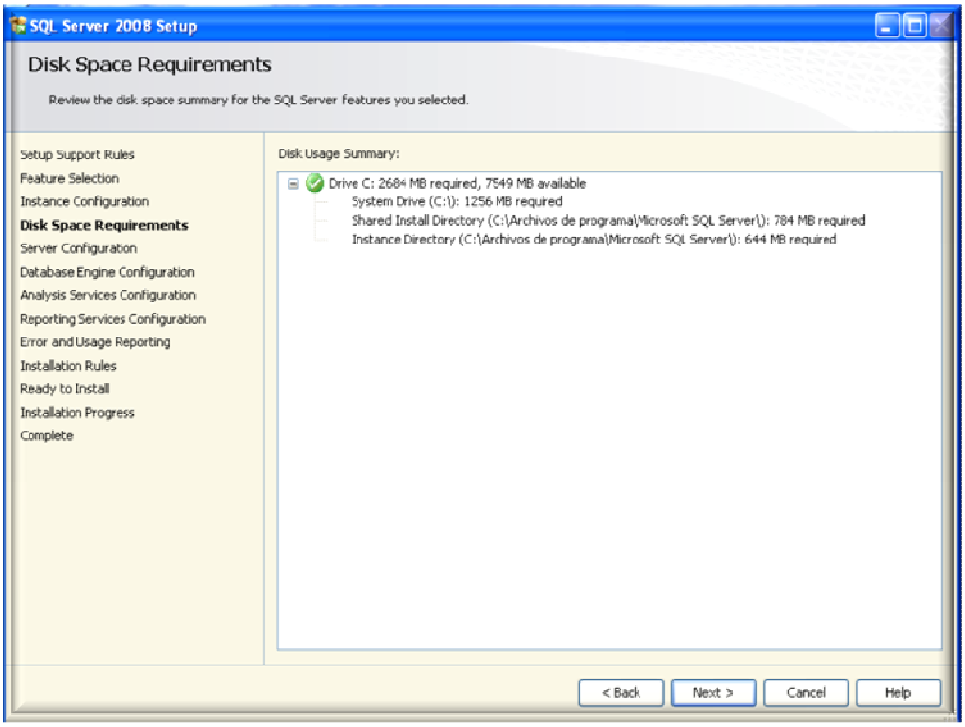
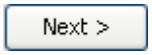


Figura 1.7. Requisitos de Espacio de Disco

Indicamos que cuenta va a levantar los servicios y especificamos la opción manual o automático, se recomienda seleccionar la opción automático, otra manera de levantar los servicios es atreves de servicios del panel de control de Windows,
presionamos el botón. 

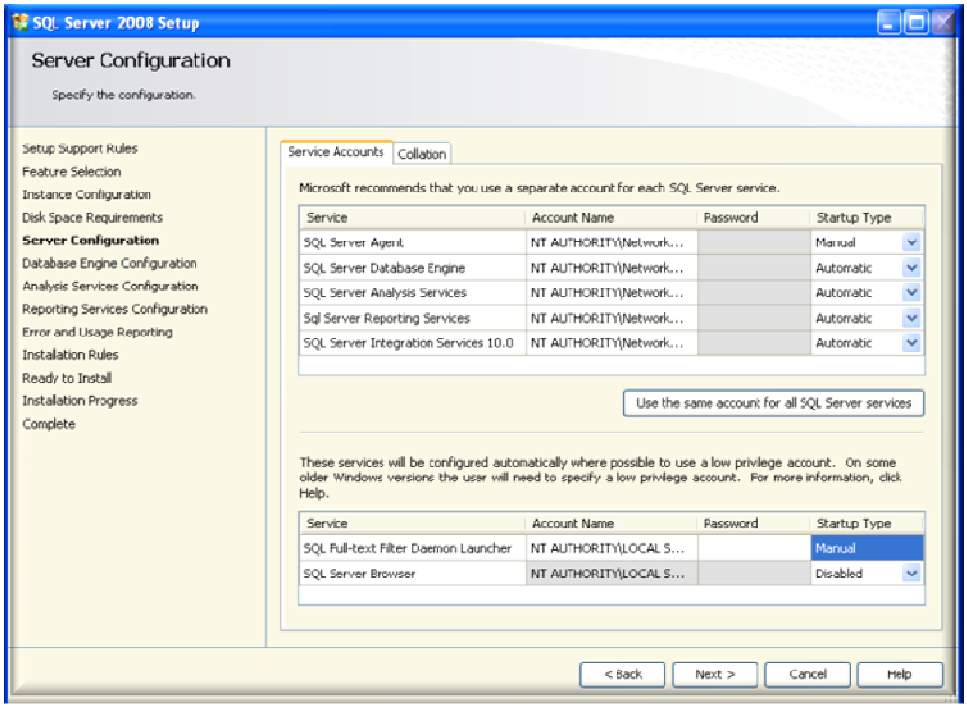


Figura 1.8. Configuración de Servicios

Especificamos el modo de seguridad y los administradores del motor de base de datos, para el modo de seguridad tenemos dos opciones.

Windows authentication mode Esta opción utiliza la seguridad de Windows para iniciar la sesión de SQL Server.

Mixed Mode (SQL Server authentication and Windows authentication)

El modo mixto utiliza la seguridad de Windows o de SQL Server definido ID de usuario y contraseña. En nuestra instalación seleccionamos **Windows authentication mode,** de esta manera el sistema operativo se encarga de validar el inicio de sesión**.**

Seleccionamos el administrado (cuenta especial) de SQL Server, presionamos el botón Add...

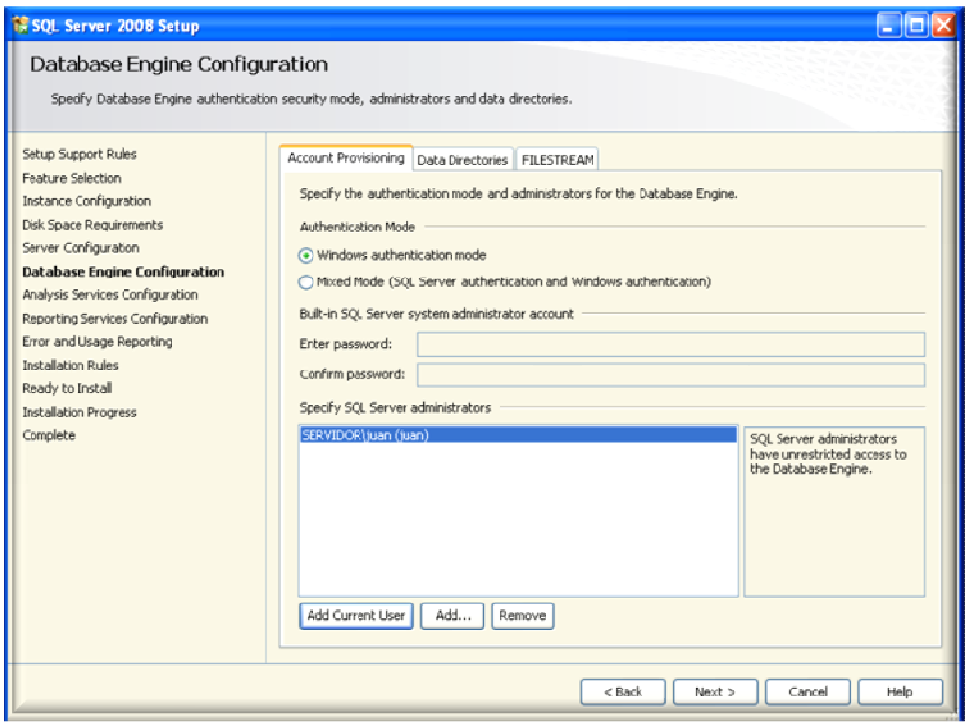


Figura 1.9. Modo de Seguridad y Administradores

Detallamos que usuario tiene los derechos administrativos para el análisis de servicios, este usuario no tiene restricciones, para agregar un usuario presionamos el botón. Add...

Pulsamos el botón Next >

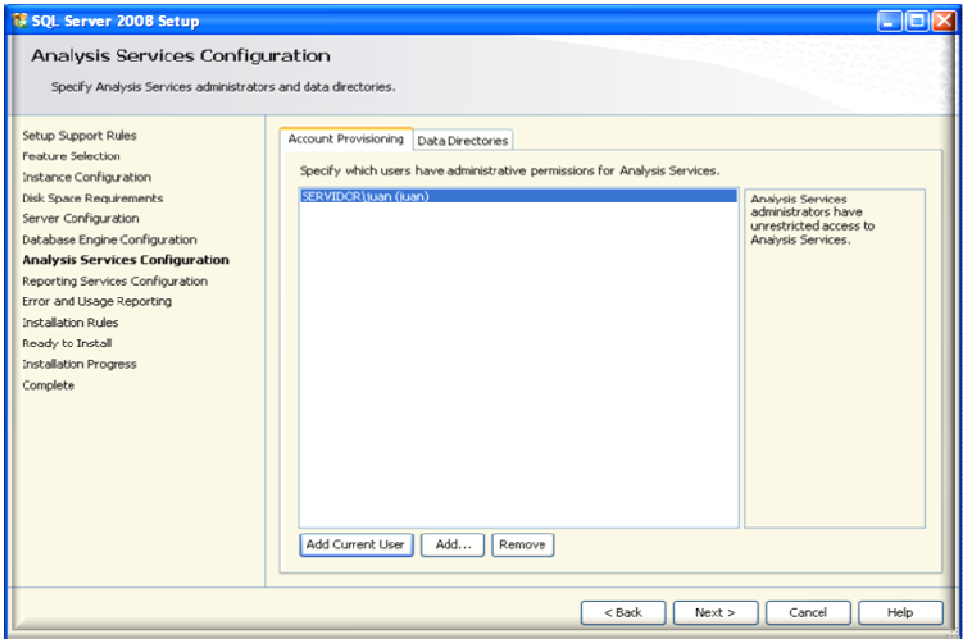
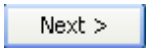


Figura 1.10. Configuración de Usuarios

Creación de la base de datos de Reporting Services (presentación de informes), seleccionamos la opción 

presionamos el botón 

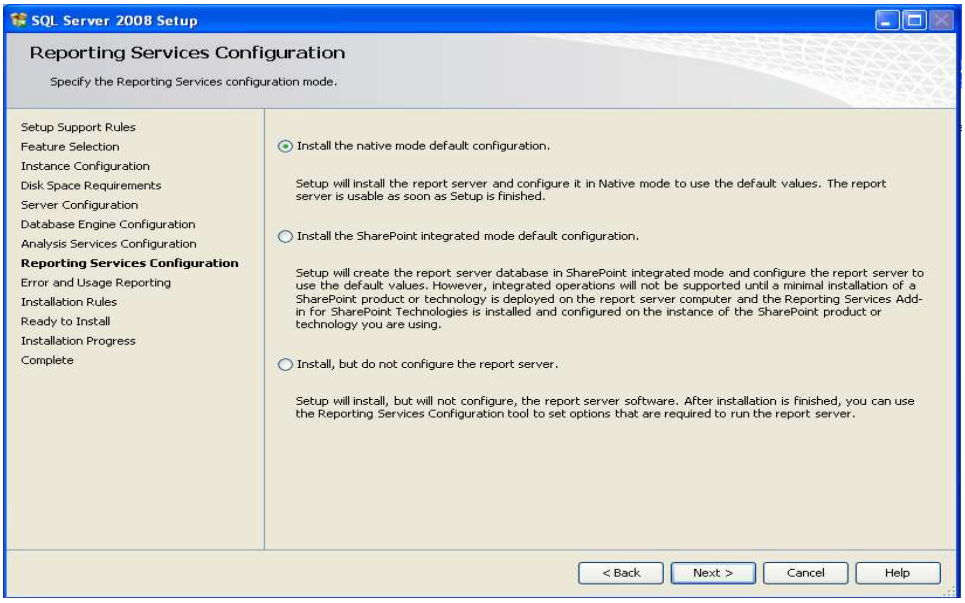
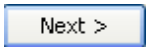


Figura 1.11 Configuración Reporting Services

La activación de envio de errores a la Microsoft es opcional en nuestro caso no lo activaremos, pulsalmos el boton 

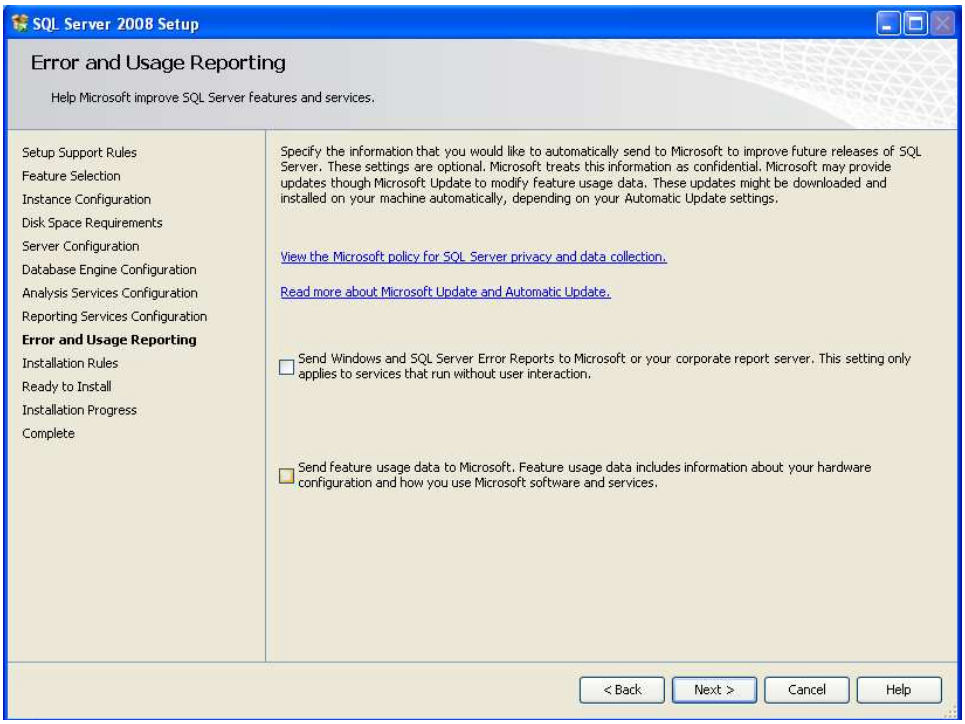
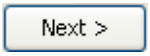


Figura 1.12. Activación de Errores

En la siguiente pantalla muestra si las reglas de instalación se han cumplido, si no es asi, la instalación se detendrá, pulsamos el botón 

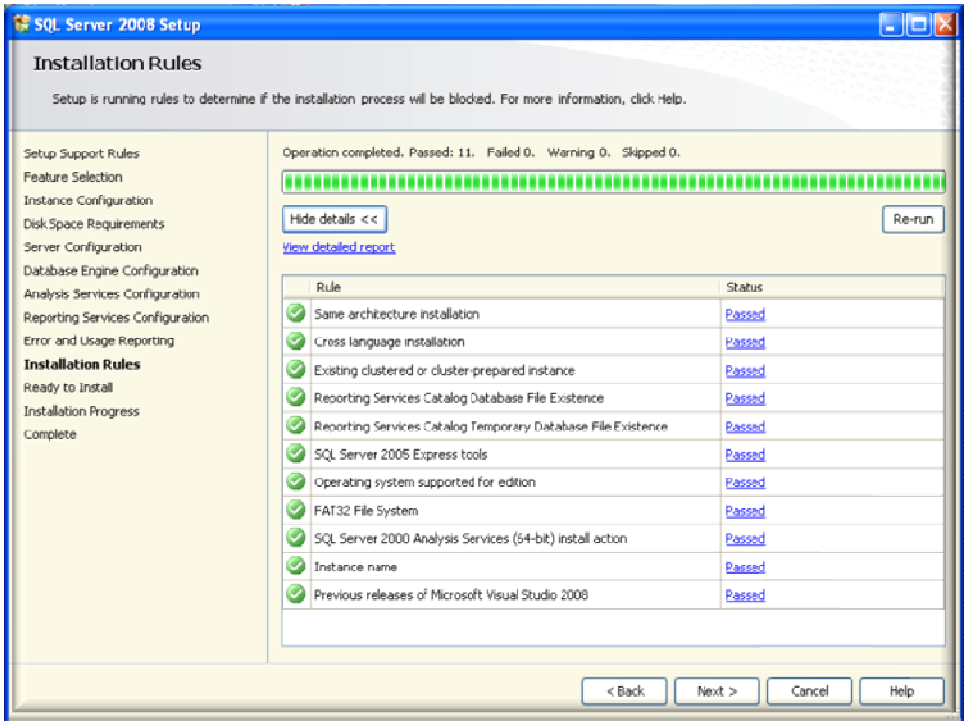


Figura 1.13. Reglas de Activación

Se verifica las características a instalar, pulsamos Install

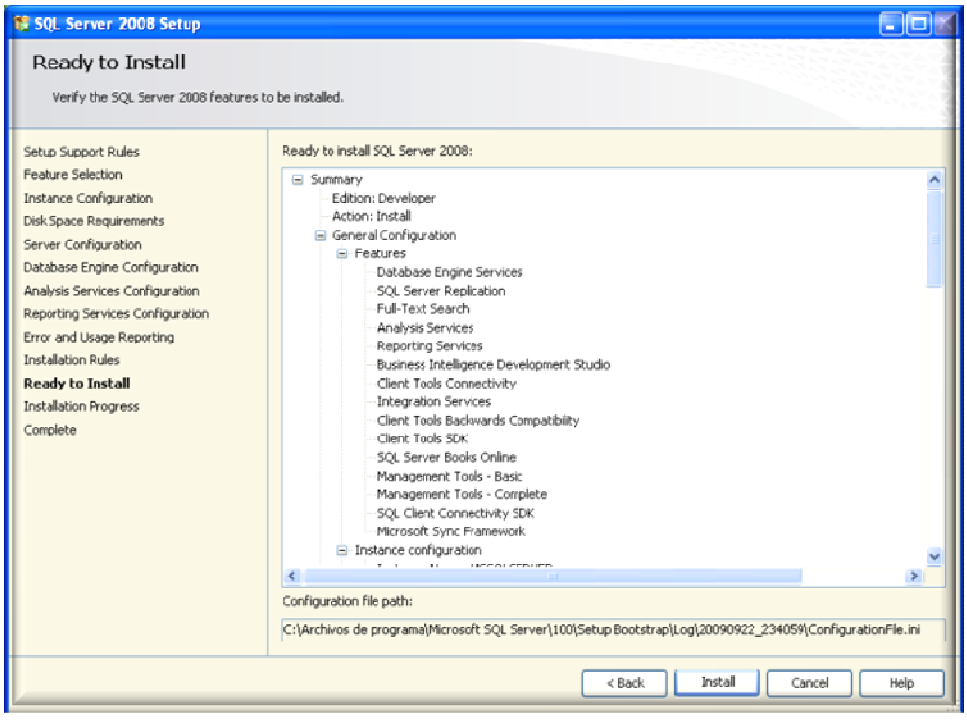
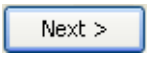


Figura 1.14. Verificación de Características a Instalar

A continuacion nos muestra el estado de cada componente, si sea instalado correctamente, pulsamos el botón 

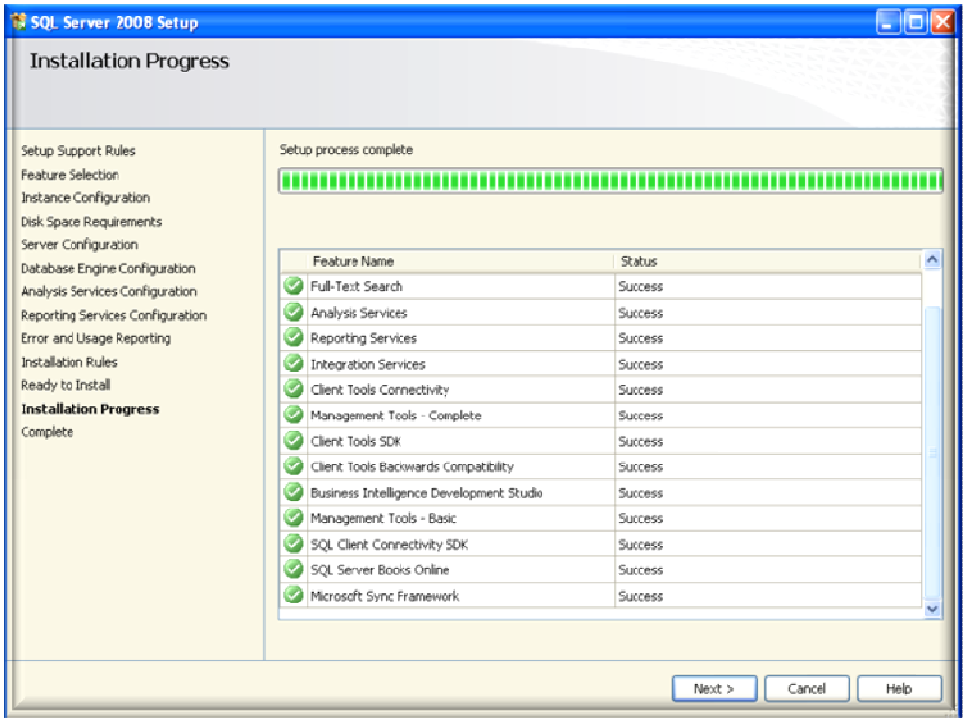
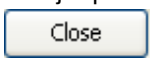


Figura 1.15. Estado de Instalación de cada Componente

Al final nos muestra un mensaje que la instalación de Sql Server 2008 se ha instalado con éxito, pulsamos el botón 

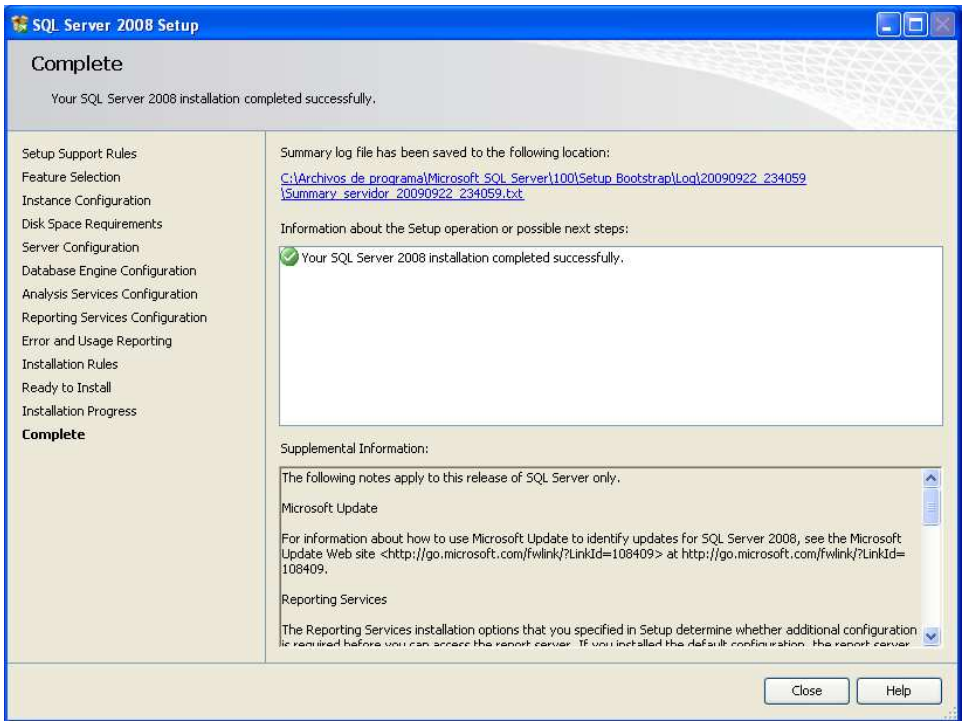


Figura 1.16. Mensaje de Instalación Exitosa

1.3 Conexión de SQL Server 2008

1.3.1 SQL Server Management

La mayor parte de las tareas de administración de base de datos SQL Server 2005 se realizan con SQL Server Management Studio, es un entorno integrado para acceder, configurar, administrar y manejar todos los componentes de SQL Server. Para ingresar nos ubicamos en el botón Inicio> Programas> Microsoft SQL Server 2008> SQL server Management 2008, aparece una pantalla la cual nos pide el tipo de servidor seleccionamos “DataBase Engine”, el nombre del servidor en este caso el nombre con el cual configuramos y la autentificación que sería la autentificación de Windows ya que de esta manera se configuro, y pulsamos el botón.

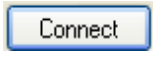


Figura 1.17 Conexión con el Servidor

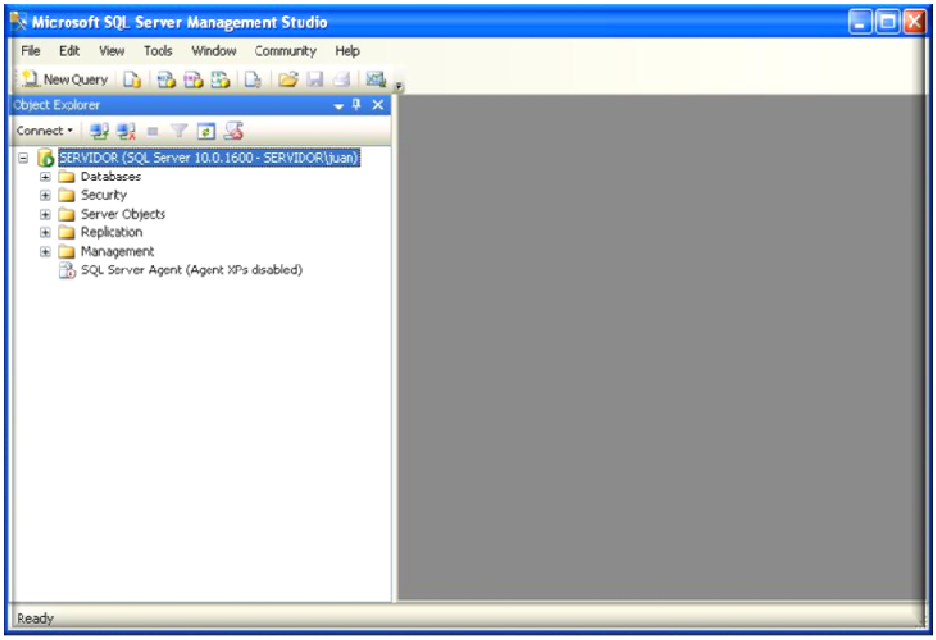


Figura 1.18 SQL Server Management

En la ventana de SQL Server Management tenemos varios elementos los cuales los más relevantes detallamos a continuación:

Object Explorer: Administra los objetos en la base de datos de SQL Server. Object Explorer presenta los objetos en una manera jerárquica, usando una vista tipo árbol agrupado por tipos de objetos.

- Databases: Visualiza las bases de datos del sistema y del usuario.
- Security: Provee accesos a los logins del servidor, roles del servidor, servidores conectados, y a servidores remotos.
- Server Object: Detalles de objetos tales como los dispositivos de copia de seguridad y proporciona una lista de servidores vinculados.
- Replication: Muestra la información relativa a los datos de replicación de una base de datos de este servidor a otra base de datos.
- Management: Detalles de los planes de mantenimiento, gestión de políticas, recopilación de datos.
- SQL Server Agent: Establece y ejecuta las tareas dentro de SQL Server en determinados momentos, con los detalles de los éxitos o de fallos.

Botones de Object Explorer

-  Connect: Este botón muestra el cuadro de Connect to Server que le permite conectarse a una instancia de SQL Server.
-  Disconnect: Este botón cierra la conexión al servidor seleccionado.
-  Stop: Se puede presionar este botón para que el Object Explorer detenga una operación que esta tomando mucho tiempo.
-  Refresh: Este botón refresca la información mostrada en Object Explorer.
-  Filter: Este botón permite seleccionar objetos filtrados que el Object Explorer muestra, por ejemplo, mostrar solo tablas dentro de un schema específico.
-  Schema: Este botón te permite agrupar objetos, como tablas, por su shema prefijo o por tipo de objeto.

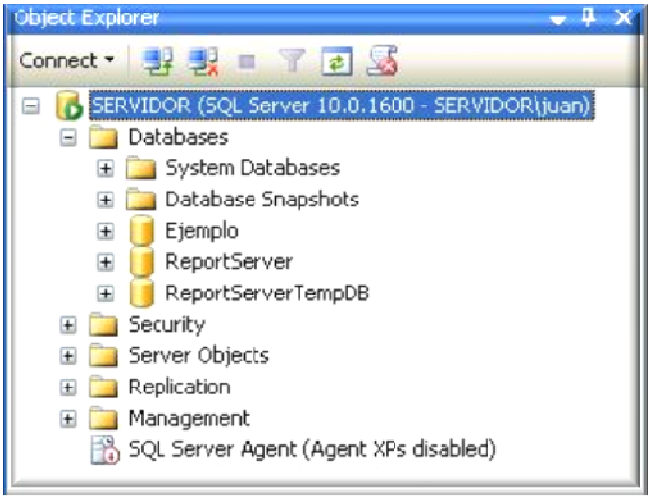


Figura 1.19. Object Explorer

Barra de Herramientas

New Query: Abre una nueva ventana de consulta sql.


DataBase Engine Query: Abre una nueva ventana de consulta con una conexión diferente.


Service Mdx Query, DMX, XMLA, estos tres botones permiten construir diferentes tipos de análisis de consultas.


SQL Server Compact Edition: Esta opción permite manejar base de datos para dispositivos móviles.

1.3.2 Query Editor

Para escribir código T-SQL (dialecto sql propio del gestor), el gestor SQL Server proporciona un editor, para ingresar al editor de consultas procedemos de la siguiente manera: Ingresar al menú Archivo> New> DataBase > Database Engine Query (nueva conexión) o Query With Current Connection (conexión actual).

Barra de Herramientas del Query Editor


Connect, Change Connection: Los primeros dos botones permiten realizar las conexiones con el servidor. El primer botón conecta con el servidor, el segundo botón permite cambiar la conexión.

Avilable Databases: Muestra todas las bases de datos residentes en el servidor, para realizar una consulta se debe seleccionar primero una base de datos.

Execute, Debug, Cancel, Parse: Los botones siguientes se ocupan de la ejecución del código introducido en el Editor de consultas. El botón rojo con el signo de exclamación ejecuta el código. El botón verde es el depurador. El botón gris que se vuelve rojo cuando el código se está ejecutando, para cancelar la ejecución se presiona este botón. El último botón analiza el código, pero no ejecuta, verifica que la sintaxis sea correcta.

Desing Query in Editor: Asistente el cual permite diseñar consultas por la selección de tablas y columnas a través de casillas de verificación.

Results to text, to grid, to File: El primer botón presenta el resultado de la consulta en formato de texto, el segundo en una cuadrícula y el último manda a guardar en un archivo.

El primer botón permite comentar líneas de código, el segundo botón des comenta el código. Los botones de tercero y cuarto insertan o eliminan espacios.

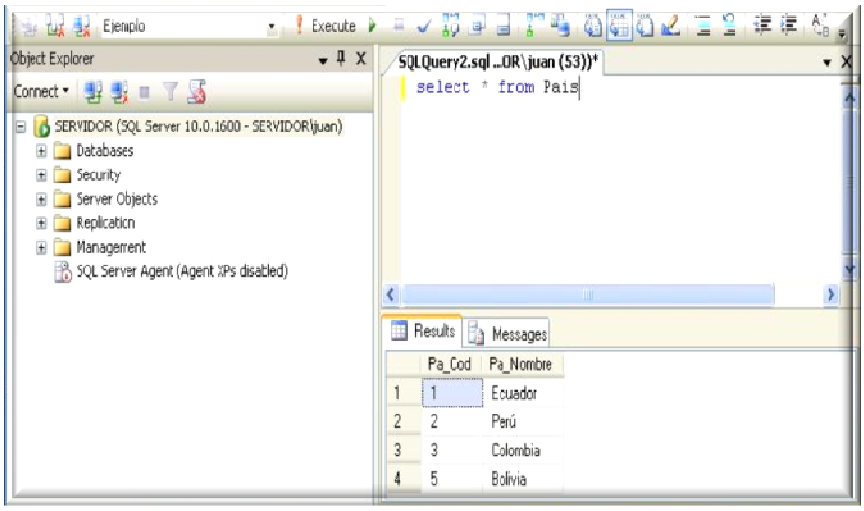


Figura 1.20 Consulta a la tabla País perteneciente a la base de datos Ejemplo

1.4 Conclusiones

Para una correcta instalación se debe seguir cada uno de los pasos, detallados anteriormente, se recomienda tener instalado Net Framework 3.5 SP1 y Windows Instaler 4 para Windows XP 32 bits, estos componentes son necesarios para proceder con la instalación de SQL Server.

CAPITULO 2

ADMINISTRACION DE BASE DE DATOS

INTRODUCCION

En este capítulo se estudia la administración de Base de Datos, utilizando SQL Server Management, se trataran puntos como crear, eliminar base de datos; creación, eliminación de tablas. Al final del capítulo se propone un ejercicio práctico en el cual se debe poner en práctica todo lo aprendido.

2.1 Base de Datos

2.1.1 Creación de Base de Datos

La base de datos a crear es tomada del ejemplo de un modelo Entidad – Relación para una Compañía del libro (Fundamentals of DataBase Systems. Elmasri/Navathe)

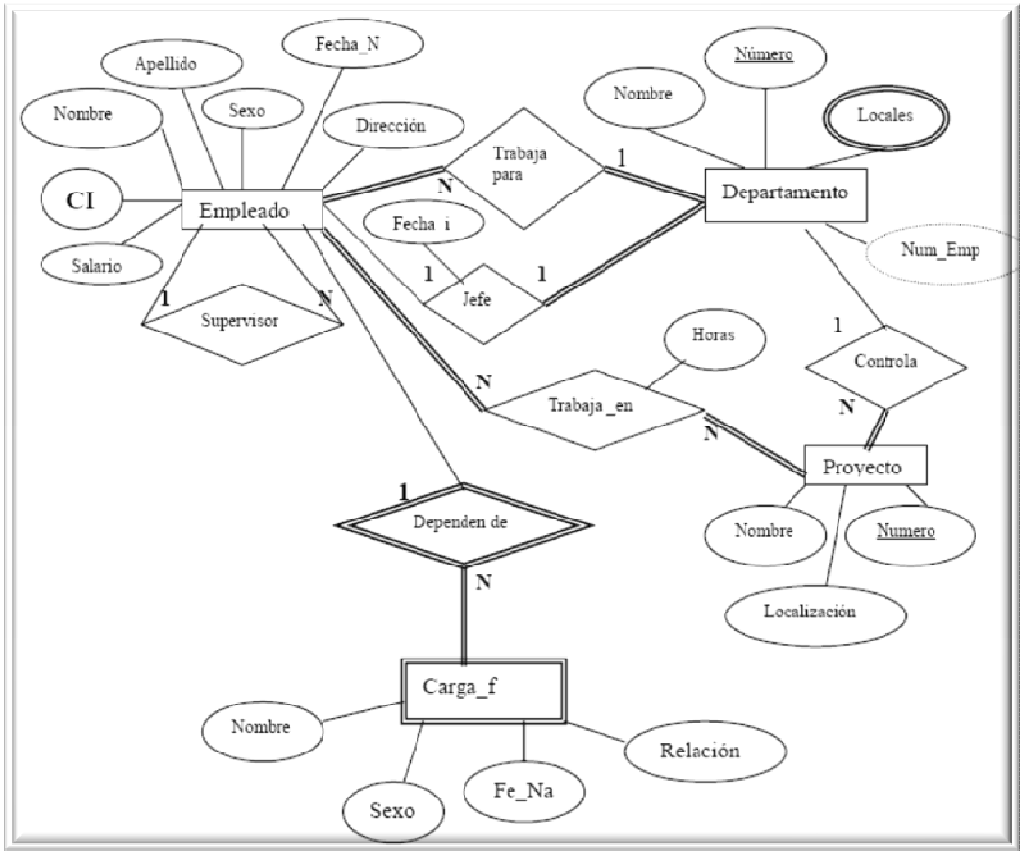


Figura 2.1 Modelo entidad relación de la base de datos Compañía

Nos conectamos al servidor y presionamos el botón, Inicio> Todos los Programas> Microsoft SQL Server 2008> SQL Server Management Studio> Presionamos el botón Connect

Figura 2.2 Conexión con el Servidor

Presionamos el botón de New Query y ejecutamos la siguiente sentencia

CREATE DATABASE nombre_base_de_datos

En el ejercicio el nombre es Compañía, cambiando la n por la ñ.

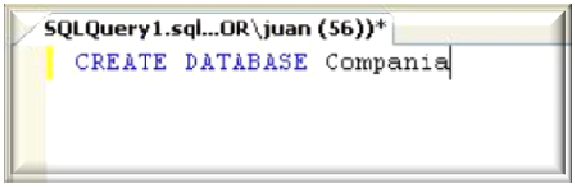


Figura 2.3 Creación de la Base de Datos Compañía

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

USE Compania

2.2 Relación entre tablas

Existen tres tipos de relaciones las cuales son:

2.2.1 Relación N:M

También llamada muchos a muchos. Cada ocurrencia de una entidad puede relacionarse con varias ocurrencias de otra entidad y viceversa, cuando existe una relación de este tipo debe crearse una nueva tabla la cual está compuesta por los atributos de la relación en caso de tenerlas, Ej.: un empleado trabaja en varios proyectos y en un proyecto trabajan varios empleados .

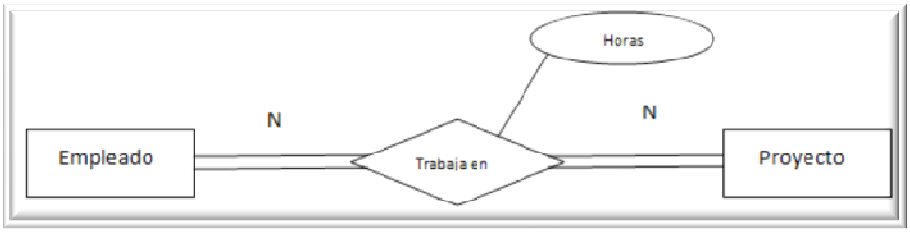


Figura 2.4. Relación N:M

2.2.2 Relación 1:N

También llamada uno a muchos. Cada ocurrencia de una entidad puede relacionarse con varias ocurrencias de otra entidad. En este caso la entidad que tiene la cardinalidad N recibe la llave foránea Ej.: en un departamento trabajan varios empleados.

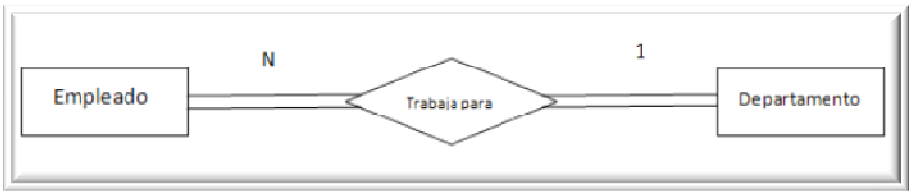


Figura 2.5 Relación 1:N

2.2.3 Relación 1:1

Cada ocurrencia de una entidad se relaciona con una ocurrencia de otra entidad, la propagación de la llave se puede hacer de forma bidireccional. Ej.: un empleado es jefe de un departamento y un departamento tiene un empleado que es jefe. Se representa como indica la Figura

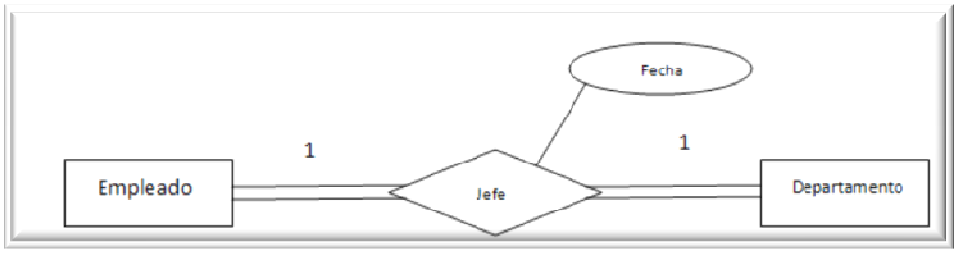


Figura 2.6. Relación 1:1

2.3 Creación de Tablas

Para crear tablas en la base de datos utilizamos la sentencia CREATE TABLE.

CREATE TABLE nombre_tabla(nombre_campo tipo Not Null, ... , nombre_campo tipo)

Not Null: Evita que se introduzcan valores nulos

Para ver la estructura de una tabla usamos el procedimiento almacenado "sp_columns" junto al nombre de la tabla:

sp_columns nombre_tabla

A continuación se creara las siguientes tablas en la base de datos compañía.

EMPLEADO

NOMBRE	APELLIDO	CI	FECHA_N	DIRECCIÓN	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	3-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 Localización

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 ejecutaremos los siguientes comandos:

```
CREATE TABLE empleado (nombre varchar(30),apellido varchar (30),
                       ci varchar (10)not null, fecha_n date,
                       direccion varchar(30),sexo char(1),
                       salario int, superci varchar(10),
```

dno int)

CREATE TABLE departamento (dnombre varchar(30),dnumero int not null,
jefeci varchar(10), jefe_fi date)

CREATE TABLE localizacion (dnumero int not null,dep_loca varchar(30))

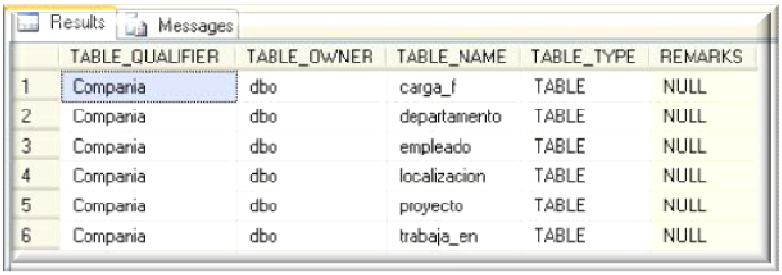
CREATE TABLE trabaja_en (eci varchar(10) not null, pno int not null, horas decimal(4,2))

CREATE TABLE proyecto (pnombre varchar(30), pnumero int not null, plocal varchar (30),dnum interger)

CREATE TABLE carga_f (eci varchar (10) not null,dep_nom varchar (30),sexo varchar (1),fechan_n date,relacion varchar(10))

Para ver las tablas existentes creadas por los usuarios en una base de datos usamos el procedimiento almacenado "sp_tables @table_owner='dbo' ":

```
sp_tables @table_owner='dbo';
```


Results Messages

	TABLE_QUALIFIER	TABLE_OWNER	TABLE_NAME	TABLE_TYPE	REMARKS
1	Compania	dbo	carga_f	TABLE	NULL
2	Compania	dbo	departamento	TABLE	NULL
3	Compania	dbo	empleado	TABLE	NULL
4	Compania	dbo	localizacion	TABLE	NULL
5	Compania	dbo	proyecto	TABLE	NULL
6	Compania	dbo	trabaja_en	TABLE	NULL

Figura 2.7 Tablas creadas en la Base de Datos Compañía

2.4 Creación de llaves Primarias y Foráneas

2.4.1 Llaves Primarias

Una llave primaria es un campo (o varios) que identifica un solo registro (fila) en una tabla. La llave primaria no puede ser un valor nulo o NULL y una tabla puede tener solo una llave primaria, la siguiente sentencia permite crear llaves primarias.

ALTER TABLE nombre_tabla ADD PRIMARY KEY (“nombre_columna”)

Las llaves primarias pueden ser definidas en el momento de la creación de la tabla, a continuación se indica la sentencia que define la llave primaria en el momento de creación de la tabla.

CREATE TABLE nombre_tabla(nombre_campo1 tipo Not Null, nombre_campo2 tipo, **PRIMARY KEY** (nombre_campo1))

Un campo numérico puede tener un atributo extra "identity". Los valores de un campo con este atributo generan valores secuenciales que se inician en 1 y se incrementan en 1 automáticamente.
Cuando un campo tiene el atributo "identity" no se puede ingresar valor para él, porque se inserta automáticamente tomando el último valor como referencia, o 1 si es el primero.

Procedemos a crear las llaves primarias para las tablas que lo requieren.

Tabla Empleado:

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

Tabla Departamento:

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

Tabla Proyecto:

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

2.4.2 Llaves Foráneas

Un campo que no es clave primaria en una tabla y sirve para enlazar sus valores con otra tabla en la cual es clave primaria se denomina clave foránea, externa o ajena. La llave foránea no puede ser un valor nulo o NULL y una tabla puede tener más una llave foránea.
La siguiente sentencia permite crear llaves foráneas.

ALTER TABLE nombre_tabla1 **ADD FOREIGN KEY** (campo_llave_foranea) **REFERENCES** nombre_tabla2 (campo_llave_primaria);

Continuando con el ejercicio crearemos las llaves foráneas en las tablas de la base de datos Compania.

Tabla Empleado:
ALTER TABLE empleado ADD FOREIGN KEY (dno) REFERENCES departamento (dnumero);
ALTER TABLE empleado ADD FOREIGN KEY (superci) REFERENCES empleado (ci);

Tabla Departamento:
ALTER TABLE departamento ADD FOREIGN KEY (jefeci) REFERENCES empleado (ci);

Tabla Localización:
ALTER TABLE localización ADD FOREIGN KEY (dnumero) REFERENCES departamento (dnumero);

Tabla Trabaja en:
ALTER TABLE trabaja_en ADD FOREIGN KEY (eci) REFERENCES empleado (ci);
ALTER TABLE trabaja_en ADD FOREIGN KEY (pno) REFERENCES proyecto (pnumero);

Tabla Proyecto:
ALTER TABLE proyecto ADD FOREIGN KEY (dnum) REFERENCES departamento (dnumero);
Tabla Carga_f
ALTER TABLE carga_f ADD FOREIGN KEY (eci) REFERENCES empleado (ci);

2.5 Restricciones

Las restricciones (constraints) son un método para mantener la integridad de los datos, asegurando que los valores ingresados sean válidos y que las relaciones entre las tablas se mantengan. Se establecen a los campos y las tablas.
Pueden definirse al crear la tabla ("create table") o agregarse a una tabla existente (empleando "alter table") y se pueden aplicar a un campo o a varios. Se aconseja crear las tablas y luego agregar las restricciones. Se pueden crear, modificar y eliminar las restricciones sin eliminar la tabla y volver a crearla.
El procedimiento almacenado del sistema "sp_helpconstraint" junto al nombre de la tabla, nos muestra información acerca de las restricciones de dicha tabla.

2.5.1 Restricción Default

La restricción "default" especifica un valor por defecto para un campo cuando no se inserta explícitamente en un comando "insert". Podemos agregar una restricción "default" a una tabla existente con la sintaxis básica siguiente:

ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción
DEFAULT valor_por_de_fecto
FOR campo;

El siguiente ejemplo se crea una restricción el cual detalla un valor por defecto “S-D” para el campo dirección.

```
ALTER TABLE empleado
ADD CONSTRAINT RD_direccion
DEFAULT 'S-D'
FOR direccion
```

2.5.2 Restricción Check

La restricción "check" especifica los valores que acepta un campo, evitando que se ingresen valores inapropiados. La sintaxis básica es la siguiente:

ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción
CHECK condicion;

Este tipo de restricción verifica los datos cada vez que se ejecuta una sentencia "insert" o "update", es decir, actúa en inserciones y actualizaciones, a continuación vemos un ejemplo:

ALTER TABLE empleado
ADD CONSTRAINT CK_salario_emp
CHECK (salario>=0);

2.5.3 Restricción Unique

La restricción "unique" impide la duplicación de campos es decir, especifica que dos o más filas no puedan tener el mismo valor en un campo. La sintaxis general es la siguiente:

ALTER TABLE nombre_tabla
ADD CONSTRAINT nombre_restricción
UNIQUE (campo);

En la tabla departamento el campo nombre debe ser único, procedemos de la siguiente manera:

ALTER TABLE departamento
ADD CONSTRAINT UQ_departamento_dnombre
UNIQUE (dnombre);

2.5.4 Restricciones Foreign key

Si intentamos eliminar un registro de la tabla referenciada por una restricción "foreign key" cuyo valor de clave primaria existe referenciada en la tabla que tiene dicha restricción, la acción no se ejecuta y aparece un mensaje de error. Esto sucede porque, por defecto, para eliminaciones, la opción de la restricción "foreign key" es "no action". Lo mismo sucede si intentamos actualizar un valor de clave primaria de una tabla referenciada por una "foreign key" existente en la tabla principal.

La restricción "foreign key" tiene las cláusulas "on delete" y "on update" que son opcionales. Estas cláusulas especifican cómo debe actuar SQL Server frente a eliminaciones y modificaciones de las tablas referenciadas en la restricción. Las opciones para estas cláusulas son las siguientes:

- "no action": indica que si intentamos eliminar o actualizar un valor de la clave primaria de la tabla referenciada (TABLA2) que tengan referencia en la tabla principal (TABLA1), se genere un error y la acción no se realice; es la opción predeterminada.
- "cascade": indica que si eliminamos o actualizamos un valor de la clave primaria en la tabla referenciada (TABLA2), los registros coincidentes en la tabla principal (TABLA1), también se eliminen o modifiquen. La sintaxis completa para agregar esta restricción a una tabla es la siguiente:

ALTER TABLE tabla1
ADD CONSTRAINT nombre_restriccion
FOREIGN KEY (llave_foranea)
REFERENCES tabla2(llave_primaria)
ON DELETE opción
ON UPDATE opción;

En la siguiente instrucción se crea una restricción a la llave foránea “dno”, la cual impide que se actualice o modifique.

ALTER TABLE empleado
ADD CONSTRAINT FK_emp_dep
FOREIGN KEY (dno)
REFERENCES departamento(dnumero)
ON DELETE no action
ON UPDATE no action;

2.5.5 Restricciones foreign key deshabilitar y eliminar

La comprobación de restricciones se puede deshabilitar para modificar, eliminar o agregar datos a una tabla sin comprobar la restricción. La sintaxis general es:

ALTER TABLE nombre_tabla
NOCHECK CONSTRAINT nombre_restriccion;

En el siguiente ejemplo se deshabilita la restricción CK_salario_emp

ALTER TABLE empleado
NOCHECK CONSTRAINT CK_salario_emp;

Para habilitar la restricción:

ALTER TABLE nombre_tabla
CHECK CONSTRAINT nombre_restriccion;

En la siguiente instrucción habilitamos la restricción CK_salario_emp:

ALTER TABLE empleado
CHECK CONSTRAINT CK_salario_emp;

Podemos eliminar una restricción la sintaxis es:

ALTER TABLE nombre_tabla
DROP CONSTRAINT nombre_restriccion;

Para eliminar la restricción CK_salario_emp procedemos de la siguiente manera:

ALTER TABLE empleado
DROP CONSTRAINT CK_salario_emp;

2.6 Edición de Base de Datos

2.6.1. Borrar Base de Datos

Para eliminar una base de datos utilizamos la sentencia Drop:

DROP DATABASE nombre_base_de_datos;

2.6.2 Renombrar Tablas de una Base de Datos

Para modificar el nombre de una tabla utilizamos la siguiente sentencia.

RENAME TABLE nombre_tabla **TO** nuevo_nombre_tabla;

2.6.3 Borrar Tablas de una Base de Datos

Para eliminar tablas utilizamos la sentencia Drop, la sintaxis es la siguiente:

DROP TABLE nombre de la tabla;

2.6.4 Borrar Columnas de una Tabla

Para eliminar columnas de una tabla utilizamos la sentencia Alter:

ALTER TABLE nombre_tabla **DROP COLUMN** nombre_columna;

2.6.5 Añadir Columnas en una Tabla

Para agregar columnas en una tabla utilizamos la siguiente sintaxis:

ALTER TABLE nombre_tabla **ADD** nombre_columna **tipo**;

2.6.6 Ingreso de Registros en una Tabla

La sentencia Insert se utiliza para agregar registros a una tabla, si sólo queremos insertar un valor para un atributo, el resto de los de la tabla deberá contener el valor nulo (NULL). Sin embargo, habrá ciertas ocasiones en que esto no será posible, cuando el atributo esté definido como NO NULO, en cuyo caso deberemos especificar un valor para éste. La sintaxis de esta sentencia es:

INSERT INTO nombre_tabla (nombres_columnas)**VALUES** (datos)

También es posible agregar múltiples filas a través del siguiente formato:

INSERT INTO nombre_tabla1 **SELECT** nombres_columna2 **FROM** nombre_tabla2;

Continuado con el ejercicio ingresaremos datos en las siguientes tablas:

Tabla Empleado:

```
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);
```

Tabla Departamento

```
 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');
```

Tabla Localización:

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

Tabla Proyecto:

```
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);
```

Tabla Trabaja_en:

```
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);
```

Tabla Carga_f:

```
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');
```

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

El objetivo de la sentencia UPDATE es actualizar los valores de una o varias filas de una tabla, sin
necesidad de borrarla e insertarla de nuevo. La sintaxis es la siguiente:

UPDATE tabla **SET** atributo1 = valor1 , atributo2 = valor2, ...
WHERE condición

A continuación se procede a actualizar la tabla empleados:

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


UPDATE empleado
SET superci='333445555', dno=5
WHERE ci='123456789';

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';
```

2.6.8 Borrar Registros de las Tablas de la Base de Datos

El objeto de la sentencia DELETE es el de borrar filas de una tabla. Para poder borrar filas en una
tabla deben de cumplirse reglas de integridad referencial. La sintaxis es la siguiente:

DELETE FROM nombre_tabla **WHERE** condición

2.7 Ejercicio Propuesto

Desarrollar el siguiente modelo Entidad Relación, la base de datos se llamará “ferreteria”, se debe utilizar todos los conocimientos adquiridos en este capítulo.

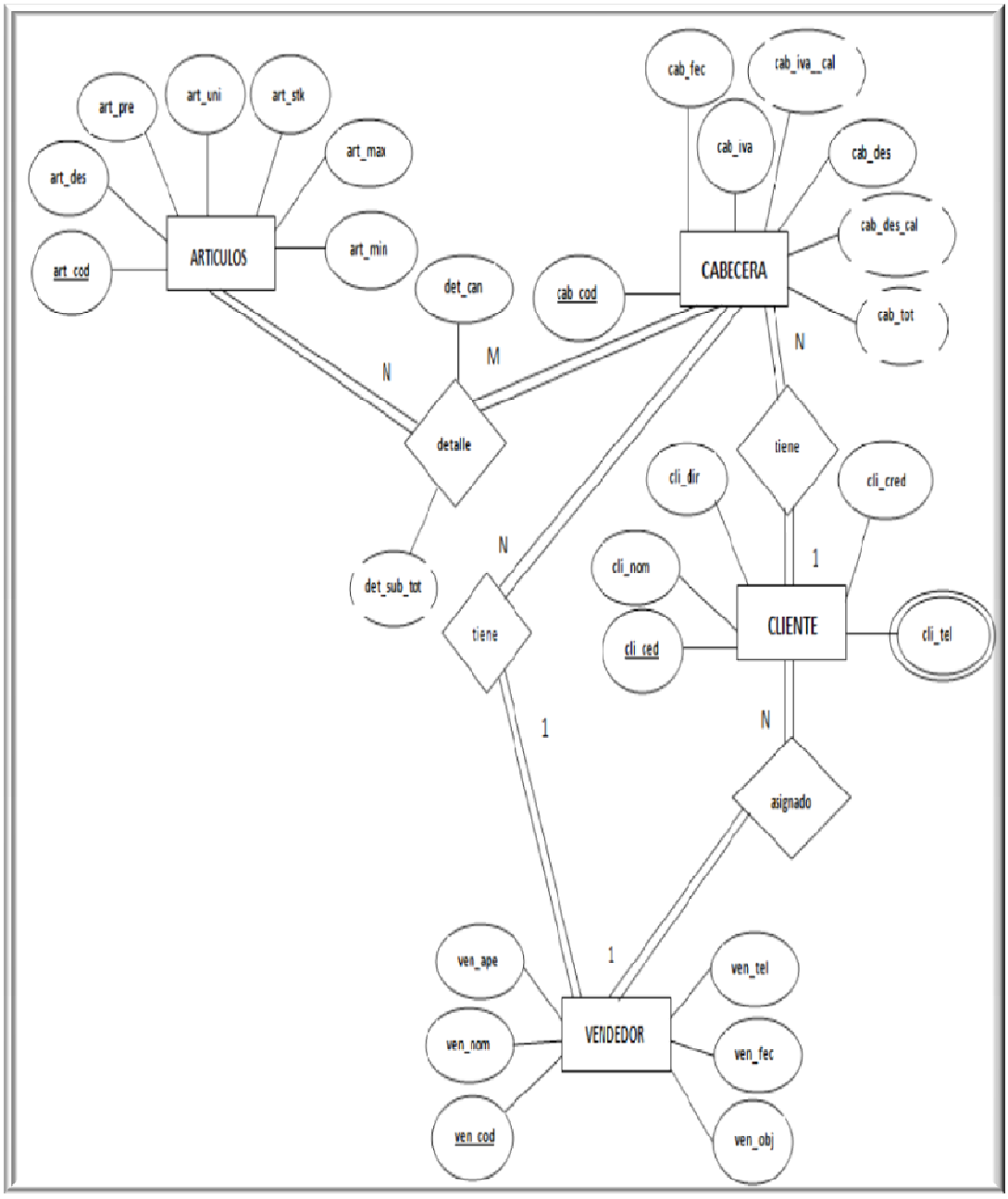


Figura 2.8 Modelo entidad relación de la base de datos Ferretería

	art_cod	art_des	art_pre	art_uni	art_stk	art_max	art_min
1	art001	Cemento	50.00	u	100.00	1000.00	1.00
2	art002	Clavos	1.00	lib	1000.00	10000.00	1.00
3	art003	Martillo	60.00	u	200.00	1000.00	1.00
4	art004	Pala	40.00	u	150.00	5000.00	1.00
5	art005	Esmeril	300.00	u	5.00	50.00	1.00

Tabla 2.7 Artículos

	cli_ced	cli_nom	cli_dir	cli_cred	cod_ven
1	0102815831	Alex Campos	Lamar 2-11	1000.00	ven001
2	0102815832	Patricio Mena	Sucre 3-11	1500.00	ven002
3	0102815833	Alex Ubago	Tarqui 2-14	1600.00	ven003
4	0102815834	Andres Cepeda	Bolivar 3-12	1400.00	ven004
5	0102815835	Francisco Cespedes	Larga 3-11	2000.00	ven004

Tabla 2.8 Cliente

	ced_cli	tel_num
1	0102815831	282821211
2	0102815831	282821212
3	0102815832	282821213
4	0102815833	282821214
5	0102815834	282821215

Tabla 2.9 Teléfono

	ven_cod	ven_nom	ven_ape	ven_tel	ven_fec	ven_obj
1	ven001	Juan	Perez	2824021	2008-01-01	1000.00
2	ven002	Carlos	Santana	2824022	2007-01-01	2000.00
3	ven003	Alan	Brito	2824023	2007-01-01	1500.00
4	ven004	Gustavo	Cerati	2824024	2006-01-01	1600.00

Tabla 2.10 Vendedor

	cod_cab	cod_art	det_cant	det_sub_tot
1	cab001	art001	3.00	150.00
2	cab001	art002	5.00	5.00
3	cab002	art003	4.00	240.00
4	cab002	art004	3.00	120.00
5	cab0013	art005	6.00	1800.00

Tabla 2.11 Cabecera

	cab_cod	ced_cli	cod_ven	cab_fec	cab_iva	cab_iva_cal	cab_des	cab_des_cal	cab_tot
1	cab001	0102815831	ven001	2009-01-10	0.12	17.67	0.05	7.75	165.00
2	cab0013	0102815833	ven003	2009-03-10	0.12	205.20	0.05	90.00	1915.20
3	cab002	0102815832	ven002	2009-02-10	0.12	41.04	0.05	18.00	383.04

Tabla 2.12 Detalle

Creación de la Base de Datos Ferreteria:

```
CREATE DATABASE Ferreteria;
USE FERRETERIA;
```

Creación de Tablas:

Creación de la Tabla Artículos:

```
CREATE TABLE articulos(
        art_cod   varchar(10) NOT NULL,
        art_des  varchar(50) NULL,
        art_pre  numeric (10, 2) NULL,
        art_uni  varchar(50) NULL,
        art_stk  numeric(10, 2) NULL,
        art_max  numeric(10, 2) NULL,
        art_min  numeric(10, 2) NULL,
  PRIMARY KEY (art_cod));
```

Creación de la Tabla Cliente:

```
CREATE TABLE cliente(
        cli_ced  varchar(10) NOT NULL,
```

```
        cli_nom  varchar(50) NULL,
        cli_dir  varchar(50) NULL,
        cli_cred  numeric(10, 2) NULL,
        cod_ven varchar (10),
PRIMARY KEY (cli_ced));
```

Creación de la tabla Teléfono:

```
CREATE TABLE telefono(
        ced_cli varchar(10) NULL,
        tel_num varchar (10)NULL);
```

Creación de la Tabla Vendedor:

```
CREATE TABLE vendedor(
        ven_cod   varchar(10) NOT NULL,
        ven_nom  varchar(50) NULL,
        ven_ape  varchar(50) NULL,
        ven_tel  varchar(15) NULL,
        ven_fec  date NULL,
        ven_obj  numeric(10, 2) NULL,
PRIMARY KEY  (ven_cod));
```

Creación de la Tabla Cabecera:

```
CREATE TABLE cabecera(
        cab_cod varchar(20) NOT NULL,
        ced_cli varchar(10)NULL,
        cod_ven varchar(10)NULL,
        cab_fec date NULL,
        cab_iva numeric(10, 2) NULL,
        cab_iva_cal numeric(10, 2) NULL,
        cab_des numeric(10, 2) NULL,
        cab_des_cal numeric(10, 2) NULL,
        cab_tot numeric(10, 2) NULL,
PRIMARY KEY  (cab_cod));
```

Creación de la Tabla Detalle:

```
CREATE TABLE detalle(
        cod_cab varchar(20)NULL,
        cod_art varchar(10)NULL,
        det_cant numeric(10, 2) NULL,
        det_sub_tot numeric(10, 2) NULL
);
```

Creación de Llaves Foráneas.

Tabla Cliente:

```
ALTER TABLE cliente ADD FOREIGN KEY (cod_ven) REFERENCES vendedor (ven_cod);
```

Tabla Telefono:

```
ALTER TABLE telefono ADD FOREIGN KEY (ced_cli) REFERENCES cliente (cli_ced);
```

Tabla Cabecera:

```
ALTER TABLE cabecera ADD FOREIGN KEY (ced_cli) REFERENCES cliente (cli_ced);
ALTER TABLE cabecera ADD FOREIGN KEY (cod_ven) REFERENCES vendedor (ven_cod);
```

Tabla Detalle:

```
ALTER TABLE detalle ADD FOREIGN KEY (cod_cab) REFERENCES cabecera (cab_cod);
ALTER TABLE detalle ADD FOREIGN KEY (cod_art) REFERENCES articulos (art_cod);
```

Restricciones.

Restricción FK_cli_ven:

```
ALTER TABLE cliente
ADD CONSTRAINT FK_cli_ven
FOREIGN KEY (cod_ven)
REFERENCES vendedor(ven_cod)
ON DELETE no action
```

```
ON UPDATE no action;
```

Restriccion FK_tel_cli:

```
ALTER TABLE telefono
ADD CONSTRAINT FK_tel_cli
FOREIGN KEY (ced_cli)
REFERENCES cliente(cli_ced)
ON DELETE no action
ON UPDATE no action;
```

Restricción FK_cab_cli:

```
ALTER TABLE cabecera
ADD CONSTRAINT FK_cab_cli
FOREIGN KEY (ced_cli)
REFERENCES cliente(cli_ced)
ON DELETE no action
ON UPDATE no action;
```

Restricción FK_cab_ven:

```
ALTER TABLE cabecera
ADD CONSTRAINT FK_cab_ven
FOREIGN KEY (cod_ven)
REFERENCES vendedor(ven_cod)
ON DELETE no action
ON UPDATE no action;
```

Restricción FK_det_cab:

```
ALTER TABLE detalle
ADD CONSTRAINT FK_det_cab
FOREIGN KEY (cod_cab)
REFERENCES cabecera(cab_cod)
ON DELETE no action
```

```
ON UPDATE no action;
```

Restriccion FK_det_art:

```
ALTER TABLE detalle
ADD CONSTRAINT FK_det_art
FOREIGN KEY (cod_art)
REFERENCES articulo(art_cod)
ON DELETE no action
ON UPDATE no action;
```

Inserción de Datos.

Tabla Artículos:

```
insert into articulos (art_cod,art_des,art_pre,art_uni
                                                ,art_stk,art_max,art_min)
values ('art001','Cemento',50,'u',100,1000,1),
        ('art002','Clavos',1,'lib',1000,10000,1),
        ('art003','Martillo',60,'u',200,1000,1),
        ('art004','Pala',40,'u',150,5000,1),
        ('art005','Esmeril',300,'u',5,50,1);
```

Tabla Vendedor:

```
insert into vendedor (ven_cod,ven_nom,ven_ape,ven_tel,
                                            ven_fec,ven_obj)
values ('ven001','Juan','Perez','2824021','2008-01-01',1000),
                ('ven002','Carlos','Santana','2824022','2007-01-01',2000),
                ('ven003','Alan','Brito','2824023','2007-01-01',1500),
                ('ven004','Gustavo','Cerati','2824024','2006-01-01',1600);
```

Tabla Cliente:

```
insert into cliente (cli_ced,cli_nom,cli_dir,cli_cred,cod_ven)
values  ('0102815831','Alex Campos','Lamar 2-11',1000,'ven001'),
                ('0102815832','Patricio Mena','Sucre 3-11',1500,'ven002'),
                ('0102815833','Alex Ubago','Tarqui 2-14',1600,'ven003'),
                ('0102815834','Andres Cepeda','Bolivar 3-12',1400,'ven004'),
                ('0102815835','Francisco Cespedes','Larga 3-11',2000,'ven004');
```

Tabla Teléfono:

```
insert into telefono (ced_cli,tel_num)
values ('0102815831','282821211'),
                ('0102815831','282821212'),
                ('0102815832','282821213'),
                ('0102815833','282821214'),
                ('0102815834','282821215');
```

Tabla Cabecera:

```
insert into cabecera (cab_cod,ced_cli,cod_ven,cab_fec
                                        ,cab_iva,cab_iva_cal,cab_des,cab_des_cal,cab_tot)
values ('cab001','0102815831','ven001','2009-01-10',0.12, 17.67,0.05, 7.75, 165.00),
                ('cab002','0102815832','ven002','2009-02-10',0.12, 41.04,0.05, 18.00, 383.04),
                ('cab0013','0102815833','ven003','2009-03-10',0.12, 205.20,0.05, 90.00,
1915.20);
```

Tabla Detalle:

```
insert into detalle (cod_cab,cod_art,det_cant,det_sub_tot)
values ('cab001','art001',3,0),
                ('cab001','art002',5,0),
                ('cab002','art003',4,0),
                ('cab002','art004',3,0),
                ('cab0013','art005',6,0);
```

2.8 Conclusiones

En este capítulo se creó la base de datos “Compania” con sus respectivas tablas, de esta manera se ponen en práctica el lenguaje SQL, para finalizar se realizo un ejercicio al final del capítulo, la creación de la base de datos “Ferretería”.

CAPITULO 3

SEGURIDAD SQL SERVER

INTRODUCCION

La seguridad es muy importante para proteger la información, existe dos tipos de seguridades, físicas y lógicas. Este capítulo hace referencia a la seguridad lógica, protección de los datos, acceso al servidor y a las bases de datos, creación de Usuarios, Logins y asignación de privilegios.

3.1 Logins

Para acceder al sistema hay que tener activado un Login (Inicio de Sesión), SQL Server tiene dos maneras de validar los logins, utilizando la autentificación de Windows la cual usa el usuario y contraseña de inicio de sesión del sistema operativo (Windows) para ingresar a SQL Server, y utilizando la autentificación de SQL Server.

3.1.1 Creación de Logins

La siguiente sintaxis crea logins con autentificación de Windows:

CREATE LOGIN (nombre_servidor\nombre_login)
FROM WINDOWS
WITH DEFAULT_DATABASE = (nombre_base_de_datos)

A continuación se crea un Login de Nombre Carlos, se debe tener a consideración que “Carlos” es usuario de Windows, para crear el usuario se debe estar posicionado en la base de datos “master”.

USER master;

```
CREATE LOGIN "SERVIDOR\Carlos"
FROM WINDOWS
WITH DEFAULT_DATABASE = Compania;
```

Creación de logins con autentificación de SQL Server:

CREATE LOGIN (nombre_login)
WITH PASSWORD = ’password’,
DEFAULT_DATABASE = (nombre_base_de_datos),
CHECK_EXPIRATION = ON,
CHECK_POLICY = ON

Creamos un Login de nombre Adm con password ‘Adm12’:

```
USE master;
CREATE LOGIN Adm
WITH PASSWORD = 'Adm12',
DEFAULT_DATABASE = Compania ,
CHECK_EXPIRATION = ON,
CHECK_POLICY = ON;
```

3.1.2 Modificación de Logins

Para desbloquear un Login que ha sido bloqueado por una clave vencida se utiliza el siguiente contacto.

ALTER LOGIN (nombre_login) **WITH PASSWORD** = ’nuevo_password’ **UNLOCK**

3.1.3 Eliminación de Logins

Para eliminar logins utilizamos la siguiente instrucción

DROP LOGIN (nombre_login)

3.2 Usuarios

Los logins son usados para permitir acceso al sistema SQL Server. Sin embargo, acceso a bases de datos individuales se lo realiza creando usuarios en esas bases de datos.

3.2.1 Creación de Usuarios

Para crear usuarios usamos la siguiente sintaxis:

CREATE USER (nombre_de_usuario) **FOR LOGIN** (nombre_login)

En el siguiente ejemplo se crea un usuario que tiene acceso a la base de datos Compañía y el nombre del usuario es "Admdb" cuyo loguin es "Adm" creado anteriormente. Para crear el usuario hay que posicionarse en la base de datos compañía, utilizando la instrucción "USE" .

```
USE Compania;
CREATE USER Admdb FOR LOGIN Adm;
```

3.2.3 Borrar Usuarios

Para borrar usuarios utilizamos el comando Drop User,.

DROP USER (nombre_de_usuario)

3.3 Permisos

Los Permisos son las reglas que gobiernan el nivel de acceso que se tiene al servidor, a la base de datos. Los permisos pueden ser concedidos, revocados o denegados.

3.3.1 Permisos a nivel de Servidor

Los permisos a nivel de servidor permiten realizar acciones como administración completa del sistema, conexión con el servidor, creación y alteración de logins, etc.

Para otorgar permisos a nivel de servidor utilizamos la sentencia:

GRANT (nombre_permiso)
TO login (nombre_login)

A continuación en el cuadro se indica los permisos a nivel de servidor:

Permiso	Descripción
CONNECT_SQL	Conecta al servidor.
CREATE LOGIN	Crea un login.
ALTER ANY LOGIN	Altera cualquier login en el rango del servidor.
CONTROL SERVER	Control completo de administración del sistema.

Figura 3.1 Permisos a Nivel de Servidor

En el siguiente ejemplo damos permiso de administración de servidor al login Adm, para crear este permiso se debe estar seleccionada la a base de datos master:

```
USER master;
GRANT CONTROL SERVER to  Adm;
```

3.3.2 Permisos a nivel de Base de Datos

Los permisos a nivel de base de datos permiten realizar acciones como administración completa de la base de datos, creación de tablas, modificación de usuario de base de datos etc.

La siguiente sintaxis permite crear permisos a nivel de base de datos:

```
GRANT (nombre_permiso)
TO  (nombre_usuario)
```

A continuación en el cuadro se indica los permisos a nivel de base de datos:

Permiso	Descripción
CREATE TABLE	Crea una tabla en la base de datos.
ALTER ANY USER	Altera cualquier usuario en la base de datos.
CONTROL	Control completo de la base de datos.

Figura 3.2 Permisos a Nivel de Base de Datos

En el siguiente ejemplo se otorga un control total de la base de datos Compania al usuario Admdb, antes de crear el permiso se debe ubicar en la base de datos “Compania” usando el comando USE.

```
USE Compania;
GRANT CONTROL  TO Admdb;
```

3.3.3 Permisos a nivel de Objetos

Permite asignar permisos sobre los objetos de la base de datos, tablas vistas etc. La siguiente sintaxis permite crear permisos a nivel de objetos de base de datos:

GRANT (nombre_permiso)
ON Objeto
TO (nombre_usuario)

A continuación en el cuadro se indica los permisos a nivel de base de datos:

Permiso	Descripción
SELECT	Selecciona rows para cualquier objeto.
ALTER	Altera cualquier objeto.

Figura 3.3 Permisos a Nivel de Objetos

En el siguiente ejemplo se da permiso de lectura sobre la tabla departamento al usuario Admdb, antes de crear el permiso se debe ubicar en la base de datos “Compania” usando el comando USE.

```
USE Compania;
        GRANT SELECT
    ON departamento
        TO Admdb;
```

3.3.4 Revocar Permisos

Para anular los permisos dados se utiliza la siguiente sentencia:

REVOKE (nombre_permiso)**TO** (nombre_usuario/loguin);

Anular el permiso de administración de servidor al login Adm, para ejecutar esta sentencia se debe ejecutar primero la base de datos “master”.

```
USE  master;
REVOKE CONTROL SERVER TO Adm;
```

Anular el control total del usuario Admdb sobre la base de datos “Compania”, para ejecutar esta sentencia se debe ubicar en la base de datos Compania:

```
USE Compania;
REVOKE control TO Admdb;
```

Anular permisos a nivel de objetos se utiliza la siguiente sentencia:

REVOKE (nombre_permiso)
ON Objeto
TO (nombre_usuario);

En el siguiente ejemplo se anula el permiso de lectura sobre la tabla departamento al usuario Admdb, antes de crear el permiso se debe ubicar en la base de datos “Compania” usando el comando USE.

```
USE Compania;
    REVOKE SELECT
        ON departamento
    TO Admdb;
```

3.4 Conclusiones

SQL Server gestiona la seguridad en tres niveles o capas, a nivel de servidor, de base de datos y a nivel de objetos. A nivel de servidor, se regula quien tiene acceso al servidor, para acceder al servidor hay que tener un inicio de sesión (Login) y a este se le asignara los permisos sobre el servidor. El siguiente nivel de seguridad es el de base de datos, para que un Login pueda acceder a una base de datos, se tiene que crear un usuario (user) en dicha base de datos. Análogamente para que un usuario tenga acceso a los objetos que componen una base de datos hay que concederle permisos, esta es la ultima capa de seguridad.

CAPITULO 4

CONSULTAS SIMPLES

INTRODUCCION

Uno de los principales motivos por el cual se guarda información, es por que posteriormente la vamos a consultar, una de las principales razones por las cuales las bases de datos relacionales lograron gran aceptación es por la forma tan sencilla de lograr acceder a los datos. Como parte de estas facilidades para poder realizar consultas, encontramos a la sentencia SELECT.

4.1 Sentencia Select

La sentencia select permite recuperar una o varias filas de una o varias tablas, para indicar de donde se va a realizar la consulta se utiliza la opción From, la sintaxis es:

SELECT (nombre_columnas) **FROM** (nombre_tablas);

El carácter **"*"** permite obtener todas la información de una tabla.
A continuación se realiza una consulta de la tabla empleados.

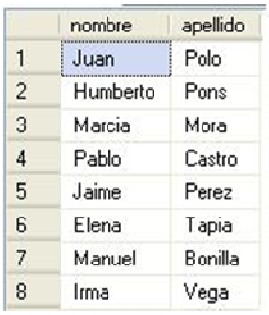
SELECT * FROM empleado

	nombre	apellido	ci	fecha_n	direccion	sexo	salario	superci	dno
1	Juan	Polo	123456789	1959-03-03	Sucre 7-12	M	3000	333445555	5
2	Humberto	Pons	333445555	1960-12-25	Bolivar 5-67	M	4000	888665555	5
3	Marcia	Mora	453453453	1960-03-29	Colombia 4-23	F	2500	333445555	5
4	Pablo	Castro	666884444	1955-09-15	Bolivar 1-50	M	3800	333445555	5
5	Jaime	Perez	888665555	1957-04-05	Sangurima 8-34	M	5500	NULL	1
6	Elena	Tapia	987654321	1961-05-03	Ordonez 7-29	F	4300	888665555	4
7	Manuel	Bonilla	987987987	1958-07-16	B. Malo 1-10	M	2500	987654321	4
8	Irma	Vega	999887777	1950-11-13	P. Cordova 3-45	F	2500	987654321	4

Figura 4.1 Consulta de la Tabla Empleado

La siguiente consulta se la realiza por columnas, obteniendo nombres y apellidos de los empleados:

SELECT nombre, apellido FROM empleado



	nombre	apellido
1	Juan	Polo
2	Humberto	Pons
3	Marcia	Mora
4	Pablo	Castro
5	Jaime	Perez
6	Elena	Tapia
7	Manuel	Bonilla
8	Irma	Vega

Figura 4.2 Consulta, Nombre y Apellido de la Tabla Empleado.

4.2 Concatenación de Datos

El operador de concatenación de cadenas es el signo más (+). Puede combinar, o concatenar, dos o más cadenas de caracteres en una única cadena. Para crear una columna temporal en la que se almacenara el resultado se utiliza la clausula As.

SELECT (nombre_columna + " separador" + nombre_columna) **AS** (columna_tem)
FROM (nombre_tabla)

En la siguiente columna concatenamos las columnas nombre y apellido de la tabla empleado.

SELECT nombre +' '+ apellido AS Empleado FROM empleado

	Empleado
1	Juan Polo
2	Humberto Pons
3	Marcia Mora
4	Pablo Castro
5	Jaime Perez
6	Elena Tapia
7	Manuel Bonilla
8	Irma Vega

Figura 4.3 Concatenación de columnas

4.3 Selección de Registros con Condiciones Específicas

La forma de especificar una condición dentro de una sentencia select es mediante la cláusula WHERE, que especifica una condición lógica que devolverá únicamente aquellos registros que la cumplan.

SELECT (nombre_columnas)
FROM (nombre_tablas)
WHERE (condición)

En la siguiente consulta, se seleccionan las filas de los empleados cuyo salario sea mayor a 3000.

```
SELECT * FROM empleado
WHERE salario > 3000
```

	nombre	apellido	ci	fecha_n	direccion	sexo	salario	superci	dno
1	Humberto	Pons	333445555	1960-12-25	Bolivar 5-67	M	4000	888665555	5
2	Pablo	Castro	666884444	1955-09-15	Bolivar 1-50	M	3800	333445555	5
3	Jaime	Perez	888665555	1957-04-05	Sangurima 8-34	M	5500	NULL	1
4	Elena	Tapia	987654321	1961-05-03	Ordonez 7-29	F	4300	888665555	4

Figura 4.4 Selección de Registro con Condición

Consultar Nombre, Apellido y Cedula de los empleados de sexo Masculino.

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

	nombre	apellido	ci
1	Juan	Polo	123456789
2	Humberto	Pons	333445555
3	Pablo	Castro	666884444
4	Jaime	Perez	888665555
5	Manuel	Bonilla	987987987

Figura 4.5 Selección de Empleados de sexo masculino.

La clausula where permite hacer consultas entre varias tablas, dependiendo de sus relaciones, en el ejemplo se consulta nombre y apellido de los jefes departamentales:

```
SELECT nombre,apellido, dnombre FROM  empleado , departamento
WHERE ci = jefeci
```

	nombre	apellido	dnombre
1	Humberto	Pons	Compras
2	Elena	Tapia	Administrativo
3	Jaime	Perez	Investigacion

Figura 4.6 Consulta de Jefes Departamentales

4.4 Eliminación de Filas Duplicadas

Elimina las filas duplicadas de una consulta, la clausula distinct se aplica a una columna.

```
SELECT DISTINCT (nombre_columna)
FROM (nombre_tabla);
```

En el siguiente ejemplo se utiliza la sentencia Distinct para listar las cédulas de los empleados que trabaja en un proyecto:

```
SELECT DISTINCT (eci) FROM trabaja_en
```

	eci
1	123456789
2	333445555
3	453453453
4	666884444
5	888665555
6	987654321
7	987987987
8	999887777

Figura 4.7 Consulta utilizando la clausula Distinct

	eci
1	123456789
2	123456789
3	666884444
4	453453453
5	453453453
6	333445555
7	333445555
8	333445555
9	333445555
10	999887777
11	999887777
12	987987987
13	987987987
14	987654321
15	987654321
16	888665555

Figura 4.8 Consulta sin la clausula Distinct

4.5 Consulta con Valores Nulos

"Null" significa "dato desconocido" o "valor inexistente". No es lo mismo que un valor "0", una cadena vacía.

A veces, puede desconocerse o no existir el dato correspondiente a algún campo de un registro. En estos casos decimos que el campo puede contener valores nulos.

SELECT (nombre_columnas) FROM (nombre_tablas)
WHERE (nombre_columna) IS NULL;

Por ejemplo en la tabla empleados podemos tener valores Null en la columna superci, esto indica que el empleado no tiene supervisor.

SELECT ci,nombre,apellido,superci FROM empleado

WHERE superci IS NULL;

	ci	nombre	apellido	superci
1	888665555	Jaime	Perez	NULL

Figura 4.9 Consulta usando la clausula Null

La siguiente sentencia selecciona valores no nulos:

SELECT (nombre_columnas) FROM (nombre_tablas)
WHERE (nombre_columna) IS NOT NULL;

En la siguiente consulta los datos de los empleados que tienen supervisor:

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

	ci	nombre	apellido	superci
1	123456789	Juan	Polo	333445555
2	333445555	Humberto	Pons	888665555
3	453453453	Marcia	Mora	333445555
4	666884444	Pablo	Castro	333445555
5	987654321	Elena	Tapia	888665555
6	987987987	Manuel	Bonilla	987654321
7	999887777	Irma	Vega	987654321

Figura 4.10 Consulta usando la clausula IS NOT NULL

4.6 Test de Correspondencia con Patrón

Para comparar porciones de cadenas utilizamos los operadores "like" y "not like". El símbolo "%" (porcentaje) reemplaza cualquier cantidad de caracteres (incluyendo ningún carácter). El carácter comodín "like" y "not like" son operadores de comparación que señalan igualdad o diferencia.

Así como "%" reemplaza cualquier cantidad de caracteres, el guión bajo "_" reemplaza un carácter.

En la siguiente consulta se lista los empleados cuyo apellido empieza con la letra “P”:

```
SELECT ci,nombre,apellido, direccion FROM empleado
WHERE apellido LIKE 'P%',
```

	ci	nombre	apellido	direccion
1	123456789	Juan	Polo	Sucre 7-12
2	333445555	Humberto	Pons	Bolivar 5-67
3	888665555	Jaime	Perez	Sanguiima 8-34

Figura 4.11 Consulta usando la Clausula LIKE y el comodín "%"

Para listar los empleados cuyo apellido no empiece con la letra "P", utilizamos la siguiente consulta:

```
SELECT ci,nombre,apellido, direccion FROM empleado
WHERE apellido NOT LIKE 'P%';
```

	ci	nombre	apellido	direccion
1	453453453	Marcia	Mora	Colombia 4-23
2	666884444	Pablo	Castro	Bolivar 1-50
3	987654321	Elena	Tapia	Ordonez 7-29
4	987987987	Manuel	Bonilla	B. Malo 1-10
5	999887777	Irma	Vega	P. Cordova 3-45

Figura 4.12 Consulta usando la clausula NOT LIKE y el comodín "%"

En la siguiente consulta se lista todos los empleados cuyo apellido tenga 4 caracteres:

```
SELECT ci,nombre,apellido, direccion FROM empleado
WHERE apellido  LIKE '____';
```

	ci	nombre	apellido	direccion
1	123456789	Juan	Polo	Sucre 7-12
2	333445555	Humberto	Pons	Bolivar 5-67
3	453453453	Marcia	Mora	Colombia 4-23
4	999887777	Irma	Vega	P. Cordova 3-45

Figura 4.13 Consulta usando la clausula LIKE y el comodín "_"

4.7 Consultas con Rango de Fechas

Microsoft SQL Server ofrece algunas funciones para trabajar con fechas y horas, en la siguiente consulta se obtiene todos los empleados nacidos desde 1959:

```
SELECT ci,nombre,apellido,fecha_n
FROM empleado
WHERE YEAR(fecha_n)>=1959;
```

	ci	nombre	apellido	fecha_n
1	123456789	Juan	Polo	1959-03-03
2	333445555	Humberto	Pons	1960-12-25
3	453453453	Marcia	Mora	1960-03-29
4	987654321	Elena	Tapia	1961-05-03

Figura 4.14 Consulta por Fecha utilizando la clausula YEAR

getdate(): Retorna la fecha y hora actuales. Sintaxis:

```
select getdate();
```

datepart(partedefecha,fecha): Retorna la parte específica de una fecha, el año, trimestre, día, hora, etc. Los valores para "partedefecha" pueden ser: year (año), quarter (cuarto), month (mes), day (dia), week (semana), hour (hora), minute (minuto), second (segundo) y millisecond (milisegundo).

```
datepart(partedefecha,fecha);
```

En el siguiente ejemplo retorna el número de mes acutual:

```
select datepart(month,getdate());
```

day(fecha): retorna el día de la fecha especificada. Ejemplo:

```
select day(getdate());
```

month(fecha): retorna el mes de la fecha especificada. Ejemplo:

```
select month(getdate());
```

year(fecha): retorna el año de la fecha especificada. Ejemplo:

```
select year(getdate());
```

4.8 Consultas Usando alias

Cuando dos columnas de diferentes tablas tienen el mismo nombre se utiliza un alias para facilitar la consulta, la sintaxis es:

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

En la siguiente consulta se lista todos los departamentos ubicados en Quito:

```
SELECT departamento.dnumero, dnombre
FROM departamento, localizacion
WHERE departamento.dnumero = localizacion.dnumero
AND dep_loca = 'Quito';
```

4.9 Consultas Renombrando Tablas

Cuando se utilizan los mismos campos de una tabla para realizar una consulta se debe renombrar la tabla, también para evitar escribir todo el nombre de la tabla, la sintaxis es:

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

Consultar el nombre y apellido de los empleados y su respectivo supervisor:

```
SELECT e.ci,e.nombre ,e.apellido , s.ci as ci_supervisor,s.nombre as
nombre_supervisor,s.apellido as apellido_supervisor
FROM empleado e, empleado s
WHERE  e.superci = s.ci
```

	ci	nombre	apellido	ci_supervisor	nombre_supervisor	apellido_supervisor
1	123456789	Juan	Polo	333445555	Humberto	Pons
2	333445555	Humberto	Pons	888665555	Jaime	Perez
3	453453453	Marcia	Mora	333445555	Humberto	Pons
4	666884444	Pablo	Castro	333445555	Humberto	Pons
5	987654321	Elena	Tapia	888665555	Jaime	Perez
6	987987987	Manuel	Bonilla	987654321	Elena	Tapia
7	999887777	Irma	Vega	987654321	Elena	Tapia

Figura 4.15 Consulta Renombrando Tablas

4.10 Conclusiones

En los ejemplos realizados en este capítulo, se puede apreciar que con simples consultas se puede acceder a los datos, las facilidades que el lenguaje SQL brinda, de acceso a los datos, ha permitido que se extienda y sea usado por los de gestores de datos.

CAPITULO 5

ATRIBUTOS DE COLUMNA

INTRODUCCION

El presente capítulo aborda temas de tratamiento y manipulación de la información, nos indica cómo realizar cálculos, ordenar y agrupar los datos bajo ciertos criterios, al final de este capítulo se plantea un ejercicio en el cual se pone en práctica todos los conocimientos adquiridos.

5.1 Funciones de Columna

Existen en SQL Server funciones que nos permiten contar registros, calcular sumas, promedios, obtener valores máximos y mínimos. Estas funciones se denominan funciones de columna y operan sobre un conjunto de valores (columna) y devuelven un único valor que resume la columna.

Contar el número de proyectos que se desarrollan en la empresa:

```
SELECT  COUNT(pnumero ) AS proyectos
FROM proyecto;
```

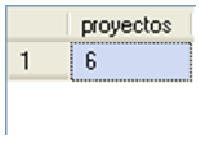
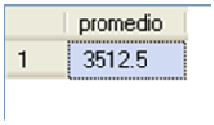


Figura 5.1 Función Count

Calcular el Salario promedio de los empleados de la empresa:

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



	promedio
1	3512.5

Figura 5.2 Función Avg

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

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

	horas
1	50.00

Figura 5.3 Función Suma

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

```
SELECT MAX (salario)AS Maximo,
          MIN (salario) AS Minimo
FROM empleado;
```

	Maximo	Minimo
1	5500	2500

Figura 5.4 Función Max y Min

5.2 Ordenamiento de los Resultados consulta (ORDER BY)

Para ordenar por algún campo el resultado de una consulta utilizamos la sentencia “Order by”, esta sentencia permite mostrar el resultado en orden ascendente o descendente.

Listar todos los empleados que trabajan en la compañía en orden alfabético descendente por apellido:

SELECT apellido, nombre, salario
FROM empleado
ORDER BY apellido **DESC**;

	apellido	nombre	salario
1	Vega	Irma	2500
2	Tapia	Elena	4300
3	Pons	Humberto	4000
4	Polo	Juan	3000
5	Perez	Jaime	5500
6	Mora	Marcia	2500
7	Castro	Pablo	3800
8	Bonilla	Manuel	2500

Figura 5.5 Ordenación de registros descendente

Cuando no se indica el tipo de ordenamiento, automáticamente el resultado será en ordenado de manera ascendente.

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

	apellido	nombre	salario
1	Bonilla	Manuel	2500
2	Castro	Pablo	3800
3	Mora	Marcia	2500
4	Perez	Jaime	5500
5	Polo	Juan	3000
6	Pons	Humberto	4000
7	Tapia	Elena	4300
8	Vega	Irma	2500

Figura 5.6 Ordenación de registros ascendente

5.3 Consultas Agrupadas (GROUP BY)

Para organizar registros en grupos y obtener un resumen de dichos grupos se utiliza “group by”, esta clausula se utiliza en conjunto con funciones de columna, de esta manera se puede realizar cálculos y agruparlos.

Listar el total de horas trabajadas por cada uno de los empleados.

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

	eci	nombre	apellido	horas
1	123456789	Juan	Polo	28.10
2	333445555	Humberto	Pons	50.00
3	453453453	Marcia	Mora	20.00
4	666884444	Pablo	Castro	14.70
5	888665555	Jaime	Perez	NULL
6	987654321	Elena	Tapia	22.00
7	987987987	Manuel	Bonilla	32.00
8	999887777	Irma	Vega	35.00

Figura 5.7 Agrupación de registros

5.4 Condiciones de Búsqueda en Grupos (Having)

La clausula “having”permite condicionar la búsqueda de los agrupamientos definidos por “group by”, la diferencia entre where y heaving radica en que "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros.

Listar los empleados que tienen más de dos cargas familiares:

```
SELECT   nombre,apellido, count (eci)as cargas
FROM empleado a , carga_f
WHERE ci=eci
GROUP BY nombre, apellido
HAVING count (eci)>2 ;
```

	nombre	apellido	cargas
1	Juan	Polo	3
2	Humberto	Pons	3

Figura 5.8 Condiciones de Búsqueda en grupos

5.5 Ejercicios de Consultas Simples de la Base de Datos Compañía

1. Listar el nombre y el número de todos los departamentos.

```
SELECT dnombre, dnumero
FROM departamento;
```

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

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

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

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

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

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

5. Contar las cargas familiares del empleado Juan Polo.

```
SELECT nombre, apellido, COUNT(eci) As cargas
FROM carga_f, empleado
WHERE eci=ci AND
nombre= 'Humberto' AND
apellido= 'Pons'
GROUP BY nombre, apellido;
```

6. Listar todos los empleados que tengan un salario entre 2000 y 4000 dólares.

```
SELECT nombre, apellido
FROM empleado
WHERE  salario BETWEEN  2000 AND 4000;
```

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, nombre, apellido;
```

8. Calcular el promedio de horas trabajadas por Juan Polo.

```
SELECT nombre, apellido, AVG(horas)
```

```
FROM empleado, trabaja_en
WHERE ci=eci AND
nombre='Juan' AND
apellido='Polo'
GROUP BY  nombre, apellido;
```

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

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

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

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

11. Listar todos los proyectos localizados en Cuenca.

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

12. Listar todos los empleados que tengan cargas familiares cuyo nombre comience con la letra M.

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

13. Listar todos los proyectos que pertenecen al departamento ubicado en Guayaquil.

```
SELECT pnombre, dep_loca
FROM proyecto, localizacion
```

```
WHERE dnumero=dnum AND
dep_loca='Guayaquil';
```

14. 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;
```

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

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

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

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

17. 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 ci = 999887777 AND
dno = dnumero AND
dnum = dnumero;
```

18. 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);
```

19. Listar el nombre, apellido y el departamento en donde trabajan los empleados.

```
SELECT  nombre, apellido, dnombre
FROM empleado, departamento
WHERE  dno=dnumero;
```

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

```
SELECT SUM(salario)as Suma,  AVG(salario) as Promedio,
            MAX(salario)as Maximo, MIN(salario)as Minimo
FROM empleado;
```

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

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

22. Calcular el salario máximo y mínimo de cada departamento.

```
SELECT dnombre, MAX(salario) As maximo , MIN(salario) As minimo
FROM empleado, departamento
WHERE dno=dnumero
GROUP BY dnombre;
```

23. Calcular el promedio de los salarios por departamento.

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

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

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

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

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

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

```
SELECT a.nombre,a.apellido,b.nombre as nom_sup, b.apellido as ape_sup
 FROM empleado a, empleado b
 where a.superci = b.ci;
```

27. Listar los supervisores que tienen más de dos cargas familiares.

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

28. Contar cuantos empleados existen en cada departamento.

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

29. Listar el departamento cuyo jefe es Humberto Pons.

```
SELECT nombre, apellido, dnombre
FROM empleado, departamento
WHERE nombre= 'Humberto' AND
apellido='Pons' AND
```

```
dno= dnumero;
```

30. Listar todos los proyectos localizados en Guayaquil.

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

5.6 Ejercicios de Consultas Simples de la Base de Datos Ferretería

1. Listar el número, el nombre del cliente, total y la fecha de las facturas emitidas.

```
SELECT cab_cod,cli_nom, cab_tot, cab_fec
FROM cabecera, cliente
WHERE          cli_ced= ced_cli;
```

2. Listar el vendedor y la fecha en la que se emitió la factura cab0013.

```
SELECT ven_nom, ven_ape, cab_fec
FROM cabecera, vendedor
WHERE cab_cod ='cab0013'
and cod_ven = ven_cod;
```

3. Listar los clientes que tengan más de una factura a su nombre.

```
SELECT ced_cli, cli_nom, COUNT(ced_cli) as numero
FROM cabecera, cliente
WHERE ced_cli= cli_ced
GROUP BY ced_cli, cli_nom
HAVING COUNT(ced_cli)>1;
```

4. Listar la factura con mayor valor por cliente.

```
SELECT  cli_nom,MAX(cab_tot) as total
FROM cabecera, cliente
WHERE cli_ced = ced_cli
GROUP BY cli_nom;
```

5. Listar el promedio de las facturas por cliente.

```
SELECT  cli_nom, AVG(cab_tot) as promedio
FROM cabecera, cliente
WHERE cli_ced = ced_cli
GROUP BY cli_nom;
```

6. Listar el stock, el stock máximo, stock mínimo de los artículos cuyo nombre empiece con “C”.

```
SELECT art_des, art_stk, art_max, art_min
FROM articulos
WHERE  art_des LIKE 'C%';
```

7. Listar los teléfonos de los clientes cuyo nombre empiece con la letra “A”;

```
SELECT cli_ced, cli_nom, tel_num
FROM cliente, telefono
WHERE cli_ced= ced_cli
AND cli_nom LIKE 'A%';
```

8. Listar el nombre de los clientes ordenados alfabéticamente con su respectivo vendedor asignado.

```
SELECT cli_nom, ven_nom
FROM cliente, vendedor
WHERE cod_ven= ven_cod
ORDER BY cli_nom;
```

9. Listar la suma, el promedio, el valor máximo y el valor mínimo del total de todas las facturas.

```
SELECT SUM(cab_tot)AS SUMA
                , AVG(cab_tot) AS PROMEDIO
                , MAX(cab_tot)AS MAXIMO
                ,MIN(cab_tot) AS MINIMO
FROM cabecera;
```

10. Listar las facturas cuyo total haya sobrepasado los 1000 con su respectivo cliente.

```
SELECT cab_cod, cli_nom, cab_tot
FROM cabecera, cliente
WHERE cab_tot>1000
AND cli_ced= ced_cli;
```

11. Listar todos los artículos en orden descendente, cuyo precio sea más de $20

```
SELECT art_des, art_pre
FROM articulos
WHERE art_pre > 20
ORDER BY art_des DESC
```

12. Listar el número, el nombre y apellido del vendedor, la fecha y el total de las facturas, que fueron emitidas en el mes de febrero del 2009.

```
SELECT c.cab_cod, v.ven_nom, v.ven_nom, c.cab_fec, cab_tot
FROM cabecera c, vendedor v
WHERE YEAR(c.cab_fec)= 2009
AND MONTH(c.cab_fec)= 3
AND c.cod_ven= v.ven_cod;
```

13. Listar el número, el nombre del cliente, la fecha y el total, de las facturas cuyo nombre de cliente termine con la letra “s”.

```
SELECT cab_cod, cli_nom, cab_fec, cab_tot
FROM cabecera , cliente
WHERE cli_nom LIKE '%s'
AND cli_ced = ced_cli;
```

14. Listar el número, el nombre y apellido del vendedor y el total, de las facturas cuyo total este entre 1000 y 5000.

```
SELECT cab_cod, cab_fec, cab_tot,
ven_nom, ven_ape
FROM cabecera, vendedor
WHERE  cab_tot BETWEEN 1000
```

```
AND  5000
AND ven_cod = cod_ven;
```

15. Listar el total de compras por cliente.

```
SELECT cl.cli_ced,cl.cli_nom,SUM(c.cab_tot)TOTAL
FROM cabecera c , cliente cl
WHERE c.ced_cli= cl.cli_ced
GROUP BY cl.cli_ced, cl.cli_nom;
```

16. Listar el número, el total, la fecha y el vendedor de las facturas emitidas el 10 de Enero del 2009.

```
SELECT c.cab_cod, c.cab_tot, c.cab_fec, v.ven_nom, v.ven_ape
FROM cabecera c, vendedor v
WHERE YEAR(c.cab_fec)= 2009 AND
MONTH(c.cab_fec)= 1 AND
DAY (c.cab_fec)= 10 AND
c.cod_ven= v.ven_cod;
```

17. Listar el nombre, apellido y teléfono de los vendedores que fueron contratados el 1 de Enero del 2007.

```
SELECT v.ven_nom, v.ven_ape, v.ven_tel
FROM vendedor v
WHERE YEAR(v.ven_fec)= 2007 AND
MONTH (v.ven_fec)= 1 AND
DAY (v.ven_fec)= 1;
```

18. Listar el nombre, apellido y teléfono de los vendedores cuyo objetivo en ventas sea mayor a 1000.

```
SELECT v.ven_nom, v.ven_ape, v.ven_tel, v.ven_obj
FROM vendedor  v
WHERE v.ven_obj >1000;
```

5.7 Conclusiones

En este capítulo se indico como realizar cálculos con la información, las funciones de columna pueden calcular el promedio, la suma, el valor máximo y mínimo de una columna, contar el número de valores de datos de una columna; también se realizo consultas agrupadas bajo ciertas condiciones.

CAPITULO 6

SUBCONSULTAS Y SUBCONSULTAS ANIDADAS

INTRODUCCION

Las subconsultas se emplean cuando una consulta es muy compleja, entonces se la divide en varios pasos lógicos y cuando la consulta depende de los resultados de otra consulta. En el presente capítulo se muestra la utilidad de las subconsultas y al final del capítulo se presenta una serie de ejercicios prácticos.

6.1 Subconsultas

Las subconsultas son sentencias select dentro de otra sentencia select, la subconsulta se encuentra dentro de la clausula where o having de otra consulta superior. Las subconsultas se deben incluir entre paréntesis.

Hay tres tipos básicos de subconsultas: Consultas que retornan un solo valor escalar que se utiliza con un operador de comparación o en lugar de una expresión. Consultas que retornan una lista de valores, que se combinan con "in", o los operadores "any", "some" y "all". Consultas que testean la existencia con "exists".

La extructura de una subconsulta es:

SELECT nombres_columnas
FROM nombres_tablas
WHERE <expresión><condición>**(SELECT** nombres_columnas
FROM nombres_tablas
WHERE condición**);**

Reglas a tener en cuenta al emplear subconsultas:

- La lista de selección de una subconsulta que va luego de un operador de comparación puede incluir sólo una expresión o campo (excepto si se emplea "exists").

- Si el "where" de la consulta exterior incluye un campo, este debe ser compatible con el campo en la lista de selección de la subconsulta.

- Las subconsultas luego de un operador de comparación (que no es seguido por "any" o "all") no pueden incluir cláusulas "group by" ni "having".

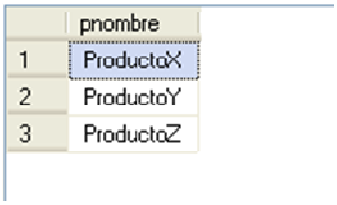
- "Distinct" no puede usarse con subconsultas que incluyan "group by".

- Una subconsulta puede estar anidada dentro del "where" o "having" de una consulta externa o dentro de otra subconsulta.

- Si una tabla se nombra solamente en un subconsulta y no en la consulta externa, los campos no serán incluidos en la salida (en la lista de selección de la consulta externa).

Listar los nombres de los proyectos controlados por el departamento de Investigación:

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



	pnombre
1	ProductoX
2	ProductoY
3	ProductoZ

Figura 6.1 Subconsulta, listado de los proyectos controlados por el departamento de Investigación

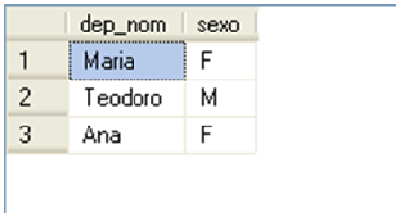
6.2 Condiciones de Búsqueda en las Subconsultas

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

Este test compara el valor de una expresión con un valor resultado de una subconsulta, si la condición se cumple devuelve un valor de true, si la subconsulta no devuelve valor el resultado del test de comparación devuelve NULL.

Listar los nombres y el sexo de las cargas familiares del empleado Humberto Pons:

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



	dep_nom	sexo
1	Maria	F
2	Teodoro	M
3	Ana	F

Figura 6.2 Subconsulta, cargas familiares del empleado Humberto Pons

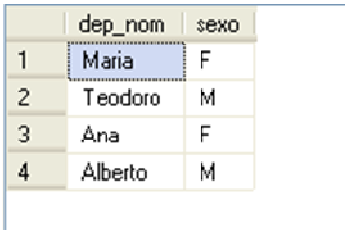
6.2.2 Test de inclusión (IN)

Cuando una subconsulta devuelve una lista de valores de un solo campo (columna), se la puede comparar con un único valor utilizando el test de inclusión IN, si el único valor coincide con algunos de los valores de la columna, la subconsulta retorna un valor de TRUE.

Listar el nombre y el sexo de las cargas familiares de todos los empleados que ganen más de $3000.

```
SELECT dep_nom, sexo
FROM  carga_f
WHERE eci  IN(SELECT ci
```

```
FROM empleado
WHERE salario > 3000);
```



	dep_nom	sexo
1	Maria	F
2	Teodoro	M
3	Ana	F
4	Alberto	M

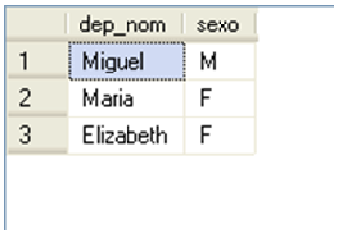
Figura 6.3 Subconsulta, cargas Familiares de

Empleados que ganan más de $3000

También se puede buscar valores No coincidentes con una lista de valores que retorna una subconsulta utilizando NOT IN

Listar el nombre y el sexo de las cargas familiares de todos los empleados que ganen menos de $3000.

```
SELECT dep_nom, sexo
FROM  carga_f
WHERE eci NOT IN (SELECT ci
        FROM empleado
        WHERE salario > 3000);
```



	dep_nom	sexo
1	Miguel	M
2	Maria	F
3	Elizabeth	F

Figura 6.4 Cargas Familiares de

Empleados que ganan menos de $3000

6.2.3 Test de Existencia (EXISTS)

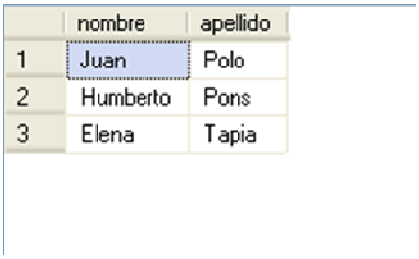
Los operadores "exists" y "not exists" se emplean para determinar si hay o no datos en una lista de valores.

Estos operadores pueden emplearse con subconsultas correlacionadas para restringir el resultado de una consulta exterior a los registros que cumplen la subconsulta (consulta interior).

Estos operadores retornan "true" (si las subconsultas retornan registros) o "false" (si las subconsultas no retornan registros).

Listar el nombre de los empleados que tienen cargas familiares:

```
SELECT nombre, apellido
FROM empleado
WHERE EXISTS (SELECT *
              FROM carga_f
              WHERE ci=eci);
```

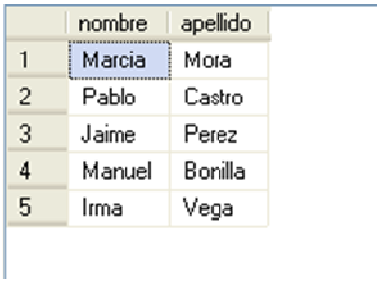


	nombre	apellido
1	Juan	Polo
2	Humberto	Pons
3	Elena	Tapia

Figura 6.5 Subconsulta, empleados que tienen cargas familiares

Listar el nombre de los empleados que no tienen cargas familiares:

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



	nombre	apellido
1	Marcia	Mora
2	Pablo	Castro
3	Jaime	Perez
4	Manuel	Bonilla
5	Irma	Vega

Figura 6.6 Subconsulta, Empleados que no tienen cargas familiares

6.2.4 Test Cuantificados

6.2.4.1 Test ANY

Revisa si alguna fila de la lista resultado de una subconsulta se encuentra el valor especificado en la condición. Se utiliza conjuntamente con uno de los seis operadores de comparación SQL (=, <>, <, <=, >, >=).

Compara un valor escalar con los valores de un campo y devuelven "true" si la comparación con algún valor de la lista de la subconsulta es verdadera, sino "false". El tipo de datos que se comparan deben ser compatibles.

Listar el nombre de los empleados que trabajan en el proyecto 10:

```
SELECT nombre, apellido
FROM empleado
WHERE ci = ANY (SELECT eci
        FROM trabaja_en
        WHERE pno= 10);
```

	nombre	apellido
1	Humberto	Pons
2	Manuel	Bonilla
3	Irma	Vega

Figura 6.7 Subconsulta, empleados que Trabajan en el proyecto 10

6.2.4.2 Test ALL

Recupera los registros de la consulta principal que cumplan con la comparación con todas las filas obtenidas en la subconsulta. Los operadores de comparación que se utilizan conjuntamente con el testa ALL son (=, <>, <, >, <=, >=).

Si individualmente todas las comparaciones se cumplen, el test ALL retorna un resultado TRUE.

Listar el nombre de los empleados que por cada proyecto han trabajado más horas, que las trabajadas por cada empleado en el proyecto 10.

```
SELECT nombre, apellido
FROM empleado, trabaja_en
WHERE ci=eci AND
HORAS > ALL(SELECT horas
                    FROM trabaja_en
                    WHERE pno=10);
```

	nombre	apellido
1	Juan	Polo
2	Humberto	Pons
3	Irma	Vega
4	Manuel	Bonilla

Figura 6.8 Subconsulta, empleados que trabajan más horas que los empleados que trabajan el proyecto 10

6.3 Subconsultas Anidadas

Cuando una subconsulta está dentro de otra subconsulta (más de dos niveles) se le denomina subconsulta anidada.

Listar los empleados que trabajan en el proyecto computadora:

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

	nombre	apellido
1	Humberto	Pons
2	Manuel	Bonilla
3	Irma	Vega

Figura 6.9 Subconsulta, empleados que trabajan

en el proyecto computadora

6.4 Ejercicios de Subconsultas de la base de datos Compañía

1. Listar los nombres de los proyectos controlados por el departamento de Investigación:

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

2. Listar los nombres y el sexo de las cargas familiares del empleado Juan Polo.

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

3. Listar el nombre de todos los empleados que trabajan en el proyecto ProductoY.

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

4. Listar el nombre de los empleados que trabajan en el departamento de investigación.

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

5. Listar el nombre de los empleados que trabajan en dos o más proyectos.

```
SELECT nombre, apellido
FROM empleado
WHERE EXISTS(
        SELECT COUNT(eci),eci
        FROM trabaja_en
        WHERE ci=eci
        GROUP BY eci
        HAVING COUNT(eci)>=2);
```

6. Encontrar el departamento que tiene el mayor salario promedio.

```
SELECT dnombre
FROM empleado, departamento
```

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

7. Listar el nombre y apellido del empleado que tiene únicamente cargas familiares de sexo masculino.

```
SELECT nombre, apellido
FROM empleado
WHERE ci IN (
        SELECT eci
        FROM carga_f
        WHERE sexo = 'M');
```

8. Listar los departamentos en donde todos los empleados de cada departamento ganen más de $3000.

```
SELECT dnombre
FROM departamento
WHERE dnumero IN (
        SELECT dno
        FROM empleado
        GROUP BY dno
        HAVING MIN (salario) > 3000);
```

9. Listar los nombres y apellidos de los jefes departamentales.

```
SELECT nombre, apellido
FROM empleado
WHERE ci IN (
        SELECT jefeci
        FROM departamento) ;
```

10. Listar el nombre del empleado que tiene más cargas familiares

```
SELECT nombre, apellido
FROM empleado, carga_f
WHERE eci=ci
GROUP BY  ci, nombre, apellido
HAVING COUNT (ci) >= ALL (
        SELECT  COUNT (eci)
        FROM carga_f
        GROUP BY eci);
```

6.5 Ejercicios de Subconsultas de la base de datos Ferretería

1. Listar los clientes que tiene asignado el vendedor Juan Perez.

```
SELECT cli_nom
FROM cliente
WHERE cod_ven IN (
        SELECT ven_cod
        FROM vendedor
        WHERE ven_nom = 'Juan' AND
        ven_ape = 'Perez');
```

2. Listar el número, la fecha y el total, de las facturas del cliente Patricio Mena.

```
SELECT cab_cod, cab_fec, cab_tot
FROM cabecera
WHERE ced_cli = (
        SELECT cli_ced
        FROM cliente
        WHERE cli_nom = 'Patricio Mena');
```

3. Listar el nombre del vendedor que no tenga asignado Clientes.

```
SELECT *
FROM vendedor
WHERE ven_cod NOT IN  (
        SELECT cod_ven
        FROM cliente);
```

4. Sumar el total de todas las facturas del cliente Patricio Mena.

```
SELECT SUM (cab_tot)
FROM cabecera
WHERE ced_cli =(SELECT cli_ced FROM cliente
                    WHERE cli_nom = 'Patricio Mena');
```

5. Sumar el total de todas las facturas por cliente, que haya emitido el vendedor Juan Pérez.

```
SELECT SUM (cab_tot)
FROM cabecera
GROUP BY cod_ven
HAVING cod_ven= (
        SELECT ven_cod
        FROM vendedor
        WHERE ven_nom = 'Juan' AND
        ven_ape = 'Perez');
```

6. Listar los vendedores que tienen asignados más de un vendedor.

```
SELECT ven_nom, ven_ape
FROM vendedor
WHERE EXISTS (
        SELECT cod_ven, COUNT (cod_ven)
        FROM cliente
        WHERE cod_ven = ven_cod
        GROUP BY cod_ven
        HAVING COUNT (cod_ven)>= 2);
```

7. Listar el nombre del cliente que mas facturas tiene a su nombre.

```
SELECT cli_nom
```

```
FROM cliente, cabecera
WHERE cli_ced = ced_cli
GROUP BY cli_nom
HAVING COUNT(cli_ced) >= ALL (
        SELECT COUNT (ced_cli)
        FROM cabecera
        GROUP BY ced_cli);
```

8. Listar el nombre de los clientes que realizaron compras entre enero y febrero del 2009.

```
SELECT cli_nom
FROM cliente
WHERE cli_ced IN (
        SELECT ced_cli
        FROM cabecera
        WHERE MONTH (cab_fec) BETWEEN 01 AND 02);
```

9. Listar el nombre del vendedor que haya emitido facturas de más $ 1000

```
SELECT *
FROM vendedor
WHERE ven_cod IN
        (SELECT cod_ven
        FROM cabecera
        WHERE cab_tot > 1000);
```

10. Listar el nombre y apellido del vendedor que más facturas ha emitido.

```
SELECT ven_nom, ven_ape
FROM vendedor, cabecera
WHERE ven_cod = cod_ven
GROUP BY ven_nom, ven_ape
HAVING COUNT(ven_cod) >= ALL (
        SELECT COUNT (cod_ven)
        FROM cabecera
        GROUP BY cod_ven);
```

6.6 Conclusiones

Hay que tener en consideración que el gestor de base de datos no soporta anidamientos de funciones “max(avg(….))”, debido a esta falencia se buscaron otras alternativas al realizar las consultas, un ejemplo es el ejercicio 6 del punto “6.4 Ejercicios de Subconsultas de la base de datos Compañía”.

CAPITULO 7

PROCEDIMIENTOS ALMACENADOS Y TRIGGERS

INTRODUCCION

Los procedimientos almacenados son un conjunto de sentencias que se almacenan en el servidor, realizan una tarea específica por ejemplo, recuperar los datos de una tabla, realizar cálculos etc. Los procedimientos pueden ser llamados las veces que sean necesarios.

Los triggers conocidos también como disparadores, son una especie de procedimientos, que se ejecutan cuando ocurre un evento (inserción, borrado, actualización) sobre alguna tabla.

7.1 Procedimientos almacenados

7.1.1 Creación de Procedimientos almacenados

Un procedimiento almacenado es un conjunto de instrucciones a las que se les da un nombre,. Permiten encapsular tareas repetitivas.

La sintaxis básica es:

```
CREATE PROCEDURE nombre_procedimiento
AS instrucciones;
```

Crear un procedimiento que liste el nombre, apellido y el salario de los empleados de la base de datos Compañía.

```
USE  Compania;
CREATE PROCEDURE pa_empleados
```

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

Con la siguiente sentencia ejecutamos el procedimiento almacenado.

```
EXEC pc_empleados;
```

	nombre	apellido	salario
1	Juan	Polo	3000
2	Humberto	Pons	4000
3	Marcia	Mora	2500
4	Pablo	Castro	3800
5	Jaime	Perez	5500
6	Elena	Tapia	4300
7	Manuel	Bonilla	2500
8	Irma	Vega	2500

Figura 7.1 Listado de los empleados de la base de datos Compañía.

7.1.2 Eliminación de Procedimientos Almacenados

Los procedimientos almacenados se eliminan con la sentencia “drop procedure”, la sintaxis es la siguiente:

```
DROP PROCEDURE nombre_procedimiento;
```

Eliminar el procedimiento pa_empleados:

```
DROP PROCEDURE pa_empleados;
```

7.1.3 Procedimientos (Parámetros de Entrada)

Los procedimientos pueden recibir información a través de los parámetros de entrada, para que un procedimiento admita parámetros de entrada, se debe crear variables como parámetros, la sintaxis es:

```
CREATE PROCEDURE nombre_procedimiento
@nombre_parametro tipo [=valor_por_defecto]
AS sentencias;
```

Los parámetros se definen luego del nombre del procedimiento, el nombre del parámetro comienza con el signo de "@", los parámetros existen solo dentro del procedimiento, cuando se declaran varios parámetros estos deben ser separados por comas. Al llamar un procedimiento los valores de los parámetros deben ir en el mismo orden en que se declaro los parámetros.

Crear un procedimiento que liste el nombre, apellido y el salario de los empleados de la base de datos Compañía que trabajen en el departamento de Investigación.

```
CREATE PROCEDURE pa_empleados @nombre varchar(50)
AS
SELECT nombre, apellido
FROM empleado, departamento
WHERE dnombre = @nombre
AND dnumero = dno;
```

Para ejecutar el procedimiento utilizamos la sentencia "exec", seguido del nombre del procedimiento y un valor para el parámetro.

```
EXEC pa_empleados 'Investigacion';
```

	nombre	apellido
1	Juan	Polo
2	Humberto	Pons
3	Marcia	Mora
4	Pablo	Castro

Figura 7.2 Empleados que trabajan en el departamento de Investigación

7.1.4 Procedimientos (Parámetros de Salida)

Para que el procedimiento devuelva información se debe declarar parámetros de salida (output), la sintaxis es:

```
CREATE PROCEDURE nombre_procedimiento
@parametro_entrada  tipo,
@parametro_salida      tipo  output
AS
Sentencias;
```

Crear un procedimiento que sume el salario de todos los empleados que trabajan en el departamento de investigación.

```
CREATE PROCEDURE pa_empleados_suma
@nombre varchar(50),
@suma numeric (10) output
AS
SELECT  @suma = SUM (salario)
FROM empleado, departamento
WHERE dnombre = @nombre
AND dnumero = dno;
```

Para ejecutar el procedimiento se procede de la siguiente manera:

```
DECLARE @sum numeric (10);
EXEC pa_empleados_suma 'Investigacion',@sum output;
SELECT @sum as suma;
```

Declaramos una variable para guardar el valor devuelto por el procedimiento; ejecutamos el procedimiento enviando un valor.
La instrucción que realiza la llamada al procedimiento debe contener un nombre de variable para almacenar el valor retornado.

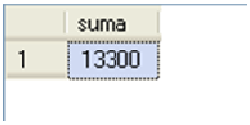


Figura 7.3 Salario acumulado de los empleados que

Trabajan en el departamento de investigación.

7.1.5 Modificación de Procedimientos Almacenados

Los procedimientos almacenados pueden modificarse, por necesidad de los usuarios o por cambios en la estructura de las tablas que referencia.

Un procedimiento almacenado existente puede modificarse con "alter procedure". Sintaxis:

```
ALTER PROCEDURE nombre_procedimiento
@nombre_parametro tipo
AS sentencias;
```

Modificar el procedimiento pa_empleados, para que liste el nombre, apellido, salario y dirección de los empleados de la base de datos Compañía que trabajen en el departamento de investigación cuyo apellido empiece con la letra “P”.

```
ALTER PROCEDURE pa_empleados
@nombre varchar(50),
@ape    varchar(5)
AS
SELECT nombre, apellido, direccion
FROM empleado, departamento
WHERE dnombre = @nombre
AND dnumero = dno
AND apellido LIKE @ape;
```

Para ejecutar el procedimiento se procede de la siguiente manera:

```
EXEC pa_empleados 'Investigacion','P%';
```

	nombre	apellido	direccion
1	Juan	Polo	Sucre 7-12
2	Humberto	Pons	Bolivar 5-67

Figura 7.4 Empleados que trabajan en el departamento

de investigación, cuyo apellido empieza con la letra 'P'.

7.2 Triggers (Disparadores)

7.2.1 Creación de Triggers

Son un conjunto de sentencias que se ejecutan cuando ocurre algún evento en la base de datos. Los triggers se disparan cuando se modifican (agregar, modificar, eliminar) los datos de una tabla, a diferencia de los procedimientos, los triggers se ejecutan automáticamente y no reciben parámetros.

Sintaxis básica:

```
CREATE  TRIGGER  nombre_trigger
ON nombre_tabla
FOR   [insert, update o delete]
AS
Sentencias;
```

7.2.2 Inserción Triggers

La sintaxis para crear un trigger que se ejecute cuando insertamos datos en una tabla es:

```
CREATE TRIGGER nombre_trigger
```

ON NOMBRETABLA
FOR insert
AS
Sentencias;

Crear un trigger que muestre un mensaje cada vez que se ingresa un nuevo trabajador en la tabla empleados.

```
CREATE TRIGGER di_empleados_insert
ON empleado
FOR insert
AS
print 'Se ha ingresado un nuevo empleado';
```

7.2.3 Eliminación Triggers

La sintaxis para crear un disparador, que se ejecute siempre que una instrucción "delete" elimine datos en una tabla es:

CREATE TRIGGER nombre_trigger
ON nombre_tabla
FOR delete
AS
Sentencias;

Crear un disparador que controle que no se elimine más de un registro de la tabla proyecto.

```
CREATE TRIGGER di_proyecto_delete
ON proyecto
FOR delete
AS
if (select count (*) from deleted)>1
begin
        print ('No puede borrar más de un proyecto');
```

```
        rollback transaction;
end;
```

Cuando se activa un disparador "delete", los registros eliminados en la tabla del disparador se agregan a una tabla llamada "deleted". La tabla "deleted" es una tabla virtual que conserva una copia de los registros eliminados; tiene una estructura similar a la tabla en que se define el disparador.

La sentencia rollback indica que la transacción (la eliminación) no tuvo éxito y el gestor de base de datos restaura la base de datos a su estado antes de que la transacción comenzara.

7.2.4 Actualización Triggers

Podemos crear un disparador para que se ejecute siempre que una instrucción "update" actualice los datos de una tabla.

Cuando se ejecuta una instrucción "update" en una tabla que tiene definido un disparador, los registros originales (antes de ser actualizados) se mueven a la tabla virtual "deleted" y los registros actualizados (con los nuevos valores) se copian a la tabla virtual "inserted". Dentro del trigger se puede acceder a estas tablas.

Sintaxis:

```
CREATE TRIGGER nombre_trigger
ON nombre_tabla
FOR update
AS
 sentencias;
```

En el cuerpo de un trigger se puede emplear la función "update(campo)" que recibe un campo y retorna verdadero si el evento involucra actualizaciones (o inserciones) en ese campo; en caso contrario retorna "false".

Crear un triggers que evite que se modifique la fecha de nacimiento de los empleados:

```
CREATE TRIGGER di_empleado_fecha
```

```
ON empleado
FOR update
AS
if update (fecha_n)
begin
        print('La Fecha de nacimiento no puede modificarse');
        rollback transaction;
end;
```

7.2.5 Eliminación de Triggers

Los triggers se eliminan con la siguiente instrucción:

```
DROP TRIGGER  nombre_trigger;
```

Eliminare el trigger “di_empleado_fecha”

```
DROP TRIGGER di_empleado_fecha di_empleado_fecha;
```

7.2.6 Modificación de Triggers

Al modificar un trigger se utiliza la siguiente sintaxis:

```
ALTER TRIGGER  nombre_trigger
nueva_definición;
```

Modificar el disparador “di_proyecto_delete”, para que permita eliminar hasta tres registros de la tabla proyecto.

```
ALTER TRIGGER di_proyecto_delete
ON proyecto
FOR delete
AS
        if (select count (*) from deleted)>3
        begin
```

```
            print ('No puede borrar más de tres  proyectos');
            rollback transaction;
    end;
```

7.3 Conclusiones

En este capítulo se ha visto la importancia de los procedimientos y triggers, lo cuales permiten reutilizar código. Cabe recalcar que los triggers generan tablas virtuales (inserted, deleted) en las cuales se almacenan una copia de los nuevos o antiguos registros.

CAPITULO 8

COMPARACION ENTRE SQL SERVER Y MY SQL

INTRODUCCION

El escogimiento del gestor de base de datos es de las partes más importantes, al momento de realizar un proyecto. Por esta razón en este capítulo, realiza una comparación entre SQL Server 2008 con la versión de MySQL 5.0, en cuanto a, comparación de plataformas soportadas, requerimientos en cuanto a hardware y software, dialectos SQL.

8.1 Comparación de la Plataforma

SQL Server 2008 sólo funciona en plataformas basadas en Windows, incluyendo Windows XP, Windows NT y Windows Vista.
En comparación con SQL Server, MySQL 5.0 soporta las plataformas basadas en Windows, AIX sistemas basados en sistemas HP-UX, Linux, Sun Solaris.

8.2 Requerimientos en cuanto a Hardware

SQL SERVER

- Procesador:
  Como mínimo Pentium III de 1 GHz o superior.
- Memoria:
  Desde 512 MB
- Disco:
  1 Gb de espacio, depende de las características a instalar.

MYSQL

- Procesador:
  Desde Pentium III de 512 MHz
- Memoria:
  Desde 256 MB
- Disco:
  512 Mb de espacio.

8.3 Requisitos de software (para instalar en el S.O. Windows)

SQL SERVER DEVELOPER

- Windows XP SP2
- Net Framework 3.5 SP1
- Windows Instaler 4
- Internet Explorer 6.0 o superior

MYSQL

- Windows XP SP2

8.4 T-SQL vs MySQL lenguaje

El lenguaje de SQL compatible con Microsoft SQL Server 2008 se denomina Transact-SQL (T-SQL). El lenguaje de SQL soportados en MySQL 5.0 se denomina dialecto de MySQL. El lenguaje. A continuación se realiza una breve comparación de T-SQL y el dialecto de MySQL:

Transact-SQL:

- Vistas: Soporta vistas, vistas indexadas.
- Triggers: AdminteTriggers.
- Procedimientos almacenados: Admite procedimientos.
- Funciones: Admite funciones, no admite anidamiento de funciones.
- Foreign Key: Permite actualizar y eliminar en cascada.
- Secuencias / Auto Número: Sí a través de la propiedad IDENTITY.

Dialecto de MySQL

- Vistas: Soporta vistas, no soporta indexadas.
- Triggers: Admite Triggers.
- Procedimientos almacenados: Admite procedimientos.
- Funciones: Admite funciones, no admite anidamiento de funciones.
- Foreign Key: Permite actualizar y eliminar en cascada en el motor tipo InnoDB, no en el motor MyISAM.
- Secuencias / Auto Número: Sí mediante AUTO_INCREMENT.

8.5 Conclusiones

Ambos productos pueden ser utilizados para construir el sistema estable y eficiente. SQL Server ocupa más recursos de hardware y solo funciona en plataformas Windows a diferencia de MySQL que funcionas en diferentes plataformas. La interfaz gráfica de SQL Server es más sencilla y más fácil de utilizar.

CAPITULO 9

CONCLUCIONES

9.1 Conclusiones Teóricas

La monografía desarrollada abarca varios temas, desde la instalación y configuración del gestor de base de datos, administración de base de datos. Seguridades a nivel de servidor, de base de datos y objetos. Al final del tutorial se indica la utilización del ambiente grafico de SQL Server Management en cuento a administración y gestión de base de datos.

9.2 Conclusiones Metodológicas

En el desarrollo de esta monografía se utilizo información de diversas fuentes, como el internet, libros, artículos. También se utilizo los conocimientos adquiridos a lo largo de la carrera universitaria, concretamente en la materia de base de datos.

9.3 Conclusiones Pragmáticas

El objetivo principal de esta monografía ha sido la de crear un tutorial, el cual se un apoyo para el estudiante, facilitando el aprendizaje de este gestor de base de datos. Ahorrando tiempo y recursos.

La creación de ejercicios prácticos es otro objetivo, por medio de los ejercicios el alumno podrá afianzar sus conocimientos y potencializar sus destrezas.

BIBLIOGRAFIA

- **Robin Dewson,** Beginning SQL Server 2008 for Developers From Novice to Professional, **Apress.**
-
- **Robert E. Walters Michael Coles, Fabio Ferracchiati, Robert Rae, Donald Farmer;** Accelerated SQL Server 2008; **Apress.**
- **Jorge Moratalla,** Base de Datos con SQL Server Transact SQL, **Grupo Eidos.**
- **James R. Groff, Paul N. Weinberg**. Guía Lan Times de SQL (1998). **McGraw-Hill.**
- **William R. Stanek,** SQL Server 2008. Guía del Administrador, **Anaya Multimedia. Primera edición.**
- **Hotek, Mike,** SQL Server 2008, **Anaya Multimedia. Primera edición.**
- **Cuaderno Docente,** Fundamentos de Base de Datos.
- http://www.sqlserversi.com/2009/01/seguridad-en-sql-server.html.
- http://www.sqlserversi.com/2009/01/seguridad-en-sql-server.html
- http://www.solovb.net
- http://www.mysql.com

ANEXO I

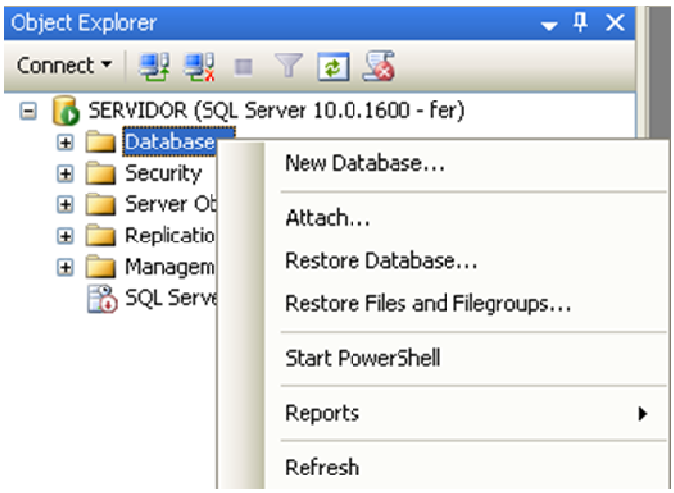
CREACION DE UNA BASE DE DATOS

USANDO SQL SERVER MANAGEMENT

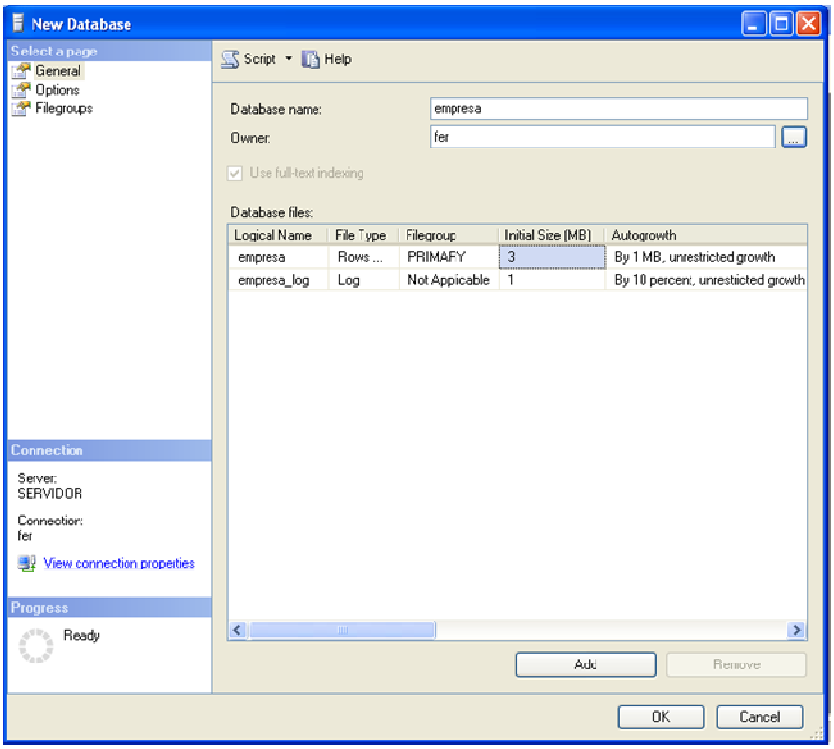
Creación de Base de Datos:

Para ingresar a SQL SERVER MANAGEMENT, nos ubicamos en el botón Inicio> Programas> Microsoft SQL Server 2008> SQL server Management 2008, aparece una pantalla la cual nos pide el tipo de servidor seleccionamos “DataBase Engine”, el nombre del servidor en este caso el nombre con el que configuramos “Servidor” y la autentificación de SQL Server, ingresamos el usuario “fer” y el password “fer” y pulsamos el botón “Connect”.

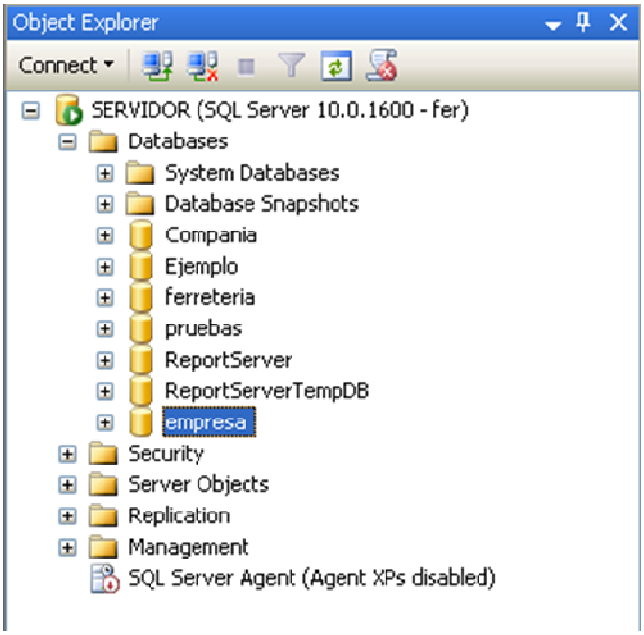
En el explorador de objetos damos un clic derecho en “Database” y seleccionamos la opción “New Database”.



Ingresamos el nombre de la base de datos en nuestro ejemplo “empresa” y seleccionamos el Owner “fer” y pulsamos el botón “OK”.



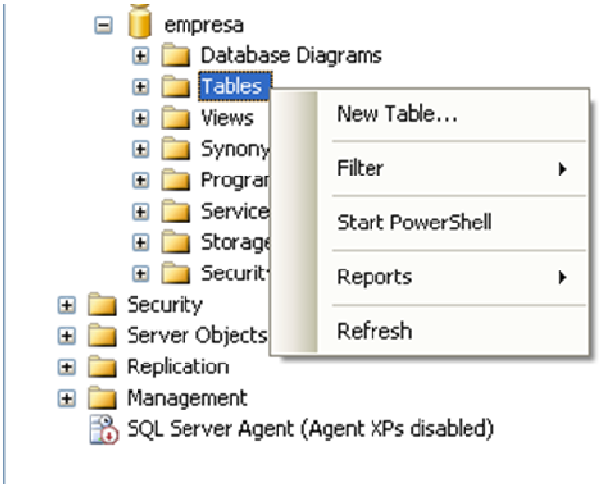
En el explorador de objetos en la sección de base de datos se verifica que se ha creado la base de datos “empresa”.



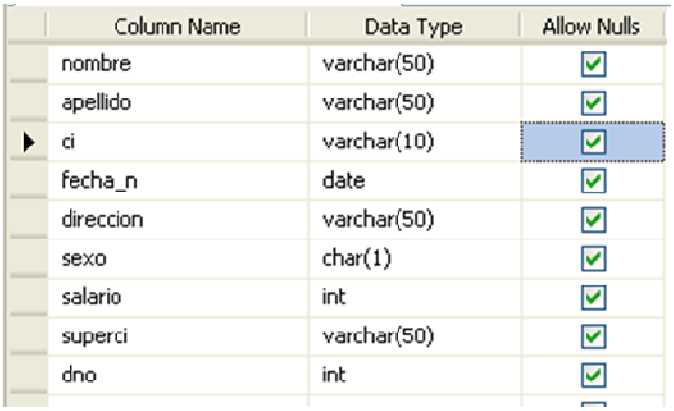
Creación de Tablas:

- Tabla empleado

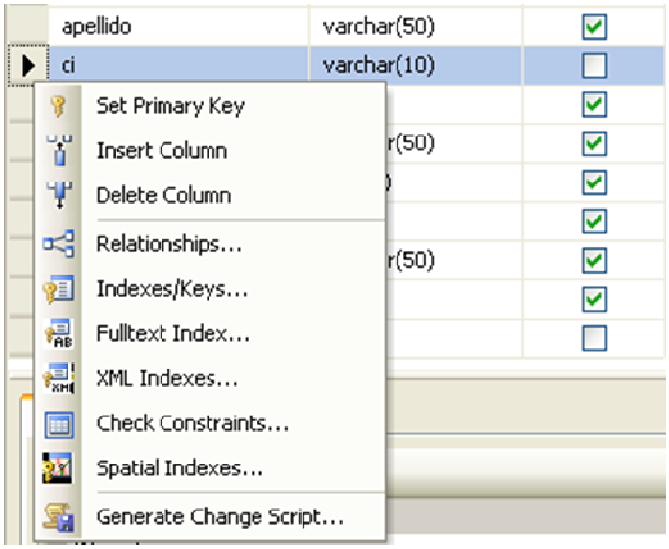
Desplegamos la base de datos empresa y damos un clic derecho sobre la opción “Tables”, damos un clic en la opción “New Table”.



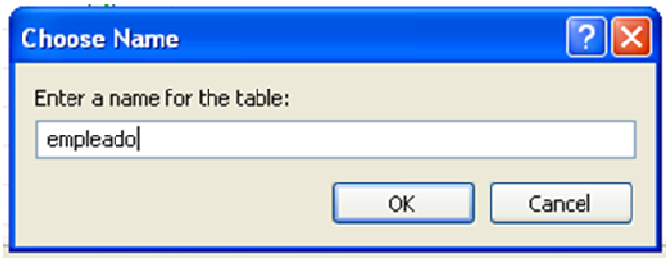
Procedemos a ingresar el nombre de cada columna y el tipo de dato.



Para indicar que la columna “ci” es llave primaria, primero quitamos el visto bueno de “Allow Nulls” y luego damos clic derecho sobre el campo “ci” y seleccionamos la opción “Set Primary Key”.



Presionamos el botón guardar, en una nueva ventana ingresamos el nombre de la tabla “empleado” y damos clic en el botón “OK”.



Seguimos el mismo procedimiento para crear las siguientes tablas.

- Tabla Departamento

Column Name	Data Type	Allow Nulls
dnombre	varchar(50)	☑
dnumero	int	☐
jefeci	varchar(10)	☑
jefe_fi	date	☑

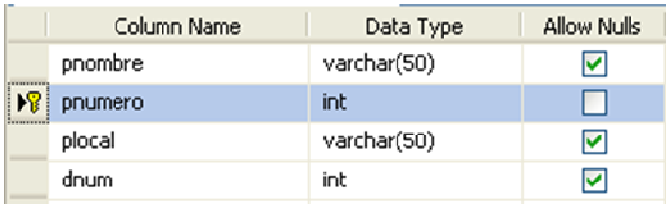
- Tabla Localización

Column Name	Data Type	Allow Nulls
dnumero	int	☐
dep_loca	varchar(50)	☑

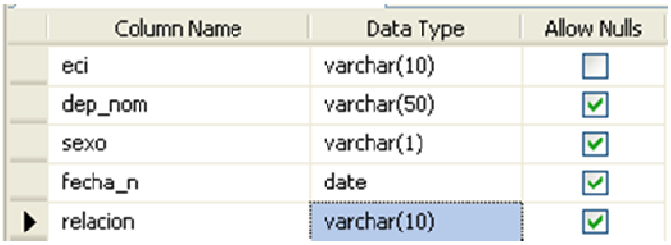
- Tabla Trabaja_en

Column Name	Data Type	Allow Nulls
eci	varchar(10)	☐
pno	int	☐
horas	decimal(4, 2)	☑

- Tabla Proyecto



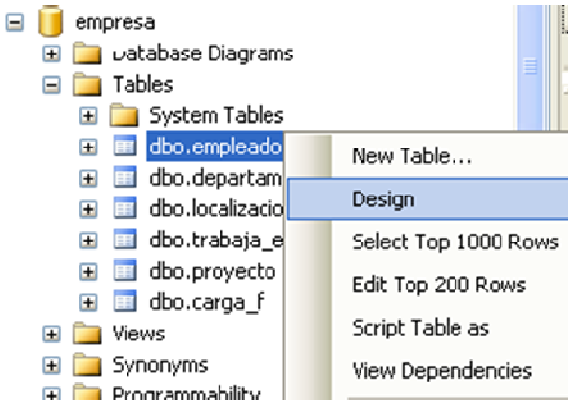
- Tabla Carga_F



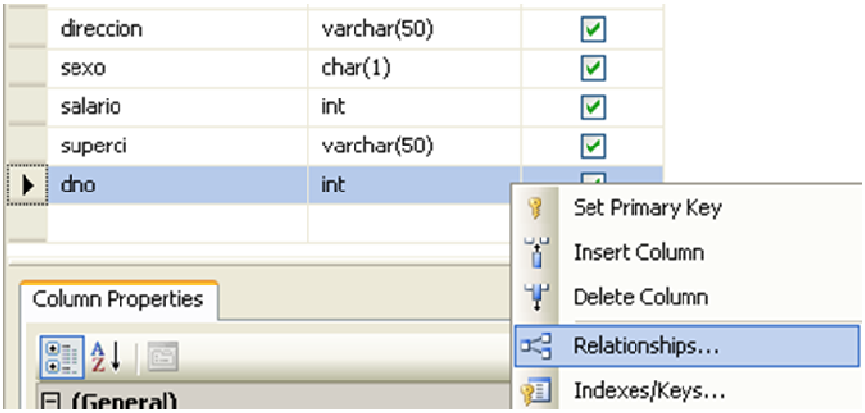
Creación de Llaves Foráneas:

-Tabla Empleado

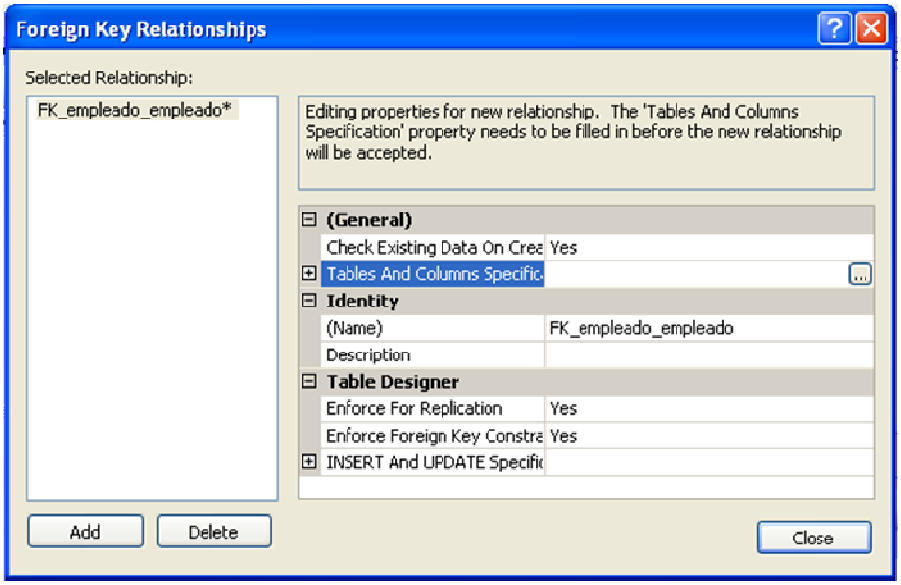
Desplegamos la opción “Tables” de la Base de datos Empresa, damos un clic derecho sobre la tabla empleado y seleccionamos la opción “Design”.



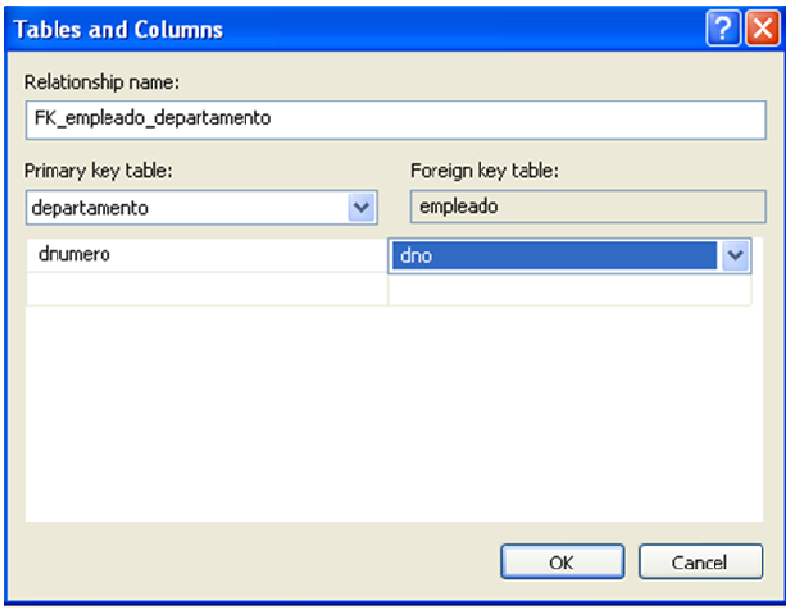
Damos un clic derecho en el campo que será la llave foránea y seleccionamos la opción “Relationships”



Agregamos una nueva relación presionando el botón “Add” y especificamos las tablas y las columnas.



Seleccionamos la tabla referenciada y la llave primaria, y la tabla principal con la llave foránea y presionamos el botón “Ok”.



En la opción “Insert and Update Specification” nos permite indicar que acciones tomar cuando se elimine (DeleteRule) o modifique (Update Rule) la llave principal de la tabla que está siendo referenciada.

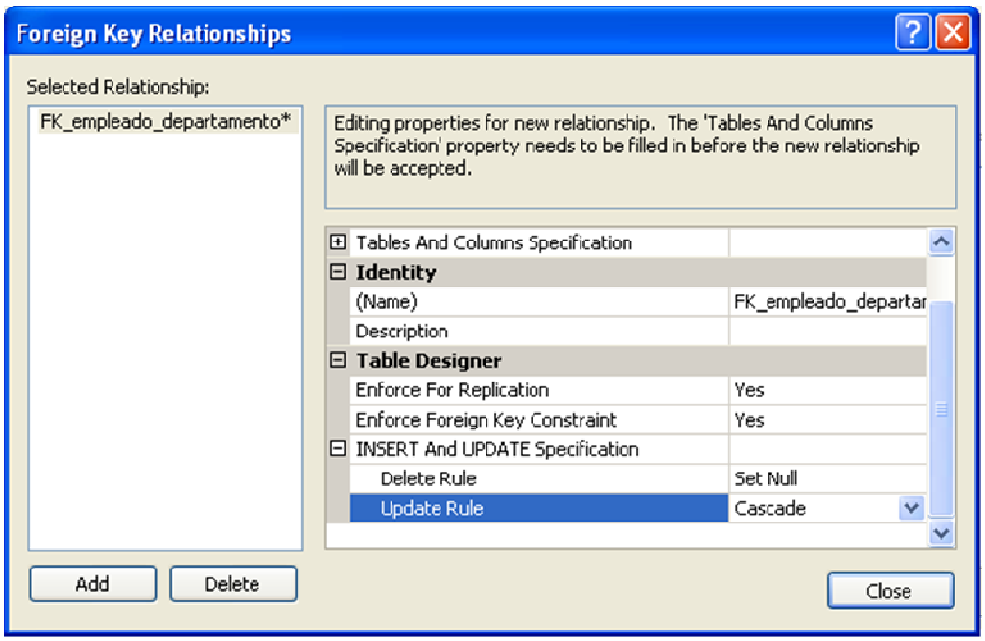
No Action: Indica que se intenta eliminar o actualizar un valor de la llave primaria de la tabla referenciada, se genere un error y la acción no se realice.

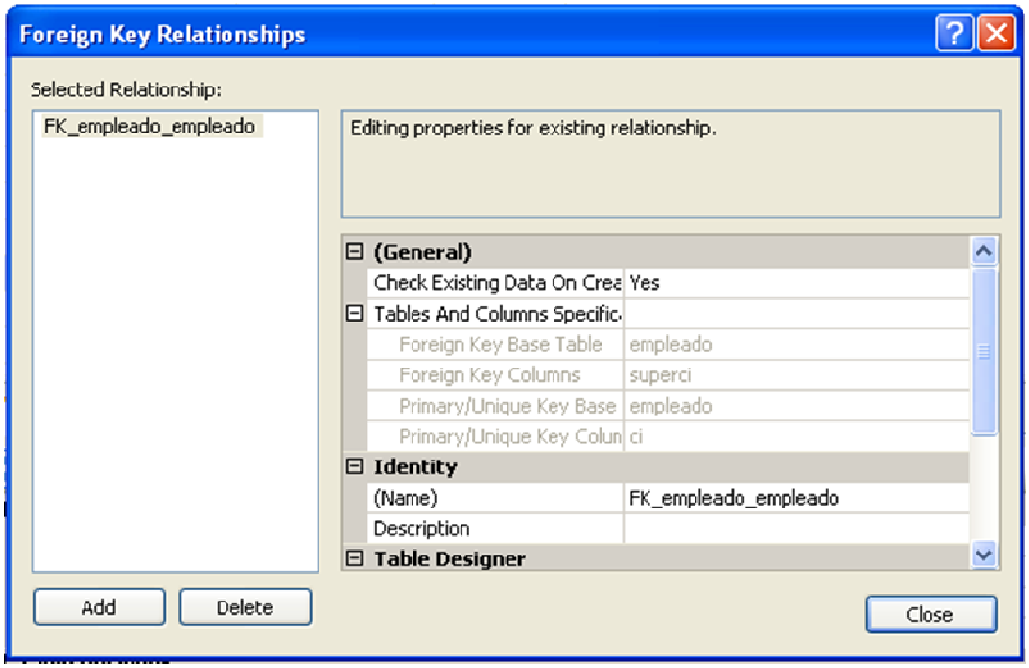
Cascade: Si se elimina o actualiza un valor de la llave primaria en la tabla referenciada, los registros coincidentes en la tabla principal, también se eliminen o modifiquen

Set Null: Si se elimina o actualiza un valor de la llave primaria de la tabla referenciada, se colocara el valor de NULL en el campo de la llave foránea de la tabla principal.

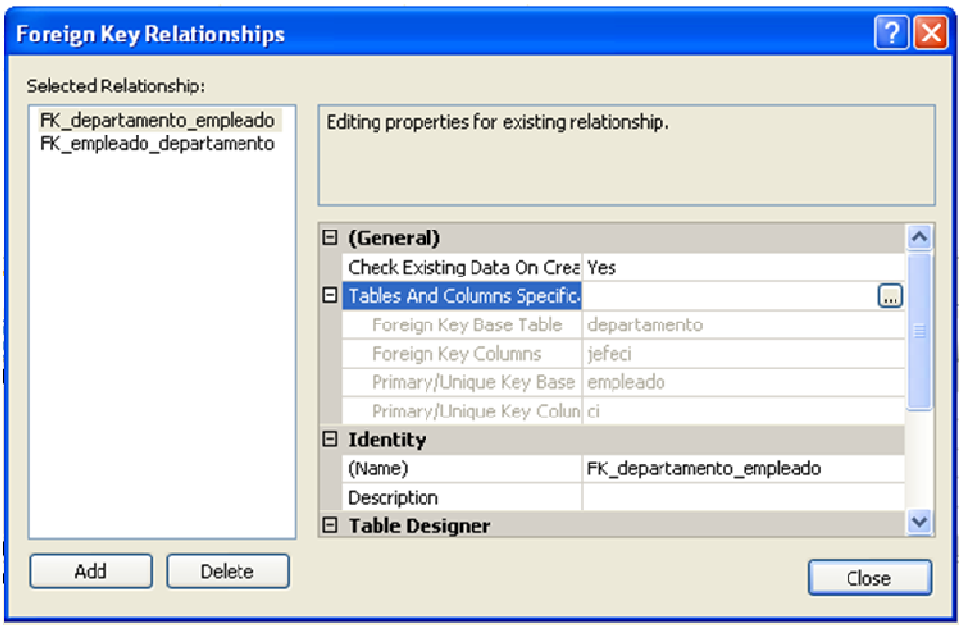
Set Default: Cuando se elimina o actualiza un valor de la llave primaria de la tabla referenciada, se colocara el valor por defecto en el campo de la llave foránea de la tabla principal.

Una vez seleccionado estas opciones Set Null para la eliminación y Cascade para la modificación, presionamos el botón “Close”.

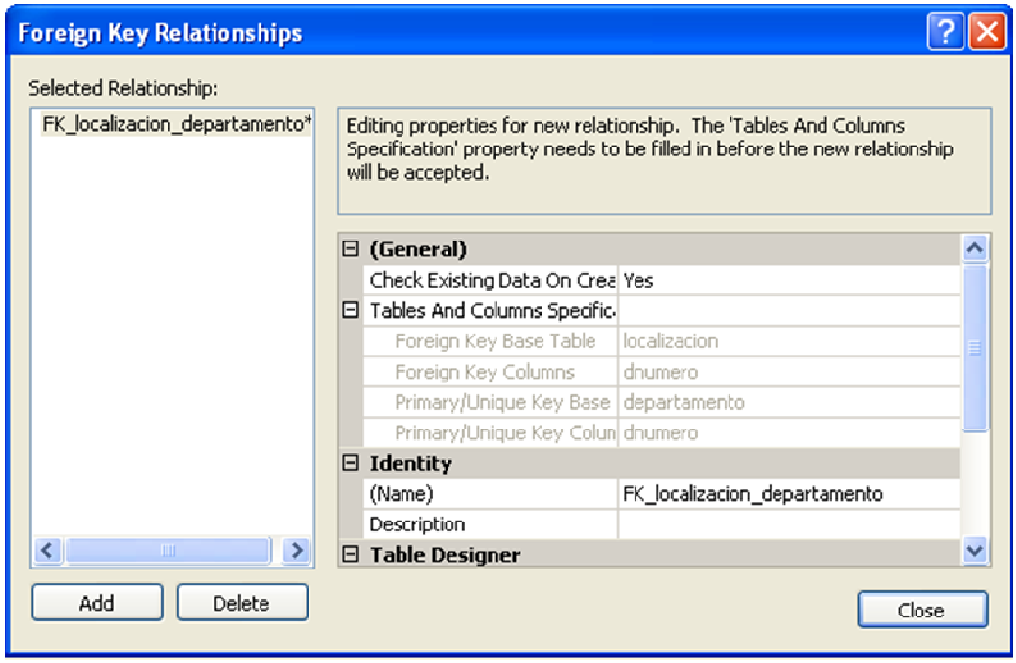




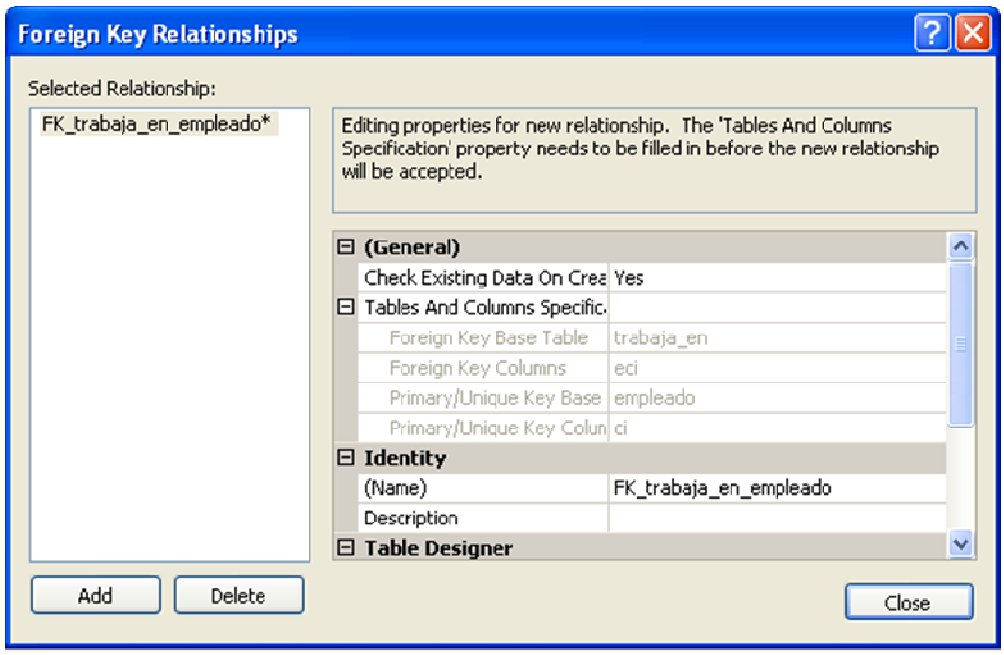
-Tabla Departamento

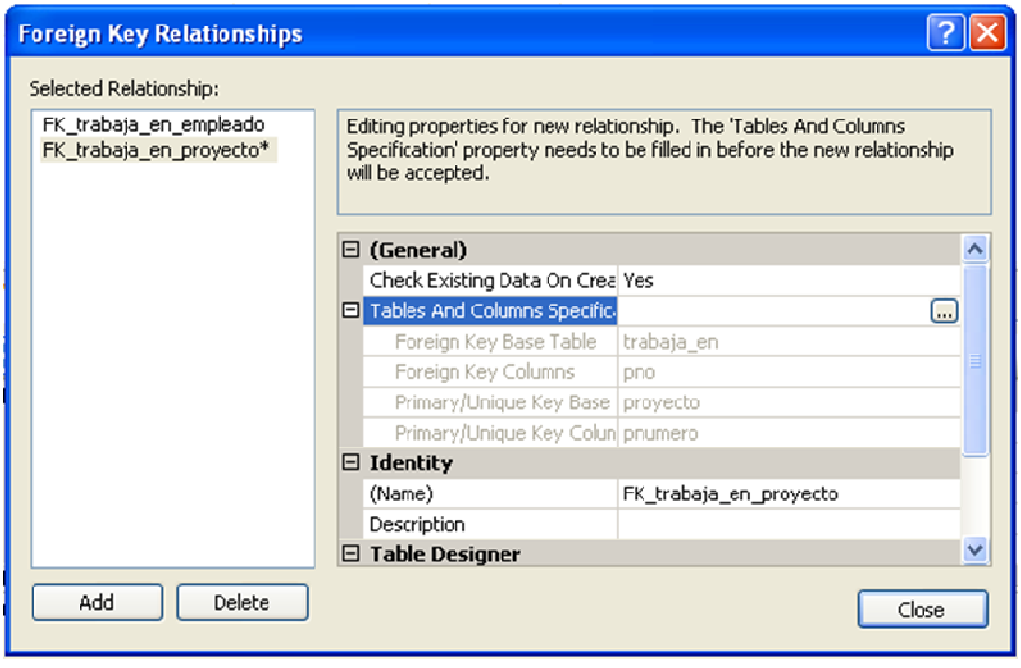


-Tabla Localización

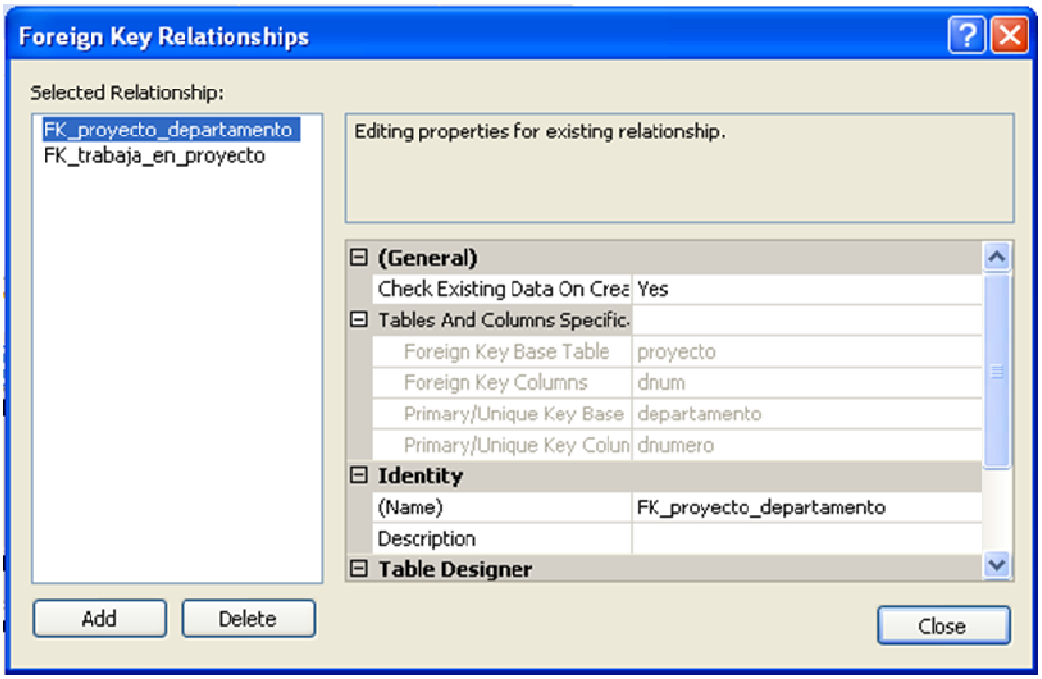


-Trabaja en

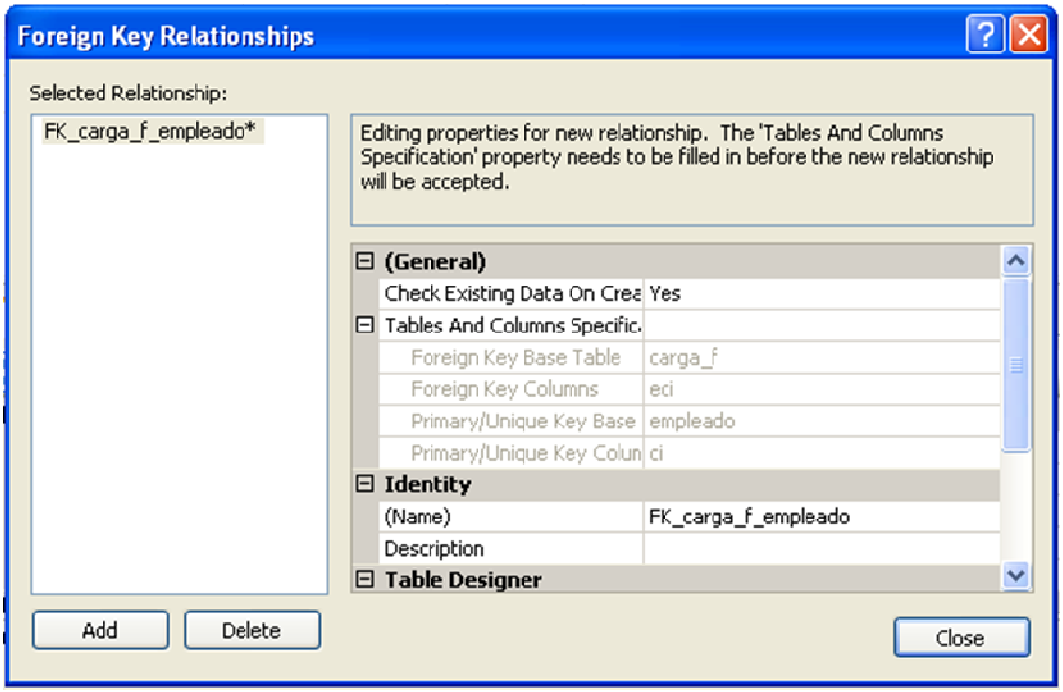




-Tabla Proyecto



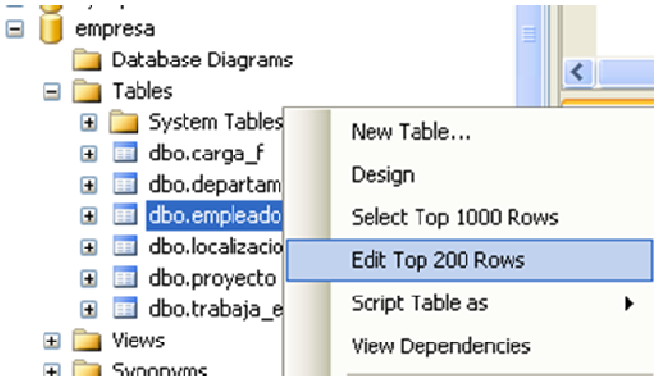
-Tabla Carga-f



Ingreso de Información:

-Tabla Empleado

En el explorador de objetos desplegamos la opción “Databases” y seleccionamos la base de datos empresa, desplegamos la opción “Tables”, damos un clic derecho sobre la tabla empleado y seleccionamos la opción “Edit Top 200 Rows”



Procedemos a ingresar los datos en la tabla, en la figura se puede visualizar que no existen datos en las columnas “superci” y “dno”, se debe a que estos dos campos son llaves foráneas y no existen por el momento estos datos, si intentáramos ingresar información nos daría un error debido a la integridad referencial. Más adelante se ingresara lo datos faltantes. Para que los datos sean ingresados se debe presionar el botón (ejecutar sql).

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
Marcia	Mora	453453453	1960-03-29	Colombia 4-23	F	2500	NULL	NULL
Pablo	Castro	666884444	1955-09-15	Bolivar 1-50	M	3800	NULL	NULL
Jaime	Perez	888665555	1957-04-05	Sangurima 8-34	M	5500	NULL	NULL
Elena	Tapia	987654321	1961-05-03	Ordonez 7-29	F	4300	NULL	NULL
Manuel	Bonilla	987987987	1955-07-16	B. Malo 1-10	M	2500	NULL	NULL
Irma	Vega	999887777	1950-11-13	P. Cordova 3-45	F	2500	NULL	NULL

-Tabla Departamento

dnombre	dnumero	jefeci	jefe_fi
Compras	1	333445555	1978-06-06
Administrativo	4	987654321	1982-12-05
Investigacion	5	888665555	1980-12-05

-Tabla Localización

dnumero	dep_loca
1	Guayaquil
5	Quito
5	Manta
5	Cuenca
1	Cuenca

-Trabaja en

eci	pno	horas
123456789	1	12,50
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*

-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

-Carga_f

	eci	dep_nom	sexo	fecha_n	relacion
▶	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	NULLElizabeth	F	1960-12-12	Conyuge

-Empleado (Actualización)

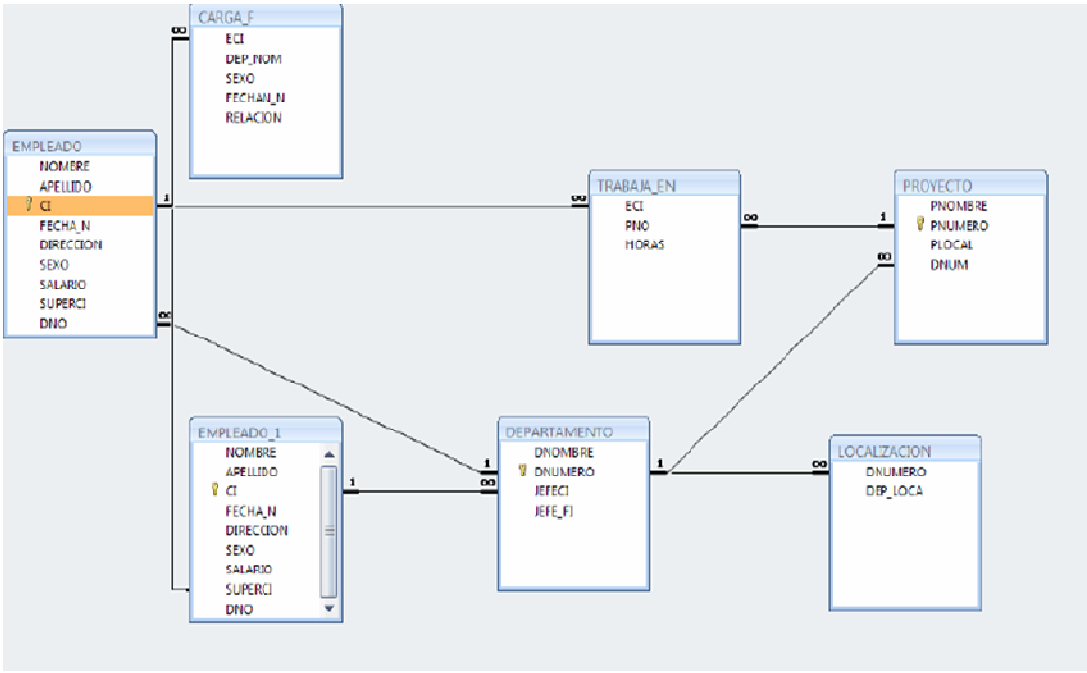
	nombre	apellido	ci	fecha_n	direccion	sexo	salario	superci	dno
	Juan	Polo	123456789	1959-03-03	Sucre 7-12	M	3000	333445555	5
	Humberto	Pons	333445555	1960-12-25	Bolivar 5-67	M	4000	888665555	5
	Marcia	Mora	453453453	1960-03-29	Colombia 4-23	F	2500	333445555	5
	Pablo	Castro	666884444	1955-09-15	Bolivar 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	Ordonez 7-29	F	4300	888665555	4
	Manuel	Bonilla	987987987	1958-07-16	B. Malo 1-10	M	2500	987654321	4
▶	Inna	Vega	999887777	1950-11-13	P. Cordova 3-45	F	2500	987654321	4

ANEXO II

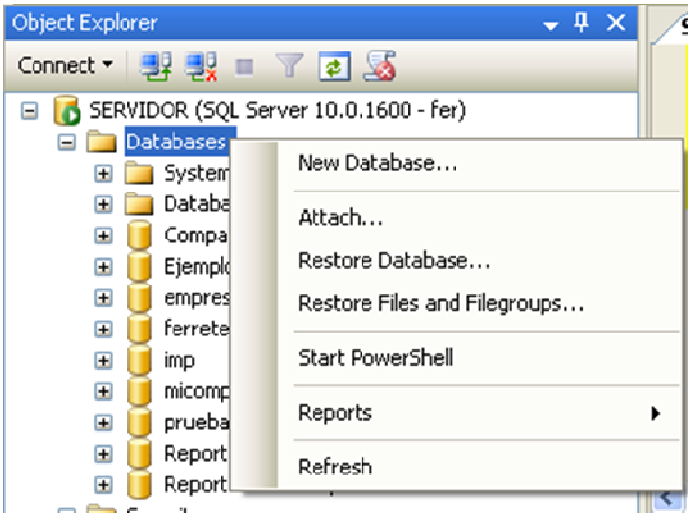
MIGRAR UNA BASE DE DATOS

En este anexo indicamos como migrar una base de datos Access 2003 a Sql Server 2008, para poder migrar debe estar instalado “SQL Server Management Studio”.

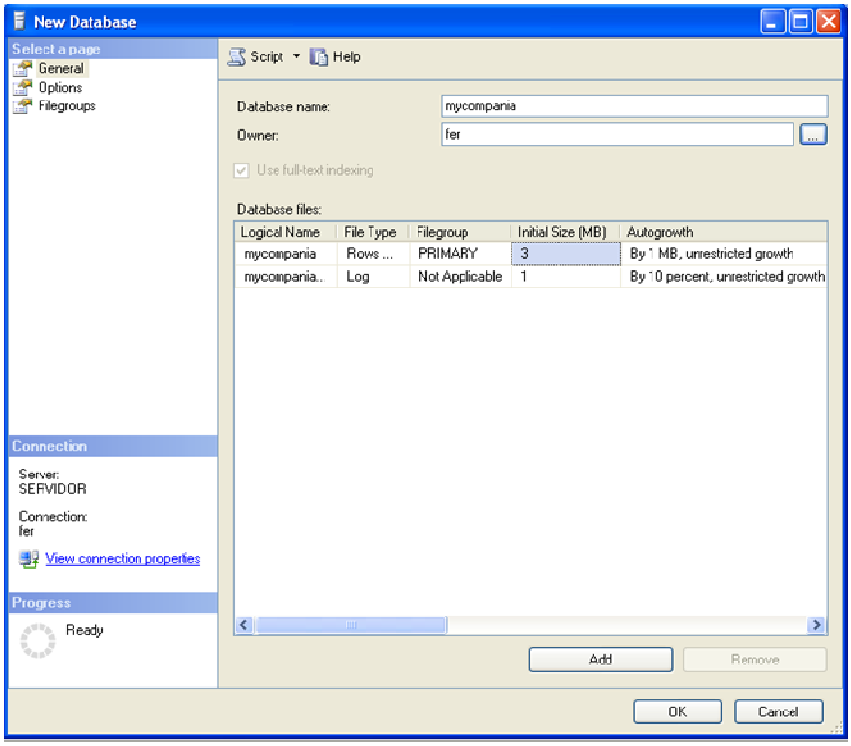
Base de datos a migrar “mycompania”, creada en Microsoft Access.



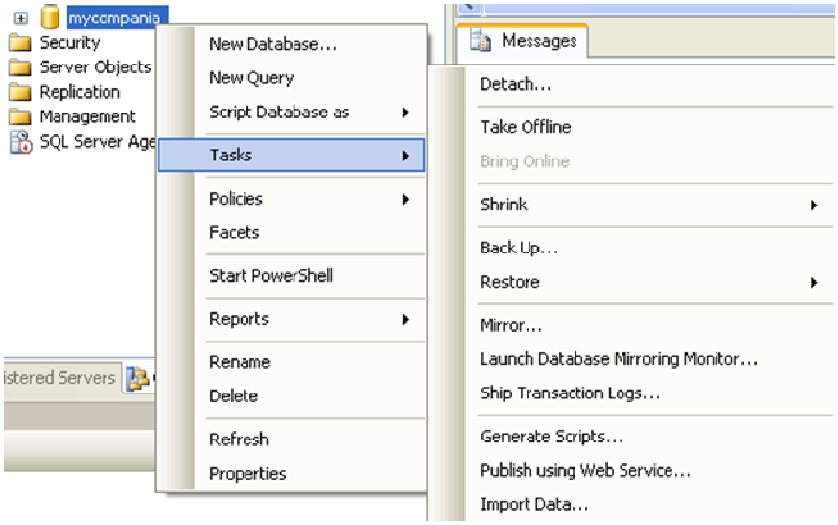
Ejecutamos SQL Server Management Studio y nos conectamos con el servidor de base de datos. En el explorador de objetos damos en clic derecho en “Databases” y seleccionamos la opción “New Database”.



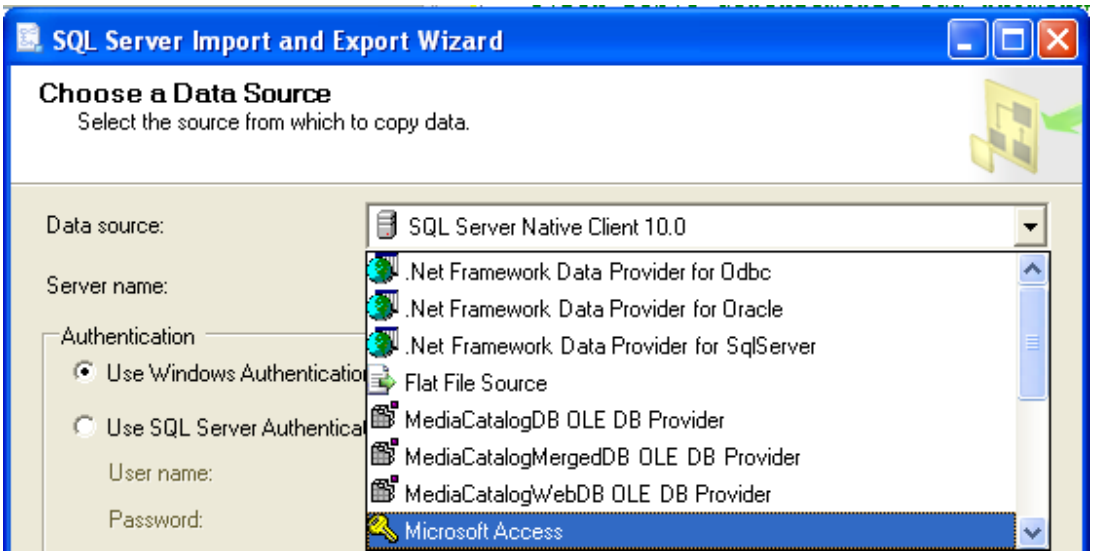
Le damos un nombre a la base de datos en nuestro ejemplo es “mycompania” y le asignamos un owner “fer” y presionamos el botón “Ok”.



Damos un clic derecho sobre la nueva base de datos “mycompañia”, seleccionamos la opción “Tasks” (tareas) y luego “Import Data” (Importar datos).

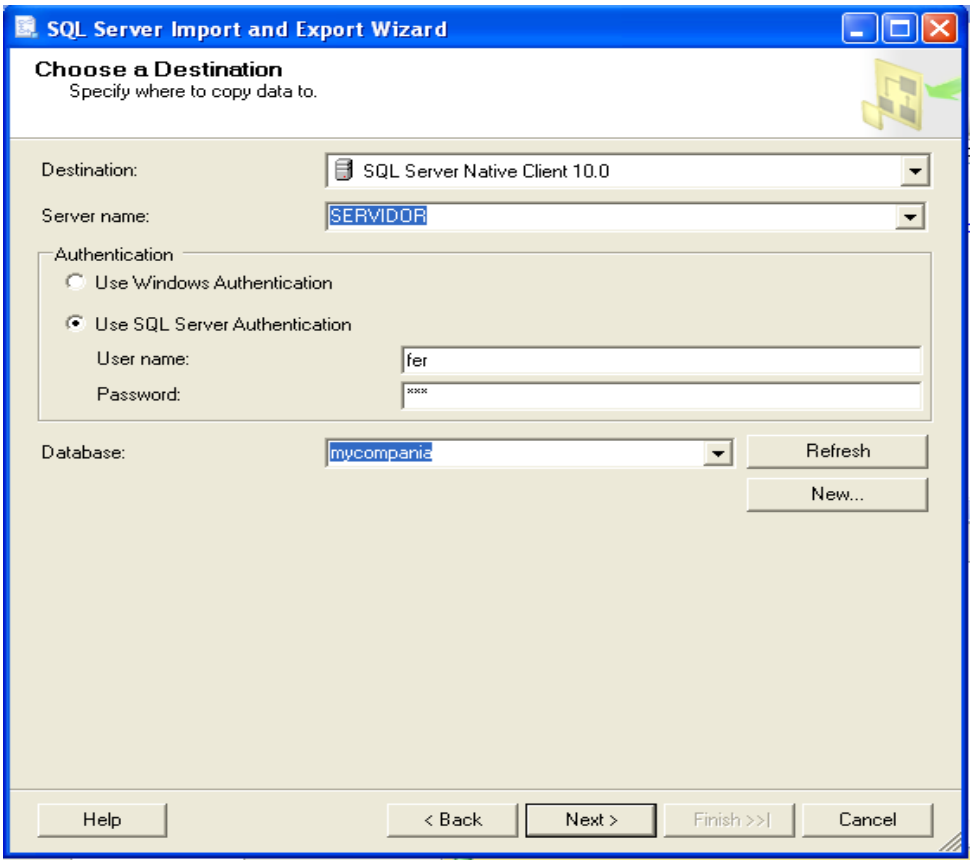


En Fuente de datos (Data Source) que muestra el asistente seleccionamos “Microsoft Access”.

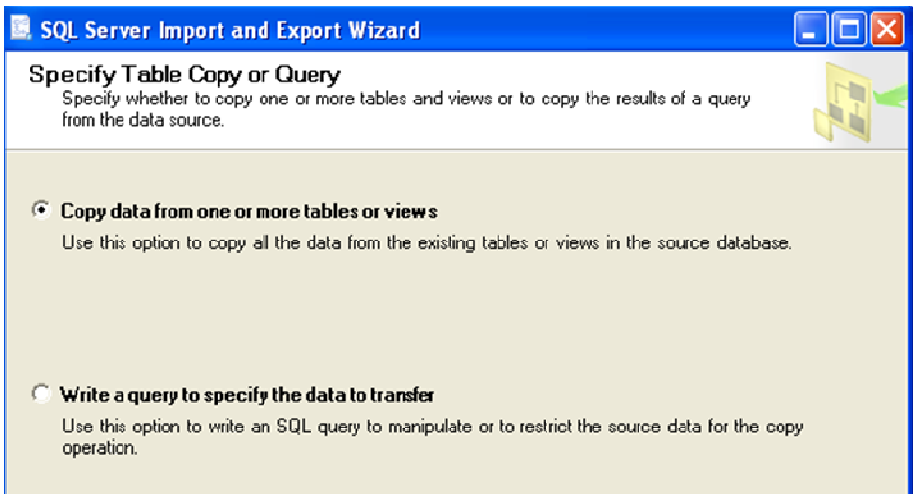


A continuación indicamos la ubicación de la base de datos que va ser migrada y presionamos el botón “Next”

En la siguiente ventana del asistente, indicamos el nombre del servidor, el usuario en nuestro caso “fer” y la contraseña “fer”, y pulsamos el botón “Next”.

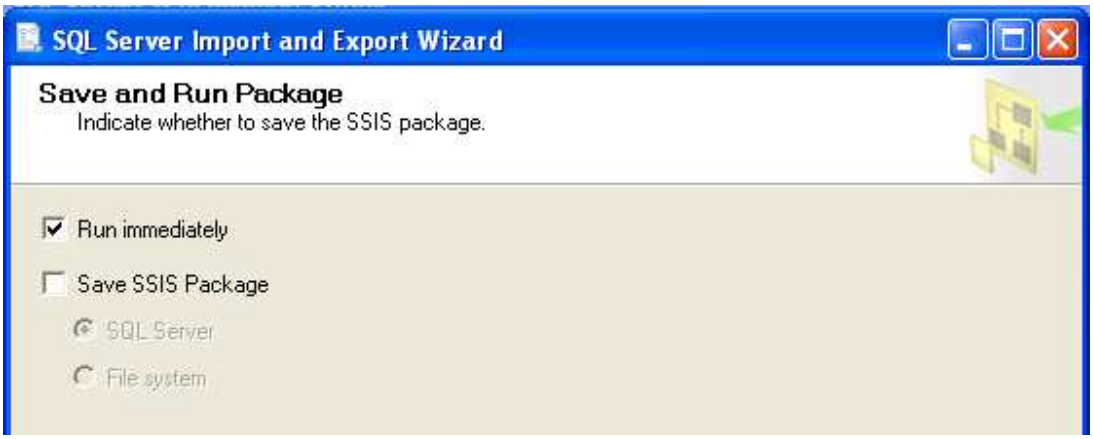


A continuación seleccionamos la opción “Copy data from one or more tables or views” para transferir toda la base de datos, si se desea transferir algo en particular se debe seleccionar la opción “Write a query to specify the data to transfer”, y pulsamos el botón “Next”.



Seleccionamos las tablas que van a ser transferidas y pulsamos “Next”.

Indicamos que se ejecute inmediatamente, pulsamos el botón “Next” y después el botón “Finalizar”.



Al terminar la transferencia se muestra una ventana de resumen en la cual indica el número de registros transferidos por cada tabla.

	Action	Status	Message
✓	Initializing Connections	Success	
✓	Setting SQL Command	Success	
✓	Setting Source Connection	Success	
✓	Setting Destination Connection	Success	
✓	Validating	Success	
✓	Prepare for Execute	Success	
✓	Pre-execute	Success	
✓	Executing	Success	
ⓘ	Copying to [dbo].[carga_f]	Success	7 rows transferred
ⓘ	Copying to [dbo].[departamento]	Success	3 rows transferred
ⓘ	Copying to [dbo].[empleado]	Success	8 rows transferred
ⓘ	Copying to [dbo].[localizacion]	Success	5 rows transferred
ⓘ	Copying to [dbo].[proyecto]	Success	6 rows transferred
ⓘ	Copying to [dbo].[trabaja_en]	Success	16 rows transferred
ⓘ	Post-execute	Success	

Terminada la migración, procedemos a revisar la base de datos en el explorador de objetos y verificamos la existencia de las seis tablas.

Para finalizar procedemos a revisar la estructura y los datos de la tabla “empleado”.

Column Name	Data Type	Allow Nulls
nombre	nvarchar(30)	☑
apellido	nvarchar(30)	☑
ci	nvarchar(10)	☑
fecha_n	datetime	☑
direccion	nvarchar(30)	☑
sexo	nvarchar(1)	☑
salario	int	☑
superci	nvarchar(10)	☑
dno	int	☑

	nombre	apellido	ci	fecha_n	direccion	sexo	salario	superci	dno
1	Juan	Polo	123456789	1959-03-03 00:00:00.000	Sucre 7-12	M	3000	333445555	5
2	Humberto	Pons	333445555	1960-12-25 00:00:00.000	Bolivar 5-67	M	4000	888665555	5
3	Marcia	Mora	453453453	1960-03-29 00:00:00.000	Colombia 4-23	F	2500	333445555	5
4	Pablo	Castro	666884444	1955-09-15 00:00:00.000	Bolivar 1-50	M	3800	333445555	5
5	Jaime	Perez	888665555	1957-04-05 00:00:00.000	Sangurima 8-34	M	5500	NULL	1
6	Elena	Tapia	987654321	1961-05-03 00:00:00.000	Ordonez 7-29	F	4300	888665555	4
7	Manuel	Bonilla	987987987	1958-07-16 00:00:00.000	B. Malo 1-10	M	2500	987654321	4
8	Irma	Vega	999887777	1950-11-13 00:00:00.000	P. Cordova 3-45	F	2500	987654321	4

Como se puede visualizar, la estructura es similar con pequeños cambios, la llave primaria y las llaves foráneas han sido eliminadas, respecto a la información esta se mantiene intacta.