MANUAL DE PRACTICAS

Taller de Base de Datos

Retícula ISIC-2004-296

Elaborado por:

M.C. Martha Escamilla Zepeda

ÍNDICE

INTRODUCCION

Sistema de Nómina

Para todas las prácticas de este Manual, se manejará una misma base de datos llamada NOMINA, esto facilitará el aprendizaje de cada concepto. En este capítulo introductorio se definen los siguientes puntos importantes para el desarrollo de cada una de las prácticas y el mejor entendimiento del Sistema sobre el cual se está trabajando.

- **Objetivo del Sistema**
- **Función de las tablas de la base de datos NOMINA**
- **Políticas de la Empresa**
- **Definición de cada tabla de la base de datos NOMINA**
- **Diagrama Entidad-Relación**
- **Integridad Referencial y Consistencia de Datos**
- **Diagrama de Flujo del Sistema**
- **Base de Datos EJEMPLO para las prácticas de Select**

Objetivo del Sistema

El Sistema de Nómina tiene por objetivo generar los ingresos recibidos por cada empleados considerando percepciones y deducciones, obteniendo su ingreso neto.

Función de las tablas de la base de datos NOMINA

La base de datos NOMINA está integrada por las siguientes tablas:

- Empleados
- Departamentos
- Nomina
- Conceptos
- Descuentos
- Movimientos

EMPLEADOS: En esta tabla se almacena la información general de cada empleado, así como sueldo y fecha de ingreso. Esta tabla está relacionada con la tabla DEPARTAMENTOS.

DEPARTAMENTOS: En esta tabla se almacena la información referente a los diferentes departamentos que integran una empresa. Esta tabla está relacionada con la tabla EMPLEADOS.

NOMINA: En esta tabla se almacenan los ingresos finales de cada empleado, por quincena, incluyendo descuentos e ingreso neto y mantiene los movimientos sólo del año actual. Esta tabla está relacionada con EMPLEADOS.

CONCEPTOS: En esta tabla se almacena la información referente los diferentes conceptos manejados normalmente por las empresas. De manera general Percepciones y Deducciones. Esta tabla se relaciona con MOVIMIENTOS.

DESCUENTOS: Esta tabla almacena la información de los porcentajes de descuentos que se aplicarán al ingreso del empleado ISPT. Es decir, para calcular el descuento que se aplicará deberá verificarse que el sueldo se encuentre en el rango del límite máximo y mínimo de esta tabla. Esta tabla sirve para realizar el cálculo de la Nómina de la empresa y no está relacionada con ninguna tabla, se utiliza para la aplicación de los porcentajes de descuento.

MOVIMIENTOS: Esta tabla almacena la información de todos los movimientos de percepción y deducción de cada empleado en cada quincena de todo el año en curso. Es decir contiene el detalle de todos los movimientos para todos los empleados. Con esta información y la del resto de las tablas se genera la nómina para la empresa. Esta tabla está relacionada con EMPLEADOS y CONCEPTOS.

Políticas de la Empresa

- El sistema genera la Nómina sólo para aquellos empleados que laboran actualmente y de forma quincenal. (Tabla Empleados)
- Los movimientos de Nómina de cada quincena será almacenado sólo del año en curso. (Tabla de MOVIMIENTOS)
- El registro de Pago de Nómina quincenal por empleado, serán almacenados sólo por un año. (Tabla de NOMINA)
- Si algún Departamento de la empresa desaparece, los empleados serán reubicados a otros Departamentos.
- Si un concepto de Nómina ya no fuera vigente, se mantiene por el año en curso.
- No existen contrataciones de empleados externos. Siempre un empleado debe pertenecer a un Departamento de la Empresa.

Definición de cada tabla de la base de datos NOMINA

EMPLEADOS

Campo	Uso	Tipo de Dato	Constraint	Borrado en Cascada
No_emp	**Número de empleado**	**Char(5)**	**Primaria CH>99999**	
Nombre	**Nombre del empleado**	**Varchar(10)**		
Apellido_p	**Apellido paterno**	**Varchar(10)**		
Apellido_m	**Apellido materno**	**Varchar(10)**		
RFC	**RFC del empleado**	**Char(10)**	**CH>AAAA999999**	
Fecha_nac	**Fecha nacimiento**	**Date**		
Fecha_ingreso	**Fecha ingreso a la empresa**	**Date**	**D>TODAY**	
Ciudad	**Cuidad de nacimiento**	**Varchar(15)**		
Sexo	**Sexo del empleado**	**Char(1)**	**CH > F/M**	
Cve_depto	**Clave depto. donde trabaja**	**Char(3)**	**Foránea CH> D99**	**NO**
Sueldo_base	**Sueldo Base mensual**	**Integer**	**CH> mayor 0**	

NOMINA

Campo	Uso	Tipo de Dato	Llave	Borrado en Cascada
No_emp	Número de empleado	Char(5)	Foránea (no_emp) Primaria Compuesta (no_Emp, no_quincena) CH> 99999	SI
No_quincena	Número de quincena	Smallint		
Percepciones	Total de percepciones para esa quincena	Float	CH> percepciones >=deducciones	
Deducciones	Total de deducciones para esa quincena	Float		
Ingreso_Neto	Ingreso neto para esa quincena. Percepciones menos deducciones	Float		

DEPARTAMENTOS

Campo	Uso	Tipo de Dato	Llave	Borrado en Cascada
cve_depto	Clave del departamento	Char(3)	Primaria CH> D99	
Nombre	Nombre del departamento	Varchar(20)		

CONCEPTOS

Campo	Uso	Tipo de Dato	Llave	Borrado en Cascada
Cve_concepto	Clave del concepto, ya sea percepción o deducción	Char(3)	Primaria CH > 999	
Descripción	Descripción del concepto	Varchar(20)		
Tipo_concepto	Tipo de concepto. D=deducción, P=percepción	Char(1)	CH > D/P	

DESCUENTOS

Campo	Uso	Tipo de Dato
Lim_inferior	Ingreso mínimo para aplicación de porcentaje	Integer
Lim_superior	Ingreso máximo para aplicación de porcentaje	Integer
Porcentaje_descto	Porcentaje de descuento a aplicar sobre total de percepciones	Smallint

MOVIMIENTOS

Campo	Uso	Tipo de Dato	Llave	Borrado en Cascada
No_emp	Número de empleado	Char(5) CH > 99999	Foránea (no_emp) Primaria Compuesta (no_Emp, no_quincena, Cve_concepto) CH > 999 Foránea (cve_concepto)	SI
No_quincena	Número de quincena en la que aplica el movimiento	Smallint CH mayor 0 menor 25		
Cve_concepto	Clave del concepto del movimiento aplicado	Char(3) CH>999		SI
Fecha_mov	Fecha en la que aplica el movimiento, ya sea deducción o percepción	Date	D > TODAY	
Cantidad	Cantidad en pesos de movimiento realizado	Float		

Diagrama Entidad-Relación

En el diagrama Entidad-Relación de la Figura 1.1, muestra las relaciones entre las diferentes entidades comentadas anteriormente. Se puede observar para cada RELACIÓN lo siguiente:

- **EMPLEADOS-MOVIMIENTOS**: Se relacionan a través del no_emp, significa que para que existan movimientos de nómina, debe el empleado estar trabajando en la empresa.
- **EMPLEADOS-NOMINA:** Se relacionan a través del no_emp, significa que para que existan quincenas por generar o pagar, debe el empleado estar trabajando en la empresa.
- **CONCEPTOS-MOVIMIENTOS**: Se relacionan a través de la cve_concepto, estos son todos los posibles conceptos dentro de un Sistema de Nómina, incluyendo Deducciones o Percepciones. No puede existir un movimiento que no esté relacionado directamente con un concepto.
- **EMPLEADOS-DEPARTAMENTOS**: Se relacionan a través de la cve_depto; es decir, un empleado siempre debe pertenecer a un Departamento de la Empresa, no existen contrataciones externas.
- **DESCUENTOS:** Esta tabla no mantiene relación directa con ninguna tabla. Servirá durante el proceso de generación de la Nómina, para obtener el porcentaje de descuento por concepto de ISPT (Impuesto Sobre el Producto del Trabajo), que es una Deducción.

Integridad Referencial y Consistencia de Datos

Las consideraciones importantes tomadas en cuenta para garantizar la Integridad Referencial y Consistencia de Datos, son mostradas en la Figura 1.2 y son las siguientes:

- **EMPLEADOS-MOVIMIENTOS**
  - Esta definición garantiza que no se generen movimientos de empleados que no estén trabajando en la empresa. **Llave primaria y foránea (no_emp)**
  - Eliminar los movimientos de un empleado que ya no trabaje en la misma. ***Borrado en cascada***

- **EMPLEADOS-NOMINA**
  - Esta definición garantiza que no se genere la nómina de empleados que no estén trabajando en la empresa. **Llave primaria y foránea (no_emp)**
  - Eliminar los registros de Nómina de un empleado que ya no trabaje en la misma. ***Borrado en cascada***

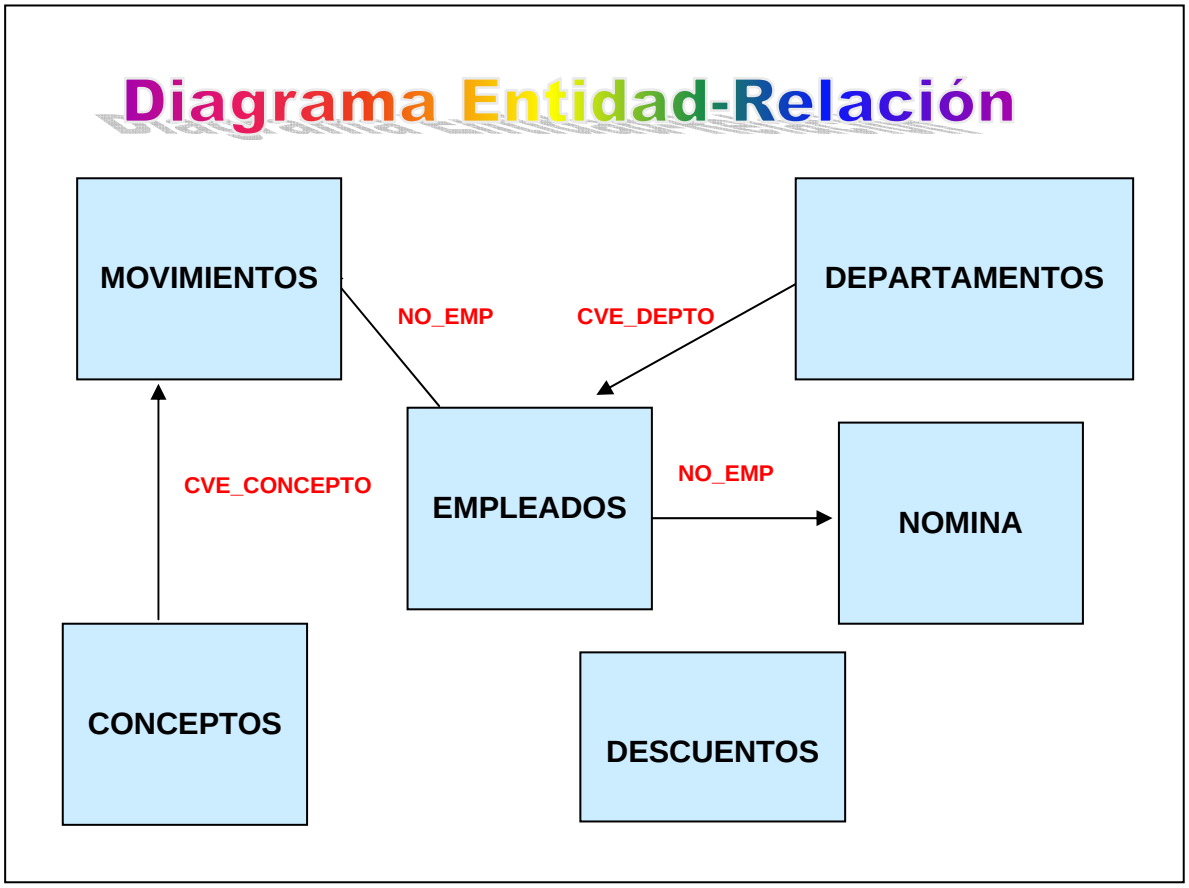


Figura 1.1 Diagrama Entidad-Relación

- **CONCEPTOS-MOVIMIENTOS**
  - Esta definición garantiza que no se generen movimientos de conceptos de deducción o percepción inexistentes. **Llave primaria y foránea (cve_concepto)**
  - No aplica el Borrado en cascada, por que si un concepto desaparece en alguna fecha del año, debe mantenerse como registro histórico por todo el año.

- **EMPLEADOS-DEPARTAMENTOS**
  - Esta definición garantiza que siempre un empleado trabaje o esté asignado a un departamento de la empresa. No hay contrataciones externas. **Llave primaria y foránea (cve_depto)**

- **MOVIMIENTOS**
  - Existe una llave compuesta por los campos (no_emp, no_quincena, cve_concepto). Con esto se garantiza que no se duplique la generación de movimientos por empleado, ni por quincena ni por concepto. Siempre habrá por empleado para dicha quincena un único concepto.

- **NOMINA**
  - Existe una llave compuesta por los campos (no_emp, no_quincena). Con esto se garantiza que sólo se generé un pago de Nómina por quincena. No se puede pagar más de una vez en una misma quincena al empleado.

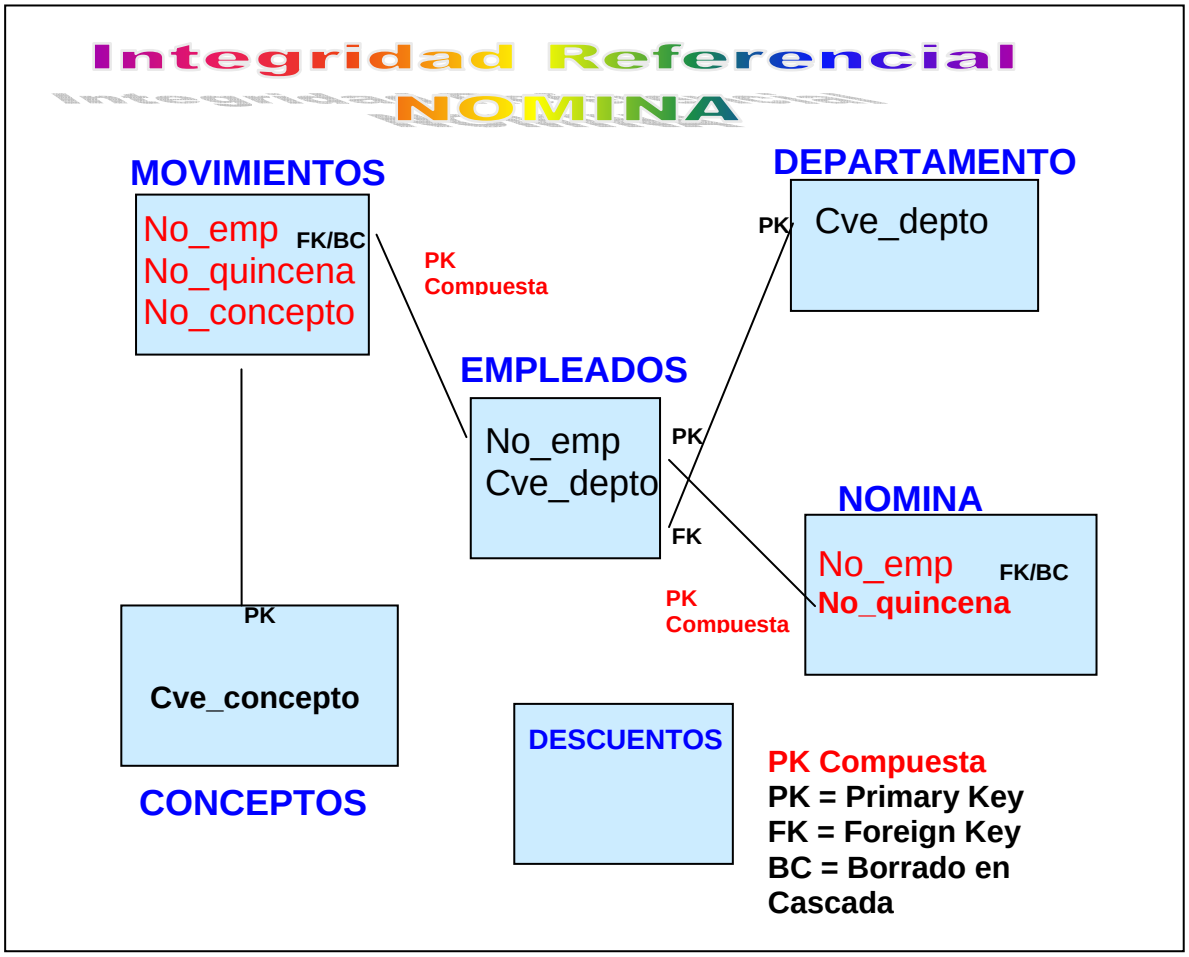


Figura 1.2 Diagrama Integridad Referencial

Diagrama de Flujo del Sistema

En el diagrama mostrado en la Figura 1.3, se muestra el flujo del Sistema de Nómina que sirve de **EJEMPLO** para este Manual de Prácticas.

Es importante hasta aquí, cojuntar todos los puntos anteriores, pues basados en ellos, se implementarán los queries de varios de los temas y subtemas del curso, así como las prácticas que se dejen al estudiante realizar por su propia cuenta y como complemento de este manual.

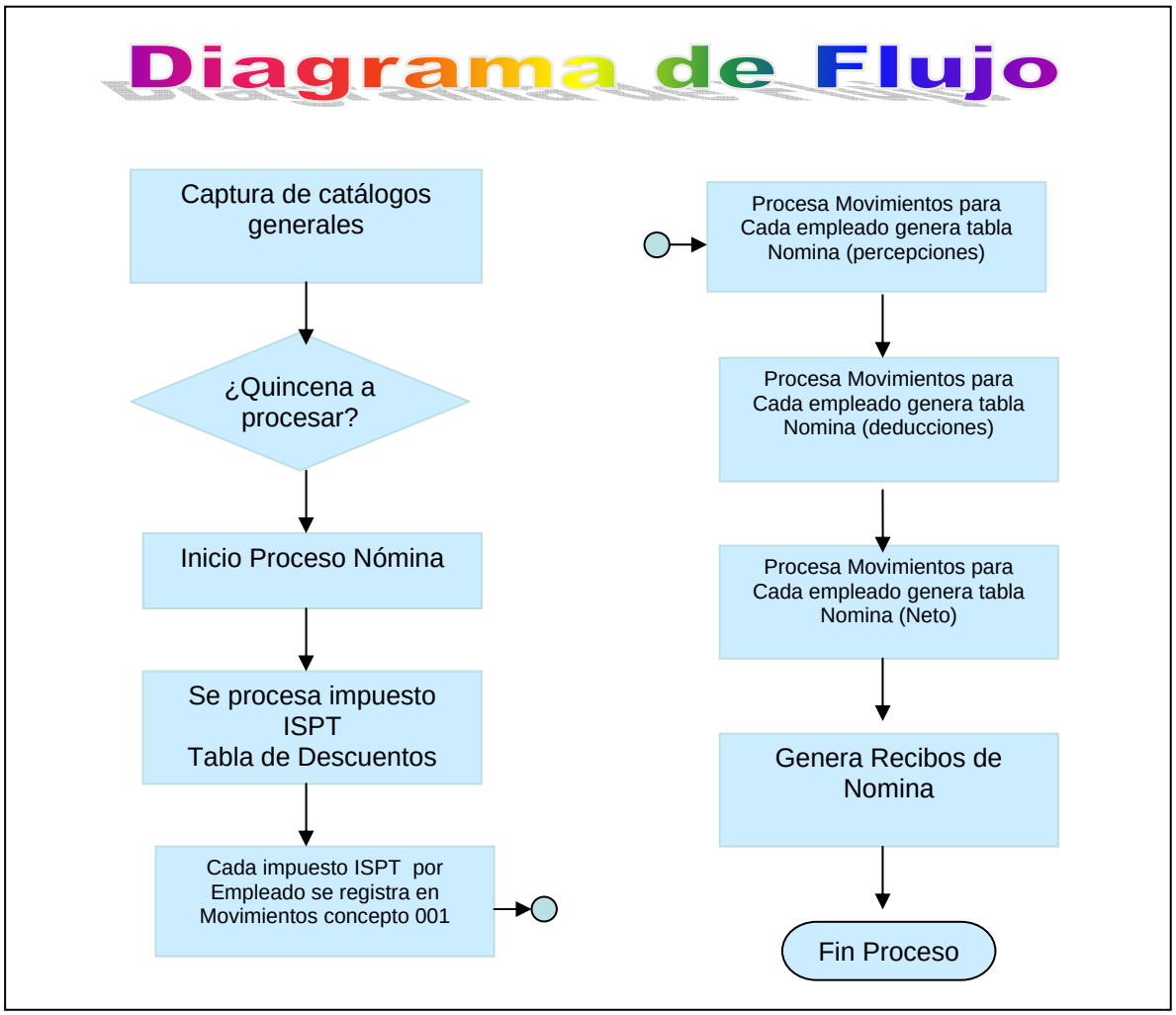


Figura 1.3 Diagrama de Flujo

Base de Datos de NOMINA, EJEMPLO para las prácticas de Select

A continuación se muestra el contenido de datos de cada tabla. Estos datos sirven para el mejor entendimiento de las Prácticas 10 a la 15, que son de acceso a la base de datos.

Es muy importante entender y saber el contenido de las bases y tablas, pues en base a esto, los queries se realizan de manera más sencilla, ya que se ha entendido el flujo del sistema y el contenido de las bases, el resto será lógica aplicada.

CONCEPTOS

cve_concepto	descripcion	tipo_concepto
001	ISR	D
002	Bono	P
003	Seguro de vida	D
004	Aguinaldo	P
005	Prestamo Auto	D
006	Seguro de vida	D
007	Seguro Social	D
008	Reparto Utilidades	P
009	Sueldo Base	P

DEPARTAMENTOS

cve_depto	nombre
D02	Mantenimiento
D01	Sistemas
D03	Personal
D04	Compras

DESCUENTOS

lim_inferior	lim_superior	porcentaje_descto
0	3000	20
3001	6000	25
6001	9000	30
9001	50000	40

NOMINA

no_emp	no_quincena	percepciones	deducciones	ingreso_neto
00001	1	6500.000000000	1425.000000000	5075.000000000
00001	2	6300.000000000	1945.000000000	4355.000000000
00002	1	4400.000000000	4400.000000000	0.00
00002	2	4400.000000000	4400.000000000	0.00
00003	1	11500.00000000	1250.000000000	10250.00000000
00003	2	11800.00000000	1270.000000000	10530.00000000
00004	1	5500.000000000	1375.000000000	4125.000000000
00004	2	5500.000000000	1375.000000000	4125.000000000
00005	1	4400.000000000	1100.000000000	3300.000000000
00005	2	4400.000000000	1100.000000000	3300.000000000
00006	1	11000.00000000	4400.000000000	6600.000000000
00006	2	11000.00000000	4400.000000000	6600.000000000
00007	1	5500.000000000	1375.000000000	4125.000000000
00007	2	5500.000000000	1375.000000000	4125.000000000
00008	1	4400.000000000	1100.000000000	3300.000000000

MOVIMIENTOS

no_emp	no_quincena	fecha_mov	cve_concepto	cantidad
00001	1	01/04/2005	002	1000.000000000
00001	1	01/03/2005	003	50.00000000000
00001	2	01/23/2005	002	800.0000000000
00001	2	01/24/2005	003	570.0000000000
00003	1	01/05/2005	002	500.0000000000
00003	1	01/04/2005	003	150.0000000000
00003	2	01/22/2005	002	800.0000000000
00003	2	01/24/2005	003	170.0000000000
00001	1	01/15/2005	009	5500.000000000
00002	1	01/15/2005	009	4400.000000000
00003	1	01/15/2005	009	11000.00000000
00004	1	01/15/2005	009	5500.000000000
00005	1	01/15/2005	009	4400.000000000
00006	1	01/15/2005	009	11000.00000000
00007	1	01/15/2005	009	5500.000000000
00008	1	01/15/2005	009	4400.000000000
00009	1	01/15/2005	009	11000.00000000
00001	2	01/30/2005	009	5500.000000000
00002	2	01/30/2005	009	4400.000000000
00003	2	01/30/2005	009	11000.00000000
00004	2	01/30/2005	009	5500.000000000
00005	2	01/30/2005	009	4400.000000000
00006	2	01/30/2005	009	11000.00000000
00007	2	01/30/2005	009	5500.000000000
00008	2	01/30/2005	009	4400.000000000
00009	2	01/30/2005	009	11000.00000000
00001	1	01/15/2005	001	1375.000000000
00001	2	01/30/2005	001	1375.000000000
00002	1	01/15/2005	001	4400.000000000
00002	2	01/30/2005	001	4400.000000000
00003	1	01/15/2005	001	1100.000000000
00003	2	01/30/2005	001	1100.000000000
00004	1	01/15/2005	001	1375.000000000
00004	2	01/30/2005	001	1375.000000000
00005	1	01/15/2005	001	1100.000000000
00005	2	01/30/2005	001	1100.000000000
00006	1	01/15/2005	001	4400.000000000
00006	2	01/30/2005	001	4400.000000000
00007	2	01/30/2005	001	1375.000000000
00007	1	01/15/2005	001	1375.000000000
00008	1	01/15/2005	001	1100.000000000
00008	2	01/30/2005	001	1100.000000000
00009	2	01/30/2005	001	4400.000000000
00009	1	01/15/2005	001	4400.000000000

EMPLEADOS

no_emp	00001	00002	00003	00004	00005
nombre	Fanny	Pedro	Guillermo	Claudia	Jaime
apellido_p	Garcia	Jimenez	Martinez	Reyes	Lopez
apellido_m	Perez	Nava	Mipa	Perez	Diaz
rfc	GAPF850501	JINP198009	MAMG500524	REPC700622	LODJ750909
fecha_nac	05/01/1985	09/22/1980	05/24/1950	06/22/1970	09/09/1975
fecha_ingreso	11/01/2005	11/01/2004	11/21/2005	01/01/2005	07/15/2004
ciudad	Metepec	Toluca	Toluca	Chihuaha	Toluca
sexo	F	M	M	F	M
cve_depto	D01	D01	D01	D02	D02
sueldo_base	5500	4400	11000	5500	4400

no_emp	00006	00007	00008	00009
nombre	Patricia	Boby	Rocio	Malu
apellido_p	Garcia	Garcia	Rios	Sanchez
apellido_m	Gonzalez	Lopez	Monroy	Paz
rfc	GAGP690707	GALB850808	RIMR630222	SAPM721908
fecha_nac	08/01/1969	08/08/1985	02/22/1963	08/19/1972
fecha_ingreso	11/21/2005	09/01/2005	12/15/2004	09/01/2005
ciudad	Toluca	Metepec	Chihuaha	Toluca
sexo	F	M	F	F
cve_depto	D02	D03	D04	D01
sueldo_base	11000	5500	4400	11000

Práctica 1
Sistema Manejador de Base de Datos

Objetivos

- Aprender la instalación de un Sistema Manejador de Base de Datos
- Aprender la configuración de un Sistema Manejador de Base de Datos
- Aprender los comandos importantes de administración de un Sistema Manejador de Base de Datos

Introducción

Para que un sistema trabaje de manera óptima es importante tener una buena instalación y configuración del Manejador de Base de Datos. Es importante realizar la configuración de acuerdo a las necesidades de espacio en disco, memoria y procesador que el tipo de sistema vaya a requerir. Es por eso que en esta práctica se darán paso a paso los detalles de instalación del Manejador.

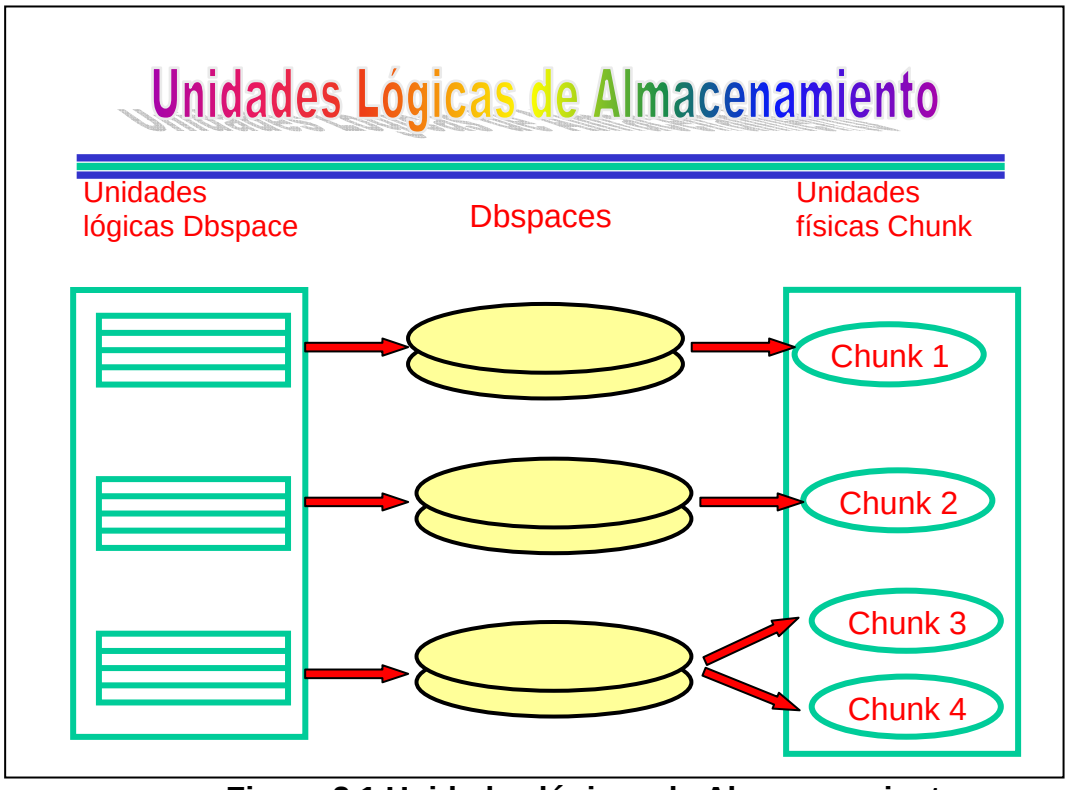


Figura 2.1 Unidades lógicas de Almacenamiento

Esta es información que el DBA debe manejar perfectamente, por eso es que sus conocimientos en el sistema operativo a utilizar y del manejador de base de datos, así como de los sistemas a

desarrollar son importantes. Los conceptos más importantes vistos en clase son los siguientes y se muestran gráficamente para su mejor entendimiento en la Figura 2.1 y la Figura 2.2.

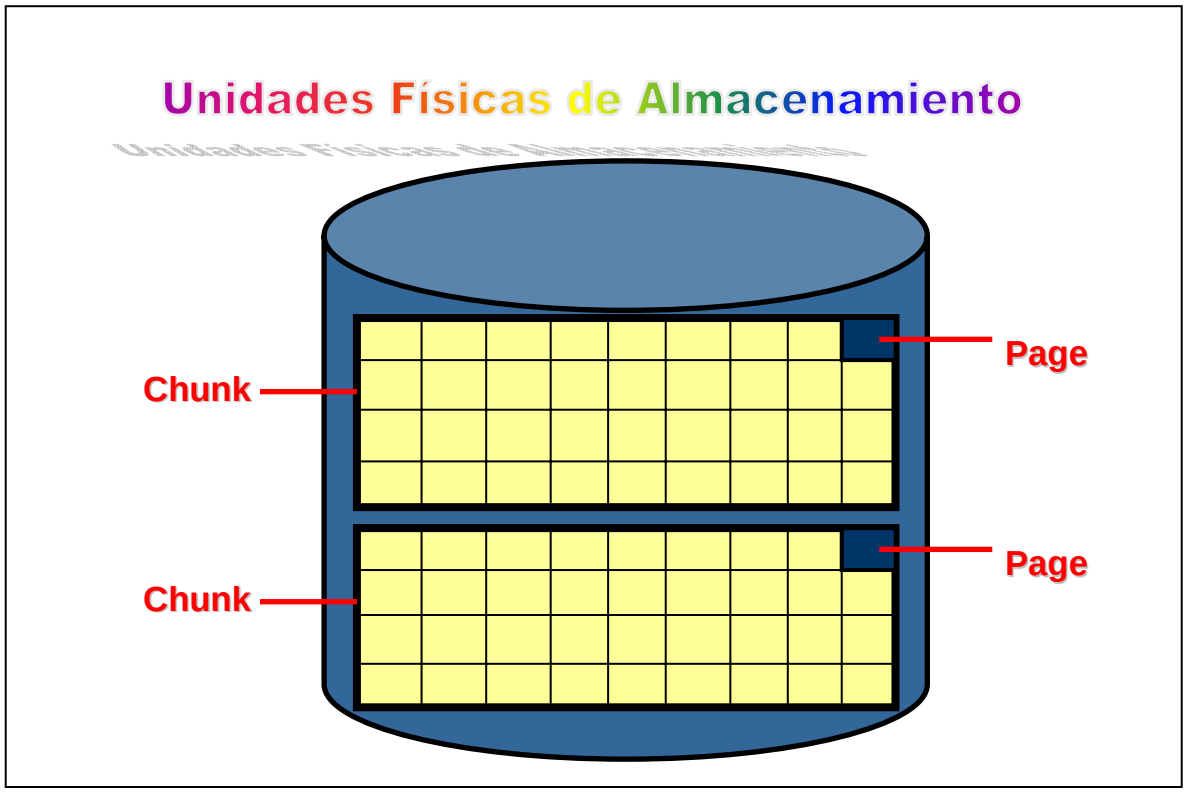


Figura 2.2 Unidades Físicas de Almacenamiento

Tema y subtema relacionado a cubrir

Tema: **Introducción al Sistema Manejador de Base de Datos(DBMS)**
Subtema: **Conceptos**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos
2. Software del Manejador de Base de Datos. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)

Metodología

Instalar Informix y configurar una instancia con las siguientes características:

Nombre de la instancia o servidor de base de datos: tbd
Rootdbs = 50 MB
Log lógicos = 6 de 3 Mb c/u

Log Físico siempre será el 20% del tamaño del rootdbs

Nota: ***Recuerda tener ya configurada la Dirección IP y el nombre del servidor. Probar que funcione dando ping al nombre del servidor.***

Los siguientes pasos deben realizarse conectado con el usuario root.

1. **Bajar el software del CD**

mkdir /opt/sw
mount /mnt/cdrom
Cambiarse al directorio donde está el software de Informix y copiarlo al directorio sw (Esta ubicación depende de cada quien, así como el formato del software)
cp DSA*.rpm /opt/sw

2. **Crear usuario y grupo de informix con id > 250 y home /opt/informix**

useradd informix –d /opt/informix
passwd informix (asignar password)

3. **Desempacar el software y asignar variables de ambiente (sólo INFORMIXDIR Y PATH para esta parte de la instalación)**

export INFORMIXDIR=/opt/informix
export PATH=$PATH:$INFORMIXDIR/bin

cd /opt/sw
rpm -i DSA*.rpm (nombre del paquete)
cd /opt/informix
#./installserver
Aquí se bajan los paquetes, debiendo dar número de serie y la llave.

4. **Habilitar servicio de Informix**

vi /etc/services
Agregar una línea con el servicio . **Ejercicio** tbdtcp 1527/tcp
y grabar el archivo.
Nota: siempre el nombre del servicio relacionado con el de la instancia.

4. **Asignar las variables de ambiente al usuario de informix recien creado.**

Editar el archivo profile del home del usuario informix y agregar las cuatro variables.

vi /opt/informix/.bash_profile

Agregar las siguientes líneas que son las variables de ambiente.
export INFORMIXDIR=/opt/informix
export PATH=$PATH:$INFORMIXDIR/bin
export INFORMIXSERVER=tbd
export ONCONFIG=onconfig.tbd

Nota: siempre el nombre debe ir relacionado con el de la instancia.

se graban los cambios y se conecta con el usuario informix

su – informix
Los siguientes pasos deben realizarse conectado con el usuario informix.

$ Verificar variables de ambiente y directorio, que sean las que configuramos previamente.
$ pwd
$ env

5. Crear directorio de respaldos y de datos

$mkdir respaldos
$ mkdir datos
$mkdir respaldos/tbd
$ mkdir datos/tbd

Nota: siempre el nombre debe ir relacionado con el de la instancia.

6. Creando archivos de respaldo

$ touch respaldos/tbd/respdatos
$ touch respaldos/tbd/resplogs

$ chmod 660 respaldos/tbd/respdatos
$ chmod 660 respaldos/tbd/resplogs

7. Creando chunks

$ touch datos/tbd/chrootdbs
$ chmod 660 datos/tbd/chrootdbs

8. Preparando y copiando archivos de configuración e instancia de Informix

$ cd etc
$ cp onconfig.std onconfig.tbd

Nota: siempre el nombre debe ir relacionado con el de la instancia.

Editar archivo de configuración.

$ vi sqlhosts
Agregar la linea de la instancia
tbd onsoctcp <nombre del host unix> tbdtcp

grabar archivo e iniciar la configuración.

Nota: siempre el nombre debe ir relacionado con el de la instancia.

9. Iniciando configuración e instancia de Informix

$ onmonitor (Completar proceso de acuerdo a lo anterior y a los visto en clase)

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL. No se recomienda aprenderse los pasos de memoria sino que se entienda que se hace en cada uno. El tiempo máximo de instalación y configuración de una instancia debe ser de 30 minutos, por lo que se sugiere la Práctica continua de este procedimiento.

Reporte del estudiante

El estudiante debe entregar un reporte de cómo quedo instalada su instancia, con las siguientes características:

Nombre de la instancia o servidor de base de datos: tbd
Rootdbs = 50 MB
Log lógicos = 6 de 3 Mb c/u
Log Físico siempre será el 20% del tamaño del rootdbs

1. ¿Cómo quedaron configurados los dbspaces?.
2. ¿Cómo quedaron configurados los logs?.
3. ¿Cómo quedaron configurados la información del archivo online.tbd y onconfig.tbd?. Explicar el detalle de que cambios fueron afectados en estos archivos y la importancia de ellos.
4. Explicar cada opción del manejo de onmonitor.
5. Para que sirven los archivos services, sqlhosts y cada una de las variables de ambiente.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 2
Manejo de SQL

Objetivos

- Aprender el uso de SQL y sus opciones de creación de queries
- Aprender los comandos generales de administración de SQL para el manejador de base de Datos correspondiente
- Conocer el uso del **dbaccess**, utilería de Informix para el acceso a SQL.

Introducción

El éxito de los sistemas de Base de Datos desarrollados es el buen manejo e implementación de los queries o sentencias SQL, es por eso que el estudiante debe conocer todas las herramientas que le permitan desarrollar queries óptimos, por lo que en esta práctica aprenderá las diferentes opciones del manejo de SQL.

Tema y subtema relacionado a cubrir

Tema: **Introducción al Sistema Manejador de Base de Datos(DBMS)**
Subtema: **Características del Manejador de Base de Datos (DBMS)**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)

Metodología

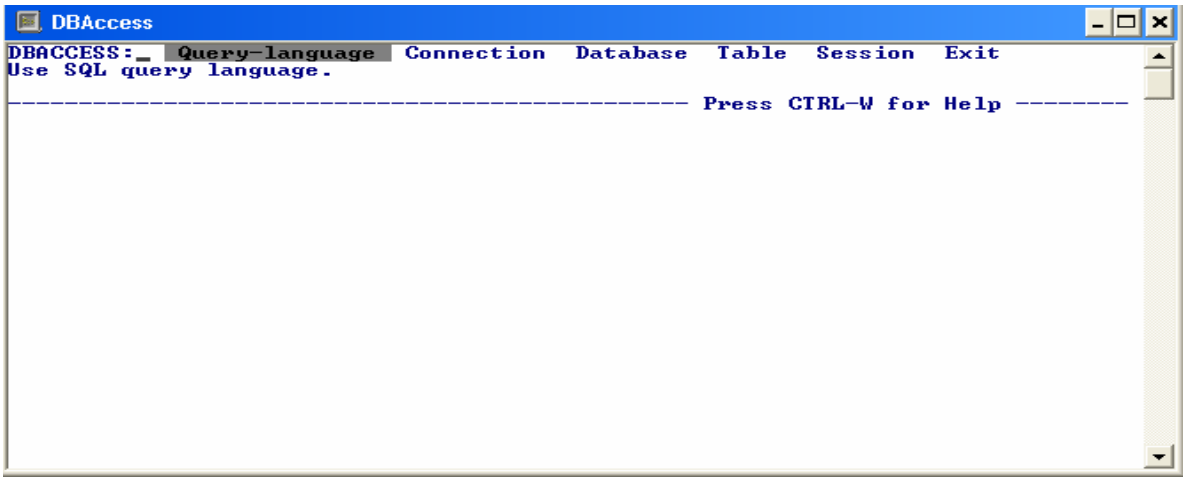
Con Informix instalado y en línea, y conectado como usuario informix, se navegará por todas las opciones de dbaccess y con la ayuda en línea se aprenderá que se hace en cada opción.

Nota: ***Recuerda tener ya configurada la Dirección IP y el nombre del servidor. Probar que funcione dando ping al nombre del servidor.***

1. Accesar SQL a través de Dbaccess o del monitor de SQL correspondiente al DBMS

Conectarse con el usuario informix y accesar dbaccess. Es importante verificar que las variables de ambiente estén levantadas así como el manejador de base de datos. En esta parte el estudiante ha accesado el menú para poder iniciar el uso de SQL.

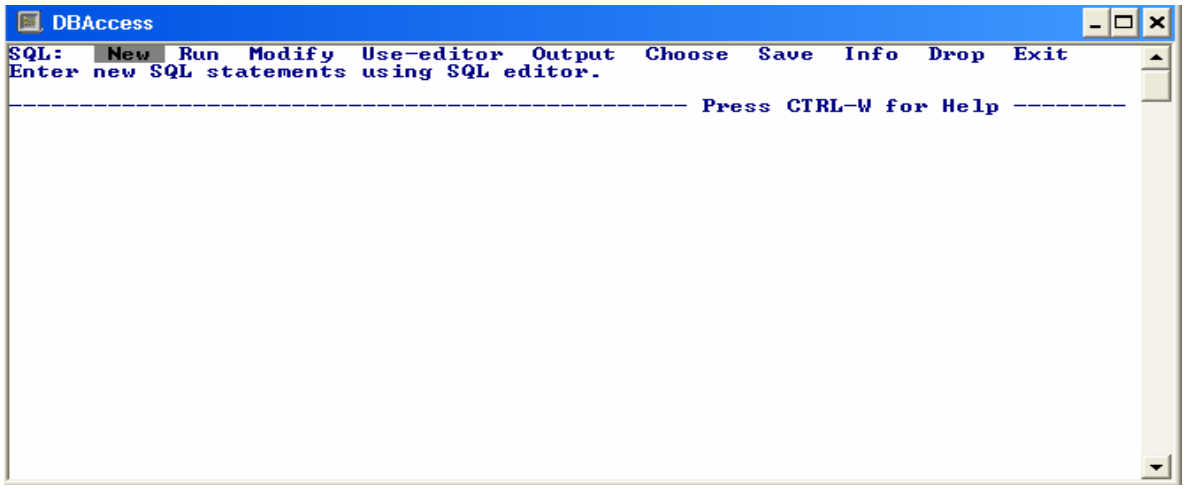
$ dbaccess



Nota: Las opciones se pueden accesar con las flechas o con la letra mayúscula que se indica en cada opción.

2. Accesar las opciones de Query-Language

La única base existente es la Sysmaster. Esta base es la del manejador de base de datos. No debe ser modificada ni eliminada. Así como la instancia es la de tbd, que es sobre la que se trabajará en todas las prácticas (sysmaster@tbd, indicando base e instancia).

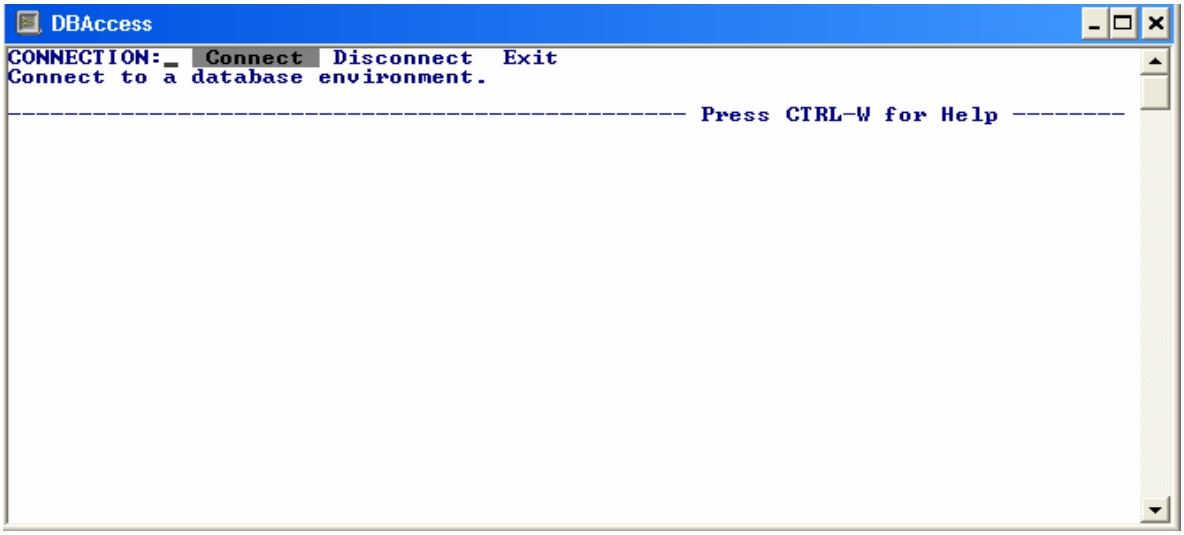


En esta parte el estudiante accesa cada opción, manejando el editor, grabando queries y accesándolos.

Aquí en esta opción se trabajará en todo el curso, creándose las bases, tablas, llaves, índices, carga de datos y queries en general. La creación será manual y no automática, con esto se asegura el aprendizaje del estudiante. Recuerda que una vez que se tiene el query hecho debe ejecutarse con **RUN.**

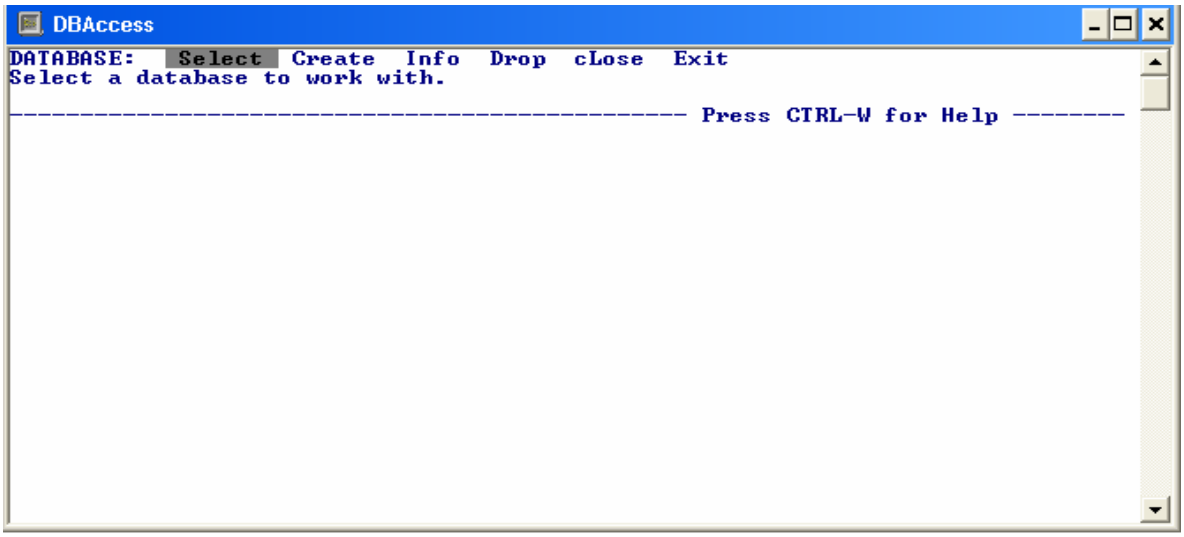
3. Accesar las opciones de Connection

En esta parte se el usuario puede conectarse o desconcectarse a una instancia determinada, en el caso que existiera más de una, se ven todas las instancias existentes dentro del servidor de base de datos.



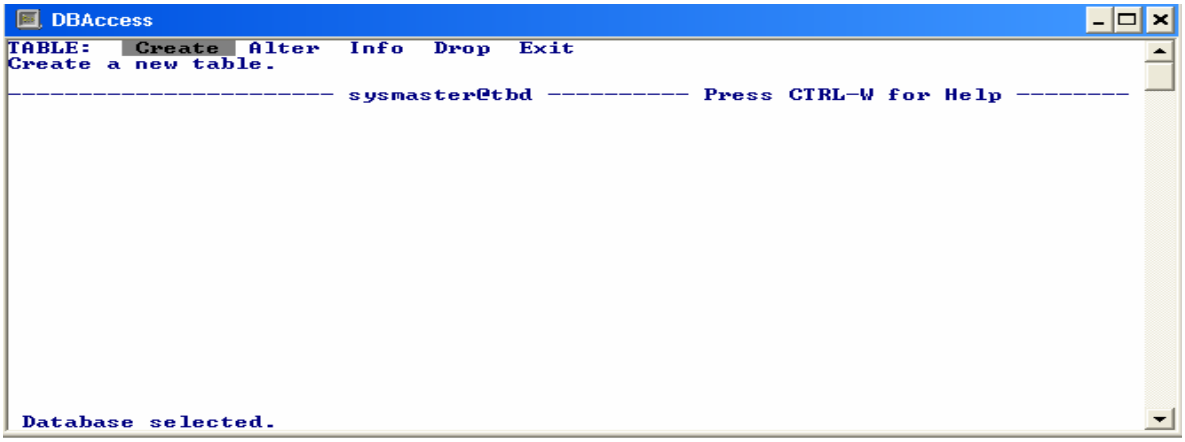
4. Accesar las opciones de Database

En esta parte el estudiante puede crear de manera amigable a través de menús diferentes tipos de bases de datos, así como decidir en que dbspace se alojará la base de datos. Podrá seleccionar una base, eliminarla, desactivarla o ver la información general referente a ella. No se crearán en el curso las bases con esta opción.



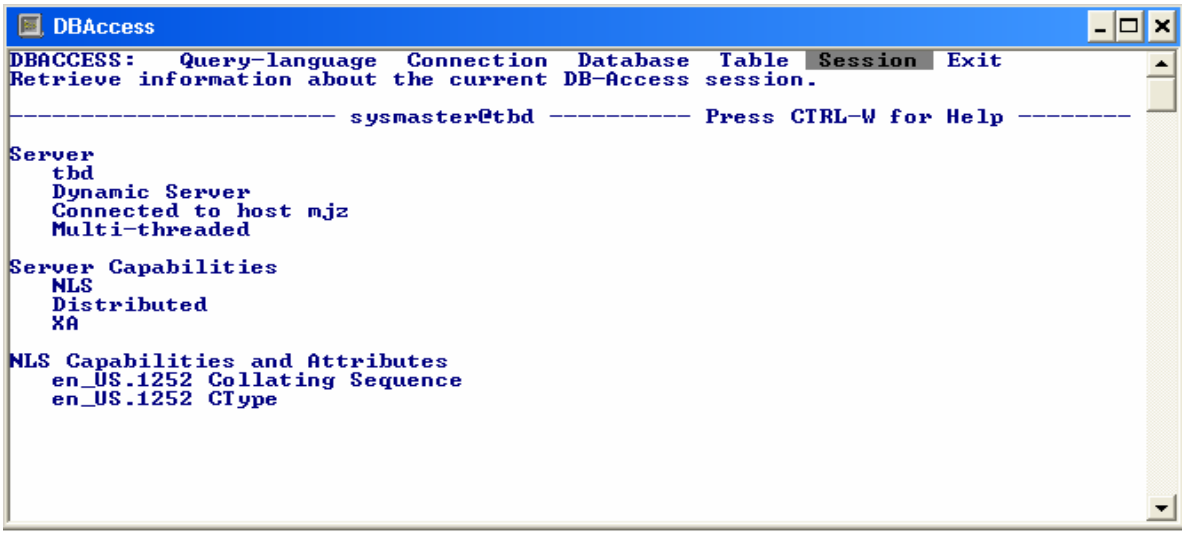
5. Accesar las opciones de Table

En esta parte el estudiante puede crear de manera amigable a través de menús las tablas de base de datos, campos y las características necesarias para cada una. Podrá seleccionar una tabla, cambiar su definición inicial, eliminarla o ver la información general referente a ella. No se crearán en el curso las tablas con esta opción.



6. Accesar las opciones de Session

En esta opción se ven las características generales del manejador de base de datos, la instancia en la que estamos conectados, servidor de sistema operativo.



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL. Recorrer cada una de las opciones del menú.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. El estudiante entregará un reporte de las funciones generales del menú, así como cada una de las opciones de cada uno, indicado su uso.

2. Para la base de Datos sysmaster, verificar de cuantas tablas está conformada.
3. Verificar las características de cinco de estas tablas.
4. Define el esquema Diagrama Entidad_relación de cinco de estas tablas.
5. Describe como está definida la integridad Referencial (Primary y Foreign Keys) de estas tablas.
6. Describe como está definida la Consistencia de datos (Checks, Defaults) de estas tablas.
7. Para estas tablas verifica la información general, en cuanto a tamaño de registro, cantidad de registros, etc. Esta información puede ser obtenida con la opción de *Status* del menú >Table.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 3
Creación de Base de Datos y Tablas en SQL

Objetivos

- Crear Bases de Datos con la herramienta SQL
- Crear tablas con la herramienta SQL
- Implementar dentro de la creación el concepto de Integridad Referencial
- Implementar dentro de la creación el concepto de Borrado en cascada

Introducción

La importancia del estatuto CREATE es la base para la creación de todos los objetos de la base como: bases, tablas, índices, etc.

SINTAXIS

CREATE DATABASE database-name [IN DBspace-name]

```
CREATE [TEMP] TABLE table-name
    (
    column-name
       {
       datatype
       | {BYTE|TEXT} [IN {TABLE | BLOBspace-name]}]
       }
       [DEFAULT default_opts]
       [table-constraint-definition][,...]
       [column-constraint-definition][,...]
       [NOT NULL] [UNIQUE [CONSTRAINT constr-name]][,...]
       [UNIQUE (unique-column-list) [CONSTRAINT constr-name]][,...]
    )
    [WITH NO LOG]
    [IN DBspace-name]
```

Tema y subtema relacionado a cubrir

Tema: **Lenguaje de Definición de Datos (DDL)**
Subtema: **Creación de Base de Datos**

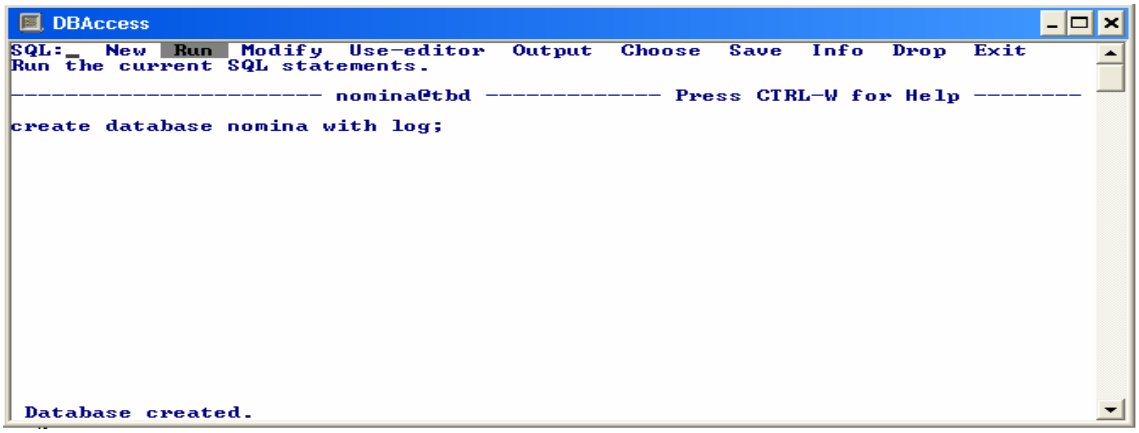
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.

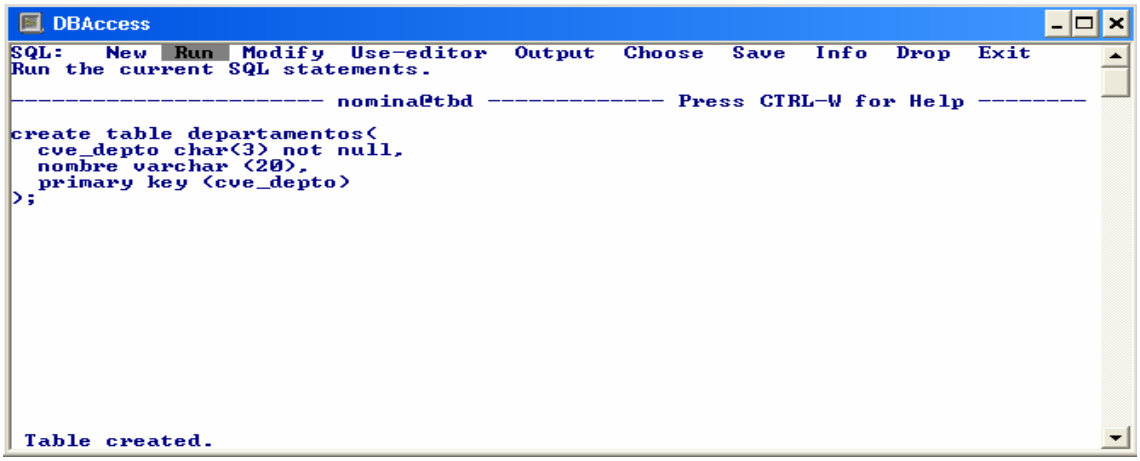
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

Metodología

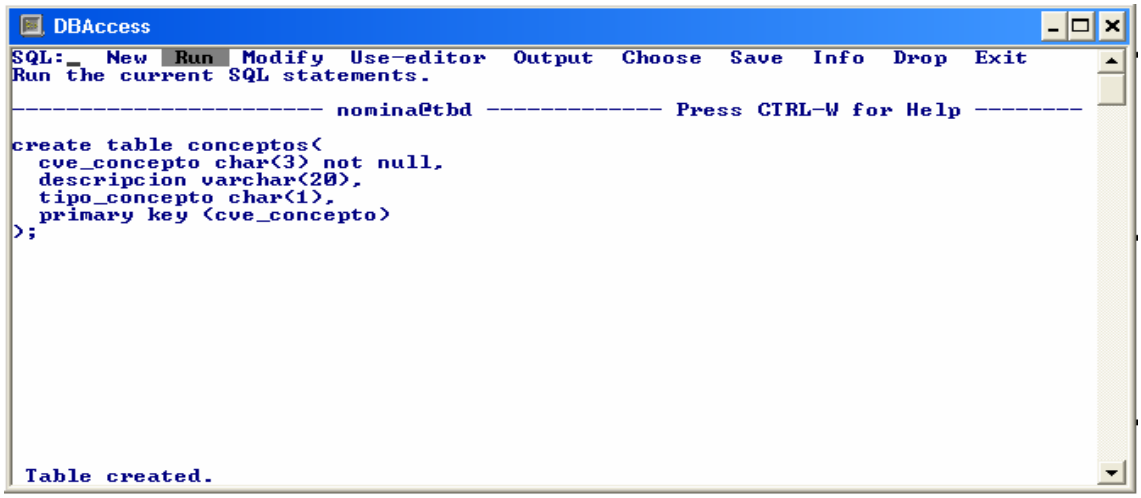
Ejercicio 1. Crear la base de Datos de NOMINA considerando borrado en cascada.



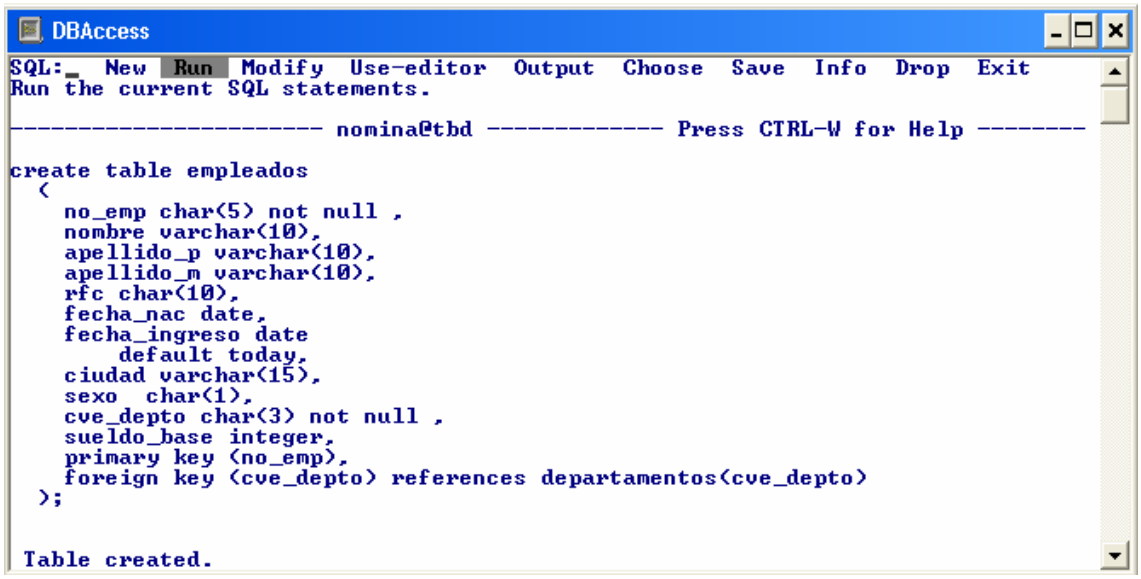
Ejercicio 2. Crear la tabla de DEPARTAMENTOS considerando integridad referencial.



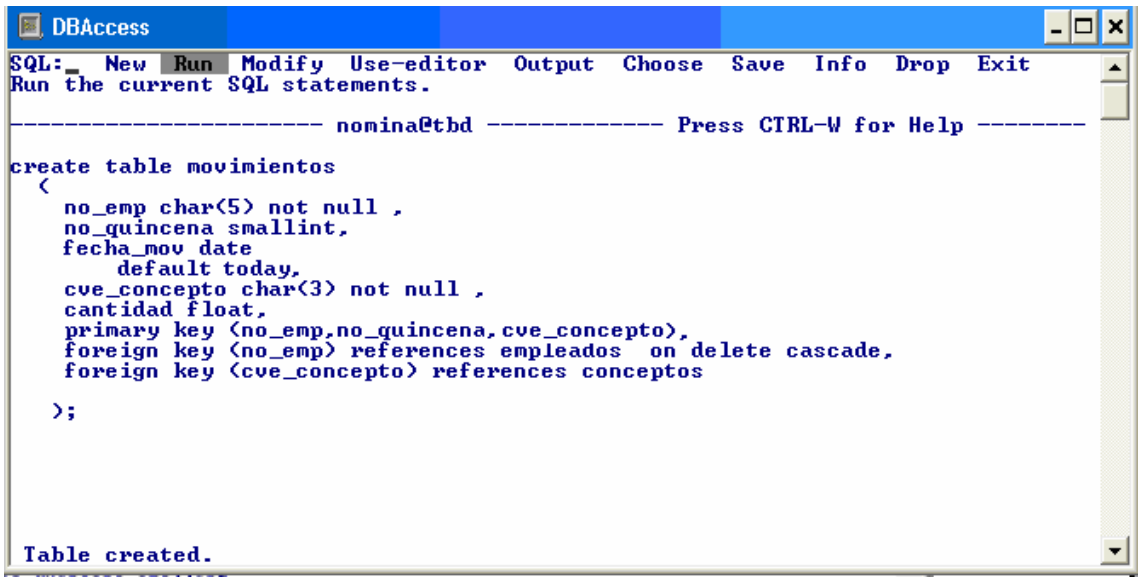
Ejercicio 3. Crear la tabla de CONCEPTOS considerando integridad referencial.



Ejercicio 4. Crear la tabla de EMPLEADOS considerando integridad referencial.



Ejercicio 5. Crear la tabla de MOVIMIENTOS considerando integridad referencial, borrado en cascada y consistencia de datos.



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama Entidad-Relación, para identificar llaves, borrados en cascada.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries de creación y ejecutarlos con la opción de **RUN.**

Crear un directorio de trabajo diferente al HOME del software del manejador de base de datos, en donde se almacenen los queries que se van desarrollando. Recuerda que a los queries automáticamente Informix les asigna la extensión .sql.

Es importante crear primero aquellas tablas que no estén relacionadas.

No olvidar las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade).

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Para qué sirve y cómo se define la integridad referencial de la base de datos?.
2. ¿Para qué sirve y cómo se define el borrado en cascada?.
3. ¿Para qué sirve y cómo se define la consistencia de datos?.
4. Crear la tabla de DESCUENTOS considerando integridad referencial.
5. Crear la tabla de NOMINA considerando integridad referencial y borrado en cascada, consistencia de datos.
6. Generar un esquema final de la base de Datos completa incluyendo todas sus tablas. Puede utilizar el comando dbschema.
7. Entregar el reporte del esquema final y preguntas anteriores.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 4
Creación de Base de Datos y Tablas en SQL con validaciones

Objetivos

- Crear tablas implementando validaciones (Check)
- Crear tablas implementando valores por default (Default)

Introducción

La importancia del estatuto CREATE es la base para la creación de todos los objetos de la base como: Tablas, índices, constraints, checks, defaults.

SINTAXIS

CREATE DATABASE database-name [IN DBspace-name]

CREATE [TEMP] TABLE table-name
(
column-name
{
datatype
| {BYTE|TEXT} [IN {TABLE | BLOBspace-name]}]
}
[DEFAULT default_opts]
[table-constraint-definition][,...]
[column-constraint-definition][,...]
[NOT NULL] [UNIQUE [CONSTRAINT constr-name]][,...]
[UNIQUE (unique-column-list) [CONSTRAINT constr-name]][,...]
)
[WITH NO LOG]
[IN DBspace-name]

DROP DATABASE database-name

DROP INDEX index-name

DROP TABLE table-name

Tema y subtema relacionado a cubrir

Tema: **Lenguaje de Definición de Datos (DDL)**
Subtema: **Creación de Tablas**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.

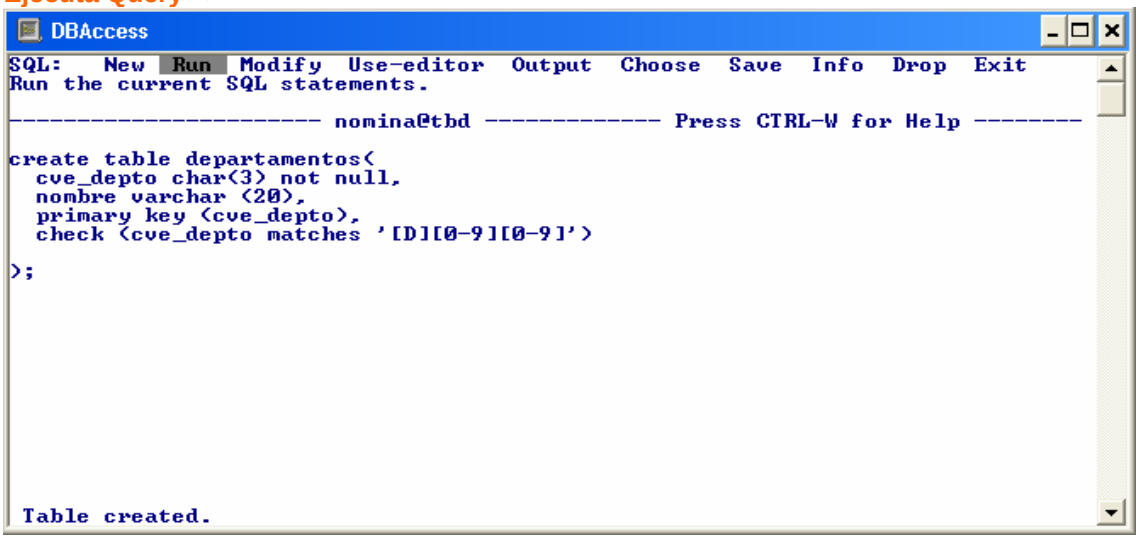
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

Metodología

*Debe utilizarse el estatuto DROP TABLE, para borrar y recrear la tabla para cada **Ejercicio,** debido a que fueron ya creadas en lal práctica anterior..*

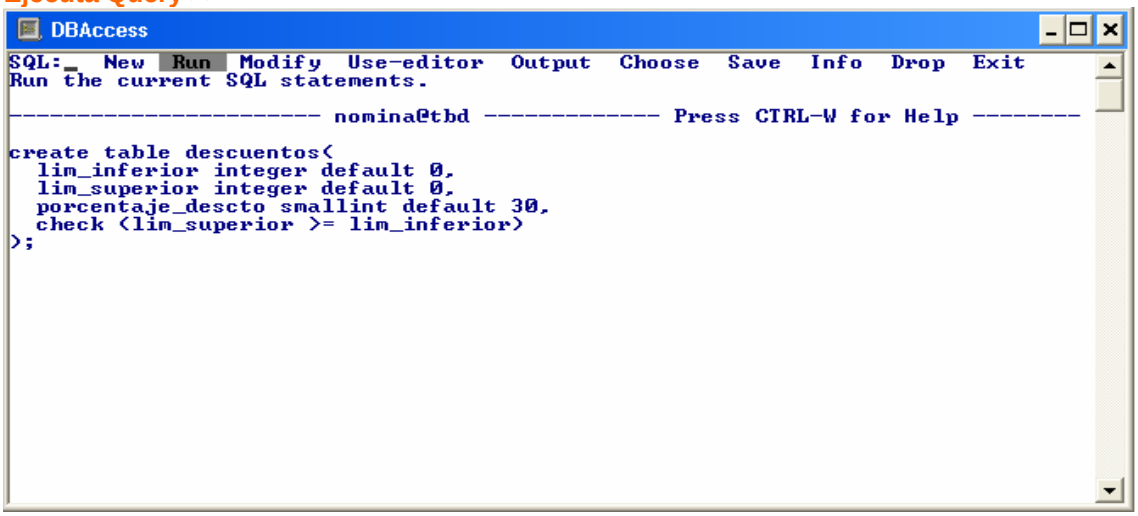
Ejercicio 1. Crear la tabla de DEPARTAMENTOS, ahora considerando los valores de check y default, manteniendo la definición integridad referencial.

Ejecuta Query >>



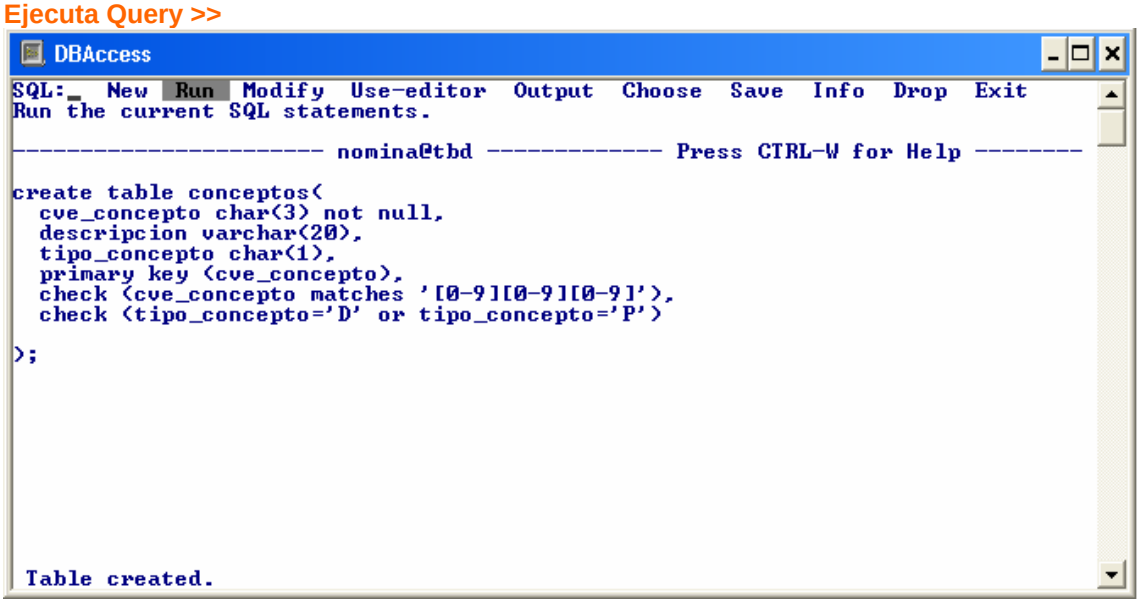
Ejercicio 2. Crear la tabla de DESCUENTOS, ahora considerando los valores de check y default, manteniendo la definición integridad referencial.

Ejecuta Query >>



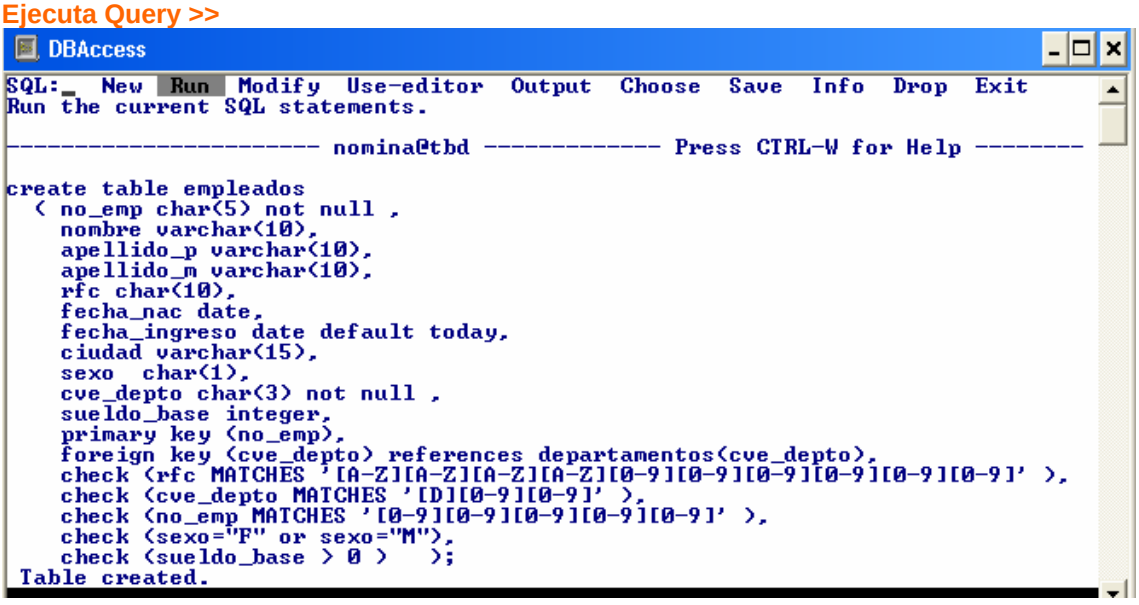
Ejercicio 3. Crear la tabla de CONCEPTOS, ahora considerando los valores de check y default, manteniendo la definición integridad referencial.

Ejecuta Query >>



Ejercicio 3. Crear la tabla de EMPLEADOS, ahora considerando los valores de check y default, manteniendo la definición integridad referencial.

Ejecuta Query >>



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama Entidad-Relación, para identificar llaves, borrados en cascada.

Es importante crear primero aquellas tablas que no estén relacionadas.

No olvidar las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade).

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries de creación y ejecutarlos con la opción de **RUN.**

Crear un directorio de trabajo diferente al HOME del software del manejador de base de datos, en donde se almacenen los queries que se van desarrollando. Recuerda que los queries automáticamente Informix les asigna la extensión .sql.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Para qué sirve y cómo se definen el CHECK para los diferentes tipos de datos?.
2. ¿Para qué sirve y cómo se definen los valores por DEFAULT para los diferentes tipos de datos?, incluir date y datetime.
3. Crear la tabla de MOVIMIENTOS considerando integridad referencial, validaciones de datos y valores por default.
4. Crear la tabla de NOMINA considerando integridad referencial, validaciones de datos y valores por default.
5. Generar un esquema final de la base de Datos completa incluyendo sus tablas. Puede utilizar el comando dbschema.
6. Entregar el reporte del esquema final y preguntas anteriores.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 5
Creación de Índices

Objetivos

- Entender el funcionamiento e importancia de los índices
- Crear tablas con índices para accesos más óptimos

Introducción

La importancia del estatuto CREATE es la base para la creación de todos los objetos de la base como: Tablas, ***índices***, constraints, checks, defaults. Se dan los estatutos de Create y Drop, debido a que aún no se ve como cambiar la estructura, por lo que si de desea cambiar una definición específica, deberá borrarse y crearse nuevamente.

Es importante entender que los índices ayudan a mejorar el tiempo de acceso de los datos de la base de datos. Por default cuando es creada una llave primaria, se crea un índice sobre ese campo. El índice CLUSTER es el único que altera el orden físico de los datos de la tabla, y sólo puede existir un índice CLUSTER por tabla. El resto de los índices que se crean, no alteran el orden físico de los datos. El índice CLUSTER es recomendado cuando se requiere que la tabla esté ordenada y el acceso sea en ese orden, mejorando el tiempo de acceso, sin embargo se debe justificar fuertemente la creación de este tipo de índices. El orden por default en la creación es ascendente. La desventaja del índice CLUSTER es que cada que se crea o se aplica, los datos son reordenados fisicamente por lo que el acceso a la tabla al inicio será lento, pero mientras ya no haya inserciones de datos el acceso será más rapido, pues ya no tiene que reordenar los datos que ya se encuentran en orden.

SINTAXIS

CREATE [UNIQUE|DISTINCT] [CLUSTER] INDEX index-name
ON table-name (column-name [ASC|DESC],...)

DROP INDEX index-name

Tema y subtema relacionado a cubrir

Tema: **Lenguaje de Definición de Datos (DDL)**
Subtema: **Creación de Índices**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

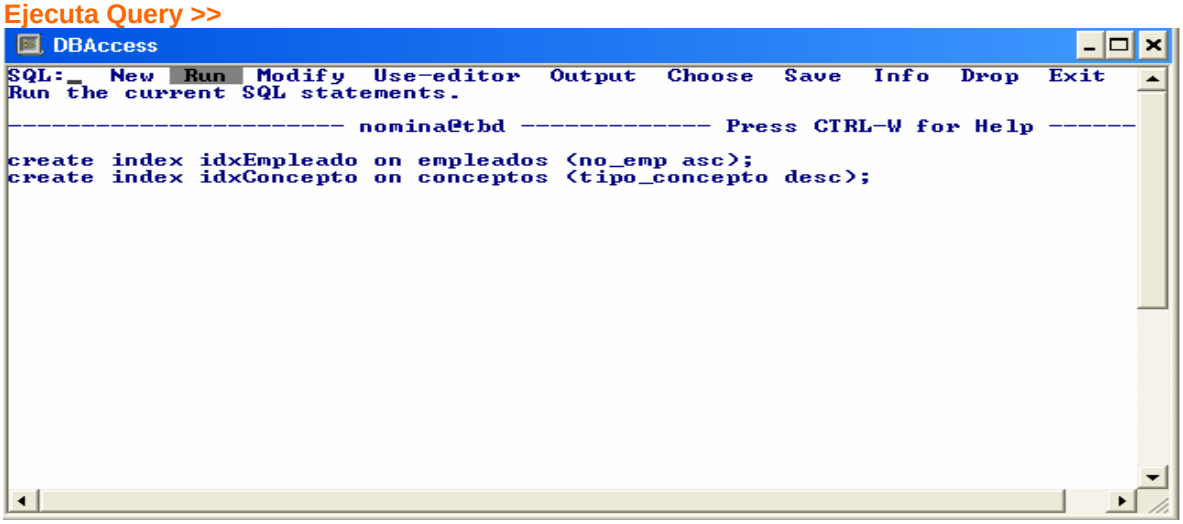
6. ***Sólo para esta práctica deberán insertarse previamente datos en la tabla de DEPARTAMENTOS. Se puede ver la práctica de INSERT para lograr este requerimiento.***

Metodología

Ejercicio 1. Creación de índices simples.

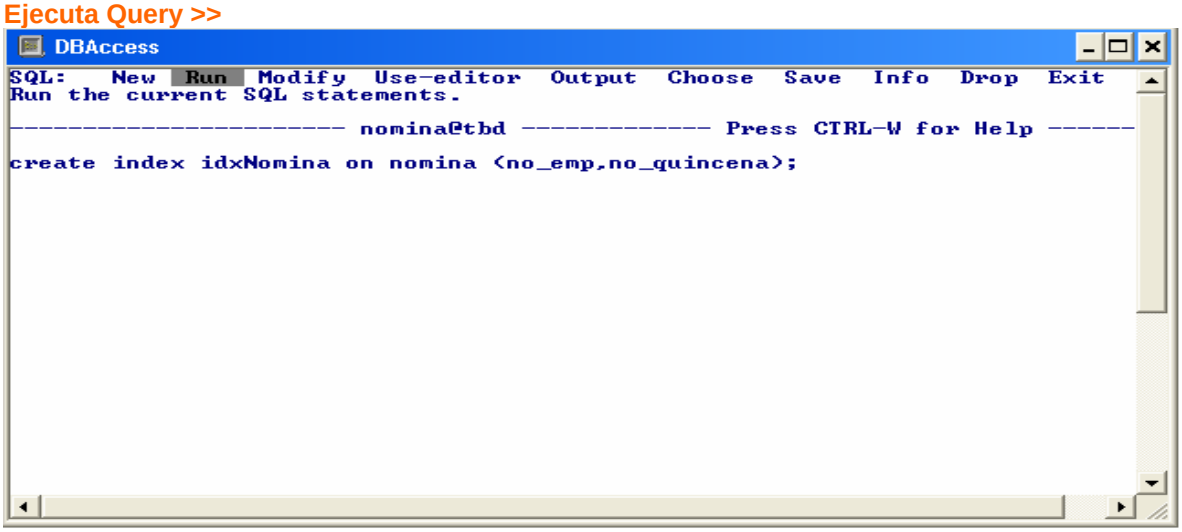
- **Insertar datos en la tabla DEPARTAMENTOS. Ver práctica 7 para Sintaxis.**
- **Crear un índice sobre el campo no_emp de la tabla de EMPLEADOS, el acceso sobre este campo normalmente es en orden ascendente.**
- **Crear un índice sobre el campo tipo_concepto de la tabla de CONCEPTOS, el acceso sobre este campo normalmente es en orden descendente.**

Ejecuta Query >>



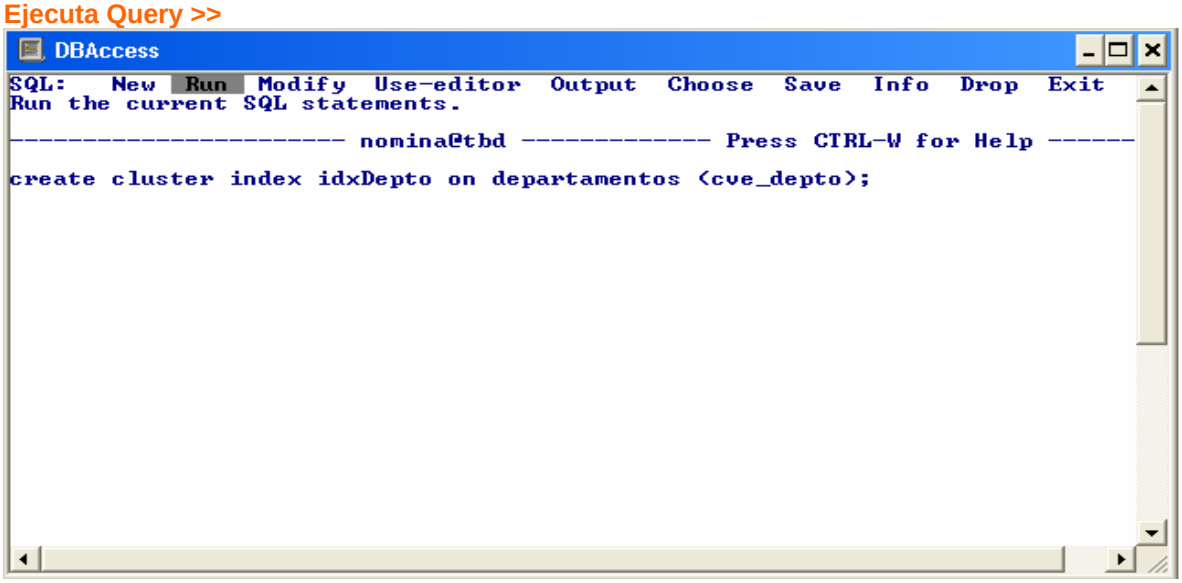
Ejercicio 2. Crear un índice compuesto sobre la tabla de NOMINA, ya que es necesario accesar por no_emp y no_quincena.

Ejecuta Query >>



Ejercicio 3. Crear un índice CLUSTER sobre la tabla de DEPARTAMENTOS por el campo de cve_depto. Observe el orden de los datos antes y después de la creación, puede utilizar el Query "SELECT cve_depto FROM departamentos".

Ejecuta Query >>



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama Entidad-Relación, para identificar llaves, borrados en cascada.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries de creación.

Es importante crear primero aquellas que no estén relacionadas.

No olvidar las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade).

Crear un directorio de trabajo diferente al HOME del software del manejador de base de datos, en donde se almacenen los queries que se van desarrollando. Recuerda que los queries automáticamente Informix les asigna la extensión .sql.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Qué problema detectaste cuando fueron creados los índices del **Ejercicio 1**?. ¿Cómo se resolvió?.
2. ¿Cómo se encontraban los datos de la tabla DEPARTAMENTOS antes de la creación del índice?. Muestra la información. Esto para el Ejercicio 3.

3. ¿Cómo se encontraban los datos de la tabla DEPARTAMENTOS después de la creación del índice?. Muestra la información. Esto para el Ejercicio 3.
4. Crear un índice CLUSTER por el campo nombre de la tabla de DEPARTAMENTOS. Generé el resultado de este Query.
5. Se observó algo en la creación del punto anterior. Explique, ¿Qué fue y porqué?.
6. ¿Cual es la importancia y uso de los índices?.
7. ¿Cuando se recomienda el uso de los índices?.
8. ¿En qué momento se usan los índices?.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 6
Modificación de la estructura de una tabla (ALTER)

Objetivos

- Modificar la estructura de una tabla existente
- Entender el grado de afectación a tablas existentes

Introducción

El estatuto ALTER debe ser manejado con mucha precaución, debido a que este estatuto modifica la estructura o definición de la base de datos, tablas, campos. Cuando la base se encuentra aún sin datos, el impacto puede no ser tan fuerte. Sin embargo cuando existe información, un cambio de tipo de datos; por ejemplo, puede destruir, borrar o alterar la información de la tabla. Por eso, esta instrucción debe ser utilizada con plena conciencia de lo que se necesita, por lo que se recomienda siempre realizar un respaldo de la base de datos antes de aplicarlo.

SINTAXIS

```
ALTER TABLE table-name
    {
    ADD
       ( new-column-name
          {
          datatype
          |
          {BYTE | TEXT} [IN {TABLE | BLOBspace-name}]
          }
          [DEFAULT default_opts]
          [NOT NULL]
          [UNIQUE [CONSTRAINT constr-name]][,...]
          [column-constraint-definition][,...]
          [BEFORE old-column-name][,...]
       )
    |
    DROP (old-column-name[,...])
    |
    MODIFY
       ( old-column-name new-datatype
          [DEFAULT default_opts]
          [NOT NULL]
          [UNIQUE [CONSTRAINT constr-name]][,...]
          [column-constraint-definition] [,...]
       )
    |
    ADD CONSTRAINT UNIQUE (old-column-name[,...])[table-constraint-definition]
    |
    DROP CONSTRAINT [owner] (constr-name[,...])}[,...]
    }
```

dbexport <database> [-X] [-c] [-q] [-d] [-ss]
[{ -o <dir> | -t <tapedev> -b <blksz> -s <tapesz> [-f <sql-command-file>] }
NOTE: arguments to dbexport are order independent.

dbimport <database> [-X] [-c] [-q] [-d <dbspace>]
[-l [{ buffered | <log-file> }] [-ansi]]
[{ -i <dir> | -t <tapedev> [-b <blksz> -s <tapesz>] [-f <script-file>] }]
NOTE: <log-file> must be a complete path
arguments to dbimport are order independent

Se puede realizar dicho respaldo con el comando dbexport/dbimport. Este es una instrucción que se ejecuta desde la línea de comandos y se muestra la sintaxis como apoyo al estudiante. La sintaxis muestra claramente las funciones que el ALTER puede realizar, como son: ADD, agregar campos nuevo, DROP eliminar campos existente, MODIFY modificar definiciones de campos existentes(aquí debe tener precaución cuando existen datos), ADD/DROP CONSTRAINT, agrega o elimina todo tipo de reglas, validaciones, llaves, índices, etc.

Tema y subtema relacionado a cubrir

Tema: **Lenguaje de Definición de Datos (DDL)**
Subtema: **Integridad**

Material y equipo necesario

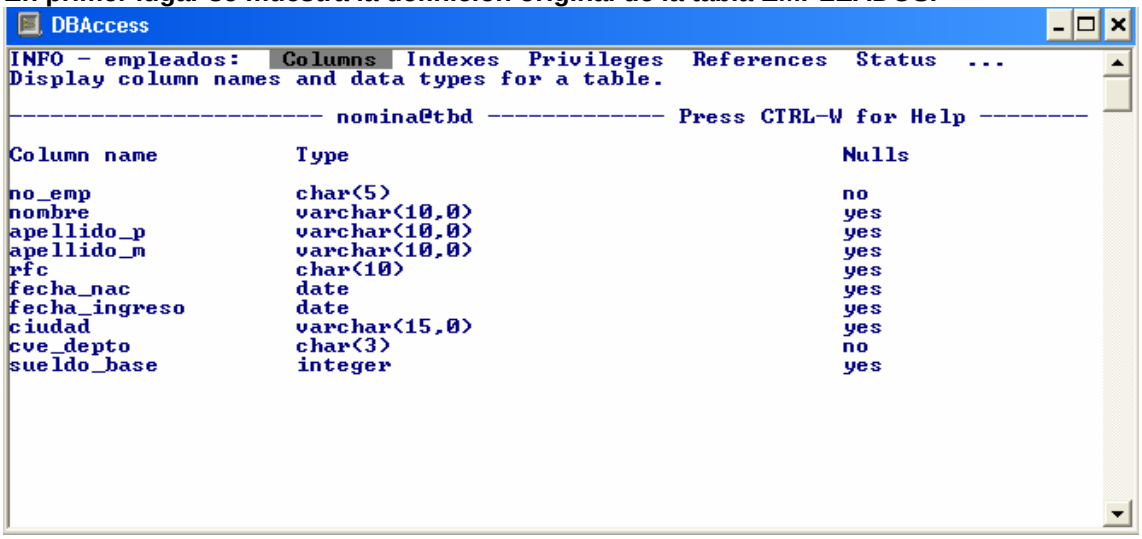
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con el esquema de la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

Metodología

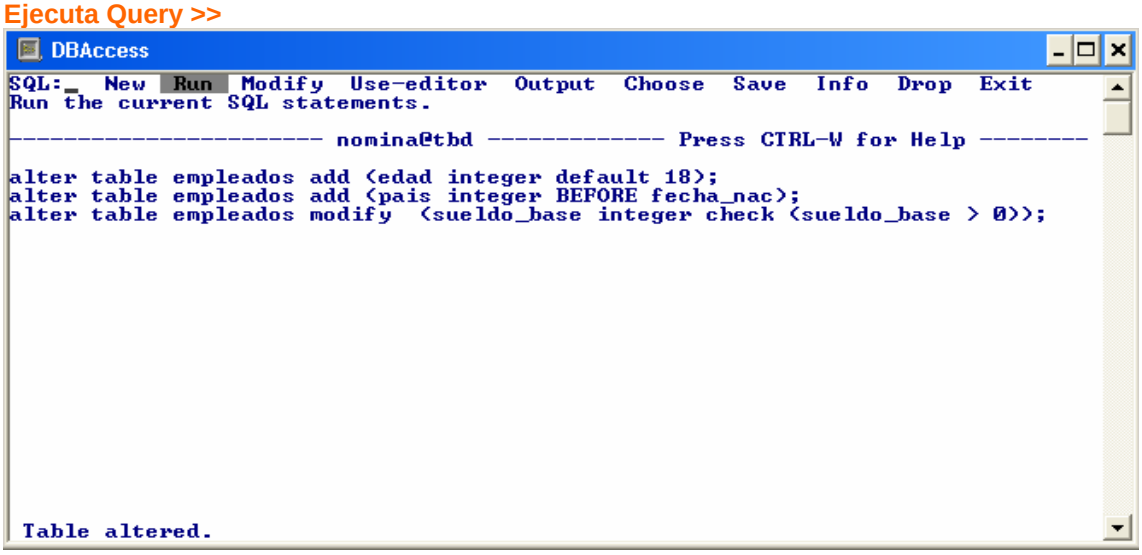
Ejercicio 1. A la tabla de EMPLEADOS, realizar los siguientes cambios de estructura.

- **Agregar el campo Edad tipo entero y valor por default 18**
- **Agregar el campo Pais tipo entero, pero que se ubique antes de la fecha de nacimiento**
- **Modificar el campo sueldo_base para que su valor sea mayor que 0**

En primer lugar se muestra la definición original de la tabla EMPLEADOS.

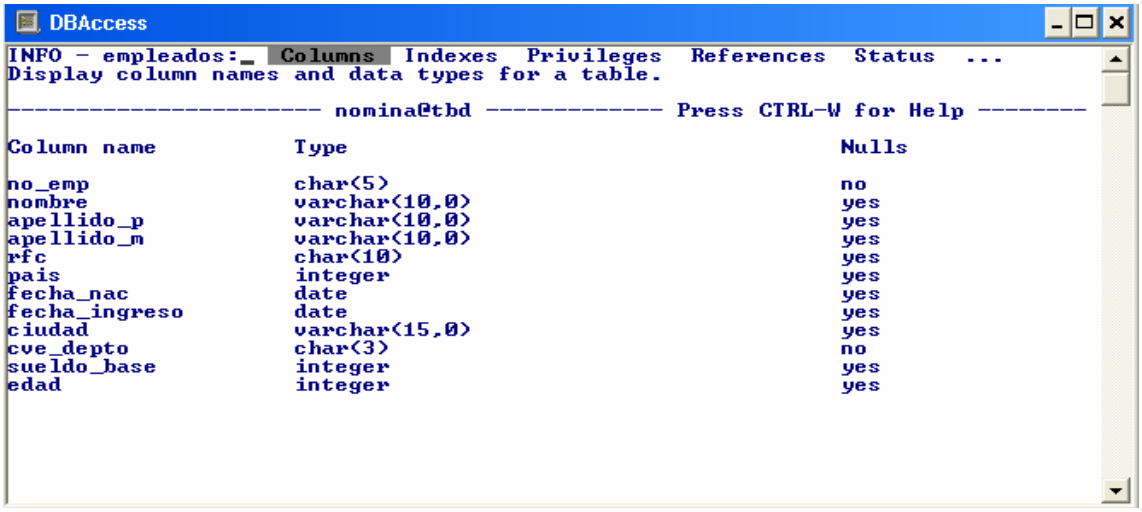


Ejecuta Query >>

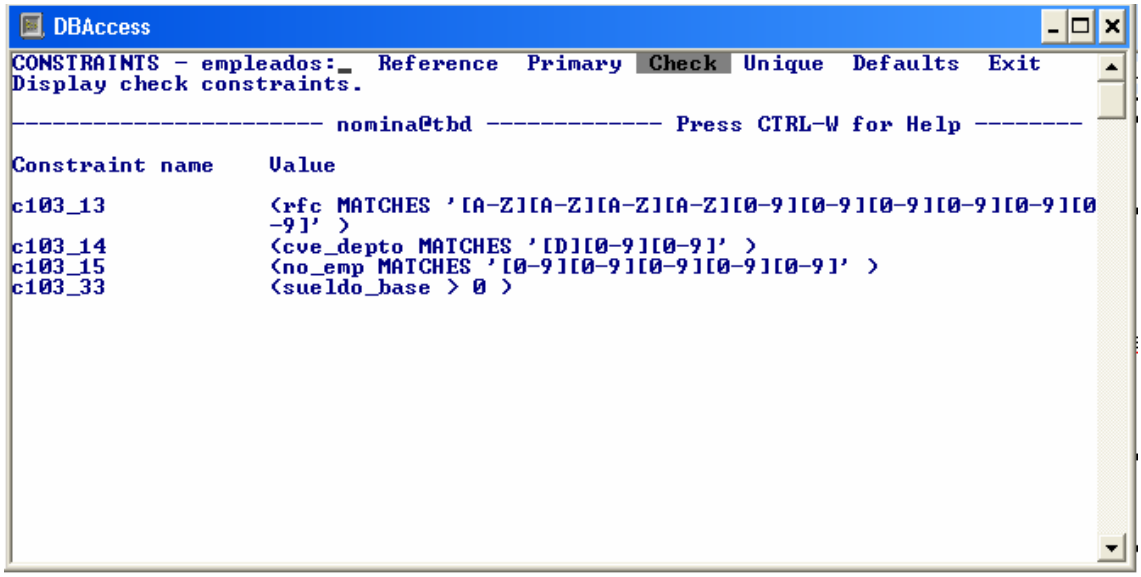


Se puede observar que se aplica la función de BEFORE para el campo de pais y para el campo sueldo_base se debe respetar y conservar el tipo de dato, sólo se agrega el CHECK para definir la validación.

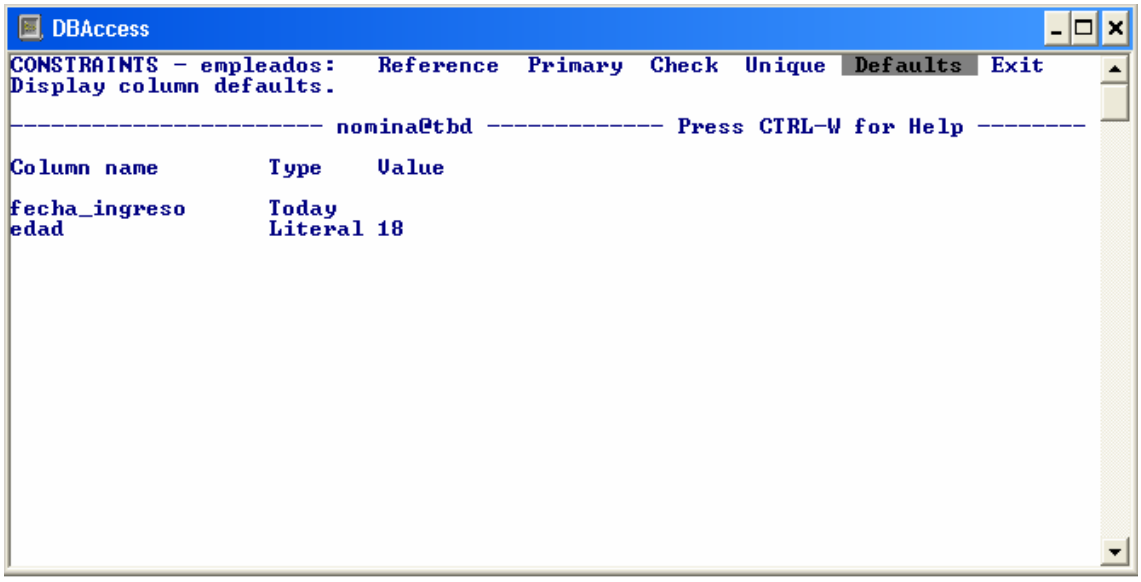
El resultado muestra ya los cambios realizados. En primer lugar la información general de la tabla, accesando Table->Info.



Para ver la validación, se accesa del menú la opción Table->Info ->cOnstraints->Check. Observándose el cambio en sueldo_base al final.

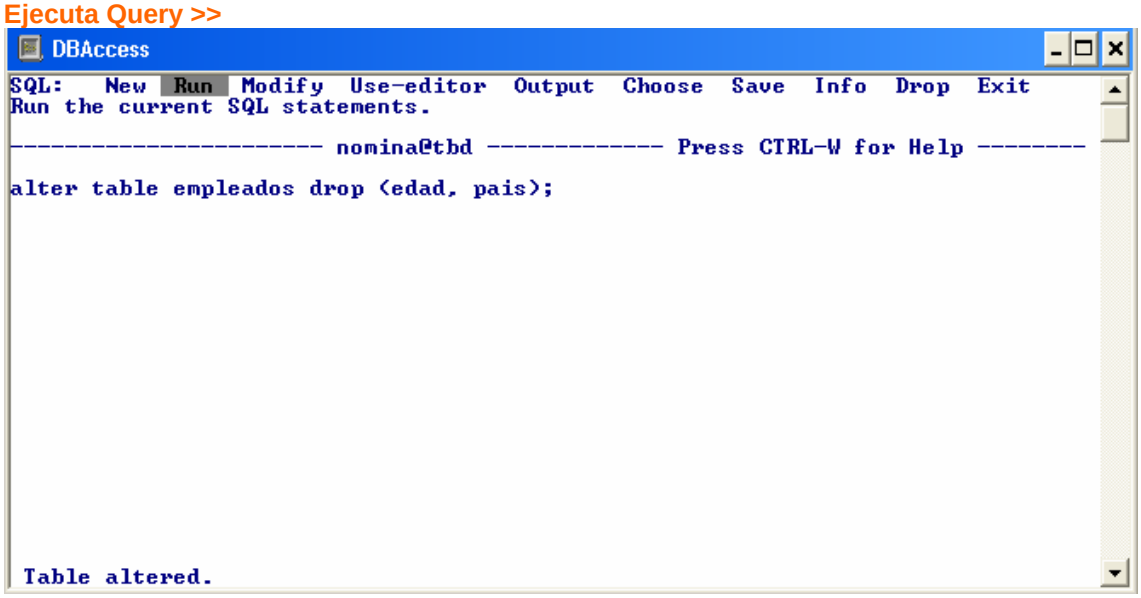


Para ver el valor por Default, se accesa del menú la opción Table->Info ->cOnstraints->Defaults. Observándose el cambio en sueldo_base.

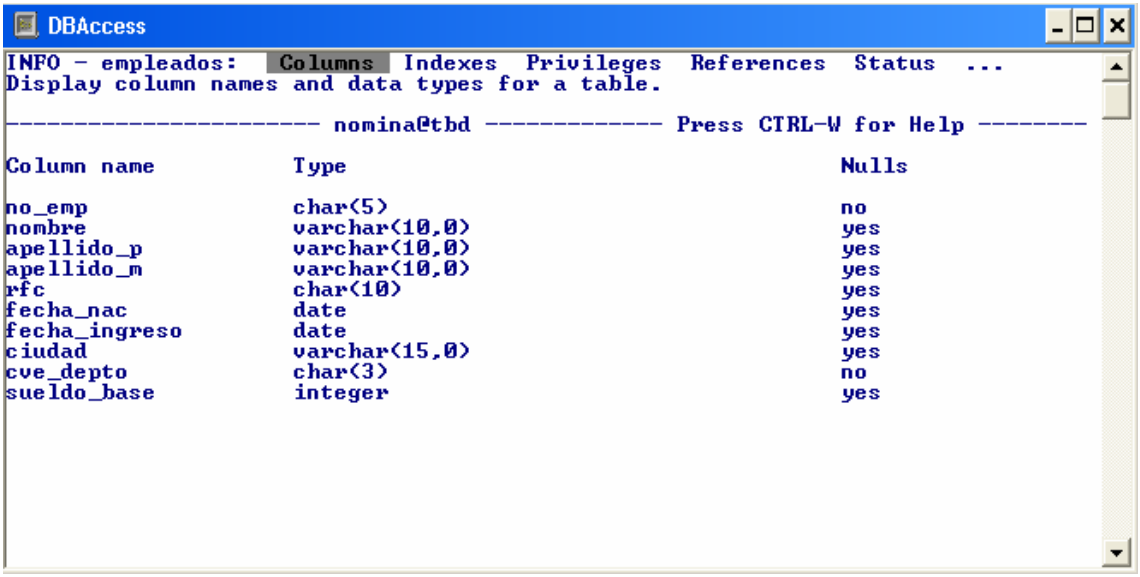


Ejercicio 2. Eliminar de la tabla EMPLEADOS los campos edad y pais.

Ejecuta Query >>



Se muestra el resultado, sólo el cambio aplicado sobre edad y pais. En este caso se elimina campo, definición y datos.



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries de creación.

Es importante considerar que en esta práctica con el estatuto ALTER, la estructura de la Base de Datos y tablas será afectada, por lo que se recomienda siempre obtener un respaldo de la Base de Datos. Puede utilizarse la herramienta dbexport /dbimport para este caso.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), pues si hay un cambio en estructura pueden perderse estas definiciones.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Cuál es la función del ALTER y qué consideraciones deben tomarse en cuenta para aplicarlo?.
2. ¿Qué es un constraint y cuáles son los diferentes tipos de constraints?.
3. ¿Cómo veo las definiciones de contraints para una tabla?.
4. Cambia el campo de ciudad al final del la tabla.
5. Elimina el índice del campo no_emp de la tabla EMPLEADOS.
6. Agrega un índice CLUSTER sobre la tabla EMPLEADOS sobre el campo RFC.
7. Cambia el check de sueldo_base que este en el rango de 1 a 50000.
8. Agrega el campo fecha_cambio_sueldo tipo datetime (Year to minute) y su valor por default que sea la fecha actual.

9. Entregar un esquema con el nuevo diseño de la base de datos y tabla EMPLEADOS

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 7
Utilizando el estatuto INSERT

Objetivos

- Implementar queries para insertar datos a las tablas
- Implementar INSERT tomando en cuenta valores por default y validaciones

Introducción

El estatuto INSERT es importante para la captura de datos en las tablas, puede resultar sencillo, pero lo importante es la definición previa de la base de datos, pues cualquier error de definición en esta parte se verá reflejada o ni siquiera será observada.

Recordar que cuando se hace un buen diseño y definición de la base de datos, el manejo de la información resulta sencillo y prácticamente el diseño va llevando al desarrollo de los queries y no los queries obligan a modificar el diseño. Entre más detalles se incluyan en el diseño, menos se programará.

Se muestra la Sintaxis del DELETE para borrar aquellos registros no deseados, así también como la del SELECT para ver las inserciones realizadas.

SINTAXIS

```
INSERT INTO table-name [(column-list)]
      {
      VALUES (value-list)
      |
      SELECT-statement
      }


DELETE FROM table-name [WHERE condition]

SELECT [ALL | DISTINCT | UNIQUE] select-list
      FROM [OUTER] table-name [table-alias] [,...]
```

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Instrucciones de INSERT**

Material y equipo necesario

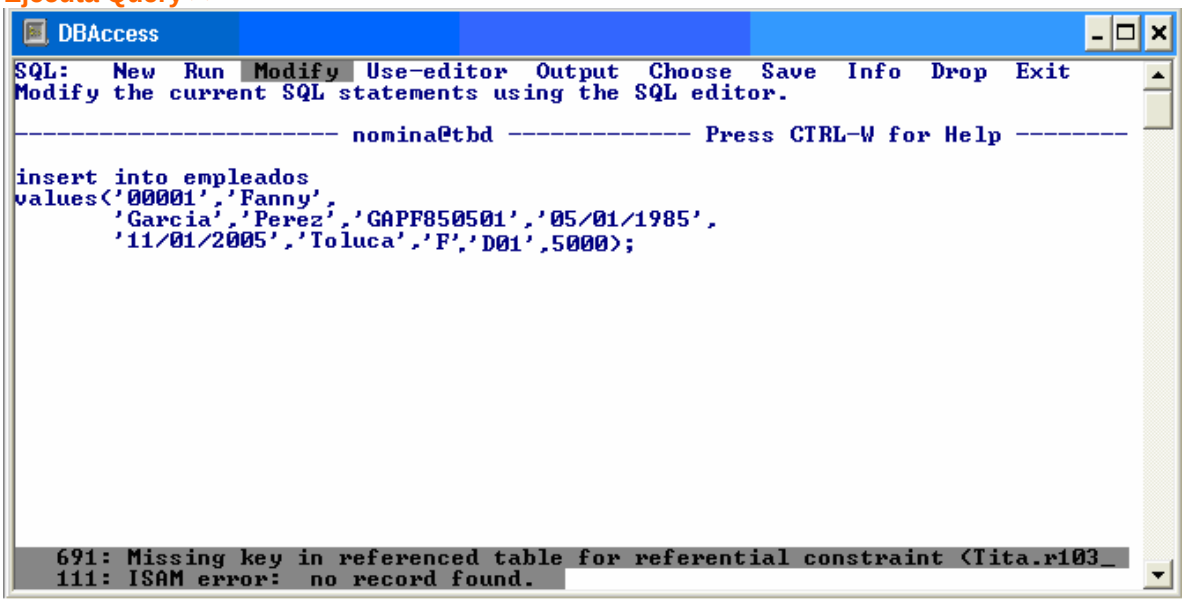
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)

4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con el esquema de la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

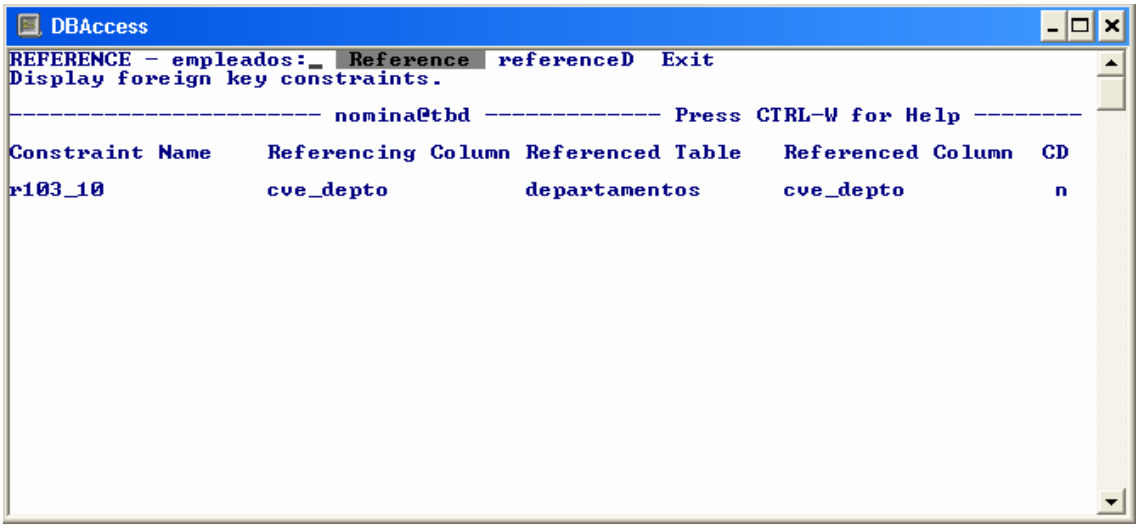
Metodología

Ejercicio 1. INSERT manejando Integridad Referencial. Inserta un registro en la tabla EMPLEADOS.

Ejecuta Query >>

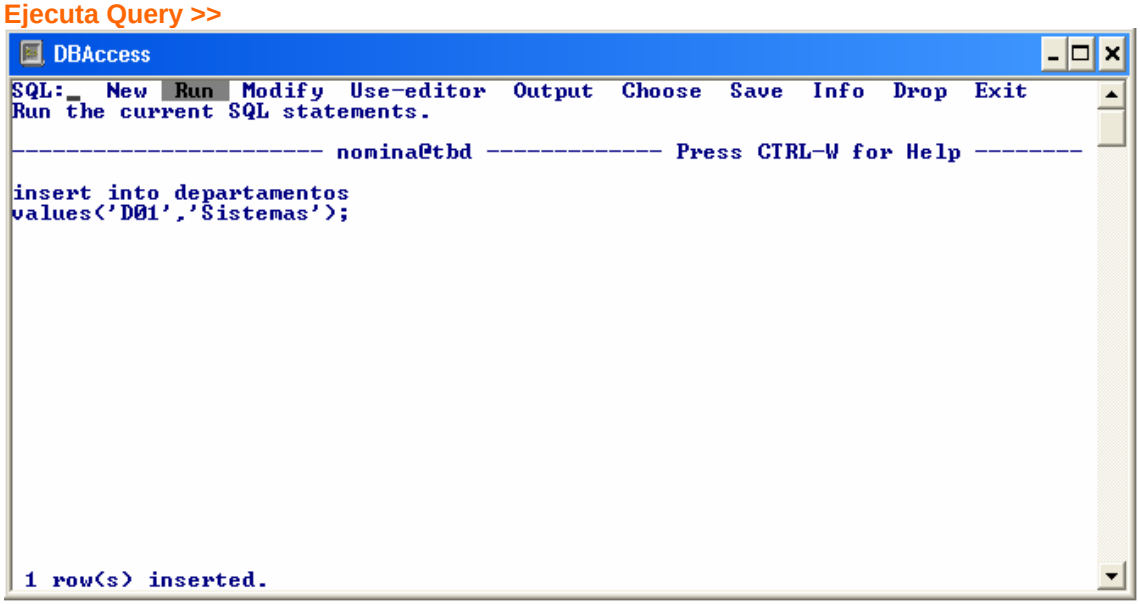


Se observa error de Integridad Referencial, dependencia de datos sobre el contraint r103_10. Esto se accesa a través del Menú con la opción Table->Info->Referente. Observándose que la dependencia es por el campo cve_depto de DEPARTAMENTOS, el valor de este campo debe encontrarse previamente en la tabla.

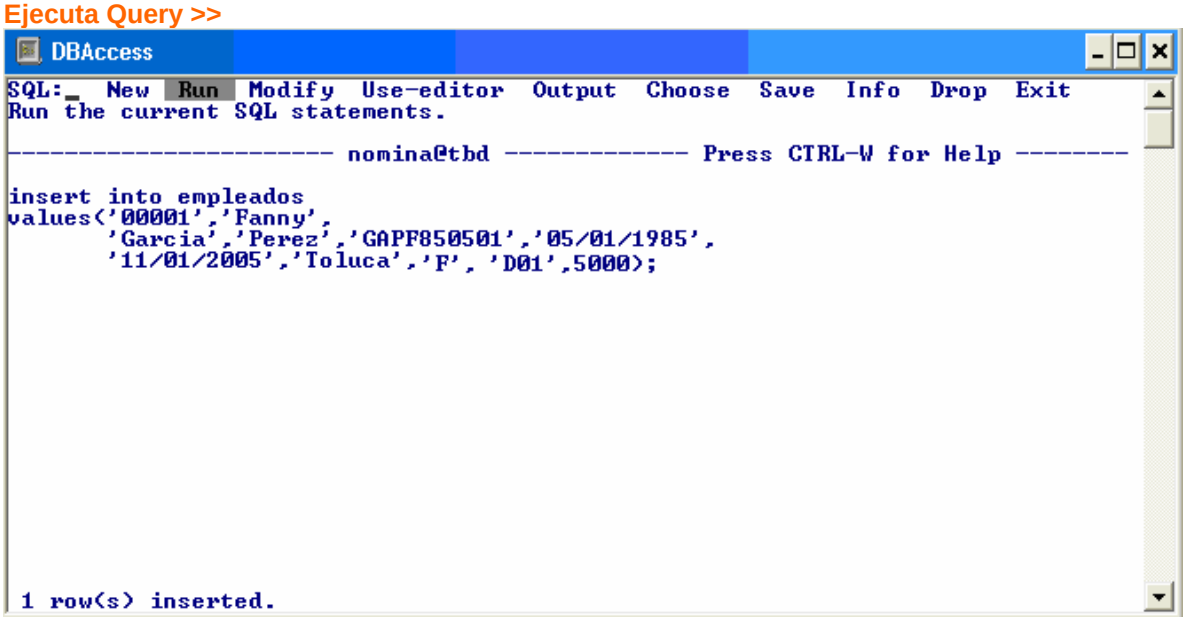


Por lo que se debe insertar primero en la DEPARTAMENTOS y ejecutar nuevamente el Query inicial.

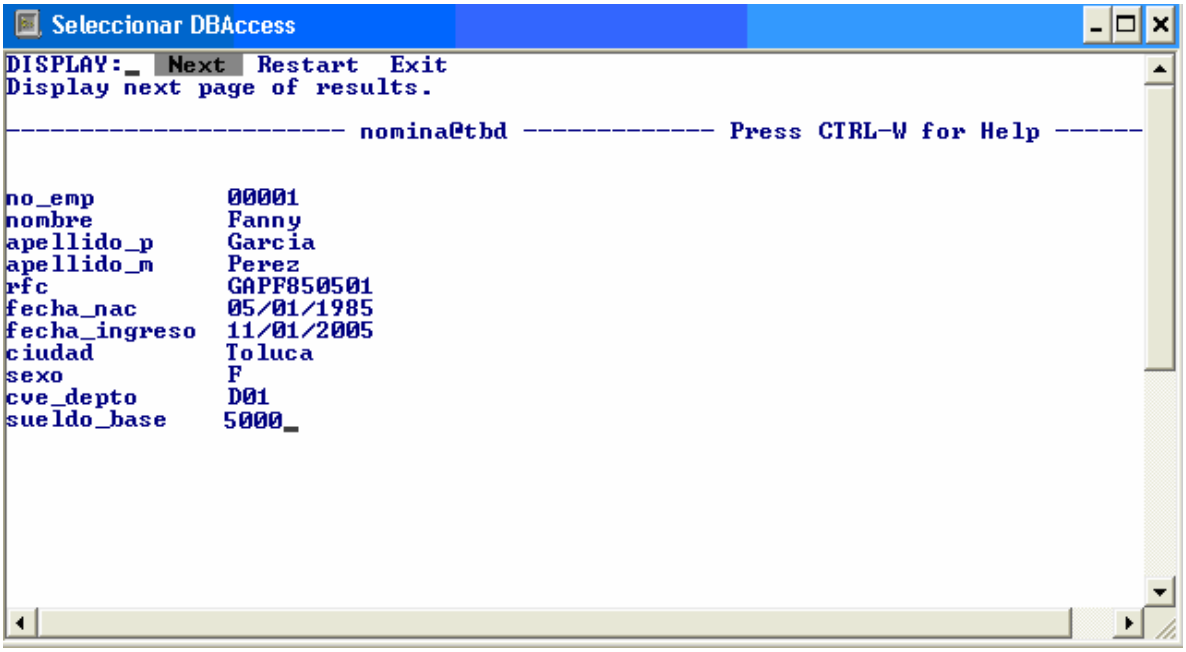
Ejecuta Query >>



Ejecuta Query >>

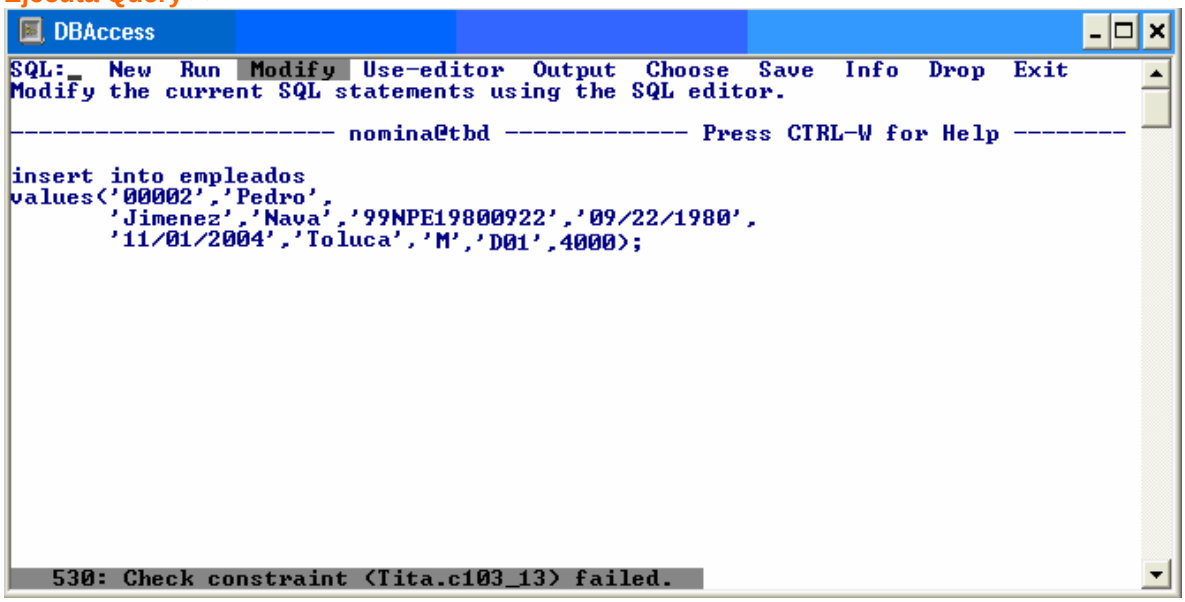


Verificando que la inserción fue exitosa.

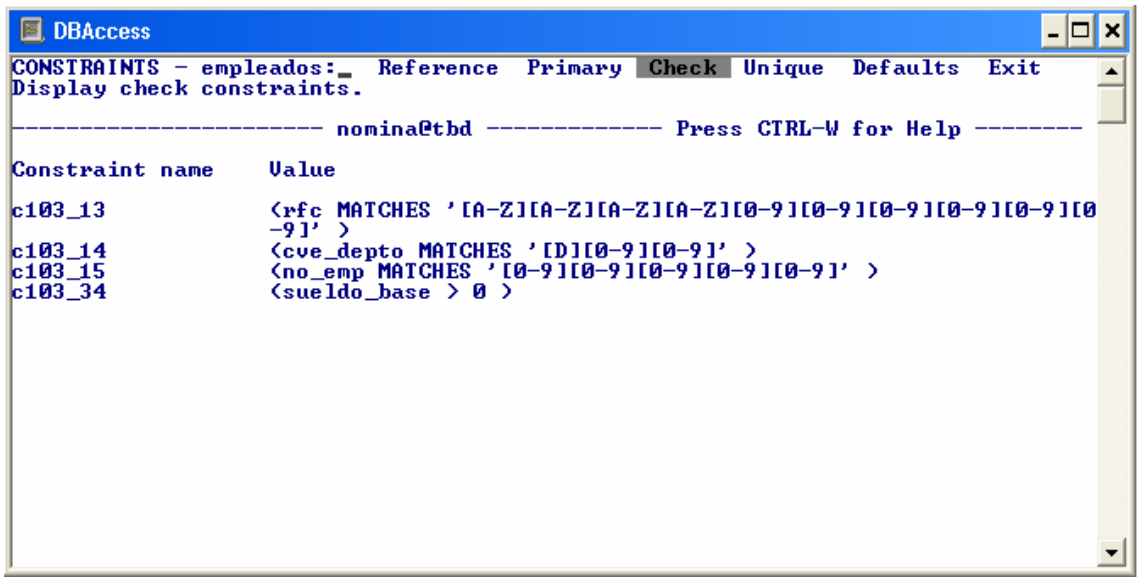


Ejercicio 2. INSERT considerando Check´s o validaciones. Inserta un registro en la tabla EMPLEADOS.

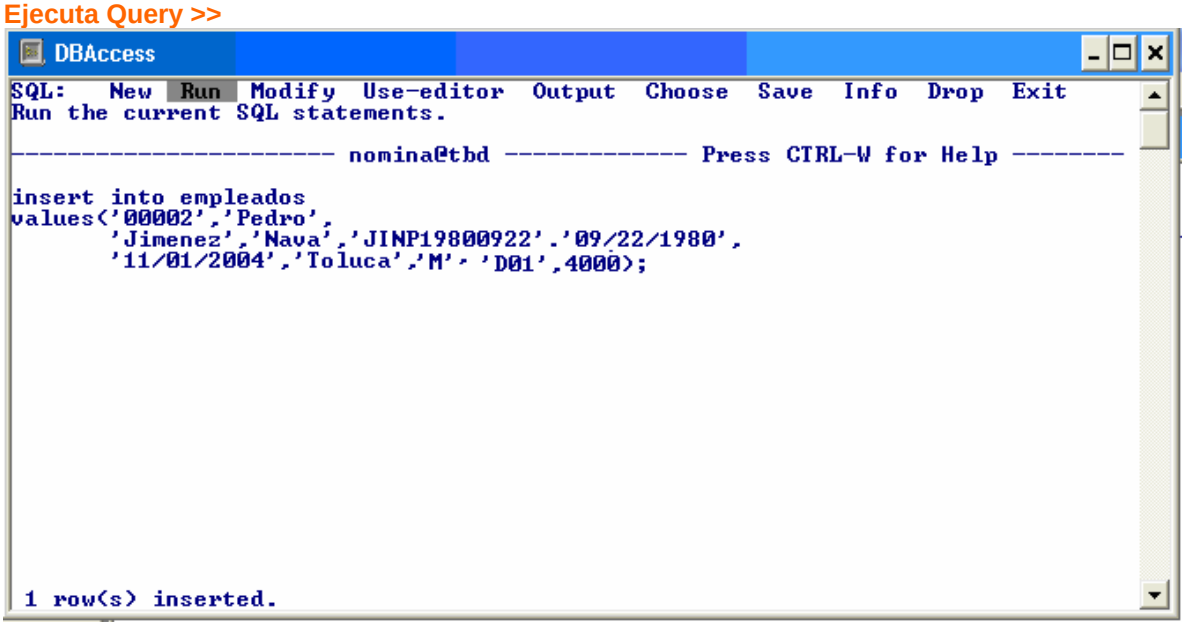
Ejecuta Query >>



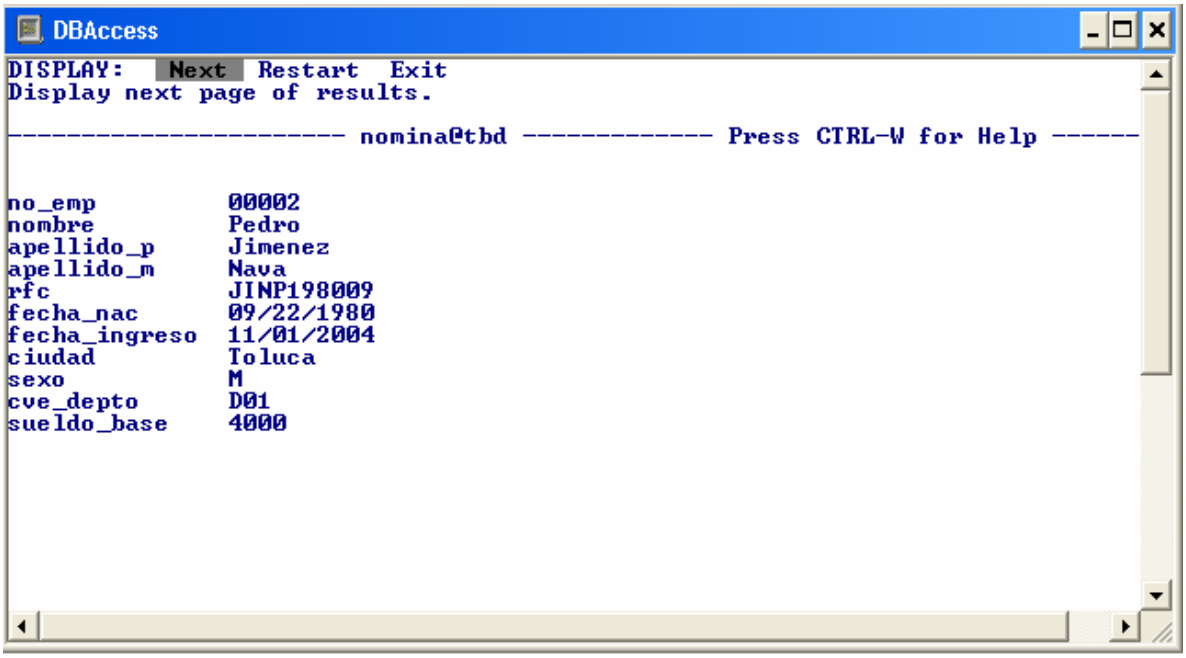
Se observa que el error de check se marca sobre el constraint c103_13. Se vuelve a verificar cual es a través del Menú, y corresponde al campo RFC, seguramente no está en el formato requerido. Por lo que debe ajustarse y ejecutarse nuevamente.



Ejecuta Query >>

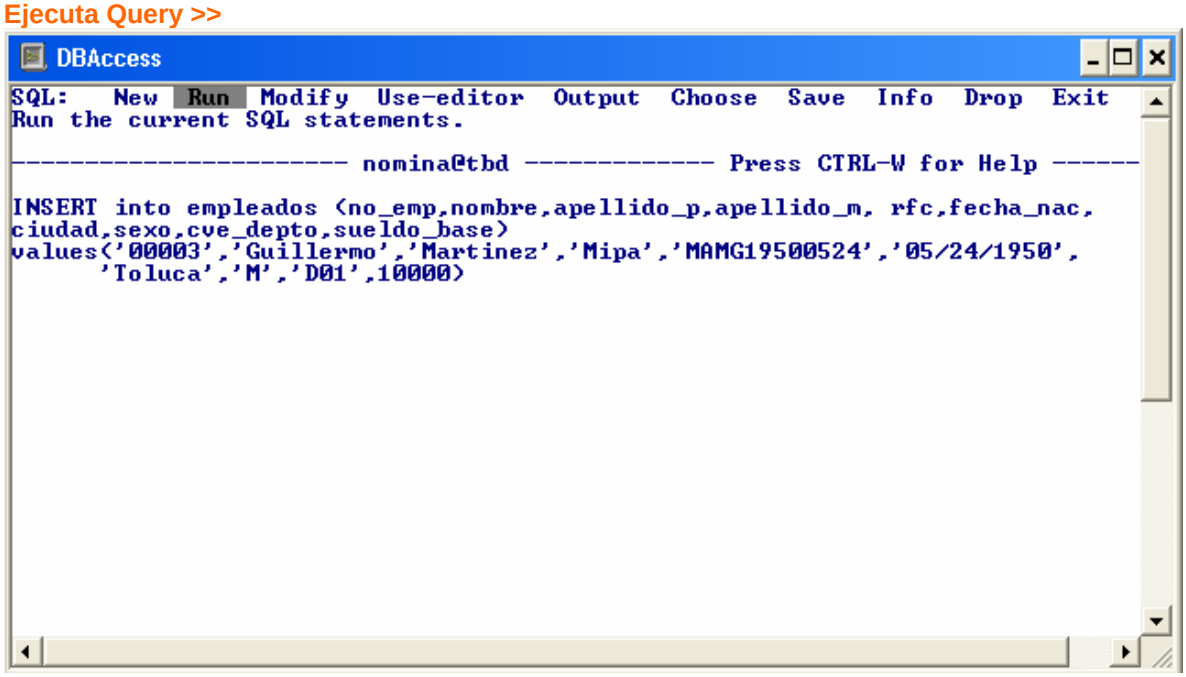


Verificando que la inserción fue exitosa.



Ejercicio 3. INSERT considerando Defaults. Inserta un registro en la tabla EMPLEADOS.

Ejecuta Query >>



Se observa que fue omitido el campo fecha_ingreso, por lo que toma el valor por default que se definió, sin marcar error. Este tipo de INSERT ayuda a mantener la consistencia de datos en la base.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Es importante considerar que en esta práctica con el estatuto INSERT, se cargará información en la Base de Datos, por lo que se sugiere siempre generar un respaldo antes y después de la carga. Puede utilizarse la herramienta dbexport /dbimport para este caso.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade) para las inserciones de datos.

La restricción CHECK y DEFAULT ayuda a mantener la consistencia de datos en la base.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Inserta los registros de la base de datos **EJEMPLO** que se encuentra en capítulo introductorio, en cada una de las tablas, el contenido debe ser el mismo.
2. ¿Cómo realizo un INSERT para comprobar que una validación está funcionando?. Toma dos Ejemplos de los datos insertados en el punto 1.
3. ¿Cómo realizo un INSERT para comprobar que una inserción por Default está funcionando?. Toma dos Ejemplos de los datos insertados en el punto 1.
4. Respaldar la base de datos(esquema y datos).
5. Entrega el esquema de la base de datos, así como los datos insertados de cada tabla.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 8
Utilizando el estatuto UPDATE

Objetivos

- Implementar queries para actualizar datos de las tablas
- Actualizar datos implementando el estatuto Where

Introducción

El uso del estatuto UPDATE es delicado, ya que realiza actualizaciones de los datos directamente, por lo que es importante tomar las consideraciones que esta práctica se comentarán. Se recomienda realizar un respaldo antes de la ejecución de los queries, este respaldo puede hacerse con el comando en línea de dbexport/dbimport. La opción de WHERE debe usarse con precaución y conocimiento claro de lo que se quiere hacer.

Se muestra la Sintaxis del DELETE para borrar aquellos registros no deseados, así también como la del SELECT para ver las actualizaciones realizadas.

SINTAXIS

```
UPDATE table-name SET {column-name = expression [,...]
      | {(col-list) | *} = (expr-list)}
      [WHERE condition]


DELETE FROM table-name [WHERE condition]

SELECT [ALL | DISTINCT | UNIQUE] select-list
      FROM [OUTER] table-name [table-alias] [,...]

dbexport <database> [-X] [-c] [-q] [-d] [-ss]
  [{ -o <dir> | -t <tapedev> -b <blksz> -s <tapesz> [-f <sql-command-file>] }
  NOTE: arguments to dbexport are order independent.

dbimport <database> [-X] [-c] [-q] [-d <dbspace>]
  [-l [{ buffered | <log-file> }] [-ansi]]
  [{ -i <dir> | -t <tapedev> [ -b <blksz> -s <tapesz> ] [-f <script-file>] }]
  NOTE: <log-file> must be a complete path
      arguments to dbimport are order independent
```

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Instrucción UPDATE**

Material y equipo necesario

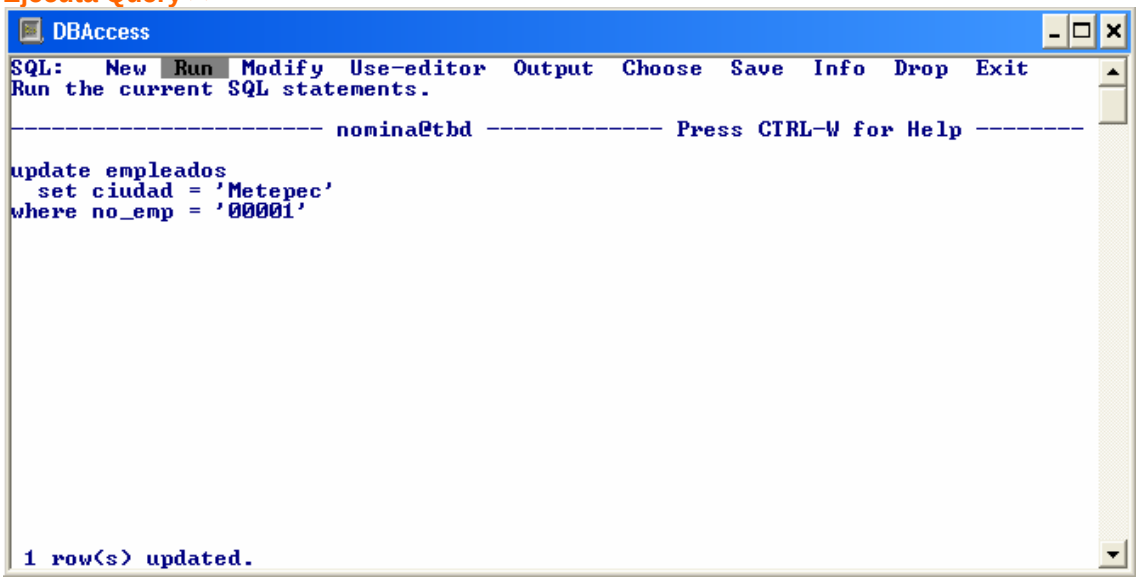
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.

2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con el esquema de la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

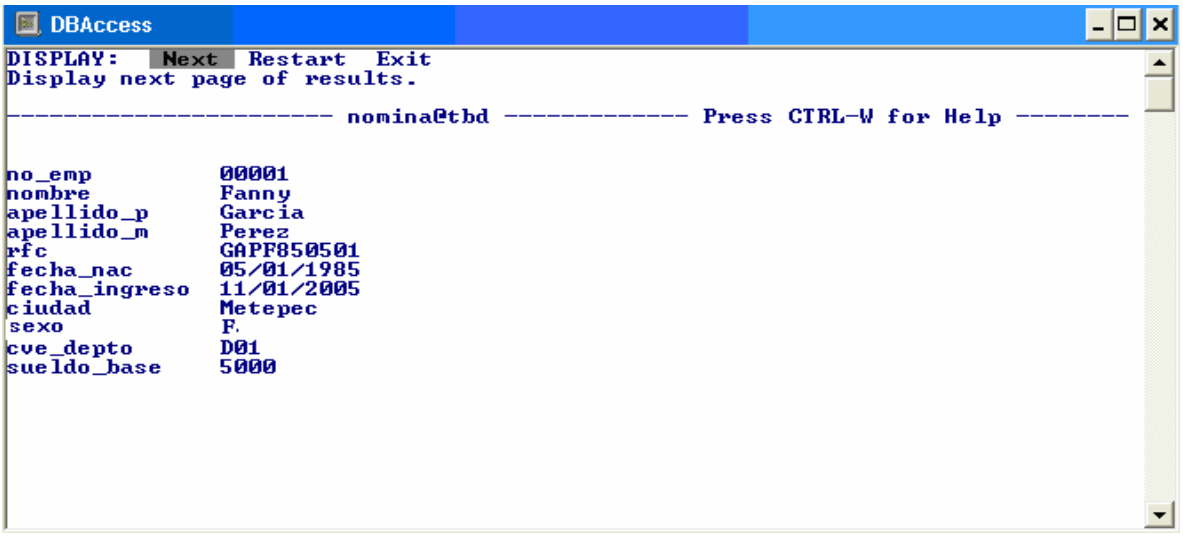
Metodología

Ejercicio 1. UPDATE con WHERE. El empleado no. 00001 cambio su ciudad de residencia a "Metepec", realizar la actualización correspondiente.

Ejecuta Query >>

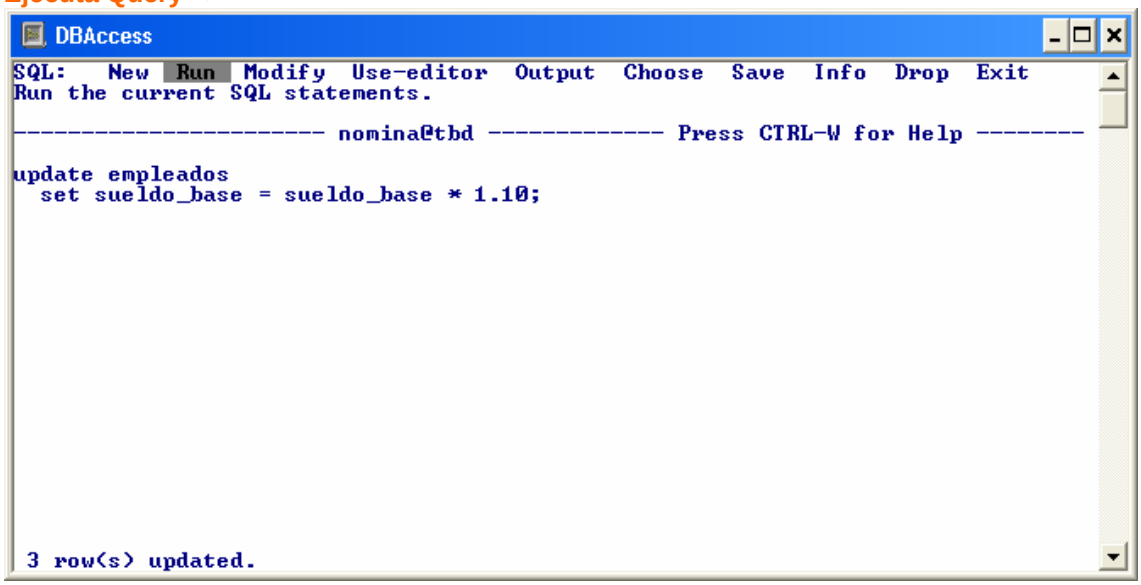


Este Query actualiza sólo los registros que cumplan con la condición en el WHERE. Se ve el resultado a continuación.



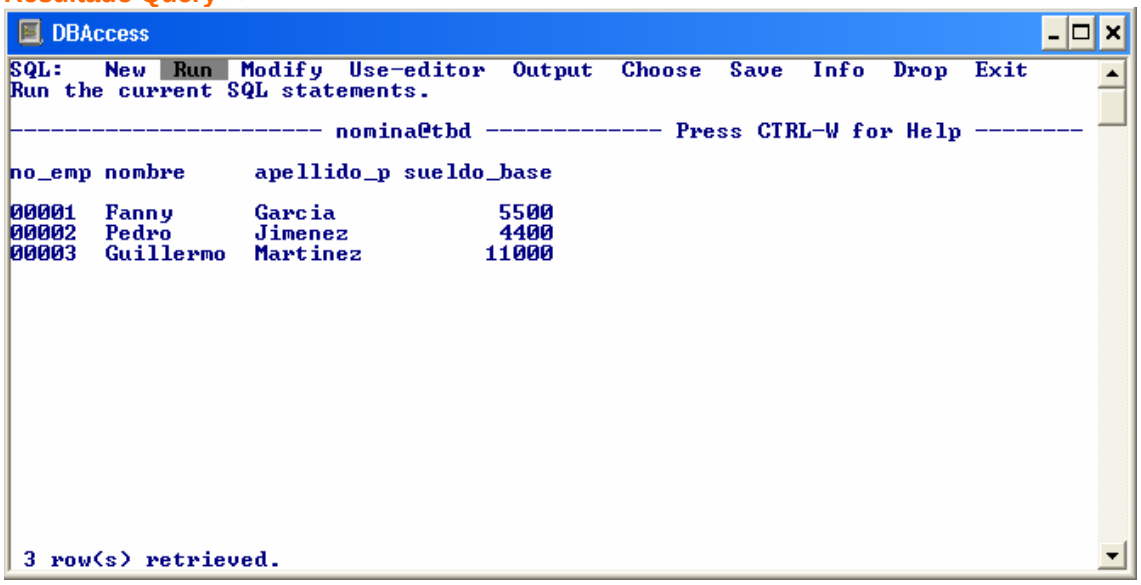
Ejercicio 2. UPDATE sin WHERE. La empresa da un aumento general del 10% al sueldo de los empleados. Realizar la actualización correspondiente.

Ejecuta Query >>



Importante: Únicamente en ejemplos como éste, donde es necesario actualizar todos los registros de una tabla, no lleva el estatuto condicionante WHERE. En cualquier otro caso se debe ser muy cuidadoso al aplicar esta instrucción sin condición, pues afecta a todos los registros.

Resultado Query >>



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Es importante considerar que en esta práctica con el estatuto UPDATE, se actualizará la información de la Base de Datos, por lo que se sugiere siempre generar un respaldo antes y después de la carga. Puede utilizarse la herramienta dbexport /dbimport para este caso.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), así como los tipos de datos para las actualizaciones de datos.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Realizar un respaldo de la base da datos.
2. Realizar un Query sobre la tabla de NOMINA, que genere el ingreso_neto = percepciones - deducciones, esto deberá realizarse sobre todos los registros de la tabla.
3. Para todos los empleados del departamento de sistemas, se les incrementa su sueldo en un 5%. Realizar Query.
4. Todos los empleados nacidos entre 1970-1980, tendrán un incremento en el sueldo del 3%.
5. Para todos los empleados habrá un incremento de sueldo del 5%, excepto para los que el año de ingreso sea el actual, o sea no tengan un año cumplido.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 9
Utilizando el estatuto DELETE

Objetivos

- Implementar queries para borrado de datos de las tablas
- Eliminar datos usando el estatuto Where
- Eliminar datos implementando el borrado en cascada

Introducción

El uso del estatuto DELETE es delicado, ya que realiza la eliminación de los registros de las tablas de la base de datos, por lo que es importante tomar las consideraciones que esta práctica se comentarán. Se recomienda realizar un respaldo antes de la ejecución de los queries, este respaldo puede hacerse con el comando en línea de dbexport/dbimport. La opción de WHERE debe usarse con precaución y conocimiento claro de lo que se quiere hacer.

Es muy importante aprovechar las ventajas de un buen diseño de base de datos para la aplicación de los borrados en cascada; y así mantener la consistencia de datos. Es por eso que un buen diseño de la base evita que se complique la realización de queries, se pierda la consistencia de datos y que al programar se eviten las validaciones innecesarias de integridad. Se muestra la Sintaxis del SELECT para ver las actualizaciones realizadas.

SINTAXIS

DELETE FROM table-name [WHERE condition]

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]

dbexport <database> [-X] [-c] [-q] [-d] [-ss]
[{ -o <dir> | -t <tapedev> -b <blksz> -s <tapesz> [-f <sql-command-file>] }
NOTE: arguments to dbexport are order independent.

dbimport <database> [-X] [-c] [-q] [-d <dbspace>]
[-l [{ buffered | <log-file> }] [-ansi]]
[{ -i <dir> | -t <tapedev> [-b <blksz> -s <tapesz>] [-f <script-file>] }]
NOTE: <log-file> must be a complete path
arguments to dbimport are order independent

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Instrucción DELETE**

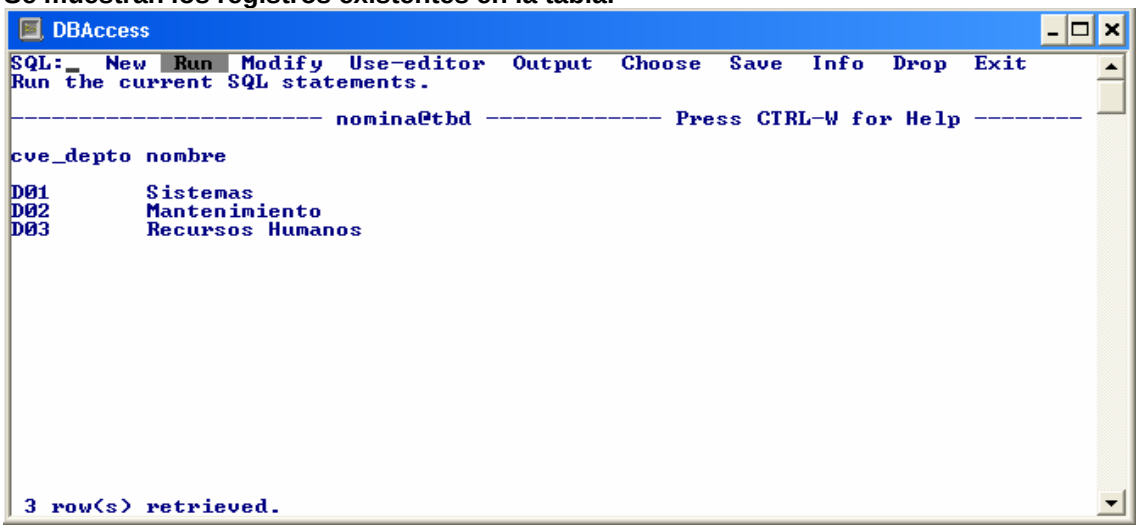
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con el esquema de la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***

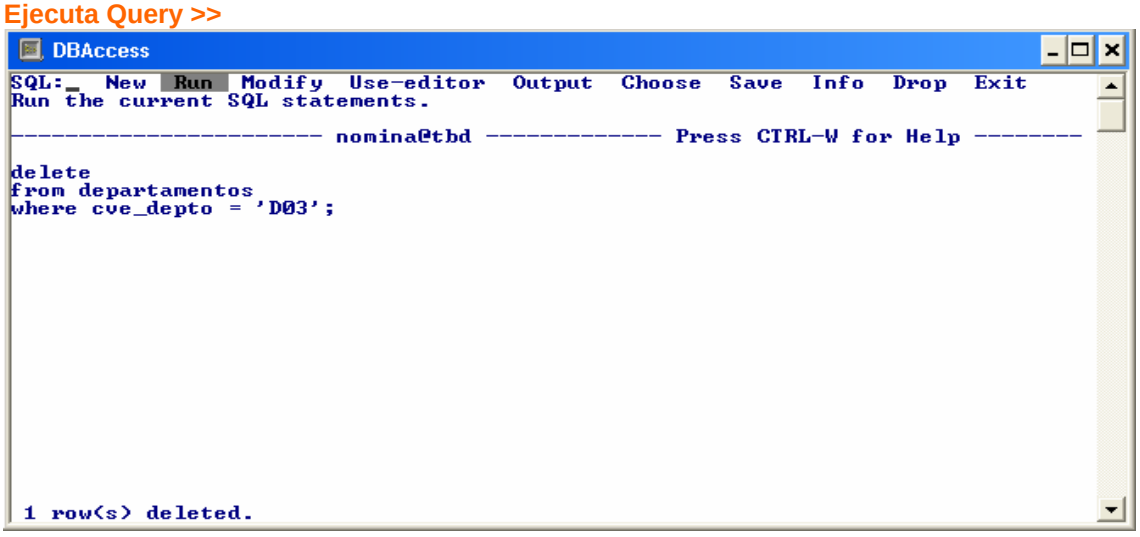
Metodología

Ejercicio 1. DELETE simple. El Departamento D03 correspondiente a Recursos Humanos desaparece de la empresa. Realizar el Query correspondiente.

Se muestran los registros existentes en la tabla.

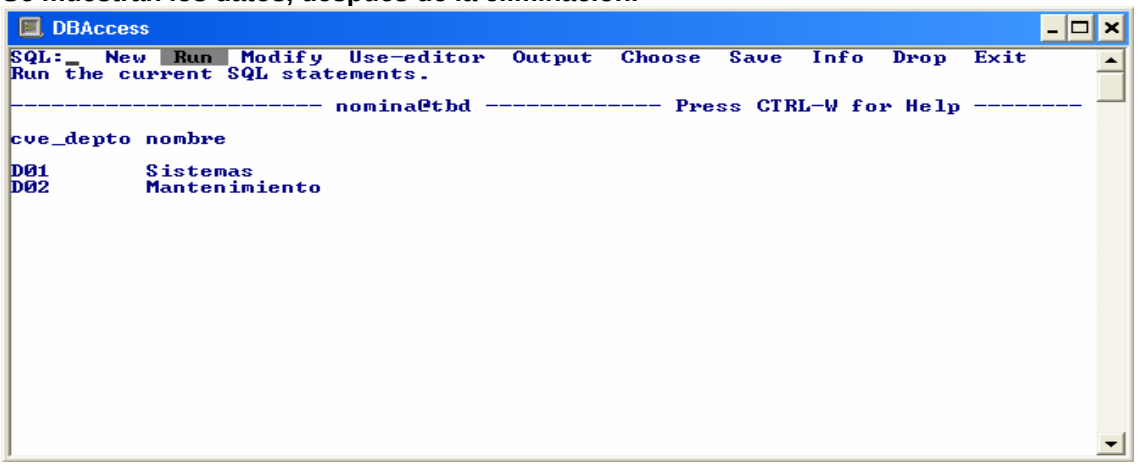


Ejecuta Query >>



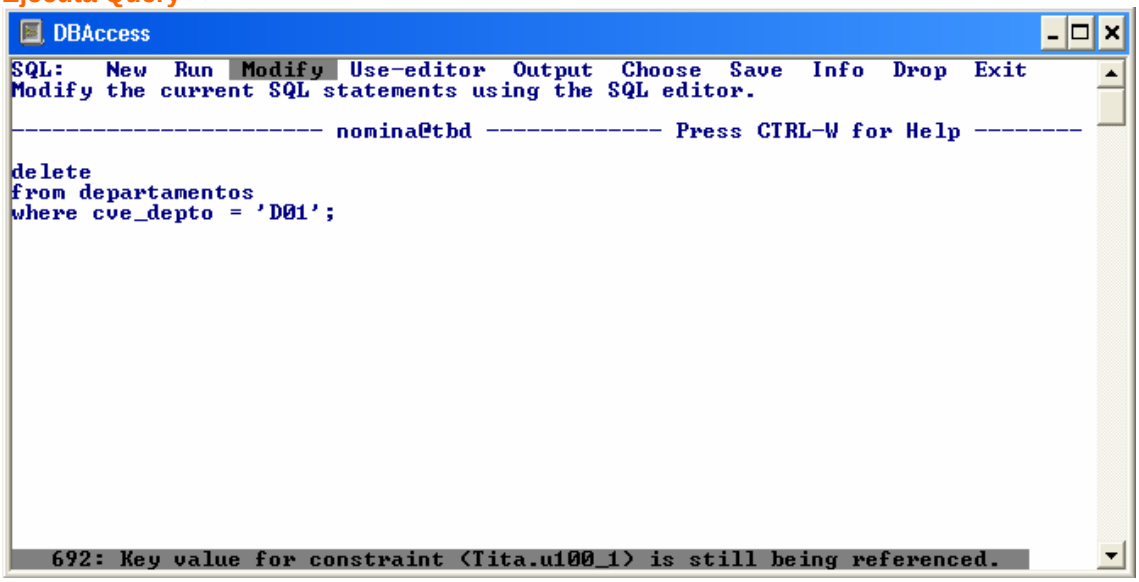
La ejecución se realiza sin problema, debido a que no hay empleados que pertenezcan a este departamento. No afecta la integridad referencial.

Se muestran los datos, después de la eliminación.



Ejercicio 2. DELETE CONSIDERANDO INTEGRIDAD REFERENCIAL. El Departamento D01 correspondiente a Sistemas desaparece de la empresa. Realizar el Query correspondiente.

Ejecuta Query >>



Se observa error en la ejecución, debido a que no puedo dar de baja un departamento al cual aún están relacionados o dados de alta empleados.

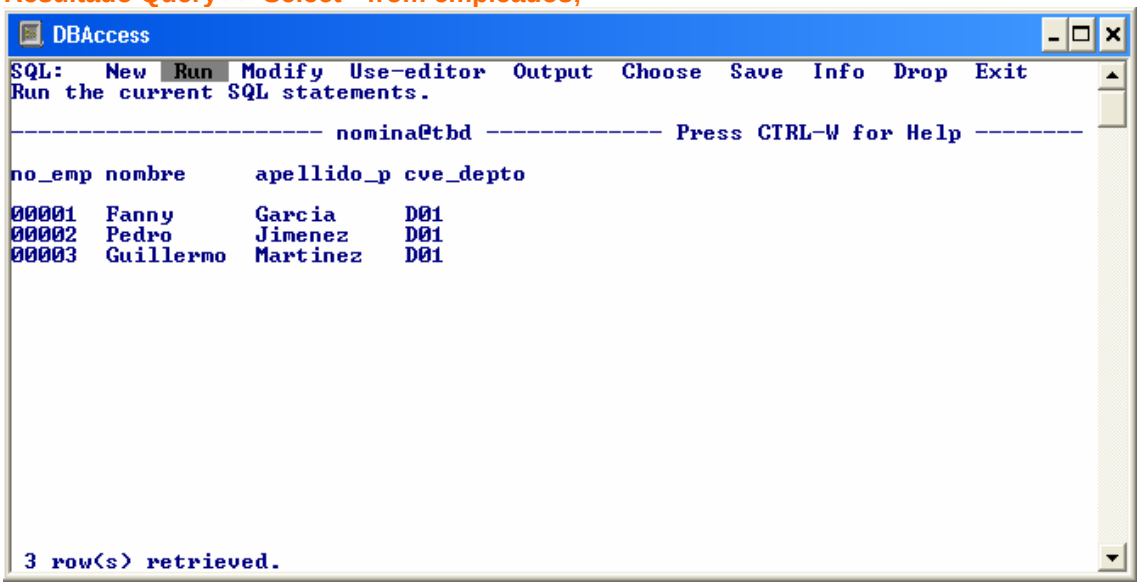
Como la política de la empresa no es dar de baja empleados si el departamento desparece, entonces primero se reubicarán los empleados de departamentos y finalmente se elimina el departamento de la base de datos.

Ejercicio 3. DELETE CONSIDERANDO BORRADO EN CASCADA. El empleado con No. 00001 se da de baja de la empresa. Realizar el Query correspondiente.

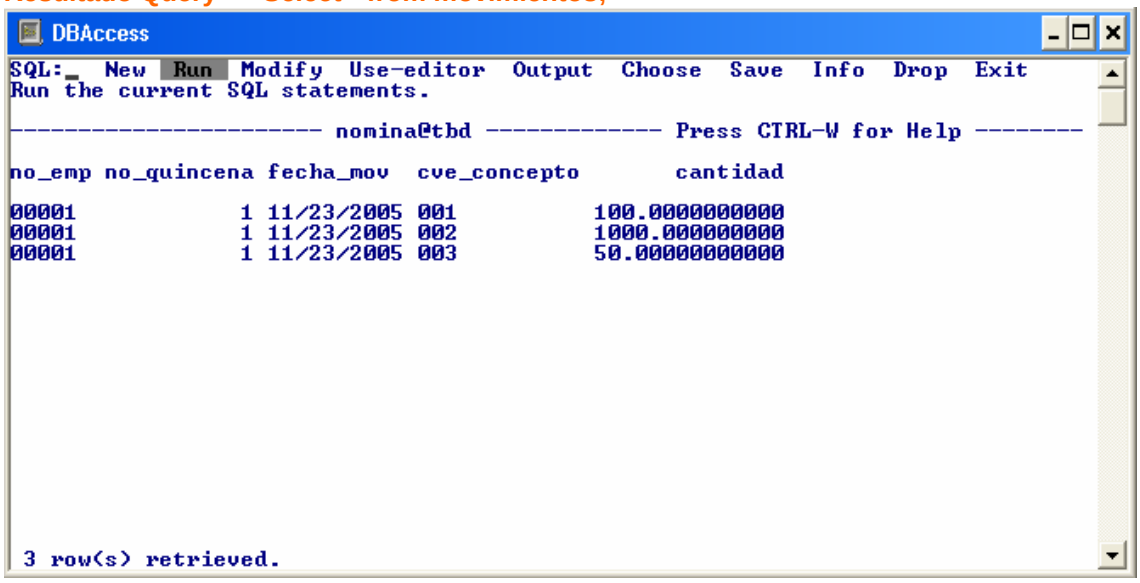
La tabla EMPLEADOS de donde se dará de baja está relacionada con MOVIMIENTOS y NOMINA, con borrado en cascada, es decir, si borro en EMPLEADOS debe borrar a ese empleado de las otras dos tablas de manera automática.

Se muestra a continuación el contenido de las tres tablas, antes de la ejecución del Query.

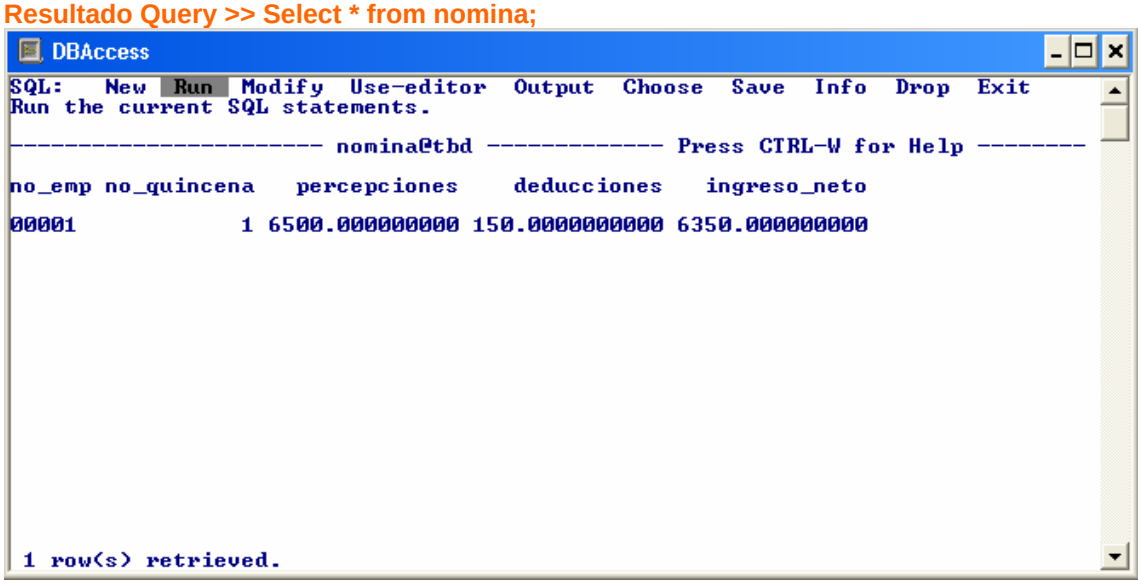
Resultado Query >> Select * from empleados;



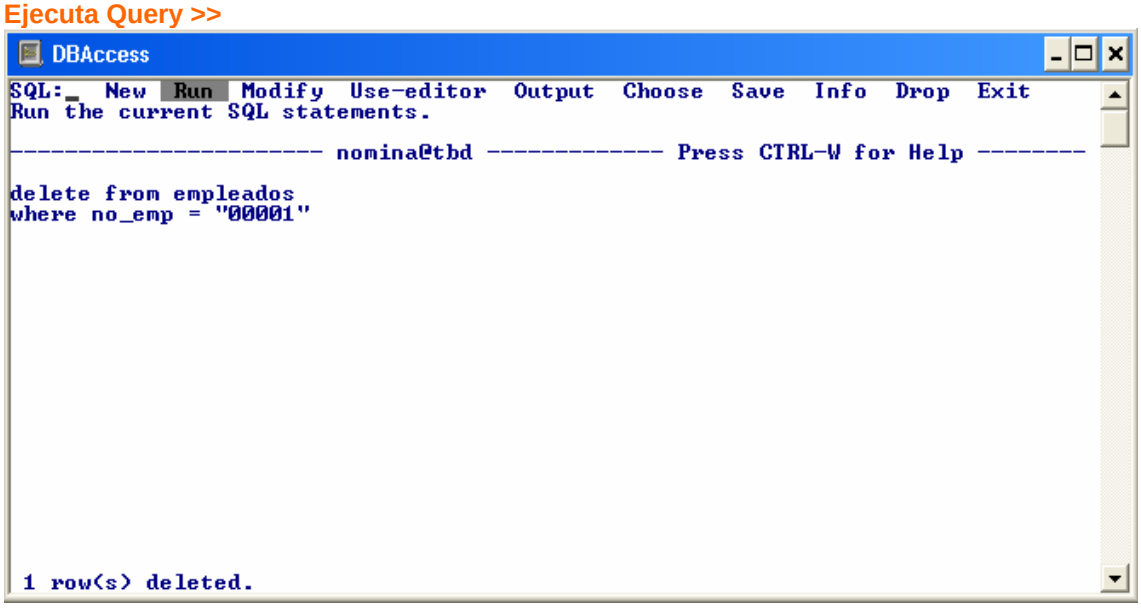
Resultado Query >> Select * from movimientos;



Resultado Query >> Select * from nomina;

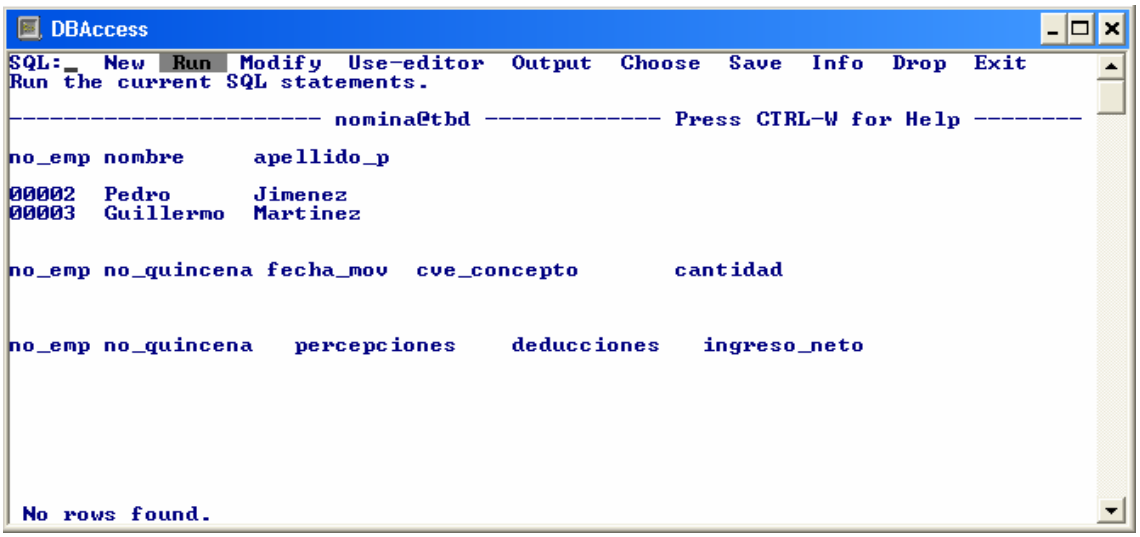


Ejecuta Query >>



Se ejecuta sin problemas eliminando en cascada los datos del empleado 00001 como se muestra nuevamente.

Resultado Query >> **Select * from empleados;**
Select * from movimientos;
Select * from nomina;



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Es importante considerar que en esta práctica con el estatuto DELETE, se eliminará la información de la Base de Datos, por lo que se sugiere siempre generar un respaldo antes del proceso. Puede utilizarse la herramienta dbexport /dbimport para este caso.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), así como los tipos de datos para la eliminación de datos.

Reporte del estudiante.

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Realizar un respaldo de la base de datos.
2. Eliminar todos los movimientos de la quincena 1.
3. Eliminar todos movimientos que el concepto sea 001 o 003.
4. Dar de baja al empleado con No. 00003, verificar que el borrado en cascada se haya realizado. Guardar resultados.
5. Eliminar el departamento D02.
6. Realizar la recuperación de la base de datos.
7. ¿Cómo verifico que hay una definición de borrado en cascada entre dos tablas?.
8. ¿Qué es el borrado en cascada y cuales son los beneficios y riesgos?.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 10
Accesando información de la Base de Datos I

Objetivos

- Implementar consultas con el estatuto Select
- Consultas usando estatuto Where

Introducción

El estatuto SELECT sirve para accesar la información de la base de datos, esta parte es muy importante entender lo que el usuario requiere para darle lo que necesita, además de realizar queries óptimos, es decir, buen diseño y rendimiento de los recursos del servidor o equipo cliente.

Es por eso, que se trabajó fuertemente en las prácticas anteriores en desarrollar un buen diseño de base de datos, esto facilita la generación de queries. De otra manera los queries generados se harían cada vez más complejos e ineficientes, tendiendo que regresar al rediseño de la base o a programarlos con otras herramientas de desarrollo.

SINTAXIS

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]
[GROUP BY column-list] [HAVING condition]
[ORDER BY column-name [ASC | DESC],...]
[INTO TEMP table-name]

SELECT-statement UNION [ALL] SELECT-statement
[UNION [ALL] SELECT-statement ...]

WHERE conditions:

expr rel-op expr

expr [NOT] BETWEEN expr AND expr

expr [NOT] IN (items)

column-name [NOT] LIKE "string" [ESCAPE escape-character]

column-name [NOT] MATCHES "string" [ESCAPE escape-character]

expr rel-op {ALL | [ANY | SOME]} (SELECT-statement)
expr [NOT] IN (SELECT-statement)

[NOT] EXISTS (SELECT-statement)

column-name IS [NOT] NULL

MATCHES Y METACARACTERES

LIKE	MATCHES	SIGNIFICADO
%	*	Evalua 0 ó más caracteres
-	?	Evalúa un character
	[ ]	Especifíca un character

OPERADORES Y PALABRAS RESERVADAS

BETWEEN
IN
LIKE
IS NULL
AND/OR
=, <>, >=,<=

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Consultas básicas SELECT, WHERE y funciones a nivel registro**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

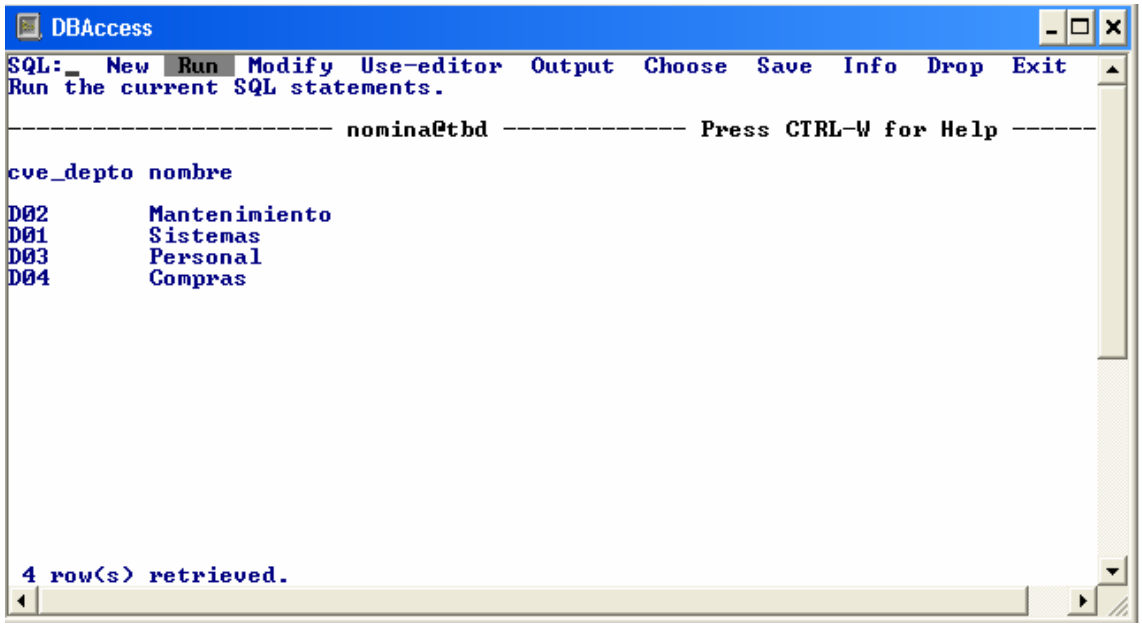
Metodología

EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. En esta práctica de SELECT, se muestra el Query que se ejecutará via dbaccess y el resultado directo, esto para mejorar el entendimiento de cada **Ejercicio**.

Ejercicio 1. SELECT simple. Accesar todos los departamentos y sus datos.

Ejecuta Query >>

```
SELECT *
FROM departamentos
```

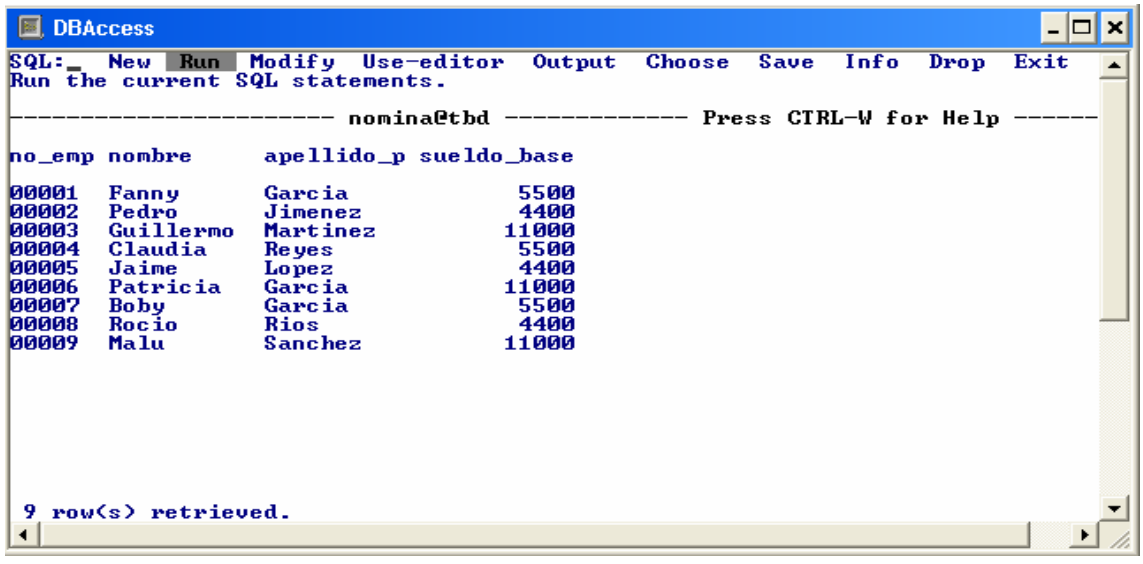


Se debe tener cuidado con el uso del metacaracter *, pues si se accesara el contenido de una tabla de miles o millones de registros, toda esta información queda en memoria y se puede tener problemas con el servidor o equipo cliente al saturarla, así como problemas de saturación de la red al viajar todos los datos de la consulta.

Ejercicio 2. SELECT simple específico. Accesar no_emp, nombre, apellido paterno y sueldo base de todos los empleados.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, sueldo_base
FROM empleados
```

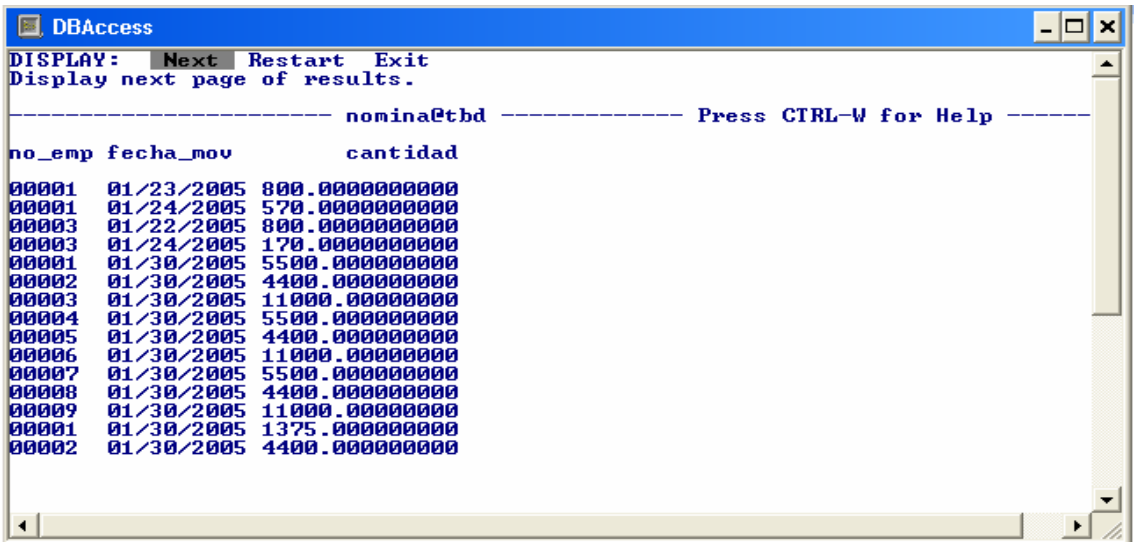


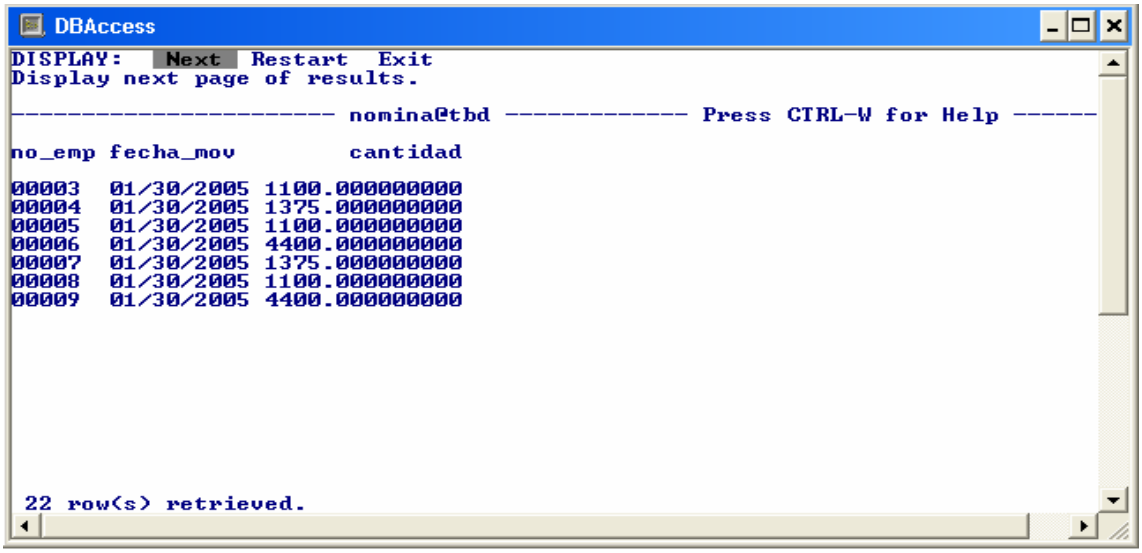
El acceso es de todos los registros de la tabla, pero sólo los campos especificados. Se van desplegando de acuerdo a como se encuentran en la tabla.

Ejercicio 3. SELECT simple con WHERE. Generar un reporte de los movimientos de la segunda quincena, desplegando sólo no_emp, fecha del mismo y cantidad.

Ejecuta Query >>

```
SELECT no_emp, fecha_mov, cantidad
FROM movimientos
WHERE no_quincena=2
```





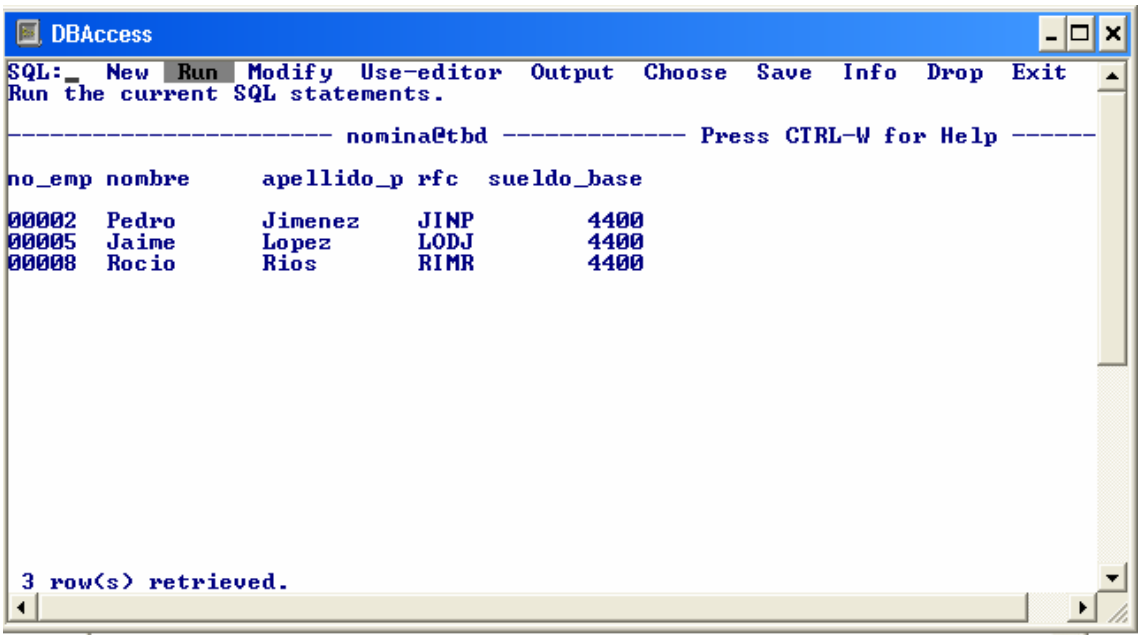
Este es un Query específico, sólo se despliegan los registros que cumplan con la condición del WHERE.

Las dos pantallas anteriores son el resultado de la tabla de movimientos para la quincena 2. Recordar que la tabla MOVIMIENTOS tiene únicamente información del presente año. Se recomienda generar siempre queries muy específicos por el ahorro de recursos de cómputo.

Ejercicio 4. SELECT simple con WHERE, usando palabras clave y Operadores. Generar un reporte de todos los empleados que su sueldo se esté entre 1000 y 5000 inclusive. Desplegar sólo no_emp, nombre, apellido paterno, los primeros cuatro caracteres del RFC y el sueldo base.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, RFC[1,4], sueldo_base
FROM empleados
WHERE sueldo_base BETWEEN 1000 AND 5000
```



Este es un Query específico, sólo se despliegan los registros que cumplan con la condición del WHERE.

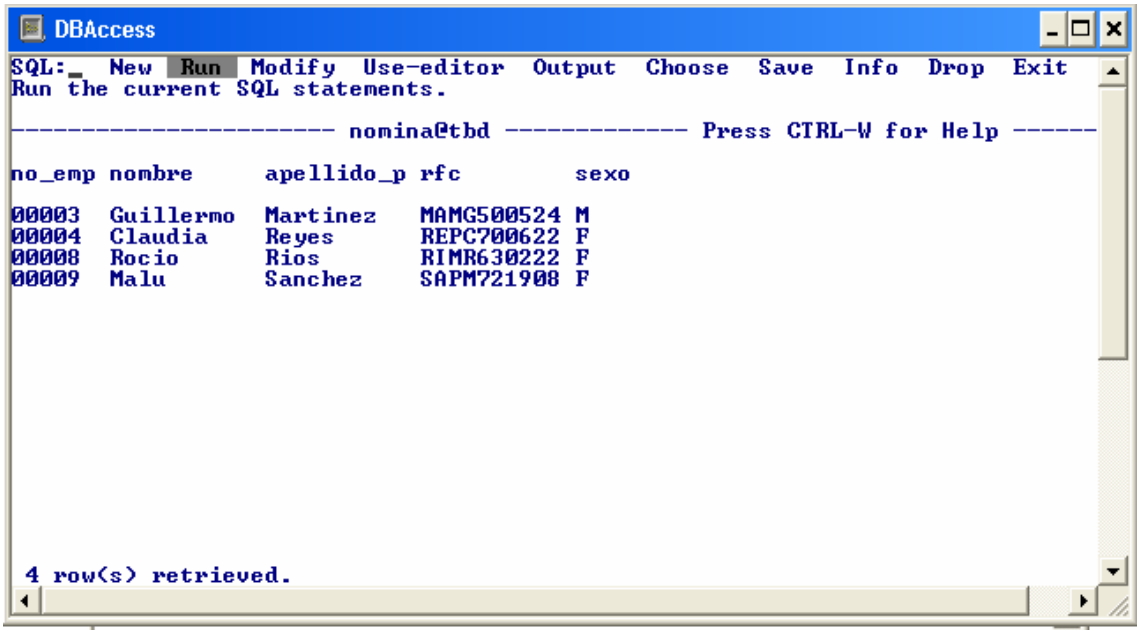
Se observa en este Query, para el RFC la aplicación de [ ] para elegir sólo una parte de string o cadena de caracteres; así también la palabra reservada BETWEEN para elegir rangos. Este también pudo realizarse con operadores <, >,=, AND.

Ejercicio 5. SELECT simple específico, usando MATCHES. Generar un reporte de todos los empleados que tengan el número "2" en cualquier parte del RCF. Desplegar no_emp, nombre, apellido paterno, RFC, sexo.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, RFC, sexo
FROM empleados
WHERE rfc matches "*2*"
```

Este es un Query específico, sólo se despliegan los registros que cumplan con la condición el WHERE.

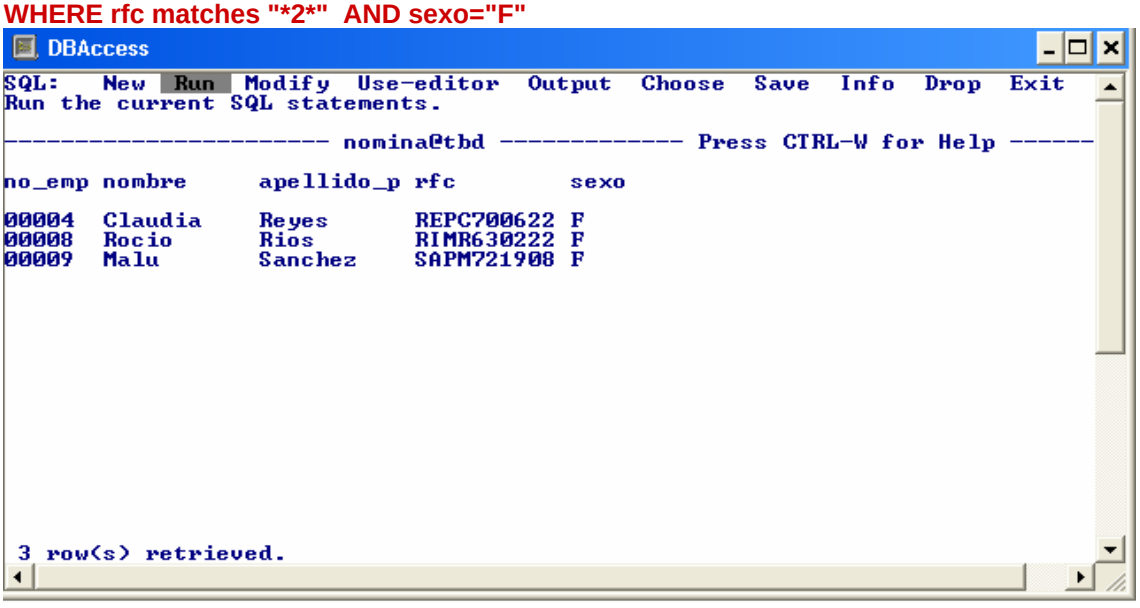


En este Query se observa el uso de metacaracteres para el MATCHES, la infinidad de accesos usando este estatuto dependerá de lo que el sistema requiera.

Ejercicio 6. SELECT simple específico, usando MATCHES y operadores. Generar un reporte de todos los empleados que tengan el número "2" en cualquier parte del RCF y que sean mujeres. Desplegar no_emp, nombre, apellido paterno, RFC, sexo.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, RFC, sexo
FROM empleados
WHERE rfc matches "*2*"  AND sexo="F"
```



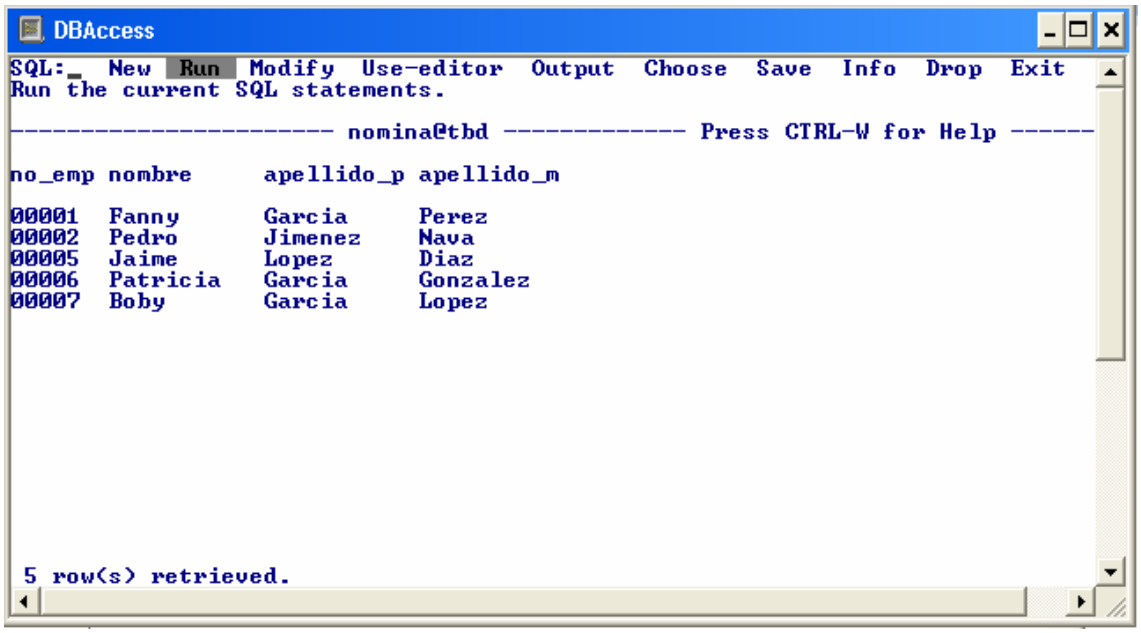
Este es un Query específico, sólo se despliegan los registros que cumplan con la condición el WHERE.

Se observa el uso de MATCHES y operador AND. La condición de WHERE puede ser tan compleja o simple como el sistema lo requiera. Sin embargo se pueden combinar todo tipo de operadores.

Ejercicio 7. SELECT simple específico, usando MATCHES y metacaracteres. Generar un reporte de todos los empleados cuyo apellido paterno inicie con cualquier letra entre A y L inclusive (mayúsculas). Desplegar no_emp, nombre, apellido paterno, apellido materno.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, apellido_m
FROM empleados
WHERE apellido_p MATCHES "[A-L]*"
```



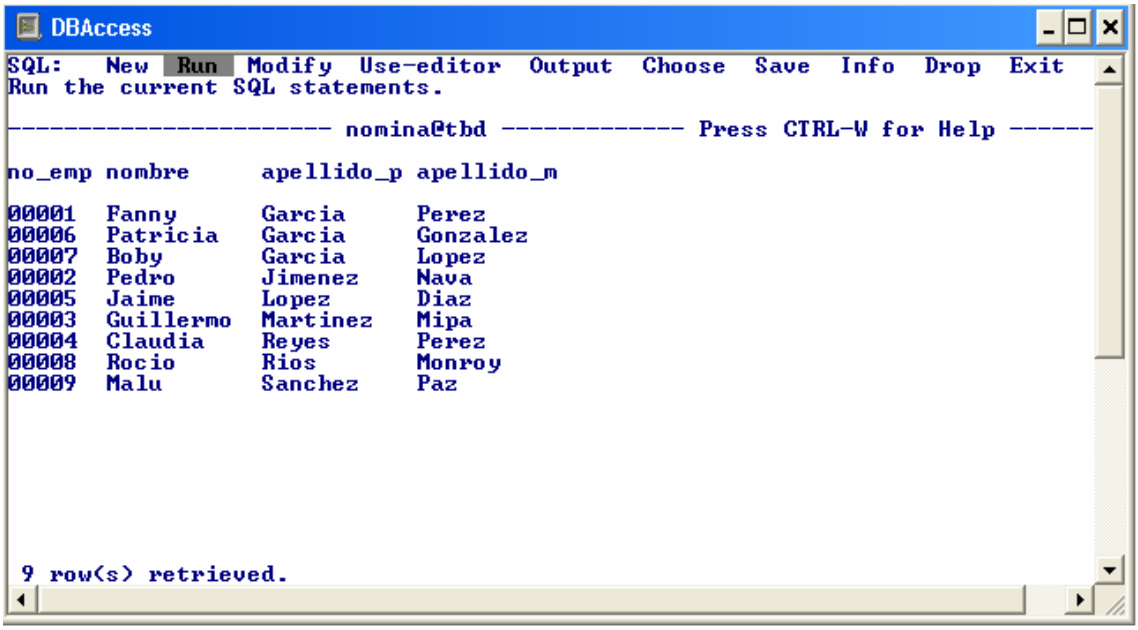
Este es un Query específico, sólo se despliegan los registros que cumplan con la condición el WHERE.

Se observa el uso de MATCHES y metacaracteres. La condición de WHERE usa la combinación alfabética [A-L]*. La condición de WHERE puede ser tan compleja o simple como el sistema lo requiera. Sin embargo se pueden combinar los caracteres alfabéticos o numéricos.

Ejercicio 8. SELECT simple específico, usando MATCHES y metacaracteres. Generar un reporte de todos los empleados cuyo apellido paterno inicie con cualquier letra entre A y L inclusive (mayúsculas ó minúsculas). Desplegar no_emp, nombre, apellido paterno, apellido materno. Ordenados por apellido paterno.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, apellido_m
FROM empleados
WHERE apellido_p MATCHES "[aA-lL]*"
ORDER BY apellido_p
```



Este es un Query específico, sólo se despliegan los registros que cumplan con la condición el WHERE. Se incluye el ORDER BY, que por default ordena los datos de forma ascendente por apellido paterno.

Se observa el uso de MATCHES y metacaracteres. La condición de WHERE usa la combinación alfabética [aA-lL]*, a diferencia del anterior, incluye letras mayúsculas o minúsculas. La condición de WHERE puede ser tan compleja o simple como el sistema lo requiera. Sin embargo se pueden combinar los caracteres alfabéticos o numéricos.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade) para el acceso de información.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Genera un reporte de todos los empleados, incluyendo no_emp, nombre, apellido paterno y suelo base.
2. Genera un reporte de los empleados que su nombre inicie con A-C, D, P. Desplegar no_emp y nombre.
3. Genera un reporte de los empleados nacido entre 1960-1980. Desplegar no_emp, nombre, apellido paterno y fecha de nacimiento.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 11
Accesando información de la Base de Datos II

Objetivos

- Implementar consultas con el estatuto Select
- Consultar información aplicando las funciones de conjunto de registro como Min, Max, Count, Sum, AVG
- Consultas usando estatuto Where

Introducción

El estatuto SELECT sirve para accesar la información de la base de datos, esta parte es muy importante entender lo que el usuario requiere para darle lo que necesita, además de realizar queries óptimos, es decir, buen diseño y rendimiento de los recursos del servidor o equipo cliente.
Es por eso, que se trabajó fuertemente en las prácticas anteriores en desarrollar un buen diseño de base de datos, esto facilita la generación de queries. De otra manera los queries generados se harán cada vez más complejos e ineficientes, tendiendo que regresar al rediseño de la base o a programarlos con otras herramientas de desarrollo.

SINTAXIS

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]
[GROUP BY column-list] [HAVING condition]
[ORDER BY column-name [ASC | DESC],...]
[INTO TEMP table-name]

SELECT-statement UNION [ALL] SELECT-statement
[UNION [ALL] SELECT-statement ...]

WHERE conditions:

expr rel-op expr

expr [NOT] BETWEEN expr AND expr

expr [NOT] IN (items)

column-name [NOT] LIKE "string" [ESCAPE escape-character]

column-name [NOT] MATCHES "string" [ESCAPE escape-character]

expr rel-op {ALL | [ANY | SOME]} (SELECT-statement)

expr [NOT] IN (SELECT-statement)

[NOT] EXISTS (SELECT-statement)

column-name IS [NOT] NULL

FUNCIONES

COUNT(*)
COUNT(DISTINCT campo)
COUNTO(campo)
SUM(campo)
AVG(campo/expresión)
MAX(campo/expresión)
MIN(campo/expresión)

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Funciones de conjunto de registros COUNT, SUM, AVG, MAX, MIN**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

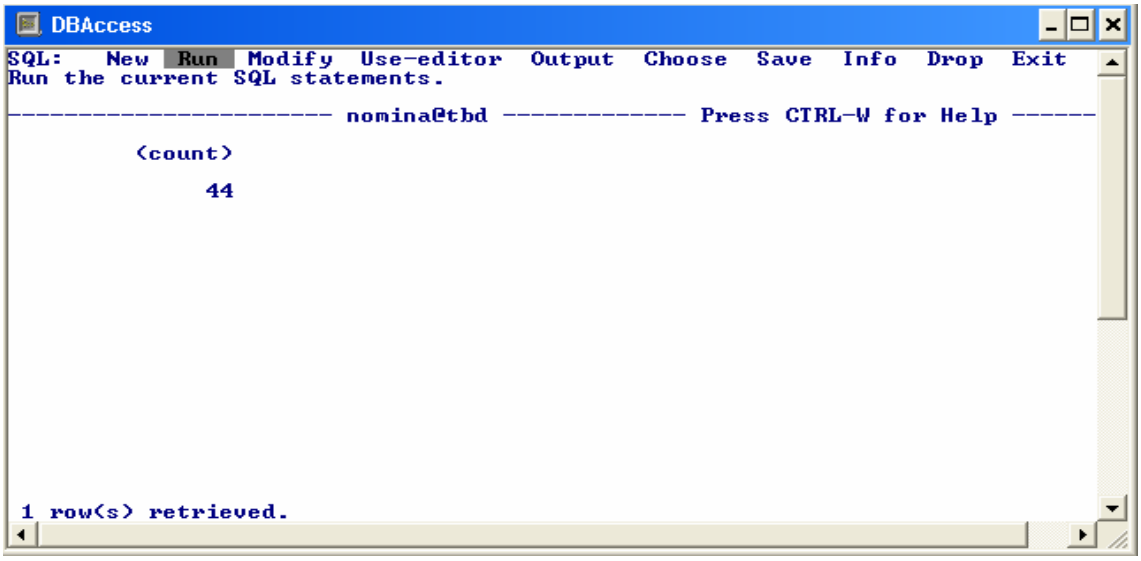
EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. En esta práctica de SELECT, se muestra el Query que se ejecutará vía dbaccess y el resultado directo, esto para mejorar el entendimiento de cada **Ejercicio**.

Ejercicio 1. SELECT simple usando COUNT. Generar un reporte que totalice el número de movimientos de la tabla de MOVIMIENTOS.

Ejecuta Query >>

```
SELECT count(no_quincena)
FROM movimientos
```

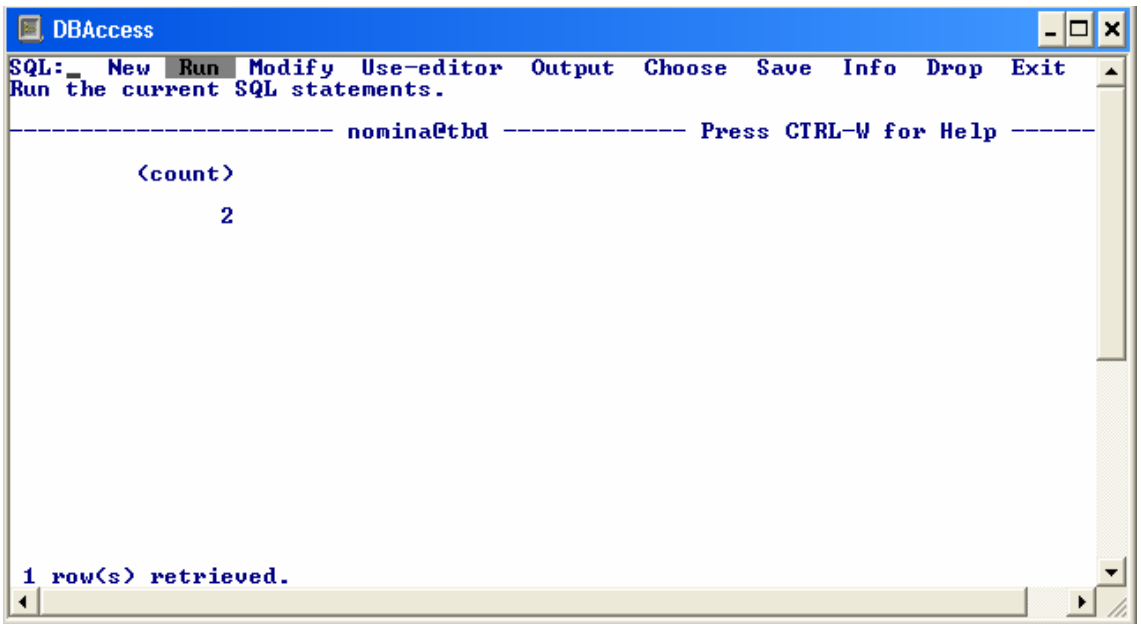
Este COUNT(no_quincena), obtiene el número de registro que se encuentran en la tabla. Funciona igual a un COUNT(*).



Ejercicio 2. SELECT simple usando COUNT(DISTINCT(campo)). Generar un reporte que totalice el número de quincenas registradas en la tabla de MOVIMIENTOS.

Ejecuta Query >>

```
SELECT count(distinct(no_quincena))
FROM movimientos;
```

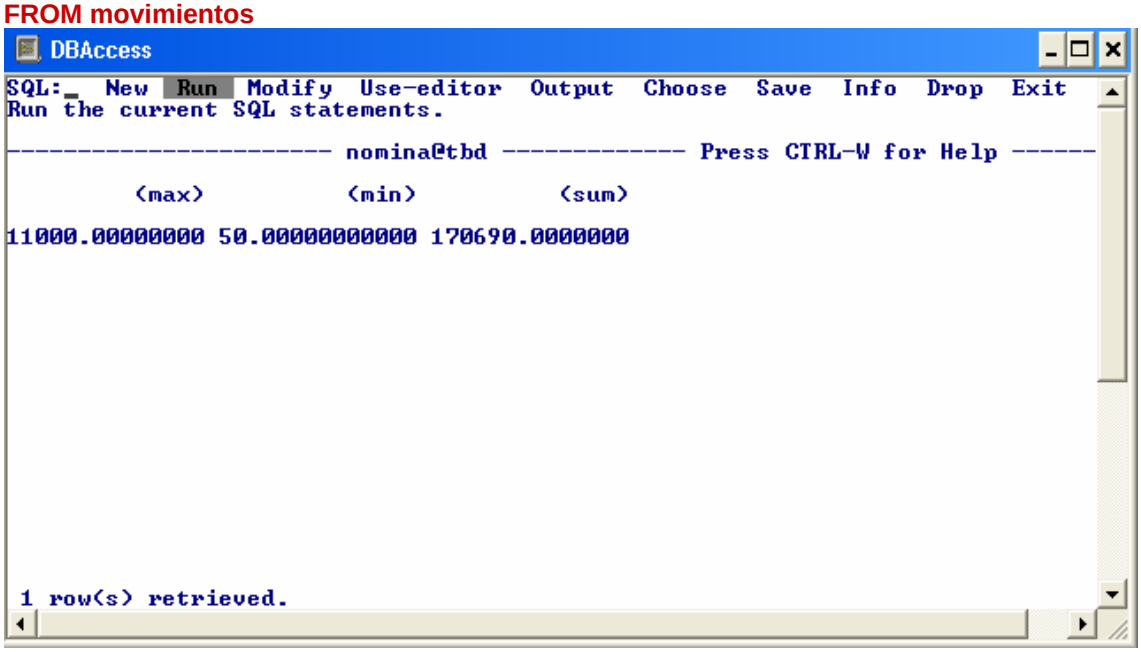


Este COUNT(no_quincena), obtiene el número de registro que se encuentran en la tabla. Aquí es importante especificar el campo, pues es el reporte que interesa en este caso, es el no. de quincenas.

Ejercicio 3. SELECT simple usando MAX,MIN,SUM. Generar un reporte que obtenga la cantidad máxima y mínima, así como el total, de los movimientos generados.

Ejecuta Query >>

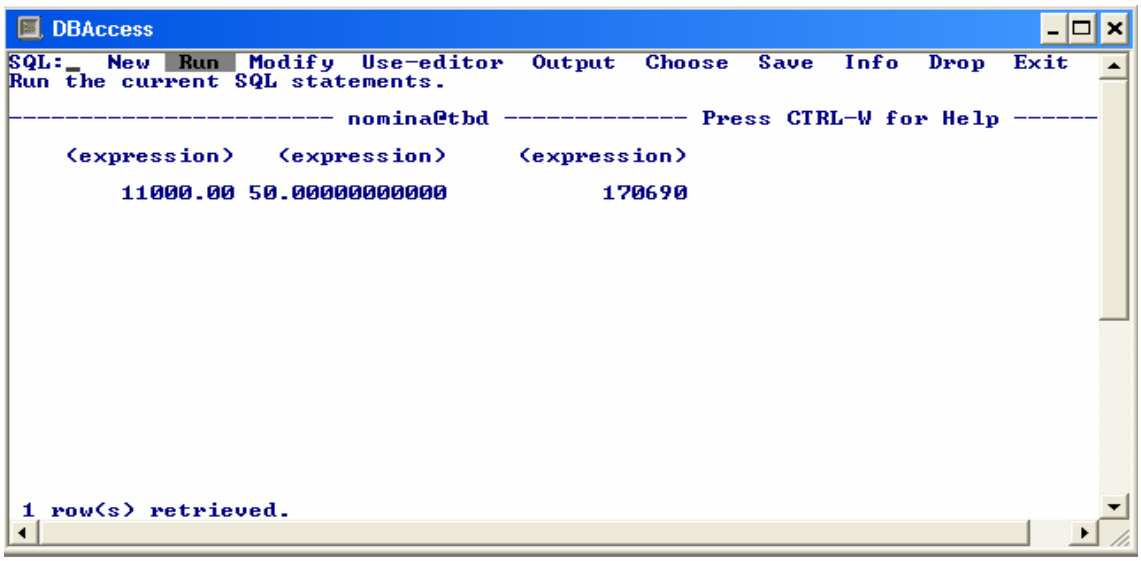
```
SELECT MAX(cantidad), MIN(cantidad), SUM(cantidad)
FROM movimientos
```



Estas funciones elijen al valor máximo y mínimo de todos los valores de la columna, así como la sumatoria de los valores de ese campo.

Ejercicio 3. SELECT simple usando más funciones. Observar lo que genera este reporte y registrarlo en el resultado de la práctica.

```
SELECT   ROUND(MAX(cantidad),2),ABS(MIN(cantidad)),
            TRUNC(SUM(cantidad))
FROM movimientos
```



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Genera un reporte que obtenga el total de percepciones, deducciones y neto que la empresa pago al año por todos los empleados.
2. Genera un reporte del número de empleados que trabajan en la empresa.
3. Genera un reporte del número de empleados que trabajan en la empresa, cuantas mujeres y cuantos hombres.
4. Genera un reporte que indique cuantos empleados tienen movimientos de Nomina, no importa en que quincena aparezcan sus movimientos. Desplegar cantidad de empleados.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 12
Accesando información de la Base de Datos III

Objetivos

- Implementar consultas con el estatuto Select
- Consultar información aplicando las funciones como Min, Max
- Consultar usando estatuto Where
- Consultar sobre una tabla

Introducción

El estatuto SELECT sirve para accesar la información de la base de datos, esta parte es muy importante entender lo que el usuario requiere para darle lo que necesita, además de realizar queries óptimos, es decir, buen diseño y rendimiento de los recursos del servidor o equipo cliente. Es por eso, que se trabajó fuertemente en las prácticas anteriores en desarrollar un buen diseño de base de datos, esto facilita la generación de queries. De otra manera los queries generados se harán cada vez más complejos e ineficientes, tendiendo que regresar al rediseño de la base o a programarlos con otras herramientas de desarrollo.

SINTAXIS

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]
[GROUP BY column-list] [HAVING condition]
[ORDER BY column-name [ASC | DESC],...]
[INTO TEMP table-name]

SELECT-statement UNION [ALL] SELECT-statement
[UNION [ALL] SELECT-statement ...]

WHERE conditions:
expr rel-op expr

expr [NOT] BETWEEN expr AND expr

expr [NOT] IN (items)

column-name [NOT] LIKE "string" [ESCAPE escape-character]

column-name [NOT] MATCHES "string" [ESCAPE escape-character]

expr rel-op {ALL | [ANY | SOME]} (SELECT-statement)
expr [NOT] IN (SELECT-statement)
[NOT] EXISTS (SELECT-statement)

column-name IS [NOT] NULL

FUNCIONES DE TIEMPO

DAY MONTH WEEKDAY YEAR DATE MDY CURRENT TODAY

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Agregación GROUP BY, HAVING**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

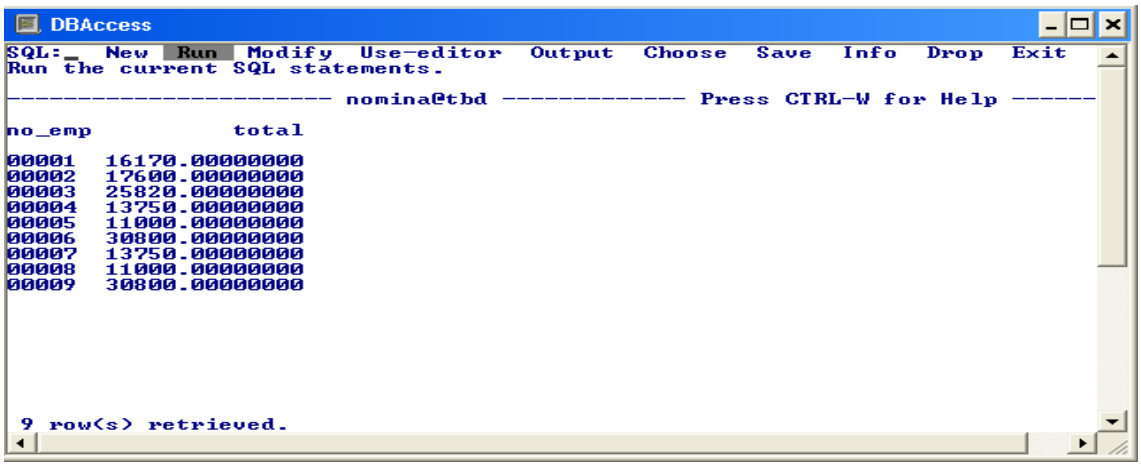
Metodología

EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. En esta práctica de SELECT, se muestra el Query que se ejecutará via dbaccess y el resultado directo, esto para mejorar el entendimiento de cada **Ejercicio**.

Ejercicio 1. SELECT simple usando GROUP BY. Generar un reporte que obtenga la cantidad total en pesos, de todos los movimientos de cada empleado. Deberá totalizar o hacer corte por no. de empleado. Desplegar no_emp y total.

Ejecuta Query >>

```
SELECT no_emp, SUM(cantidad) AS total
FROM movimientos
GROUP BY no_emp
```



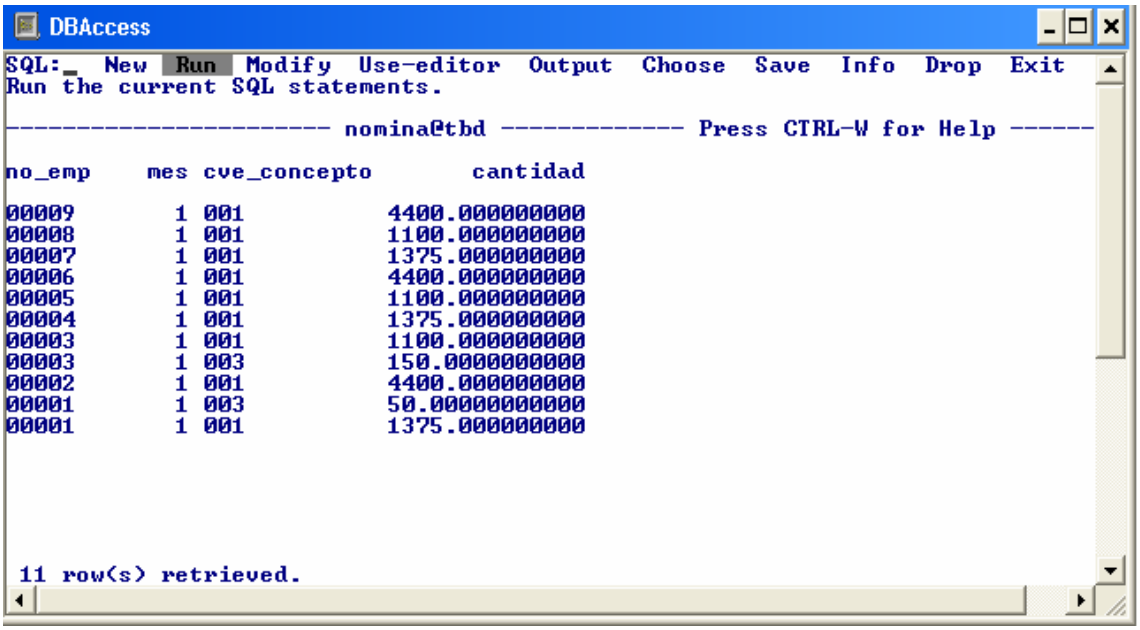
El GROUP BY normalmente va acompañado de funciones de totalización, como: COUNT, SUM,etc., este estatuto realiza sumatorias y cortes dependiendo del campo especificado. Siempre se incluirán los campos del GROUP BY todos los campos del SELECT, excepto los que sean calculados.

El GROUP BY no realiza ningún ordenamiento, va desplegando la información conforme se encuentre en la tabla.

Ejercicio 2. SELECT simple usando ORDER BY. Generar un reporte que obtenga para la quincena 1 y las claves de concepto 001 y 003, los movimientos por empleado. Incluir no_emp, mes del movimiento, cve_concepto y cantidad. Deberán estar ordenados en forma descendente por el número de empleado.

Ejecuta Query >>

```
SELECT no_emp, MONTH(fecha_mov) AS mes,cve_concepto, cantidad
FROM movimientos
WHERE no_quincena=1 AND cve_concepto IN ("001","003")
ORDER BY no_emp DESC
```

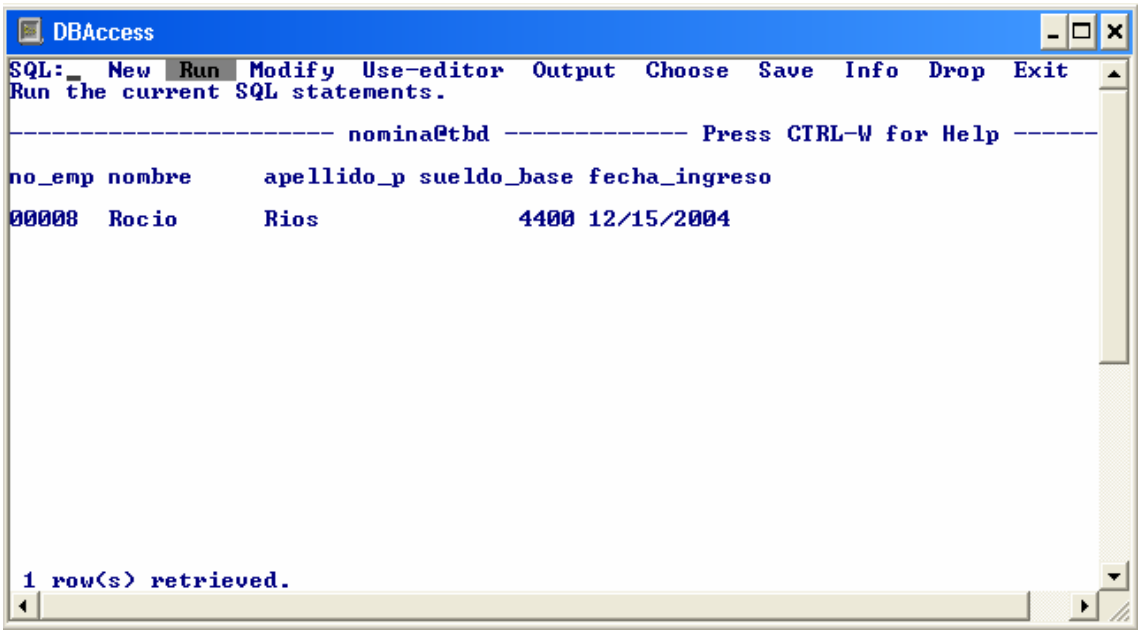


Este es un Query específico, sólo se despliegan los registros que cumplan con la condición el WHERE. Se incluye el ORDER BY, que por default ordena los datos de forma ascendente, por eso se agrega la opción DESC.

Ejercicio 3. SELECT simple usando funciones de Tiempo. Generar un reporte de empleados, que su mes de ingreso sea igual al presente mes. Desplegar no_emp, nombre, apellido paterno, sueldo base y fecha de ingreso.

Ejecuta Query >>

```
SELECT no_emp, nombre, apellido_p, sueldo_base, fecha_ingreso
FROM empleados
WHERE MONTH(fecha_ingreso)=MONTH(TODAY)
```

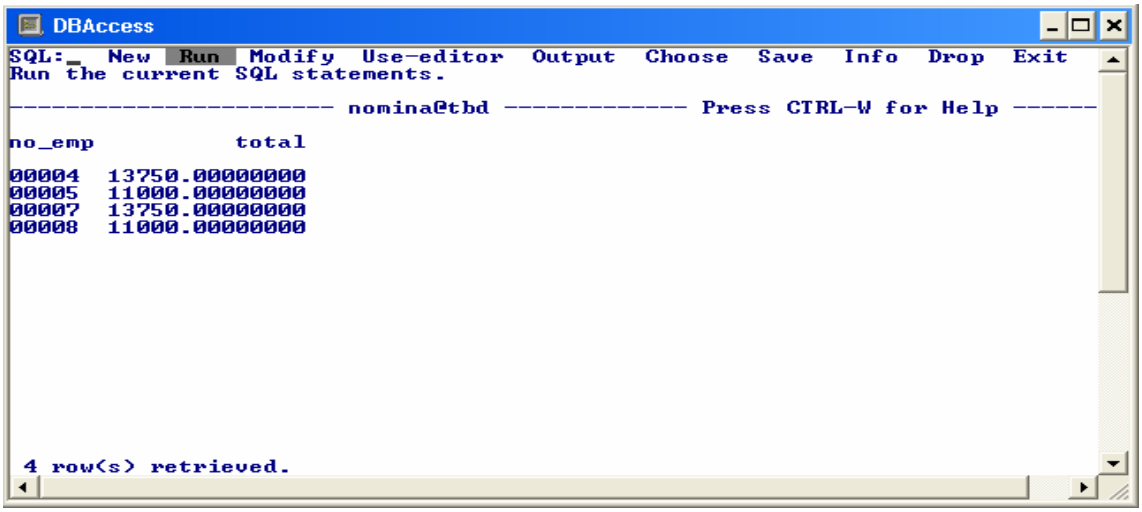


En esta práctica se muestra la aplicación de funciones de Tiempo. La combinación de ellas y su uso, dependerá de los requerimientos del sistema.

Ejercicio 4. SELECT simple usando HAVING. Generar un reporte todos los movimientos de cada empleado que su total sea menor a 15000, totalizando por empleado la cantidad. Desplegar no_emp y el total.

Ejecuta Query >>

```
SELECT no_emp, SUM(cantidad) AS total
FROM movimientos
GROUP BY no_emp
HAVING SUM(cantidad) < 15000
```



La función de HAVING, hace que sólo aquellos movimientos que totalicen menos de 15000 sean desplegados. El GROUP BY procesará todos pero el HAVING restringe cuales serán desplegados. Normalmente estas dos funciones van combinadas.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Genera un reporte que obtenga por empleado el total de sus deducciones, percepciones y neto del año actual. Desplegar No_emp, tot_percepciones, tot_deducciones, tot_neto.
2. Genera un reporte que obtenga por quincena el total de sus deducciones, percepciones y neto del año actual. Desplegar No_quincena, tot_percepciones, tot_deducciones, tot_neto.
3. Genera un reporte que totalice la cantidad de empleados hombre y mujeres.
4. Genere un reporte de empleados por sexo. Desplegar no_emp, nombre, sexo. Ordenados por sexo.
5. Genera un reporte de empleados por departamento. Ordenados por no_emp.
6. Genere un reporte por departamento indicando el total de empleados que trabajan en cada departamento. Desplegar cve_depto, total_empleados. Ordenados por departamento.
7. Genera un reporte que obtenga el total de percepciones, deducciones y neto que la empresa pago al año, el reporte debe realizar totales por mes.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 13
Accesando información de la Base de Datos IIII

Objetivos

- Implementar consultas con el estatuto Select
- Consultar información aplicando las funciones como Min, Max
- Consultar usando estatuto Where
- Consultar sobre múltiples tablas

Introducción

El estatuto SELECT sirve para accesar la información de la base de datos, esta parte es muy importante entender lo que el usuario requiere para darle lo que necesita, además de realizar queries óptimos, es decir, buen diseño y rendimiento de los recursos del servidor o equipo cliente. Es por eso, que se trabajó fuertemente en las prácticas anteriores en desarrollar un buen diseño de base de datos, esto facilita la generación de queries. De otra manera los queries generados se harán cada vez más complejos e ineficientes, tendiendo que regresar al rediseño de la base o a programarlos con otras herramientas de desarrollo.

Importante. SELECT sobre múltiples tablas. Es importante considerar que si estamos relacionando dos o más tablas se deben considerar las definiciones de integridad referencial. Cuando relacionamos dos tablas(A,B) al menos en el estatuto WHERE debe haber una condición que relacione a esas dos tablas A-B. Si relacionamos tres tablas(A, B, C) al menos en el WHERE deben existir dos condiciones, relacionando las tablas A-B, A-C.

SINTAXIS

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]
[GROUP BY column-list] [HAVING condition]
[ORDER BY column-name [ASC | DESC],...]
[INTO TEMP table-name]

SELECT-statement UNION [ALL] SELECT-statement
[UNION [ALL] SELECT-statement ...]

WHERE conditions:
expr rel-op expr
expr [NOT] BETWEEN expr AND expr

expr [NOT] IN (items)

column-name [NOT] LIKE "string" [ESCAPE escape-character]

column-name [NOT] MATCHES "string" [ESCAPE escape-character]

expr rel-op {ALL | [ANY | SOME]} (SELECT-statement)
expr [NOT] IN (SELECT-statement)
[NOT] EXISTS (SELECT-statement)
column-name IS [NOT] NULL

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Consultas sobre múltiples tablas**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

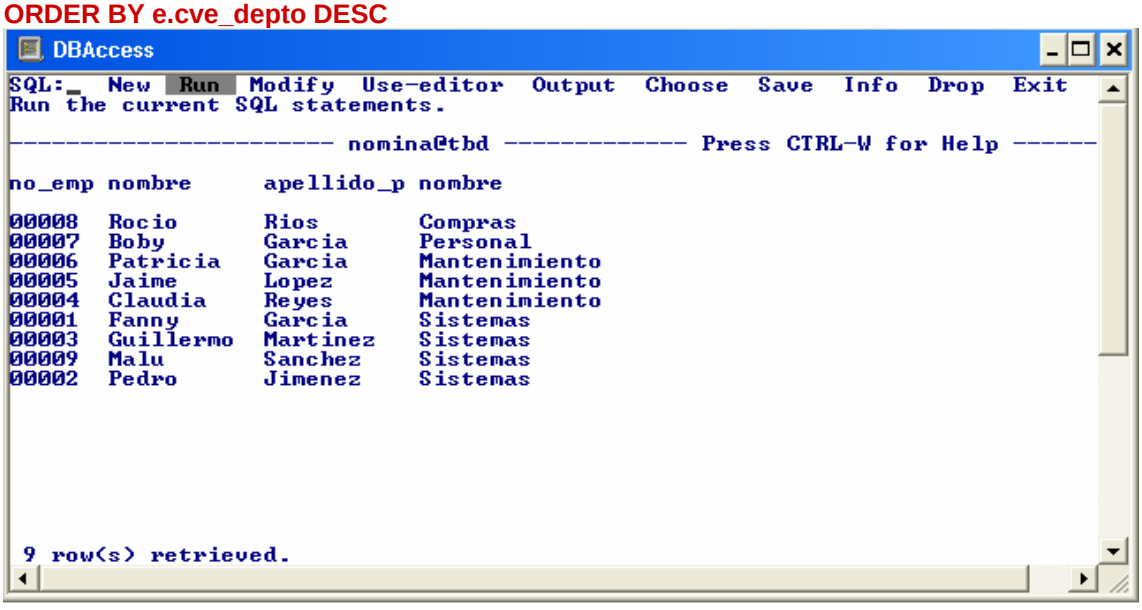
Metodología

EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. En esta práctica de SELECT, se muestra el Query que se ejecutará via dbaccess y el resultado directo, esto para mejorar el entendimiento de cada **Ejercicio**.

Ejercicio 1. SELECT sobre múltiples tablas. Generar un reporte de empleados por departamento. Desplegar no_emp, nombre, apellido paterno y nombre del departamento. Ordenar por cve_depto descendente.

Ejecuta Query >>

```
SELECT no_emp, e.nombre, apellido_p, d.nombre
FROM EMPLEADOS e, DEPARTAMENTOS d
WHERE e.cve_depto=d.cve_depto
ORDER BY e.cve_depto DESC
```

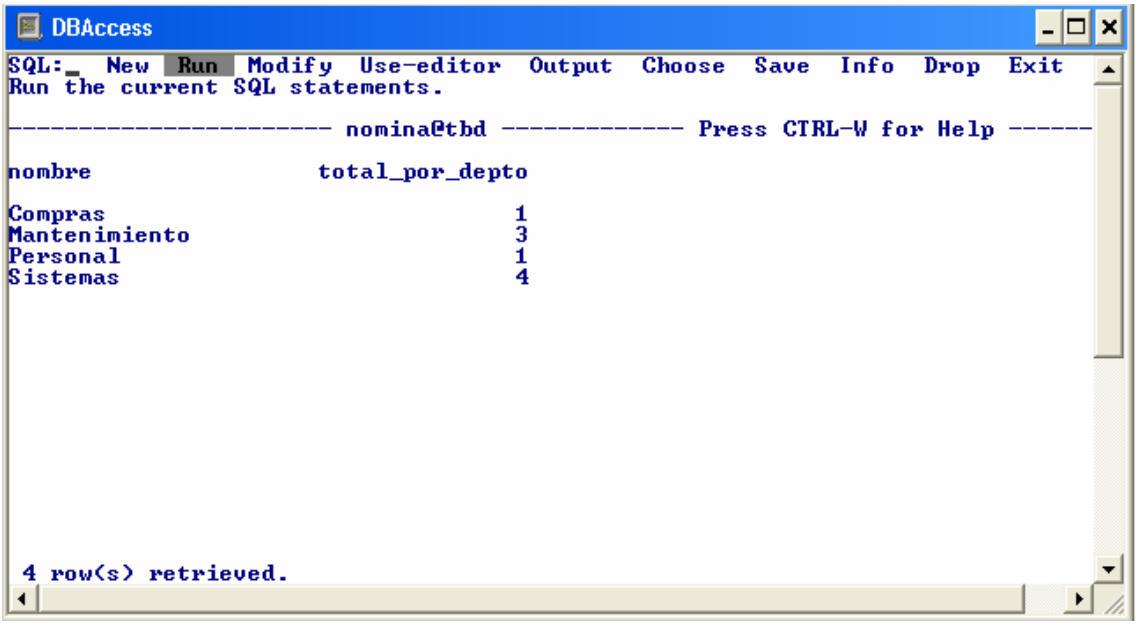


En este Query, se relaciona la tabla de EMPLEADOS Y DEPARTAMENTOS, pues los datos solicitados se encuentran en ambas tablas. Como se observa en la condición WHERE e.cve_depto=d.cve_depto, se establece la relación de integridad referencial para accesar sólo los registro requeridos.

Ejercicio 2. SELECT sobre múltiples tablas. Generar un reporte del total de empleados que trabajan en cada departamento. Desplegar nombre de departamento, total por depto. Ordenar por nombre de departamento.

Ejecuta Query >>

```
SELECT d.nombre, count(no_emp) AS Total_por_depto
FROM EMPLEADOS e, DEPARTAMENTOS d
WHERE e.cve_depto=d.cve_depto
GROUP BY d.nombre
ORDER BY d.nombre
```

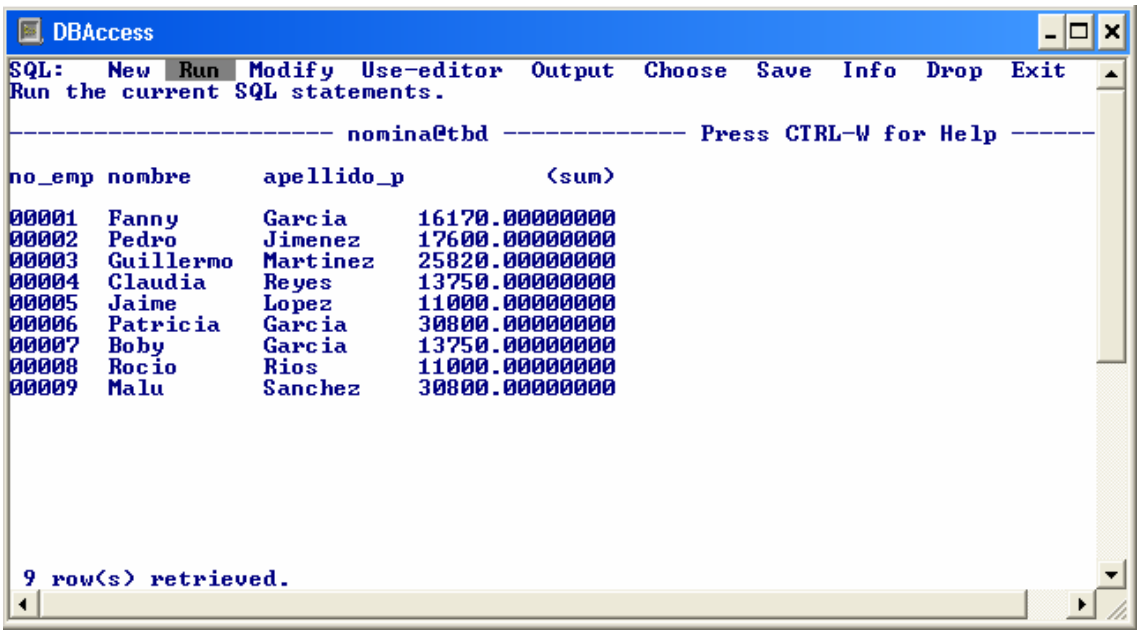


En este Query, se relaciona la tabla de EMPLEADOS Y DEPARTAMENTOS, pues los datos solicitados se encuentran en ambas tablas. Como se observa en la condición WHERE e.cve_depto=d.cve_depto, se establece la relación de integridad referencial para accesar sólo los registro requeridos, agregando un GROUP BY para totalizar por Departamento y ORDER BY.

Ejercicio 3. SELECT sobre múltiples tablas. Generar un reporte por empleado que incluya la cantidad total ($) de todos sus movimientos de nómina del año. Desplegar no_emp, nombre, apellido paterno, y total($). Ordenar por no_emp y nombre.

Ejecuta Query >>

```
SELECT e.no_emp, nombre, apellido_p, SUM(cantidad)
FROM EMPLEADOS e, MOVIMIENTOS m
WHERE e.no_emp=m.no_emp
GROUP BY e.no_emp, nombre, apellido_p
ORDER BY e.no_emp
```



En este Query, se relaciona la tabla de EMPLEADOS Y MOVIMIENTOS, pues los datos solicitados se encuentran en ambas tablas. Como se observa en la condición WHERE e.no_emp=m.no_emp, se asegura la relación de integridad referencial para accesar sólo los registro requeridos, agregando un GROUP BY para totalizar por empleadoy ORDER BY.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Genera un reporte por empleado del total de ingresos neto recibido por quincena. Desplegar no_emp, nombre, apellido paterno, no_quincena, total_neto. Ordenado por no_emp.
2. Genera un reporte por mes, y por quincena de la número de movimientos de nomina generados. Desplegar Mes, no_quincena, Total de movimientos.. Ordenado por mes.
3. Genere por empleado un reporte del total de deducciones, percepciones y neto recibidos en todo el año. Desplegar no_emp, nombre, apellido paterno, tot_perc, tot_deduc. Tot_neto. Ordenado por no_emp.
4. Genere un reporte que de acuerdo al sueldo base indique el porcentaje de descuento a aplicar. (Tabla Descuentos). Desplegar no_emp, sueldo base, porcentaje descto. Ordenado por no_emp.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 14
Accesando información de la Base de Datos V

Objetivos:

- Implementar consultas con el estatuto Select
- Consultar información aplicando las funciones como Min, Max
- Consultar usando estatuto Where
- Consultar sobre múltiples tablas
- Consultar usando estatutos Group By, Order by, Having

Introducción

El estatuto SELECT sirve para accesar la información de la base de datos, esta parte es muy importante entender lo que el usuario requiere para darle lo que necesita, además de realizar queries óptimos, es decir, buen diseño y rendimiento de los recursos del servidor o equipo cliente. Es por eso, que se trabajó fuertemente en las prácticas anteriores en desarrollar un buen diseño de base de datos, esto facilita la generación de queries. De otra manera los queries generados se harán cada vez más complejos e ineficientes, tendiendo que regresar al rediseño de la base o a programarlos con otras herramientas de desarrollo.

Importante. SELECT sobre múltiples tablas. Es importante considerar que si estamos relacionando dos o más tablas se deben considerar las definiciones de integridad referencial. Cuando relacionamos dos tablas(A,B) al menos en el estatuto WHERE debe haber una condición que relacione a esas dos tablas A-B. Si relacionamos tres tablas(A, B, C) al menos en el WHERE deben existir dos condiciones, relacionando las tablas A-B, A-C.

SINTAXIS

SELECT [ALL | DISTINCT | UNIQUE] select-list
FROM [OUTER] table-name [table-alias] [,...]
[WHERE condition]
[GROUP BY column-list] [HAVING condition]
[ORDER BY column-name [ASC | DESC],...]
[INTO TEMP table-name]

SELECT-statement UNION [ALL] SELECT-statement
[UNION [ALL] SELECT-statement ...]

WHERE conditions:
expr rel-op expr
expr [NOT] BETWEEN expr AND expr

expr [NOT] IN (items)
column-name [NOT] LIKE "string" [ESCAPE escape-character]

column-name [NOT] MATCHES "string" [ESCAPE escape-character]

expr rel-op {ALL | [ANY | SOME]} (SELECT-statement)
expr [NOT] IN (SELECT-statement)
[NOT] EXISTS (SELECT-statement)
column-name IS [NOT] NULL

Tema y subtema relacionado a cubrir

Tema: **Consultas y Lenguaje de Manipulación de Datos (DML)**
Subtema: **Consultas sobre múltiples tablas**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

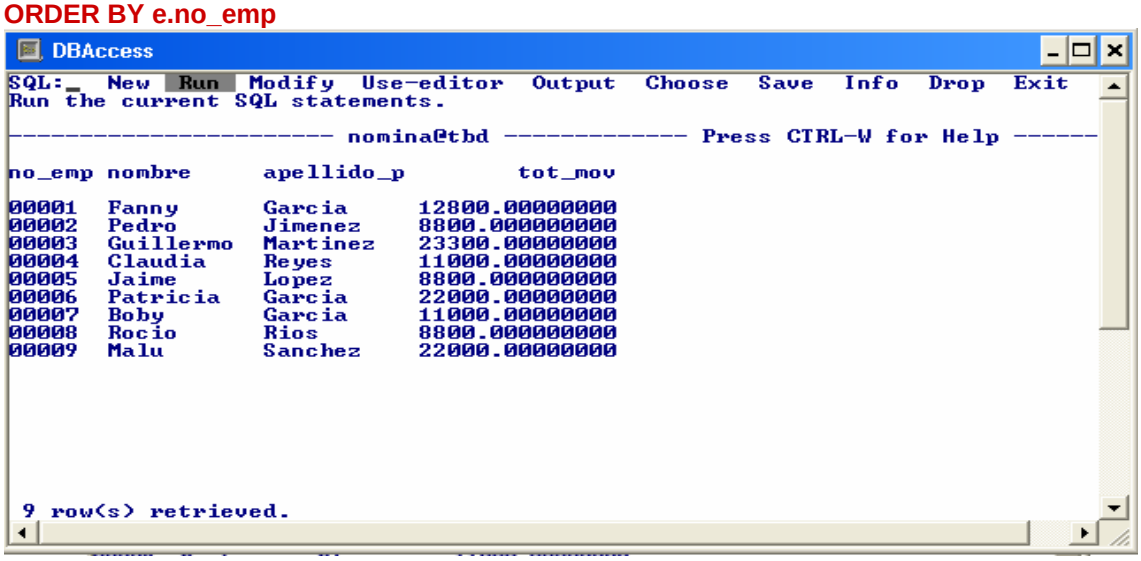
Metodología

EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. En esta práctica de SELECT, se muestra el Query que se ejecutará via dbaccess y el resultado directo, esto para mejorar el entendimiento de cada **Ejercicio**.

Ejercicio 1. SELECT de múltiples tablas. Generar un reporte por empleado, que incluya el total de movimientos ($) sólo de Percepciones. Desplegar no_emp, nombre, apellido paterno, total_mov($). Ordenado por empleado.

Ejecuta Query >>

```
SELECT e.no_emp, nombre, apellido_p, SUM(cantidad) tot_mov
FROM EMPLEADOS e, MOVIMIENTOS m, CONCEPTOS c
WHERE (e.no_emp=m.no_emp AND m.cve_concepto=c.cve_concepto)
        AND   (c.tipo_concepto="P")
GROUP BY e.no_emp, nombre, apellido_p
ORDER BY e.no_emp
```

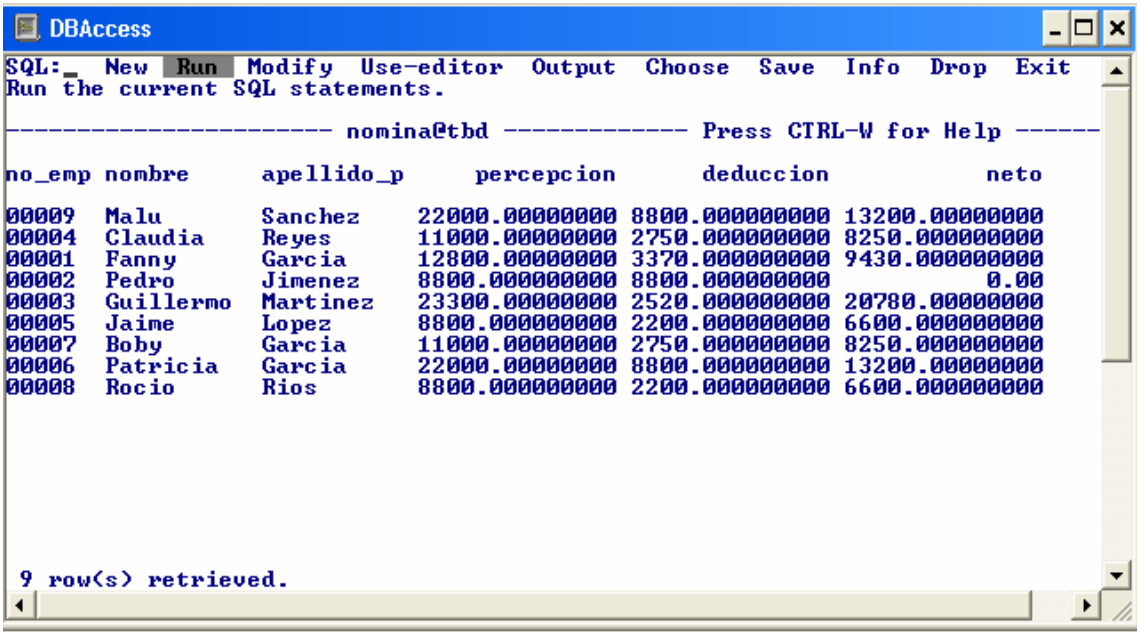


En este Query, se relacionan tres tablas EMPLEADOS, MOVIMIENTOS y CONCEPTOS, pues los datos solicitados se encuentran en ambas tablas. Por lo que al menos deben existir dos condiciones como se observa WHERE (e.no_emp=m.no_emp AND m.cve_concepto=c.cve_concepto) AND (c.tipo_concepto="P"), se asegura la relación de integridad referencial para accesar sólo los registro requeridos, agregando un GROUP BY para totalizar por empleado y ORDER BY.

Ejercicio 2. SELECT sobre múltiples tablas. Generar un reporte por empleado, que incluya el total de movimientos ($) de percepciones, deducciones y neto. Desplegar no_emp, nombre, apellido paterno, total_percep, total_deducc, total_neto.

Ejecuta Query >>

```
SELECT e.no_emp, nombre, apellido_p, sum(percepciones) AS
            percepcion,   sum(deducciones) AS deduccion,
           (sum(percepciones)-sum(deducciones))  AS neto
FROM EMPLEADOS e, NOMINA n
WHERE e.no_emp=n.no_emp
GROUP BY 1,2,3
```

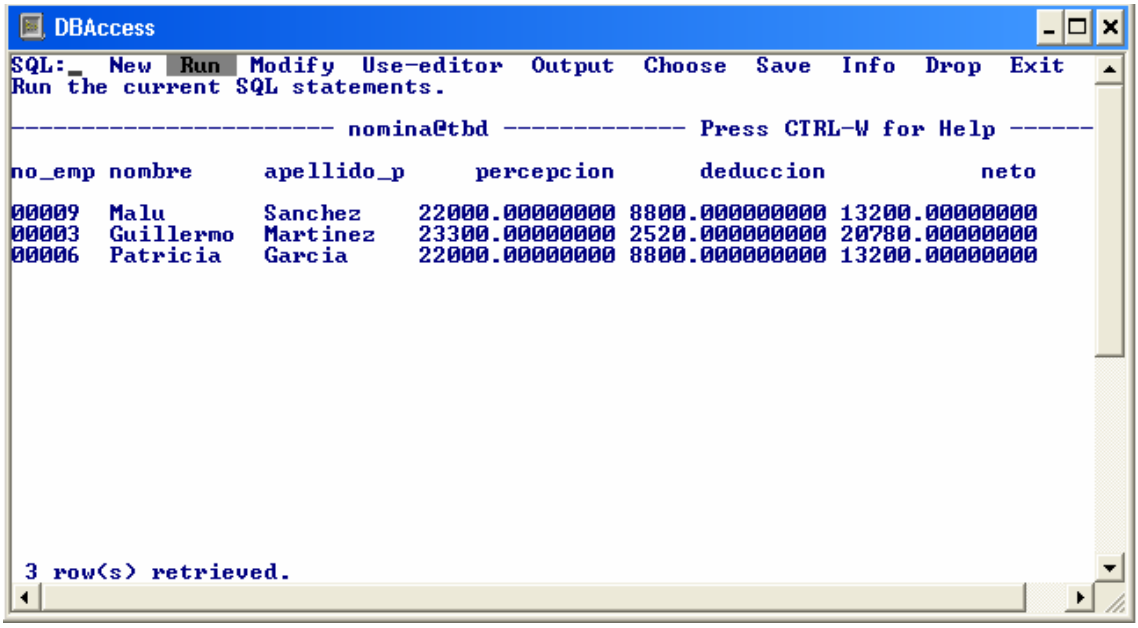


En este Query, se relacionan las tablas EMPLEADOS, NOMINA, pues los datos solicitados se encuentran en estas tablas. Por lo que al menos debe existir una condición como se observa WHERE e.no_emp=n.no_emp se asegura la relación de integridad referencial para accesar sólo los registros requeridos, agregando un GROUP BY para totalizar por empleado y se observa que se puede realizar la correspondencia de columnas del SELECT, reemplazando por números en el GROUP BY 1, 2, 3 respectivamente a los campos. También se observa el uso de campos calculados, sólo para ejemplificar, ya que pudo tomarse directamente el campo de Neto.

Ejercicio 3. SELECT sobre múltiples tablas. Generar un reporte por empleado, que incluya el total de movimientos ($) de percepciones, deducciones y neto. Desplegar no_emp, nombre, apellido paterno, total_percep, total_deducc, total_neto. El mismo Query del ejercicio anterior, pero sólo desplegará aquellos empelados donde el ingreso neto sea mayor a 10000.

Ejecuta Query >>

```
SELECT e.no_emp, nombre, apellido_p, sum(percepciones) AS
            percepcion,      sum(deducciones) AS deduccion,
            (sum(percepciones)-sum(deducciones))  AS neto
FROM EMPLEADOS e, NOMINA n
WHERE e.no_emp=n.no_emp
GROUP BY 1,2,3
HAVING SUM(percepciones)-sum(deducciones) > 10000
```



En este Query, se relacionan las tablas EMPLEADOS, NOMINA, pues los datos solicitados se encuentran en estas tablas. Por lo que al menos debe existir una condición como se observa WHERE e.no_emp=n.no_emp se asegura la relación de integridad referencial para accesar sólo los registros requeridos, agregando un GROUP BY para totalizar por empleado y se observa que se puede realizar la correspondencia de columnas del SELECT, reemplazando por números en el GROUP BY 1, 2, 3 respectivamente a los campos.

También se observa el uso de campos calculados, sólo para ejemplificar, ya que pudo tomarse directamente el campo de Neto. Se agrega HAVING para lograr la condición de desplegar sólo los ingresos netos mayores a 10000.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Genera para la quincena 2, el reporte de nomina incluyendo movimientos de percepción y deducción y neto. Desplegar no_emp, nombre, apellido paterno, percepiones(todas), deducciones (todas), total perp, total_deduc, neto. (El recibo de nómina del empleado). Ordenar por no_emp.
2. Genera reporte por empleado para quincena 1, indicando movimientos de nomina y si es percepción o deducción. Desplegar no_emp, nombre, apellido paterno, movimiento, D/P. Ordenado por no_emp.
3. Genera reporte por mes de todos los movimientos de nomina. Incluyendo por mes: Mes, no_emp, nombre, apellido paterno, cve_concepto, tipo_concepto, cantidad, total X mes. Ordenado por mes.
4. ¿Qué es lo que hace este QUERY?. Explica su función. ¿Qué diferencia con un JOIN?

```
SELECT NO_EMP ,NO_QUINCENA, SUM(CANTIDAD),TIPO_CONCEPTO
FROM MOVIMIENTOS M, CONCEPTOS C
WHERE M.CVE_CONCEPTO=C.CVE_CONCEPTO AND TIPO_CONCEPTO="P"
GROUP BY NO_EMP, NO_QUINCENA,TIPO_CONCEPTO
UNION
SELECT NO_EMP ,NO_QUINCENA, SUM(CANTIDAD),TIPO_CONCEPTO
FROM MOVIMIENTOS M, CONCEPTOS C
WHERE M.CVE_CONCEPTO=C.CVE_CONCEPTO AND TIPO_CONCEPTO="D"
GROUP BY NO_EMP, NO_QUINCENA,TIPO_CONCEPTO
```

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 15
Transacciones y sus propiedades

Objetivos

- Implementar bases de datos que trabajen con transacciones
- Entender el funcionamiento de las transacciones

Introducción

Una **transacción** es un conjunto de acciones que son ejecutadas en groupo, Si una de esas acciones no puede ser completada, todas las otras deben ser deshechas. Una transacción siempre tiene un inicio y un fin. El servidor de base de datos garantiza que las operaciones hechas dentro de los límites de la transacción sean completadas totalmente y perfectamente cerradas y escritas en disco, o si esto no es posible, la base de datos recuperará el estado que tenía antes de que la transacción iniciara.

Por ejemplo si algún cliente de un banco va a cobrar un cheque, para que la transacción esté completa, deben ejecutarse dos pasos:

- De la cuenta del dueño del cheque deberá descontar la cantidad del cheque
- De la cuenta de quien cobra el cheque debe sumarse la cantidad del cheque

Si alguna de estas acciones no se completa, la transacción no está completa y debe deshacerse. La transacción está completa si y sólo si, ambas acciones sean ejecutadas en su totalidad. Se muestran en la Sintaxis, los estatutos con los que se puede trabajar una transacción.

Es importante que la base de datos sea creada para manejar Transacciones, es decir, que sea del tipo Unbuffered o Buffered Log. También pudo crearse como: CREATE DATABASE nomina WITH LOG. Si no es creada de esta manera, las transacciones no funcionarán. Recordar que el uso de transacciones genera consistencia en la información.

SINTAXIS

BEGIN WORK

COMMIT WORK

ROLLBACK WORK

CREATE DATABASE database-name WITH LOG

Tema y subtema relacionado a cubrir

Tema: **Control de Transacciones**
Subtema: **Propiedades de la transacción**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.

2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

EL estudiante entregará los queries, el resultado de cada uno de ellos y la información solicitada en cada práctica. Para comprender mejor lo que es una transacción se darán ejemplos tipo, para que el estudiante capte mejor el concepto.

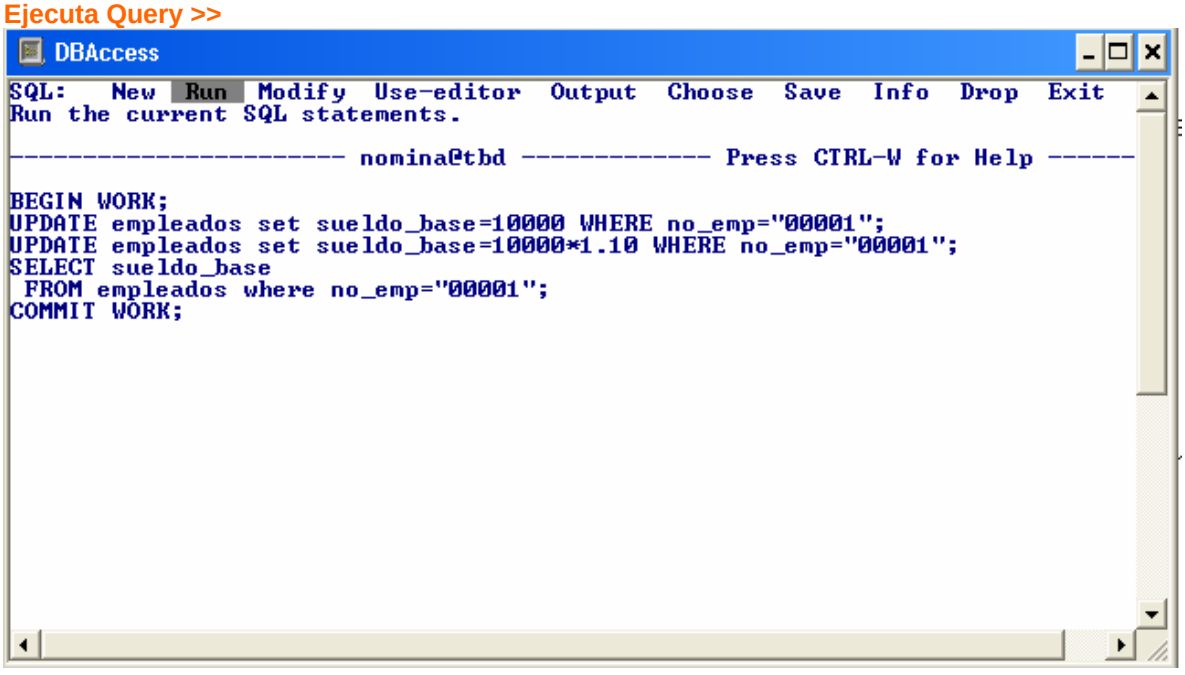
Ejercicio 1. **Transacción completada correctamente. Esquema de una transacción.**

BEGIN WORK;
Actualiza el sueldo base para empleado 00001;
Genera incremento del 10% de sueldo para empleado 00001;
Ver sueldo empleado 00001;
COMMIT WORK;

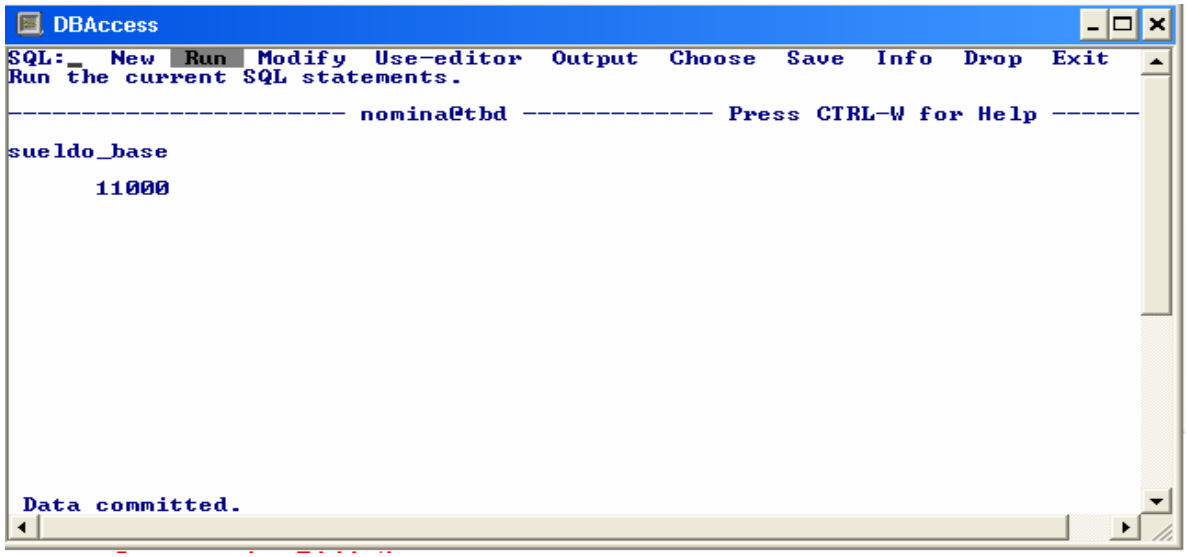
En este ejercicio al ejecutar el COMMIT, todas las acciones de la transacción fueron exitosamente ejecutadas.

Ejercicio 2. **Transacción completada correctamente, directo en SQL.**

Ejecuta Query >>

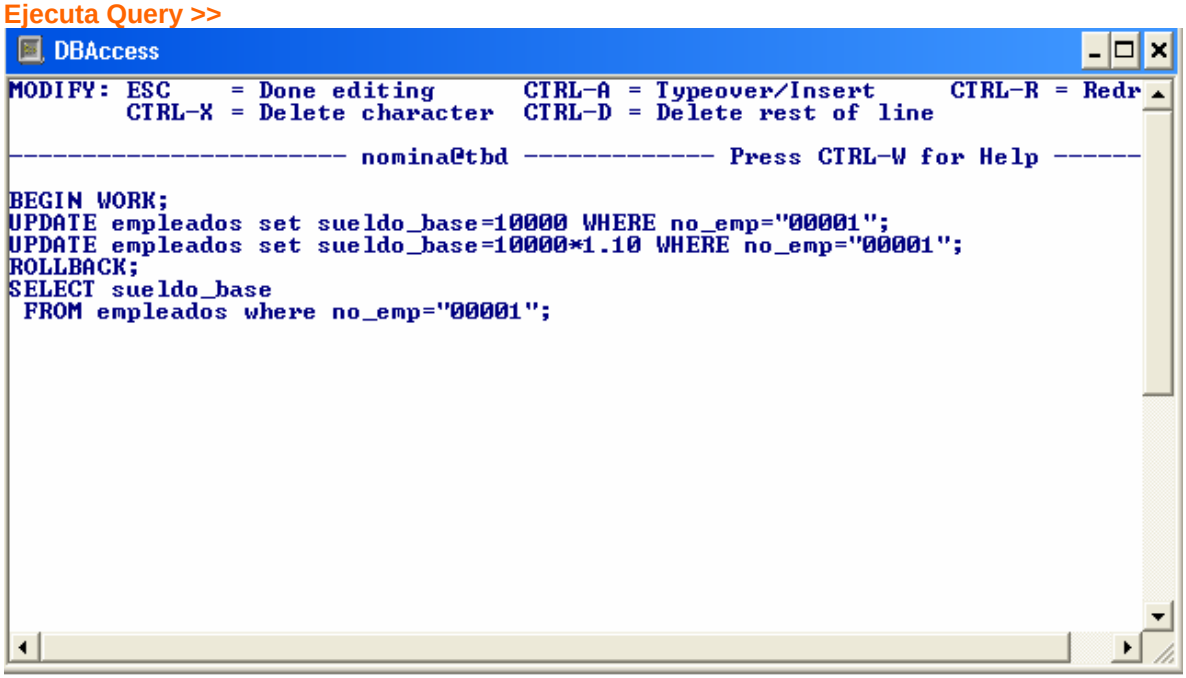


Resultado Query >> **En el resultado se refleja el incremento de sueldo y la transacción completada.**

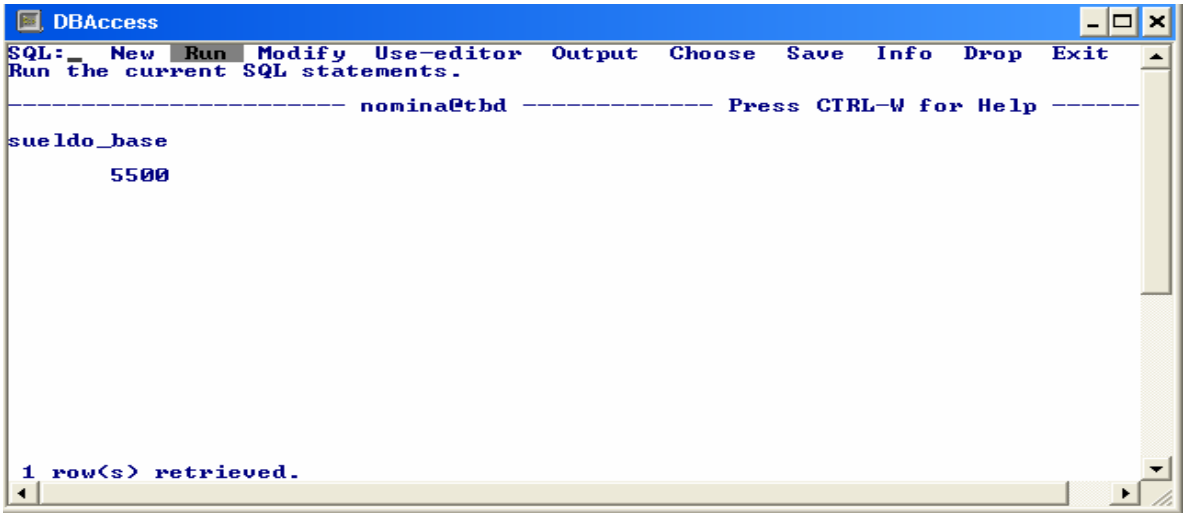


Ejercicio 3. **Transacción no completada correctamente.**

Ejecuta Query >>



Resultado Query >> **En el resultado se refleja que no logró completarse la transacción debido al ROLLBACK (que este puede significar cualquier error, esto se hace como demostración). Por lo que el sueldo se mantiene en 5500.**



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de Transacciones, niveles de aislamiento y grados de bloqueo (locks), para las diferentes consultas a realizar. Puede trabajar en más de una sesión para comprobar los resultados.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Qué es una transacción?.
2. ¿En SQL, que se requiere para armar una transacción?.
3. ¿Respecto a la base de datos, qué necesita para funcionar con transacciones y cómo se habilita?.
4. ¿Qué hacen los estatutos COMMIT, ROLLBACK; BEGIN?.
5. Para el ejercicio 3. ¿Cuál es el resultado si el ROLLBACK se encuentra entre los dos UPDATE?.
6. Para el ejercicio 3. ¿Cuál es el resultado si el ROLLBACK se encuentra al después del SELECT?.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 16
Transacciones y grados de consistencia

Objetivos

- Implementar los diferentes grados de consistencia de los datos en una base
- Apoyar la implementación con estatutos SQL
- Comprobar que se lleven a cabo los grados de consistencia de una transacción

Introducción

Una **transacción** es un conjunto de acciones que son ejecutadas en groupo, Si una de esas acciones no puede ser completada, todas las otras deben ser deshechas. Una transacción siempre tiene un inicio y un fin. El servidor de base de datos garantiza que las operaciones hechas dentro de los límites de la transacción sean completadas totalmente y perfectamente cerradas y escritas en disco, o si esto no es posible, la base de datos recuperará el estado que tenía antes de que la transacción iniciara.

Por ejemplo si algún cliente de un banco va a cobrar un cheque, para que la transacción esté completa, deben ejecutarse dos pasos:

- De la cuenta del dueño del cheque deberá descontar la cantidad del cheque
- De la cuenta de quien cobra el cheque debe sumarse la cantidad del cheque

Si alguna de estas acciones no se completa, la transacción no está completa y debe deshacerse. La transacción está completa si y sólo si, ambas acciones sean ejecutadas en su totalidad. Se muestran en la Sintaxis, los estatutos con los que se puede trabajar una transacción.

Es importante que la base de datos sea creada para manejar Transacciones, es decir, que sea del tipo Unbuffered o Buffered Log. También pudo crearse como: CREATE DATABASE nomina WITH LOG. Si no es creada de esta manera, las transacciones no funcionarán. Recordar que el uso de transacciones genera consistencia en la información.

Grados de consistencia.

Se refiere **al modo en que los datos se actualizan (aplica sólo para UPDATE; DELETE; INSERT)**, cuando varios usuarios están trabajando de manera concurrente sobre una tabla o los mismo datos. Qué reglas deben tomar para que la actualización sea exitosa sin generar bloqueos o deadlock en el acceso a los datos y así mantener la consistencia. Los grados de consistencia o bloqueo que se manejan en Informix son los siguientes:

Database Level. Este grado de bloqueo es a nivel base de datos. Se utiliza cuando se tienen muchas tablas actualizando o cuando se quiere realizar un respaldo. El bloqueo es de forma EXCLUSIVE.

Table Level. Este grado es a nivel tabla. Normalmente se aplica cuando no se desea que una tabla sea acezada por algún usuario. El acceso es de forma puede ser del tipo: EXCLUSIVE O SHARED.

Page Level. Este grado es a nivel página de datos. Esto es para no bloquear toda la tabla, pero si aseguro que si mi registro está en esta página, nadie más lo ocupe o actualice. Todos los registros dentro de la página también se bloquearán.

Row Level. Este grado es a nivel registro. Es el bloqueo a su mínimo nivel y aseguro sólo que el registro que se actualice no sea tocado por nadie.
Para cualquier tipo de grado de bloqueo es importante considerar los recursos de locks que se tienen definidos en el manejador de Base de Datos.

SINTAXIS

BEGIN WORK

COMMIT WORK

ROLLBACK WORK

ALTER TABLE table-name LOCK MODE PAGE / ROW

UNLOCK TABLE table-name

LOCK TABLE table-name IN {SHARE | EXCLUSIVE} MODE

DATABASE database-name EXCLUSIVE

CREATE DATABASE database-name WITH LOG

Tema y subtema relacionado a cubrir

Tema: **Control de Transacciones**
Subtema: **Grados de Consistencia**

Material y equipo necesario

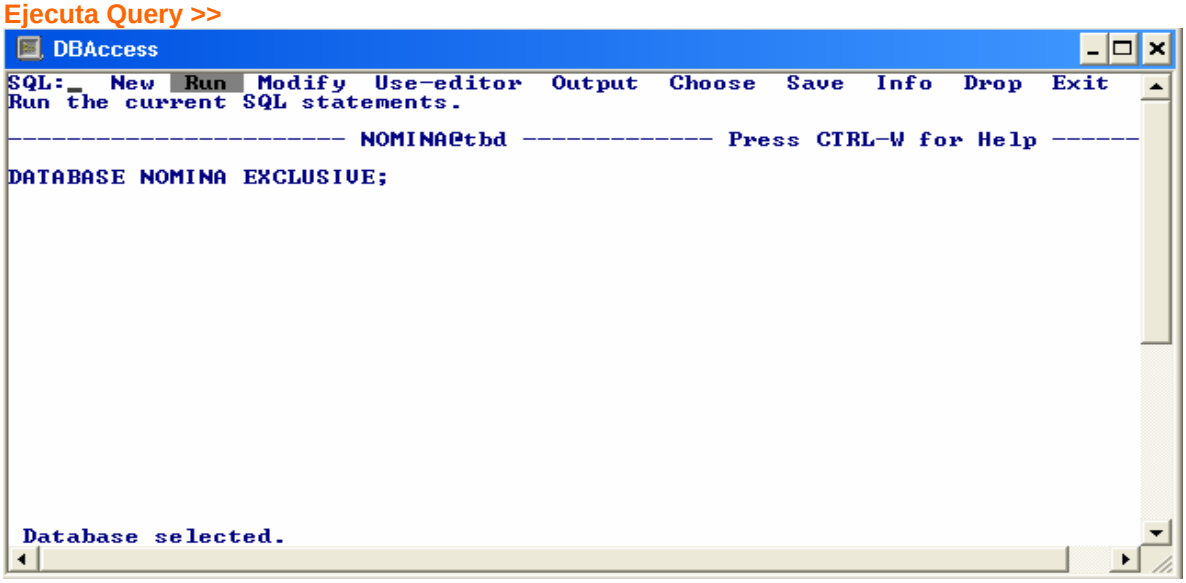
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

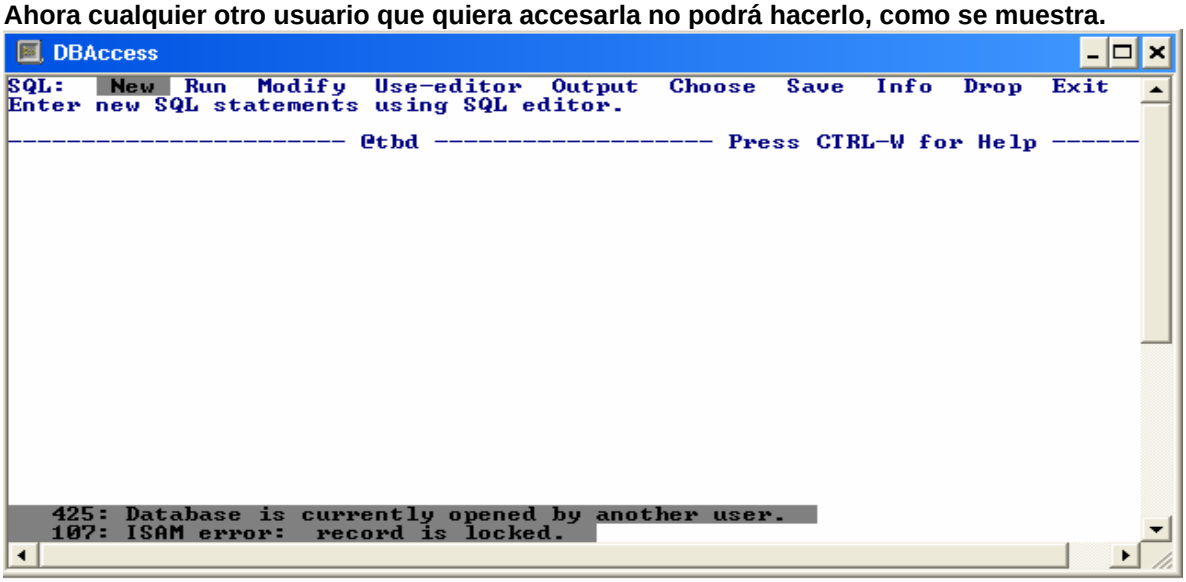
EL estudiante realizará y comprobará cada ejercicio de la práctica, el resultado de cada uno de ellos y la información solicitada.

Ejercicio 1. Bloquear la base de datos NOMINA y mostrar el bloqueo.

Ejecuta Query >>

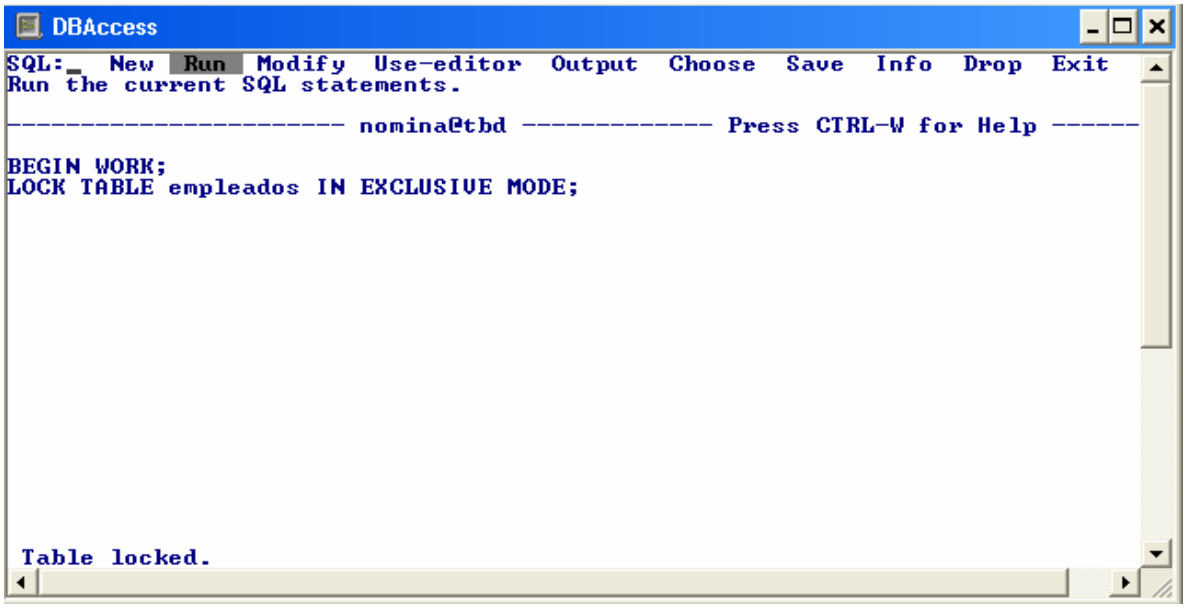


Ahora cualquier otro usuario que quiera accesarla no podrá hacerlo, como se muestra.

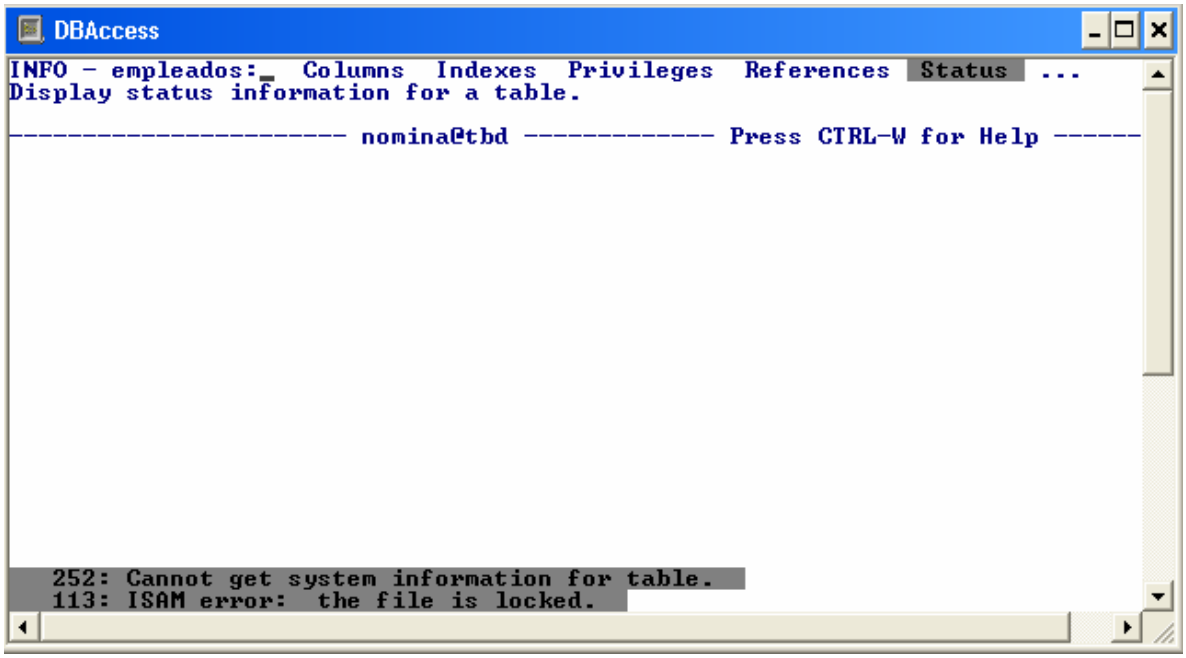


Ejercicio 2. Bloquear la tabla de empleados de modo exclusivo y mostrar el bloqueo.

Ejecuta Query >>



Es importante hacer notar, que la transacción se deja abierta para que la base quede bloqueada, pues si se aplica el COMMIT, desbloqueará todo. Ahora cualquier otro usuario que quiera accesarla no podrá hacerlo, como se muestra. Se intentó checar el estatus de la tabla EMPLEADOS, marcando error.



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de Transacciones, niveles de aislamiento y grados de bloqueo (locks), para las diferentes consultas a realizar. Puede trabajar en más de una sesión para comprobar los resultados.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Generar un bloqueo a nivel página y mostrar el error, cómo se realizó en la práctica.
2. Generar el desbloqueo del ejercicio anterior y mostrar sin error.
3. Generar un bloqueo a nivel registro y mostrar el error, cómo se realizó en la práctica.
4. Generar el desbloqueo del ejercicio anterior y mostrar sin error.
5. ¿Qué es el grado de consistencia o bloqueo y para que sirve?.
6. ¿Para que tipo de actualizaciones se aplican estos bloqueos?.
7. ¿De qué forma puede bloquearse una tabla y que ventajas tiene cada una?.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 17
Transacciones y niveles de Aislamiento

Objetivos:

- Implementar los diferentes niveles de aislamientos de datos en una base
- Apoyar la implementación con estatutos SQL
- Comprobar que se lleven a cabo los niveles de aislamiento de una transacción

Introducción

Una **transacción** es un conjunto de acciones que son ejecutadas en groupo, Si una de esas acciones no puede ser completada, todas las otras deben ser deshechas. Una transacción siempre tiene un inicio y un fin. El servidor de base de datos garantiza que las operaciones hechas dentro de los límites de la transacción sean completadas totalmente y perfectamente cerradas y escritas en disco, o si esto no es posible, la base de datos recuperará el estado que tenía antes de que la transacción iniciara.

Por ejemplo si algún cliente de un banco va a cobrar un cheque, para que la transacción esté completa, deben ejecutarse dos pasos:

- De la cuenta del dueño del cheque deberá descontar la cantidad del cheque
- De la cuenta de quien cobra el cheque debe sumarse la cantidad del cheque

Si alguna de estas acciones no se completa, la transacción no está completa y debe deshacerse. La transacción está completa si y sólo si, ambas acciones sean ejecutadas en su totalidad. Se muestran en la Sintaxis, los estatutos con los que se puede trabajar una transacción.

Es importante que la base de datos sea creada para manejar Transacciones, es decir, que sea del tipo Unbuffered o Buffered Log. También pudo crearse como: CREATE DATABASE nomina WITH LOG. Si no es creada de esta manera, las transacciones no funcionarán. Recordar que el uso de transacciones genera consistencia en la información.

Niveles de Aislamiento.

Se refiere al **modo en el se accesan o leen los datos de la base de datos (aplica sólo para SELECT)**, cuando varios usuarios están trabajando de manera concurrente sobre una tabla o los mismo datos. Qué información es la que accesa cada uno, dependerá del nivel de Aislamiento que esté definido. Informix trabaja con cuatro tipos de niveles de Aislamiento:

Dirty Read. En este tipo, no existe ningún nivel de aislamiento, no importa quien este consultando o cuantos usuarios, el acceso a los datos es directo.

Commited Read. Antes de leer el registro, garantiza que no tenga algún Lock exclusivo sobre la tabla o registro a ser consultado. Si no existe, aplica un Lock en modo Share o compartido para la lectura de sus datos. Es el nivel más bajo de asilamiento.

Cursor Stability. Es cuando tenemos múltiples registros a leer. Lo que se genera es un lock shared o compartido de cada registro que va leyendo y conforme pasa de un registro a otro va liberando el lock del anterior y lo aplica al siguiente. Esto es para asegurarse que la lectura del actual sea limpia y nadie pueda actualizarlo mientras termina su lectura.

Repeatable Read. Es el nivel más alto de aislamiento. Antes de leer los datos, aplica a todos un Shared lock o compartido y no lo libera hasta que termina la lectura de todos los datos. Asegurando la limpieza de todo el conjunto.

SINTAXIS

```
BEGIN WORK
COMMIT WORK
ROLLBACK WORK
SET ISOLATION TO {DIRTY READ | COMMITTED READ
      | CURSOR STABILITY | REPEATABLE READ}
CREATE DATABASE  database-name WITH LOG
```

Tema y subtema relacionado a cubrir

Tema: **Control de Transacciones**
Subtema: **Niveles de Aislamiento**

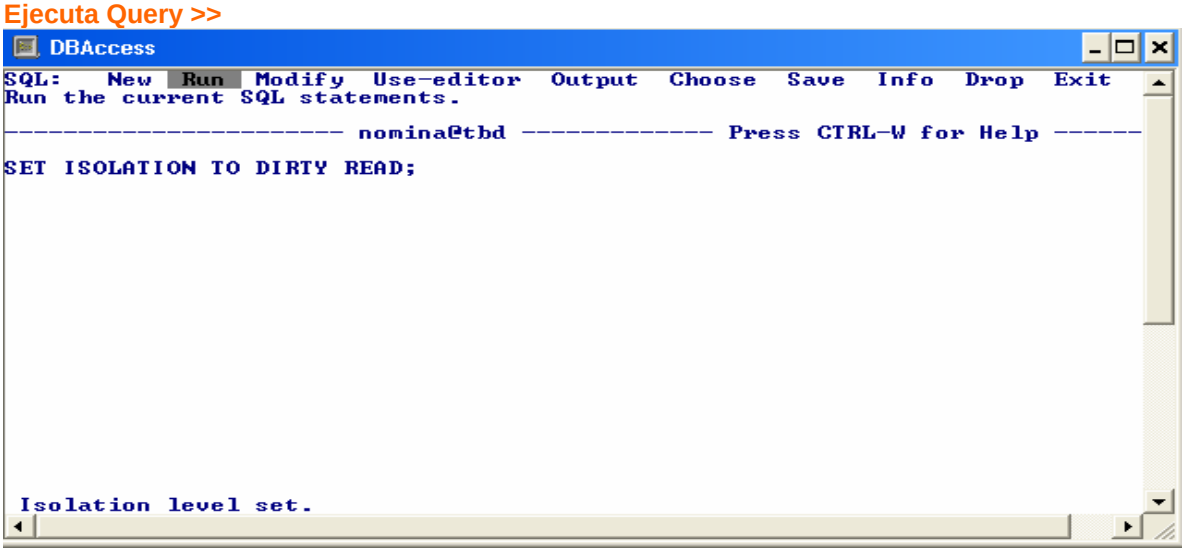
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

Ejercicio 1. Activar cada tipo de nivel de aislamiento. Dirty Read.

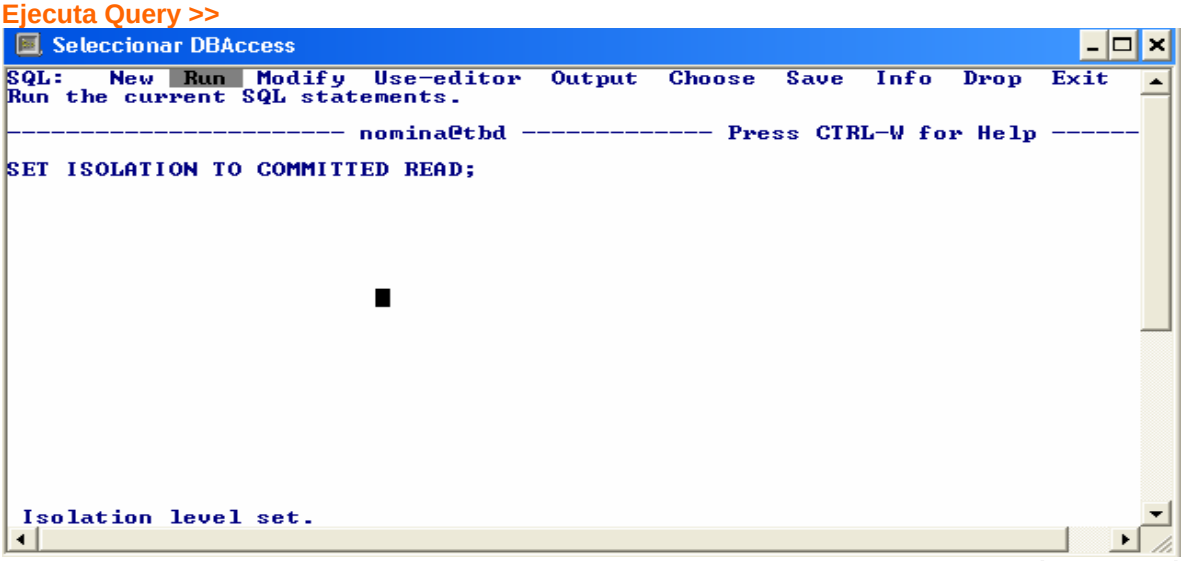
Ejecuta Query >>



Es importante recordar que el nivel de aislamiento se aplica dentro de una sesión y durará hasta que sea cambiado el tipo o la sesión sea terminada. El nivel de aislamiento se define y aplica sobre todos los queries de SELECT o lectura que se ejecuten después sobre la base de datos.

Ejercicio 2. Activar cada tipo de nivel de aislamiento. Commited Read.

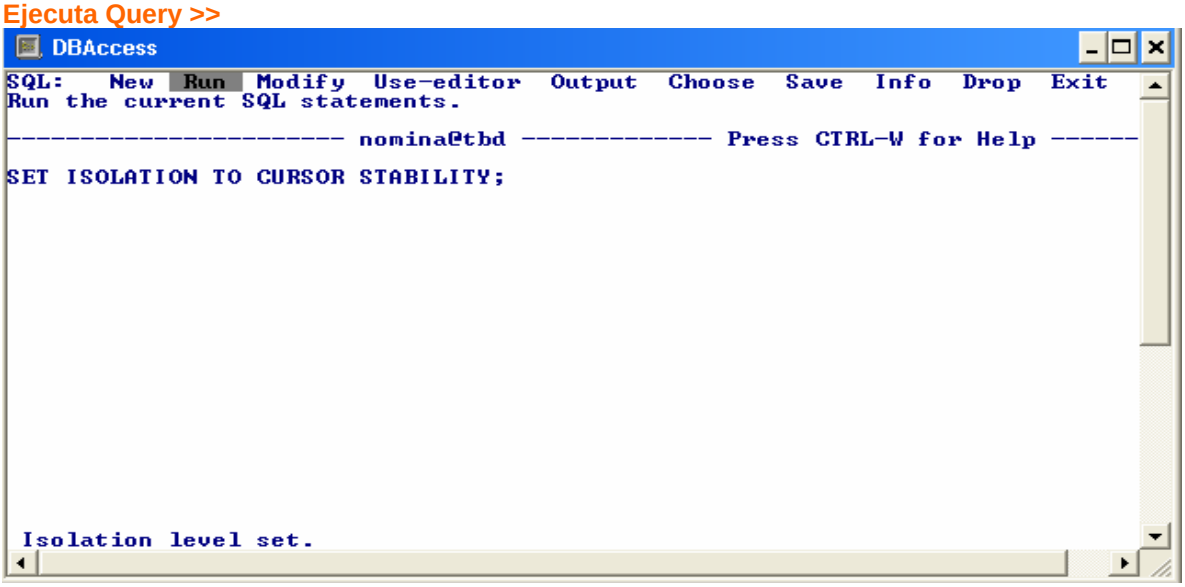
Ejecuta Query >>



Es importante recordar que el nivel de aislamiento se aplica dentro de una sesión y durará hasta que sea cambiado el tipo o la sesión sea terminada. El nivel de aislamiento se define y aplica sobre todos los queries de SELECT o lectura que se ejecuten después sobre la base de datos.

Ejercicio 3. Activar cada tipo de nivel de aislamiento. Cursor Stability.

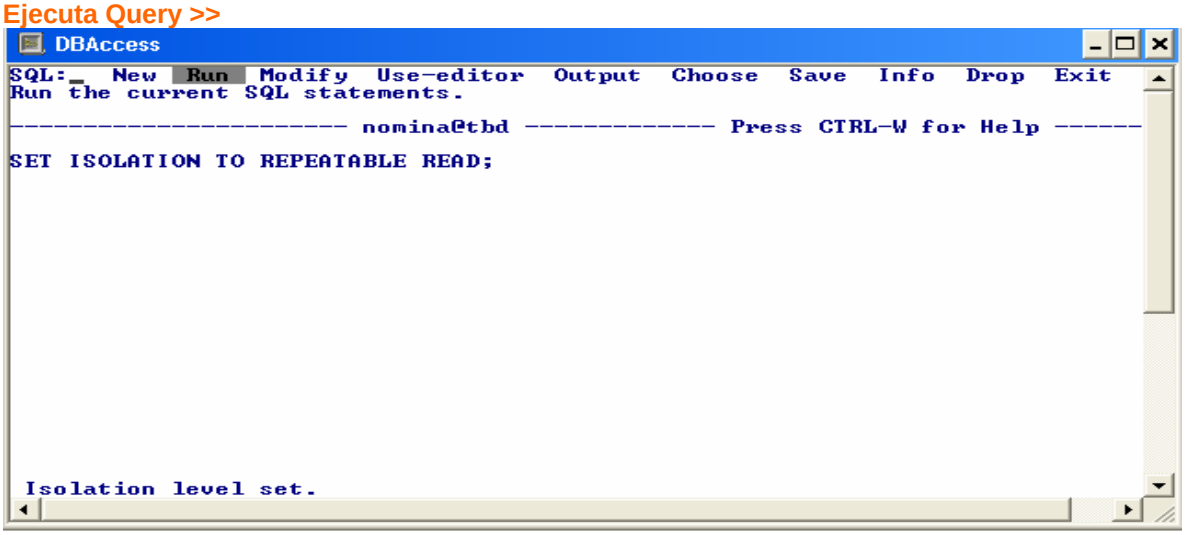
Ejecuta Query >>



Es importante recordar que el nivel de aislamiento se aplica dentro de una sesión y durará hasta que sea cambiado el tipo o la sesión sea terminada. El nivel de aislamiento se define y

aplica sobre todos los queries de SELECT o lectura que se ejecuten después sobre la base de datos.
Ejercicio 4. Activar cada tipo de nivel de aislamiento. Repeatable Read.

Ejecuta Query >>



Es importante recordar que el nivel de aislamiento se aplica dentro de una sesión y durará hasta que sea cambiado el tipo o la sesión sea terminada. El nivel de aislamiento se define y aplica sobre todos los queries de SELECT o lectura que se ejecuten después sobre la base de datos.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de Transacciones, niveles de aislamiento y grados de bloqueo (locks), para las diferentes consultas a realizar. Puede trabajar en más de una sesión para comprobar los resultados.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Defina qué es y para qué sirve un nivel de aislamiento?.
2. ¿Este nivel de aislamiento para qué tipo de queries trabaja?.
3. Genera un query SELECT para comprobar cada nivel de aislamiento, cada nivel de asilamiento. Genera pantallas y queries.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 18
Transacciones usando COMMIT y ROLLBACK

Objetivos

- Implementar transacciones utilizando estatutos de Commit y Rollback
- Apoyar la implementación con estatutos SQL
- Comprobar como trabajan las bases bajo Commit y al deshacer las transacciones
- Entender el funcionamiento general de una transacción desde que inicia hasta que termina o bien se deshace.

Introducción

Una **transacción** es un conjunto de acciones que son ejecutadas en grupo, Si una de esas acciones no puede ser completada, todas las otras deben ser deshechas. Una transacción siempre tiene un inicio y un fin. El servidor de base de datos garantiza que las operaciones hechas dentro de los límites de la transacción sean completadas totalmente y perfectamente cerradas y escritas en disco, o si esto no es posible, la base de datos recuperará el estado que tenía antes de que la transacción iniciara.

Por ejemplo si algún cliente de un banco va a cobrar un cheque, para que la transacción esté completa, deben ejecutarse dos pasos:

- De la cuenta del dueño del cheque deberá descontar la cantidad del cheque
- De la cuenta de quien cobra el cheque debe sumarse la cantidad del cheque

Si alguna de estas acciones no se completa, la transacción no está completa y debe deshacerse. La transacción está completa si y sólo si, ambas acciones sean ejecutadas en su totalidad. **Para abrir y cerrar una transacción se utilizan los estatutos BEGIN WORK y COMMIT WORK, y para deshacer una acción se emplea el ROLLBACK WORK.**

Es importante que la base de datos sea creada para manejar Transacciones, es decir, que sea del tipo Unbuffered o Buffered Log. También pudo crearse como: CREATE DATABASE nomina WITH LOG. Si no es creada de esta manera, las transacciones no funcionarán.

SINTAXIS

BEGIN WORK
COMMIT WORK
ROLLBACK WORK
CREATE DATABASE database-name WITH LOG

Tema y subtema relacionado a cubrir

Tema: **Control de Transacciones**
Subtema: **Instrucciones COMMIT y ROLLBACK**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.

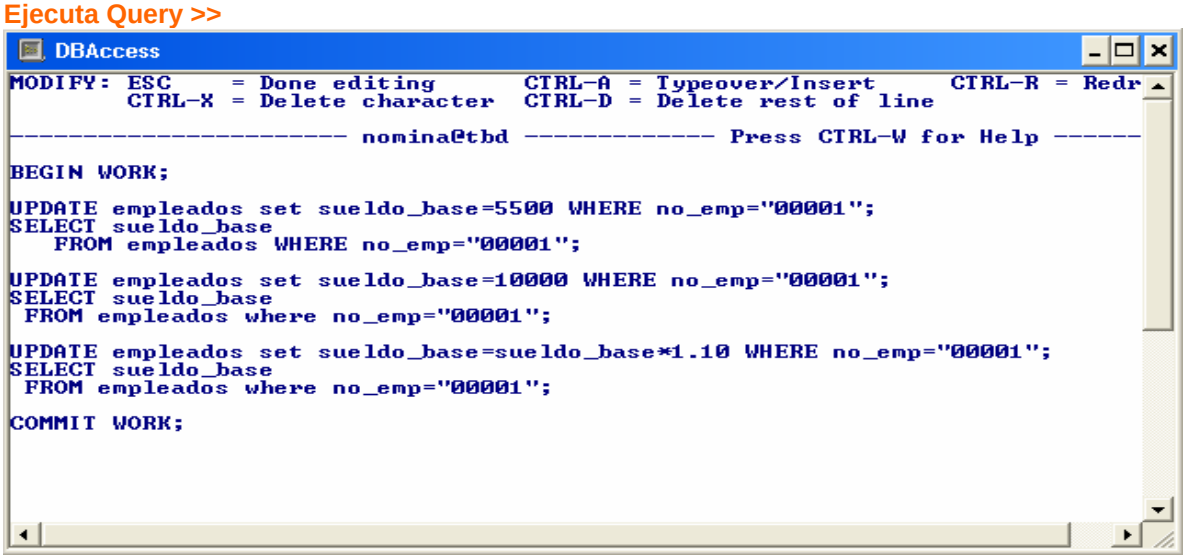
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

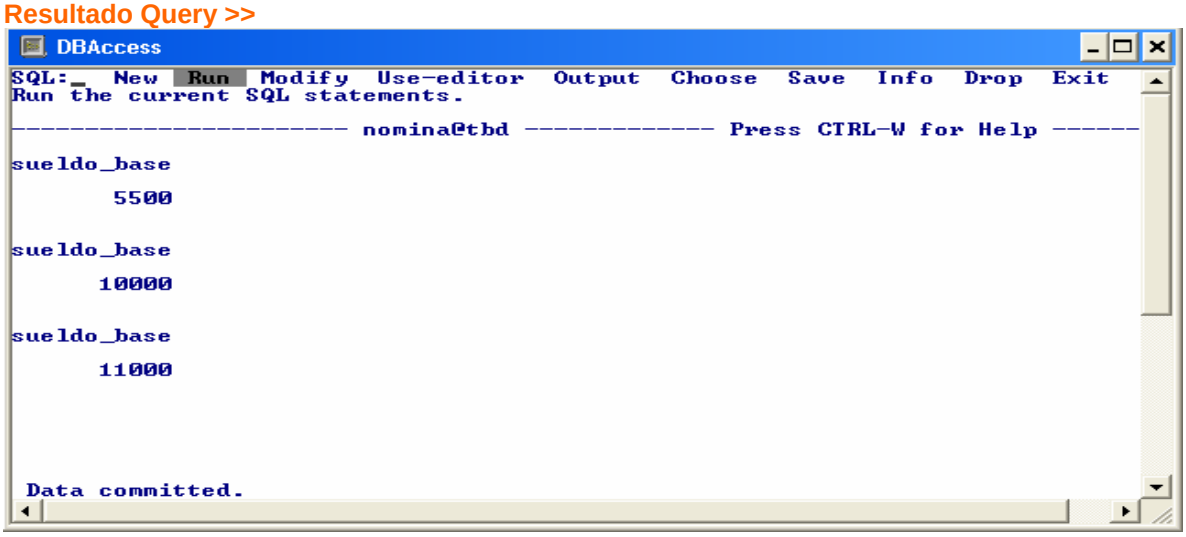
EL estudiante realizará y comprobará cada ejercicio de la práctica, el resultado de cada uno de ellos y la información solicitada.

Ejercicio 1. Transacción completa utilizando BEGIN WORK y COMMIT WORK. En este ejercicio se realizan tres actualizaciones de sueldo de 5500, 10000 y 11000 para el número de empleado "00001".

Ejecuta Query >>



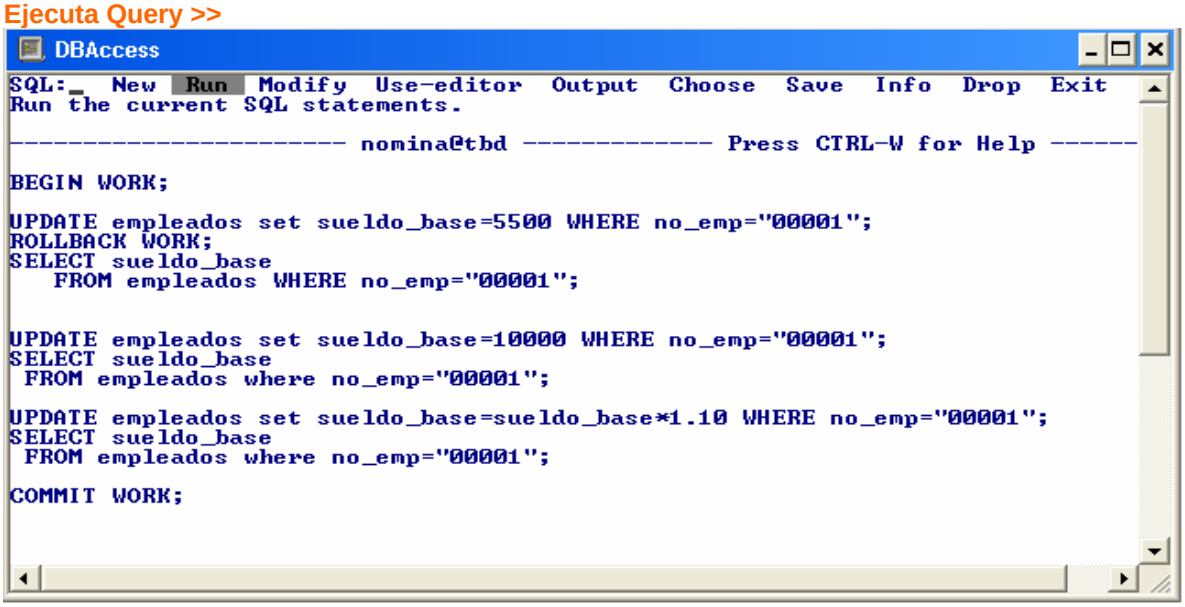
Resultado Query >>



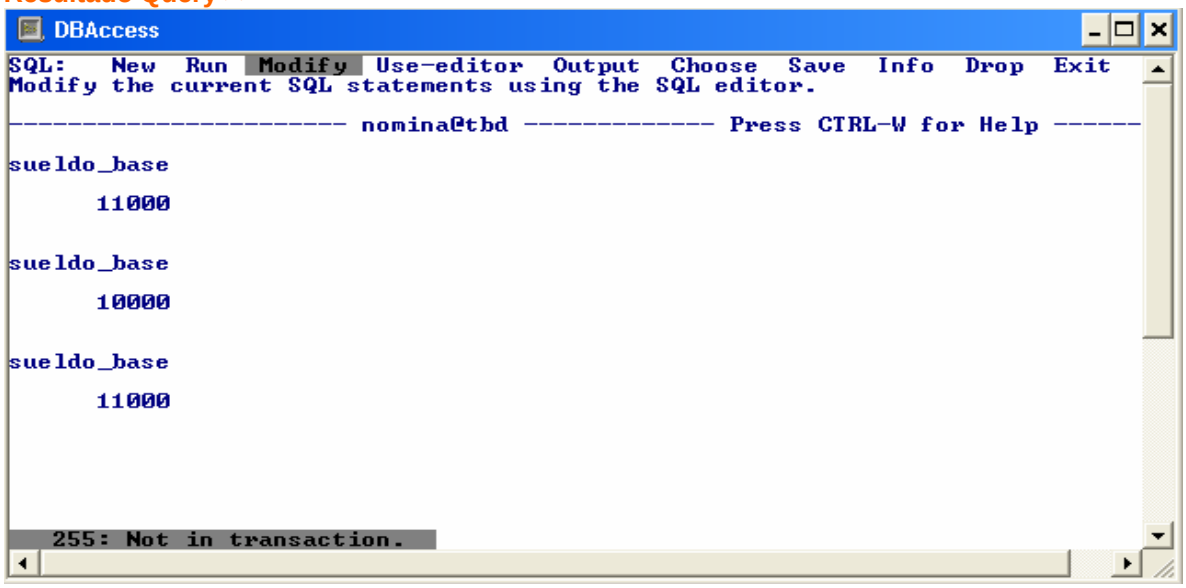
Se observa que en la ejecución de una transacción completa sin problema en alguno de los queries inmersos, el resultado y las actualizaciones se ejecutan correctamente, cerrándose la transacción. Recuerda que la transacción esta completa si todas las acciones se ejecutan bien y completas, de esta manera se mantiene la consistencia de los datos.

Ejercicio 2. Transacción incompleta utilizando BEGIN WORK, COMMIT WORK, ROLLBACK. En este ejercicio se realizan tres actualizaciones de sueldo de 5500, 10000 y 11000 para el número de empleado "00001". Sin embargo ahora se simula con un ROLLBACK que el primer UPDATE no se completó.

Ejecuta Query >>



Resultado Query >>

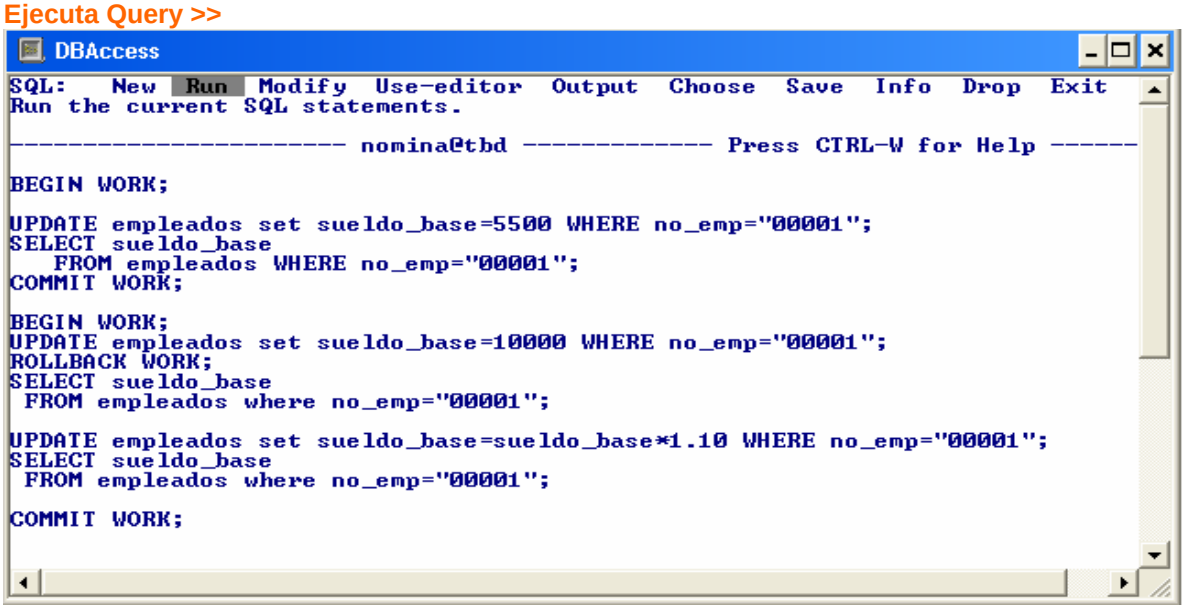


Se observa en la ejecución que al no completarse la actualización de sueldo de 5500, entonces el primer sueldo es diferente al resultado original. El error que marca se debe a que cualquier acción no completada dentro de una transacción hace que se cierre la

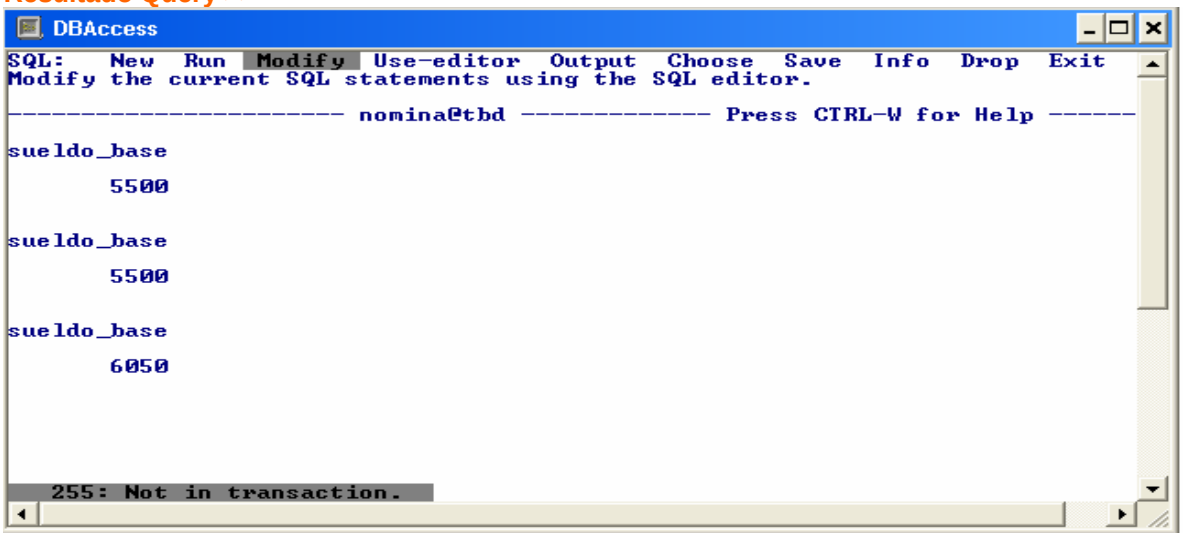
transacción y cuando se encuentra un COMMIT lo que indica es que ya el proceso no está en transacción.

Ejercicio 3. Transacción incompleta utilizando BEGIN WORK, COMMIT WORK, ROLLBACK. En este ejercicio se realizan tres actualizaciones de sueldo de 5500, 10000 y 11000 para el número de empleado "00001". Sin embargo ahora se simula con un ROLLBACK que el segundo UPDATE no se completó.

Ejecuta Query >>



Resultado Query >>



Se observa en el resultado, valores tres valores diferentes al resultado original del ejercicio 1. En esta parte al no completar la segunda actualización de sueldo a 10000, entonces aplica el incremento sobre 5500.

Los Ejercicios 2 y 3 son una muestra clara del cuidado que debe tenerse al crear y ejecutar transacciones. Se debe asegurar la ejecución completa de cada acción, pues en cada punto puede ser acezada por otros usuarios, obteniendo valores diferentes como se ha demostrado.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de Transacciones, niveles de aislamiento y grados de bloqueo (locks), para las diferentes consultas a realizar. Puede trabajar en más de una sesión para comprobar los resultados.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Para el ejercicio 1. ¿En qué momento se va registrando cada actualización a disco?.
2. Para el ejercicio 2. ¿En qué momento se va registrando cada actualización a disco?.
3. Para el ejercicio 3. ¿En qué momento se va registrando cada actualización a disco?.
4. Para el ejercicio 3. ¿Por qué se agrega un COMMIT y BEGIN después del primer UPDATE?.
5. Para las siguientes acciones dentro de la transacción realiza los queries correspondientes, como en el ejercicio 1.

   INICIO

   Sumar todas las percepciones de la tabla MOVIMIENTOS para el empleado 00001 de la quincena 1

   Sumar todas las deducciones de la tabla MOVIMIENTOS para el empleado 00001 de la quincena 1

   Generar el Ingreso Neto

   Insertar un registro nuevo en la tabla de NOMINA con todos los valores anteriormente generados

   FIN

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 19
Vistas y su creación

Objetivos

- Entender la importancia de la creación de vistas
- Crear vistas a partir de las tablas de base de datos

Introducción

Las Vistas son construidas sobre las bases de datos y complementar lo siguiente:

- Proveer a los diferentes usuarios diferentes vistas de la base de datos. Una vista sencilla puede incluir datos de diferentes tablas.
- Limitar el acceso de datos confidenciales a los distintos usuarios según su jerarquía.
- Permitir a los usuarios insertar, borrar y actualizar datos a través de vistas organizadas, respetando integridad referencial y consistencia, siempre y cuando se den estos permisos con GRANT/REVOKE.
- Se pueden crear las vistas con SELECT y el resultado de éste formará la vista.
- No se pueden usar ALTER TABLE, CREATE INDEX sobre vistas.
- La vista puede ser considerada una tabla adicional, aunque los datos en realidad se encuentran en sus tablas originales. Pero cualquier actualización a través de vistas se realiza en las tablas directamente.
- El acceso a las vistas puede hacerse con SELECT como si fuera una tabla extra.

SINTAXIS

CREATE VIEW view-name [(column-list)]
AS SELECT-statement

DROP VIEW view-name

Tema y subtema relacionado a cubrir

Tema: **Vistas**
Subtema: **Definición y objetivo de las vistas**

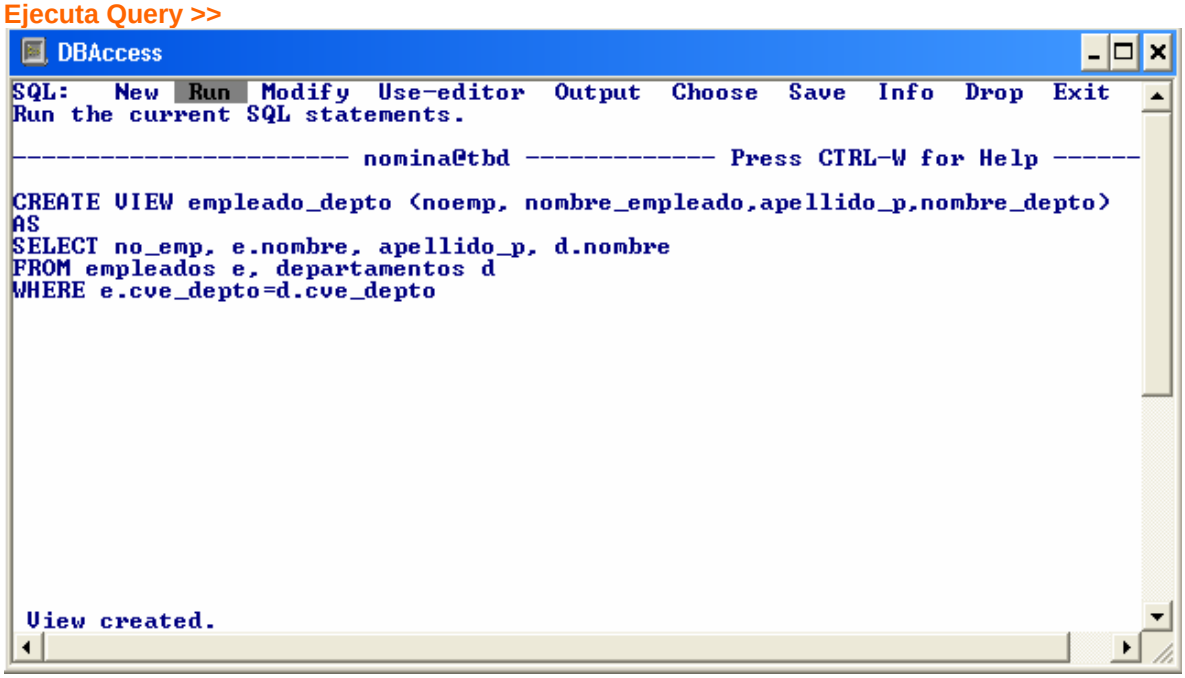
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

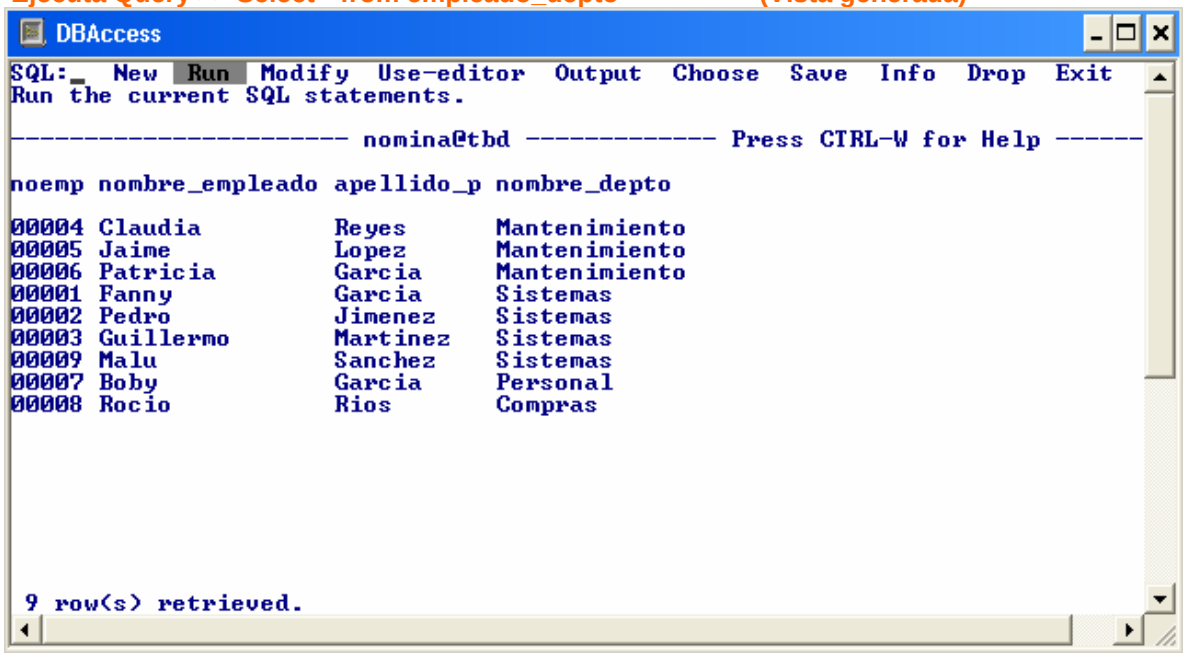
Ejercicio 1. Creación de Vistas. Crear una vista que relacione las tablas de EMPLEADOS Y DEPARTAMENTOS, pero que sólo muestre no_emp, nombre empleado, apellido paterno, nombre departamento.

Ejecuta Query >>

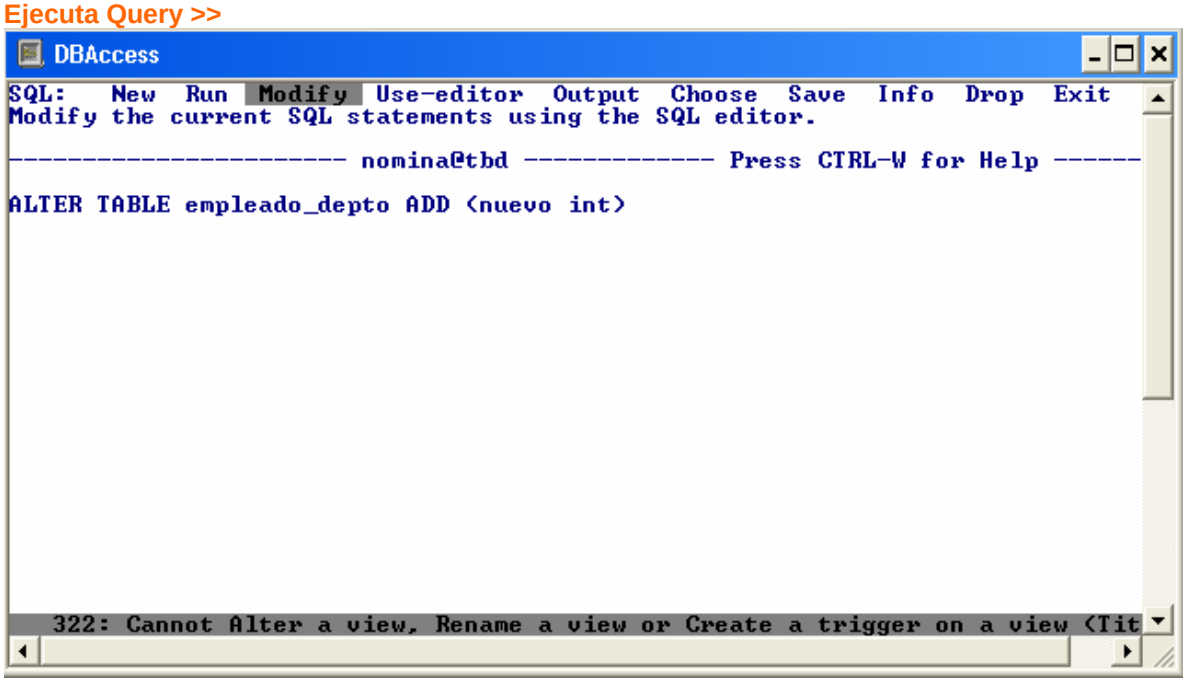


Se recomienda nombrar a las vistas de acuerdo a las tablas relacionadas y/o función que lleve a cabo. Para ver el resultado de la vista generada

Ejecuta Query >> Select * from empleado_depto (Vista generada)

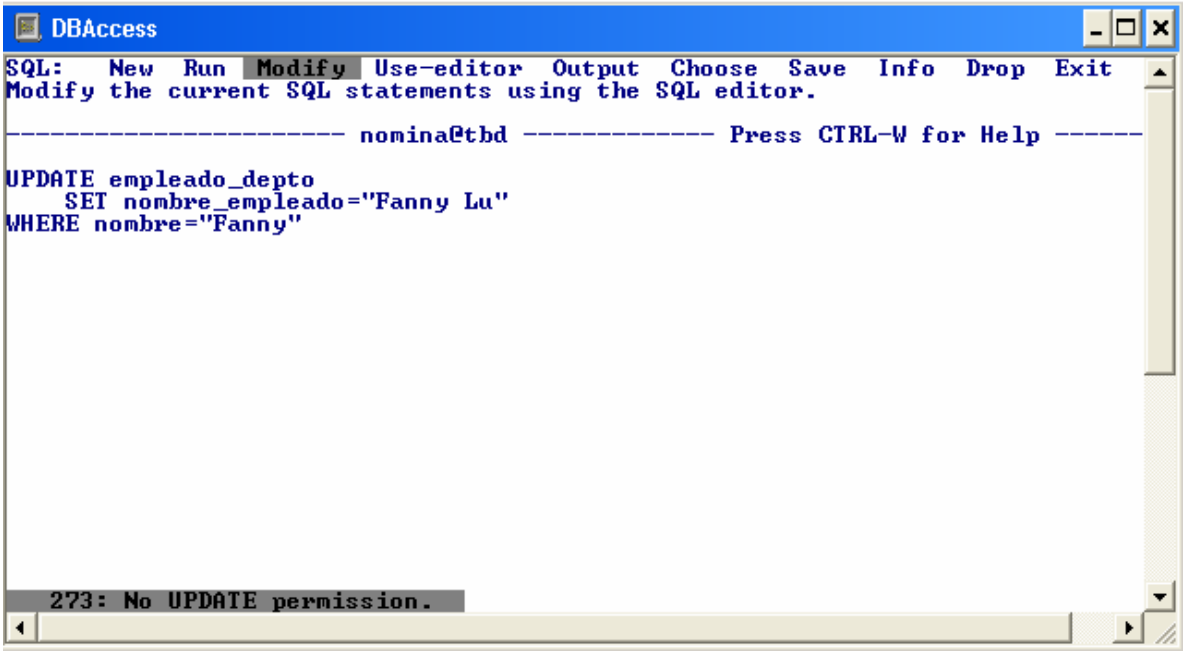


Ejercicio 2. Alterar la estructura de un Vista. Agregar a la vista el campo nuevo tipo entero.
Ejecuta Query >>



Las vistas son manipuladas como tablas, pero no pueden ser alteradas ni agregar índices.

Ejercicio 3. Actualizar datos de un Vista.
Ejecuta Query >>



Es importante recordar que estas funciones son permitidas mientras se den los permisos requeridos con GRANT/REVOKE.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Qué es una Vista y cual es su función?.
2. ¿Cuál es la diferencia entre tabla y vista?.
3. ¿Qué acciones se pueden hacer sobre una vista y cuales no y porqué?.
4. ¿Cómo puedo accesar una vista?.
5. Obtener el esquema de base de datos para ver las definiciones realizadas hasta este momento.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 20
Funciones avanzadas con Vistas

Objetivos

- Crear vistas a partir de las tablas de base de datos
- Administración de vistas

Introducción

Las Vistas son construidas sobre las bases de datos y complementar lo siguiente:

- Proveer a los diferentes usuarios diferentes vistas de la base de datos. Una vista sencilla puede incluir datos de diferentes tablas.
- Limitar el acceso de datos confidenciales a los distintos usuarios según su jerarquía.
- Permitir a los usuarios insertar, borrar y actualizar datos a través de vistas organizadas, respetando integridad referencial y consistencia, siempre y cuando se den estos permisos con GRANT/REVOKE.
- Se pueden crear las vistas con SELECT y el resultado de éste formará la vista.
- No se pueden usar ALTER TABLE, CREATE INDEX sobre vistas.
- Las vista puede ser considerada una tabla adicional, aunque los datos en realidad se encuentran en sus tablas originales. Pero cualquier actualización a través de vistas se realiza en las tablas directamente.
- El acceso a las vistas puede hacerse con SELECT como si fuera una tabla extra.

SINTAXIS

CREATE VIEW view-name [(column-list)]
AS SELECT-statement

DROP VIEW view-name

(ver detalles en prácticas 21-22)

GRANT tab-privilege ON table-name TO {PUBLIC | user-list}
[WITH GRANT OPTION] [AS grantor]

GRANT db-privilege TO {PUBLIC | user-list} [WITH GRANT OPTION] [AS grantor]

REVOKE {tab-privilege ON table-name | db-privilege} FROM {PUBLIC | user-list}

Tema y subtema relacionado a cubrir

Tema: **Vistas**
Subtema: **Instrucciones para la administración de Vistas**

Material y equipo necesario

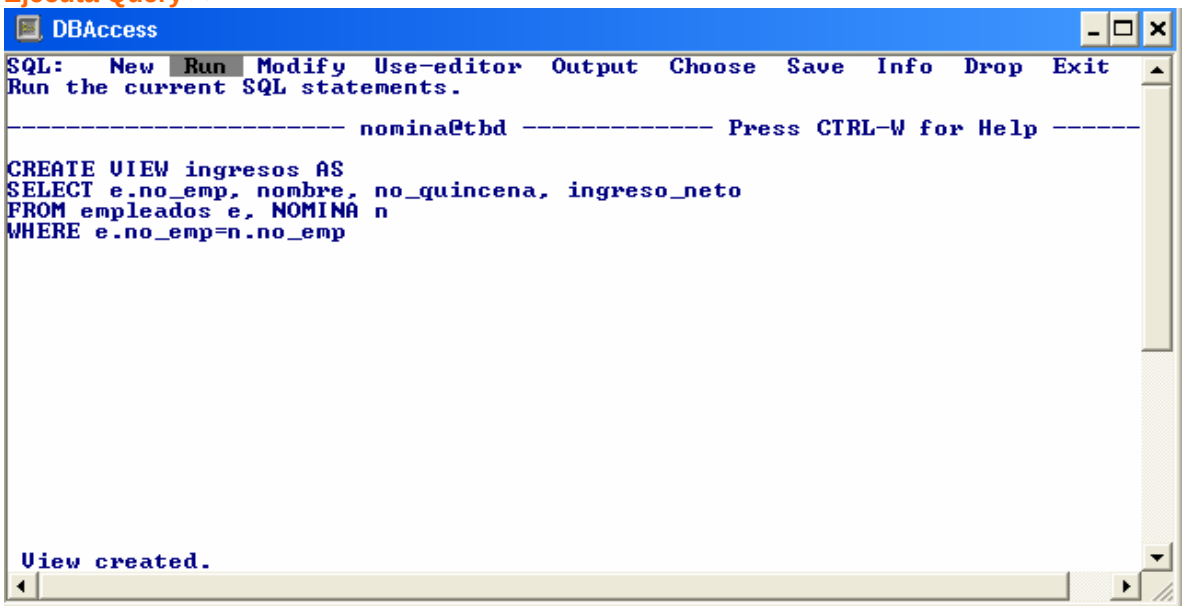
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.

2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

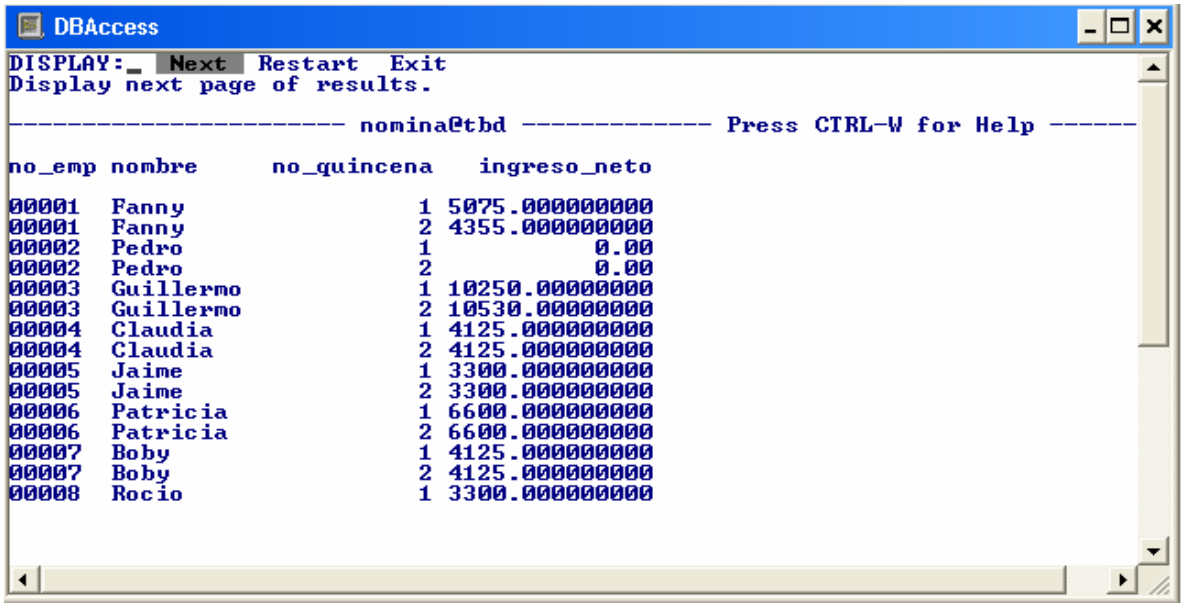
Metodología

Ejercicio 1. Crear una vista que tenga la siguiente información. No_emp, nombre, no_quincena, ingreso neto.

Ejecuta Query >>



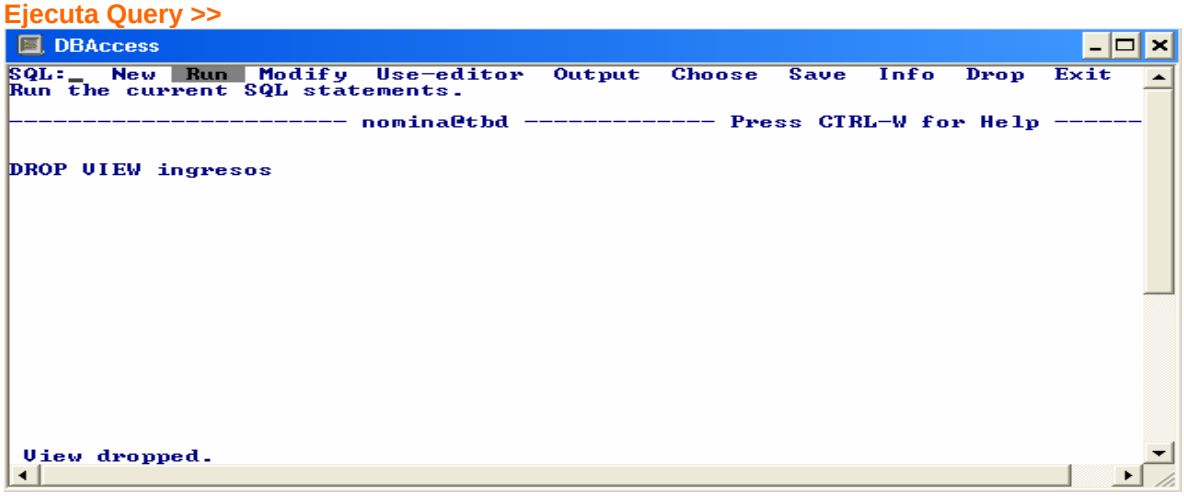
Ejecuta Query >> Para ver la vista (SELECT * from ingresos)



Sólo se muestran aquellos datos que son importantes para el usuario que requiere dichas información.

Ejercicio 2. Eliminar vista ingresos.

Ejecuta Query >>



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Generar una vista movimientos_concepto que incluya: no_emp, no_quincena, cantidad, cve_concepto, tipo_concepto, para el empleado 00001.
2. Dar los permisos necesario con GRANT para poder insertar un nuevo movimientos para el empleado 00001.
3. Insertar el registro. Verificar sus existencia en la tabla de MOVIMIENTOS.
4. Borrar el registro asignado permisos para poder hacerlo.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 21
Seguridad en Base de Datos I

Objetivos

- Entender la importancia de la seguridad y confidencialidad de datos
- Entender y consultar de una base los esquemas de autorización

Introducción

El uso del estatuto GRANT/REVOKE, sirve para asignar o negar respectivamente privilegios de acceso a las tablas o vistas de la Base de Datos. GRANT extiende privilegios a usuarios, no los elimina. REVOKE elimina privilegios a usuarios, no los extiende. Estos estatutos bien utilizados, proporcionan un alto nivel de Seguridad a las Bases de Datos, por lo que se recomienda utilizarlo siempre.

Los privilegios sobre Base de Datos son:

***CONNECT*:** Permite la conexión a la base de Datos.
RESOURCE: Permite agregar recursos a la Base, cómo crear tablas, índices, etc.
DBA: Tiene todos los niveles de privilegios sobre Bases y Tablas. Además de Eliminación de Bases y Tablas.

Los privilegios sobre Tablas son:
INSERT. Permite el la inserción de registros.
DELETE. Permite la eliminación de registros.
UPDATE. Permite actualizar registros.
SELECT. Permite el acceso a los datos.
INDEX. Permite la creación de índices sobre la tabla.
ALTER. Permite realizar cualquier tipo de modificación de estructura.

Como propietario de la tabla o DBA, son asignados estos seis privilegios. La propiedad de la tabla no puede ser transferida a otro usuario. PUBLIC se refiere al resto de los usuarios que no son ni el propietario ni el DBA. Recordar que el usuario **informix** funge como DBA en todas las Bases de Datos.

SINTAXIS

GRANT tab-privilege ON table-name TO {PUBLIC | user-list}
[WITH GRANT OPTION] [AS grantor]

GRANT db-privilege TO {PUBLIC | user-list} [WITH GRANT OPTION] [AS grantor]

REVOKE {tab-privilege ON table-name | db-privilege} FROM {PUBLIC | user-list}

DATABASE PRIVILEGES	TABLE PRIVILEGES
CONNECT	ALTER DELETE
RESOURCE	INDEX INSERT
DBA	EXECUTE
	SELECT[(cols)]
	UPDATE [(cols)]
	REFERENCES[(cols)]
	ALL [PRIVILEGES]

Tema y subtema relacionado a cubrir

Tema: **Seguridad**
Subtema: **Esquemas de autorización**

Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

Ejercicio 1. Observar para la Base de Datos NOMINA y la tabla DEPARTAMENTOS con que usuario fue creado y cuales son los permisos que actualmente tiene sobre la tabla.

- **Se genera el esquema de la base de datos, desde la línea de comandos.**
  - **$ Dbschema –d nomina –ss > archivo_salida**
  - **Editar archivo_salida**
  - **Resolver con esta información Ejercicio 1. Observar y anotar todo lo referente a permisos para la Base y la tabla especificada.**

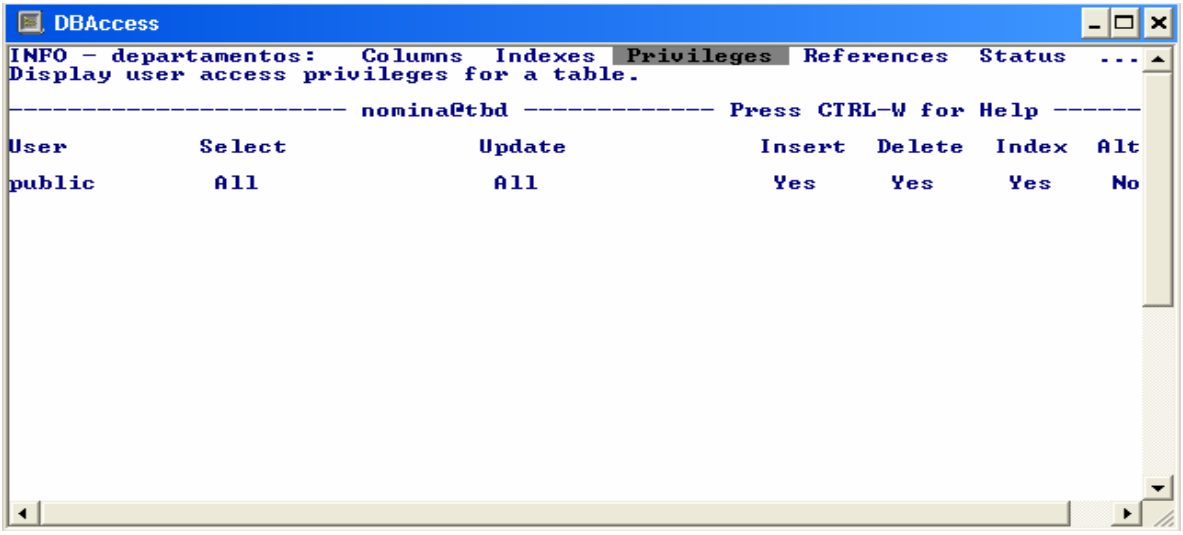
```
DBSCHEMA Schema Utility       INFORMIX-SQL Version 10.00.TC3
Copyright IBM Corporation 1996, 2004 All rights reserved
grant dba to "Tita";

{ TABLE "Tita".departamentos row size = 24 number of columns = 2 index size = 8 }
create table "Tita".departamentos
  (
    cve_depto char(3) not null ,
    nombre varchar(20),

    check (cve_depto MATCHES '[D][0-9][0-9]' ),
    primary key (cve_depto)
  )  extent size 16 next size 16 lock mode page;
revoke all on "Tita".departamentos from "public";

grant select on "Tita".departamentos to "public" as "Tita";
grant update on "Tita".departamentos to "public" as "Tita";
grant insert on "Tita".departamentos to "public" as "Tita";
grant delete on "Tita".departamentos to "public" as "Tita";
grant index on "Tita".departamentos to "public" as "Tita";
```

Se observa lo siguiente:

- **Usuario propietario Tita, teniendo además privilegios de DBA.**
- **Para la tabla DEPARTAMENTOS , su propietario es el usuario Tita, quien es el que crea la tabla.**
- **Con el REVOKE se eliminan todos los privilegios sobre la DEPARTAMENTOS al usuario PUBLIC. Esto aumenta la seguridad de la información.**
- **Finalmente el usuario Tita, da los privilegios de tabla INSERT; DELETE; UPDATE; SELECT; INDEX al usuario PUBLIC.**

Ejercicio 2. Consultar a través de dbaccess para la Base de Datos NOMINA y la tabla DEPARTAMENTOS con que usuario fue creado y cuales son los permisos que actualmente tiene sobre la tabla y compararlos con los del Ejercicio 1.

Elegir la opción del Menú-> Table->Info->Privileges



Se observan los mismos privilegios para el usuario PUBLIC. Siempre el usuario quien crea es DBA.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Para cada tabla de la Base de datos, ver los privilegios de tabla que tiene asignados. Realizarlo por dbschema y dbaccess. Mostrar resultados de cada uno.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 22
Seguridad en Bases de Datos II

Objetivos

- Entender la importancia de la seguridad y confidencialidad de datos
- Implementar seguridad en tablas de base de datos utilizando los estatutos de Grant/Revoke

Introducción

El uso del estatuto GRANT/REVOKE, sirve para asignar o negar respectivamente privilegios de acceso a las tablas o vistas de la Base de Datos. GRANT extiende privilegios a usuarios, no los elimina. REVOKE elimina privilegios a usuarios, no los extiende. Estos estatutos bien utilizados, proporcionan un alto nivel de Seguridad a las Bases de Datos, por lo que se recomienda utilizarlo siempre.

Los privilegios sobre Base de Datos son:

***CONNECT*:** Permite la conexión a la base de Datos.
RESOURCE: Permite agregar recursos a la Base, cómo crear tablas, índices, etc.
DBA: Tiene todos los niveles de privilegios sobre Bases y Tablas. Además de Eliminación de Bases y Tablas.

Los privilegios sobre Tablas son:
INSERT. Permite el la inserción de registros.
DELETE. Permite la eliminación de registros.
UPDATE. Permite actualizar registros.
SELECT. Permite el acceso a los datos.
INDEX. Permite la creación de índices sobre la tabla.
ALTER. Permite realizar cualquier tipo de modificación de estructura.

Como propietario de la tabla o DBA, son asignados estos seis privilegios. La propiedad de la tabla no puede ser transferida a otro usuario.

SINTAXIS

GRANT tab-privilege ON table-name TO {PUBLIC | user-list}
[WITH GRANT OPTION] [AS grantor]

GRANT db-privilege TO {PUBLIC | user-list} [WITH GRANT OPTION] [AS grantor]

REVOKE {tab-privilege ON table-name | db-privilege} FROM {PUBLIC | user-list}

DATABASE PRIVILEGES	TABLE PRIVILEGES
CONNECT	ALTER DELETE
RESOURCE	INDEX INSERT
DBA	EXECUTE
	SELECT[(cols)]
	UPDATE [(cols)]
	REFERENCES[(cols)]
	ALL [PRIVILEGES]

Tema y subtema relacionado a cubrir

Tema: **Seguridad**
Subtema: **Instrucciones GRANT y REVOKE**

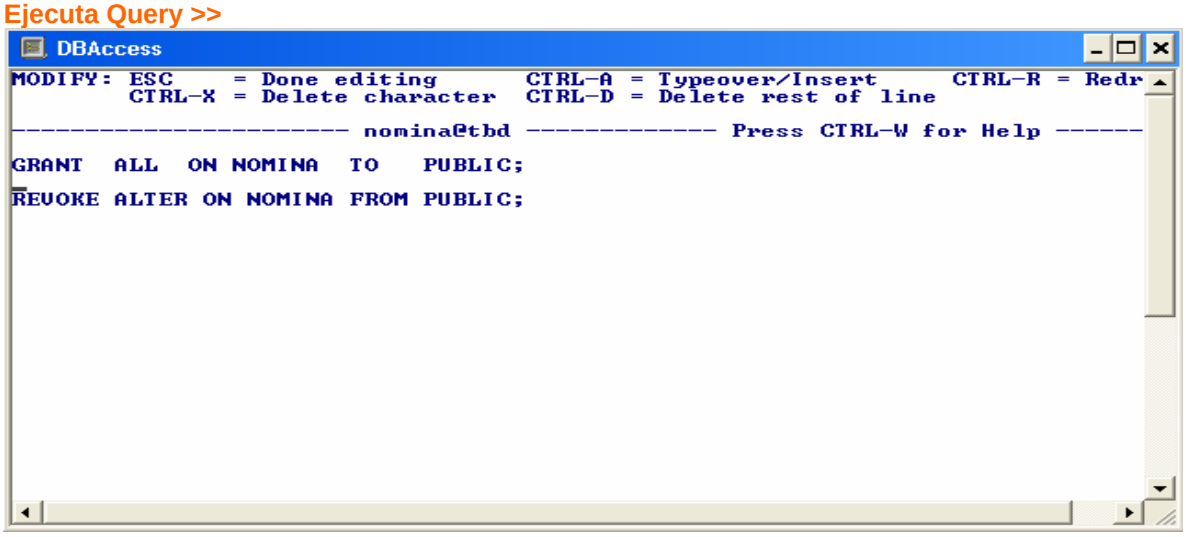
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

Ejercicio 1. PERMISOS GENERALES. Asignar y quitar todos los privilegios de tabla, a todos los usuarios sobre la tabla NOMINA.

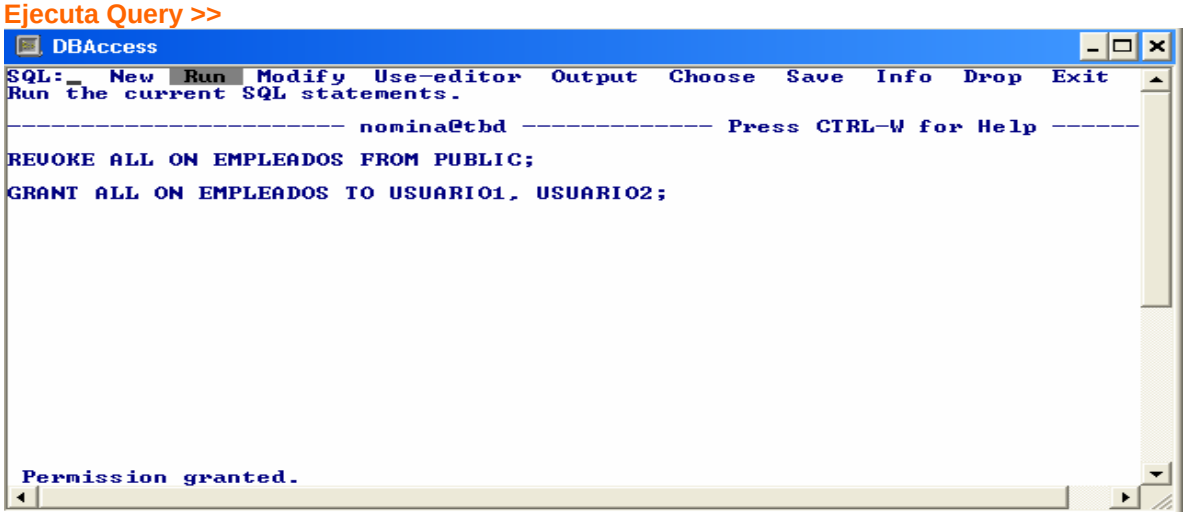
Ejecuta Query >>



En este query se asignan con GRANT ALL , todos los permisos a los usuarios que representan al usuario PUBLIC y después se eliminan. Es importante saber que se permite ver y a qué usuarios. Esto garantiza la seguridad de la Base de Datos.

Ejercicio 2. PERMISOS PARA USUARIOS ESPECÍFICOS. Elimina todos los privilegios sobre la tabla EMPLEADOS para el usuario PUBLIC, permitiendo sólo los permisos a los usuarios: USUARIO1 Y USUARIO2.

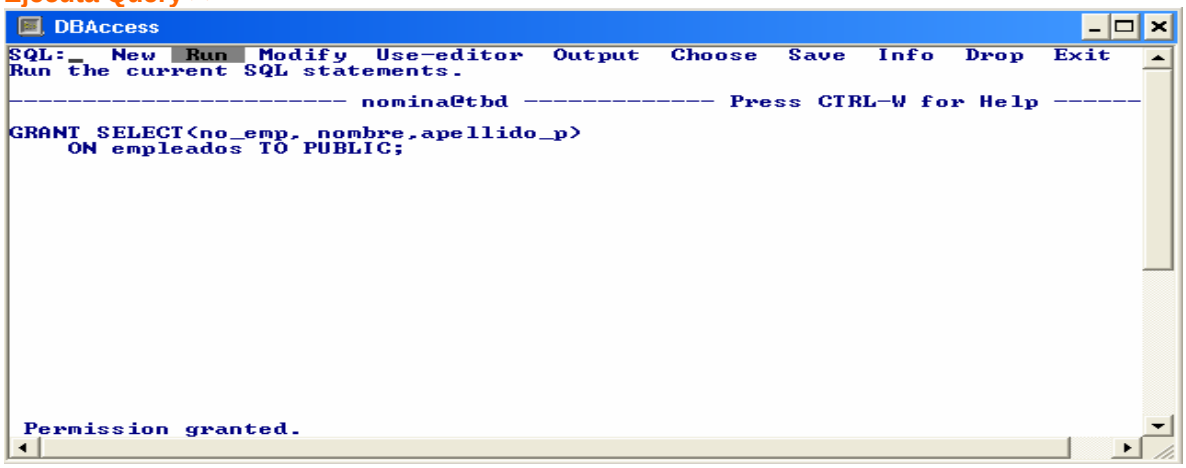
Ejecuta Query >>



Como se puede observar, la asignación así como la eliminación de permisos puede ser específica a ciertos usuarios, incrementado la seguridad.

Ejercicio 3. PERMISOS SOBRE CAMPOS ESPECÍFICOS. Para todos los usuarios (o usuario PUBLIC), dar permisos de SELECT únicamente para los campos no_emp, nombre y apellido paterno. No podrán ver más información.

Ejecuta Query >>



Para incrementar la seguridad de acceso, se observa que pude darse por ciertos campos específicos, de esta manera se protege la información confidencial.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 23
Procedimientos Almacenados o Stored Procedures

Objetivos

- Entender el concepto y ventajas del uso de Procedimientos Almacenados
- Implementar bajo SQL procedimientos almacenados y comprobar su
- funcionamiento y ventajas contra queries normales

Introducción

Los Stored Procedures o Procedimientos Almacenado, son de gran utilidad para optimizar y compartir los recursos del manejador de base de datos. Es decir un mismo procedimientos puede ser compartido por varios usuarios. Estos procedimientos no forman parte del programa del usuario, sino que son programados y almacenados en el servidor de base de datos y son referenciados desde el programa cliente. Ventajas de un procedimiento almacenado:

- Reduce la complejidad de un programa
- Mejora el rendimiento de la aplicación
- Provee un nivel extra de seguridad
- Las reglas y políticas de la empresa pueden hacerse centralizadas y programadas
- Diferentes sistemas con diferentes lenguajes de programación pueden accesar los procedimientos almacenados
- En un ambiente cliente-servidor no se necesita realizar la distribución de código
- No pueden existir dos procedimientos con el mismo nombre

SINTAXIS

```
CREATE PROCEDURE procedure-name ( [expression[,...])
      [define-stmt]
      [exception-declaration]
      [statement-list]
   END PROCEDURE [DOCUMENT string[,...]][WITH LISTING IN string]

DROP PROCEDURE procedure-name

FOREACH
END FOREACH

IF CONDITION
THEN
ELIF
THEN
ELSE
END IF;

WHILE CONDITION
END WHILE;

FOR I = 1 TO 10
….
END FOR;
```

Tema y subtema relacionado a cubrir

Tema: **Introducción al SQL procedural**
Subtema: **Procedimientos Almacenados**

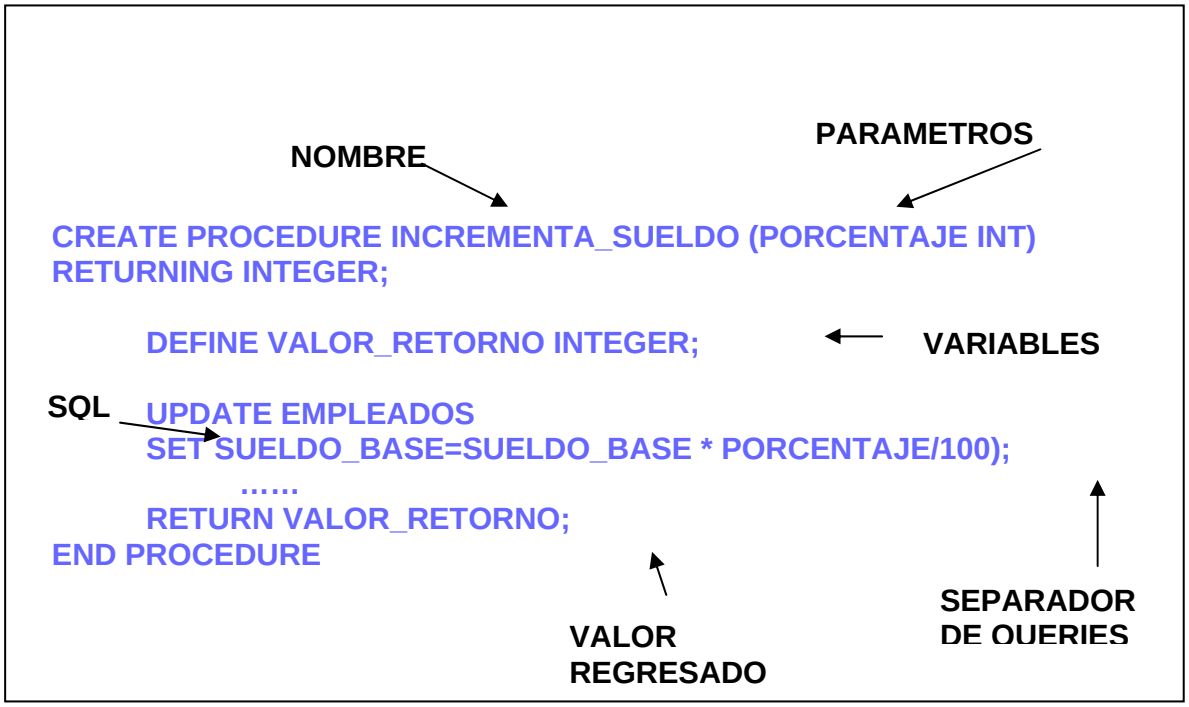
Material y equipo necesario

1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

Se debe realizar un respaldo previo para seguridad de los datos. Puede utilizarse dbexport/dbimport.

Ejercicio 1. Componentes de un Stored Procedure.



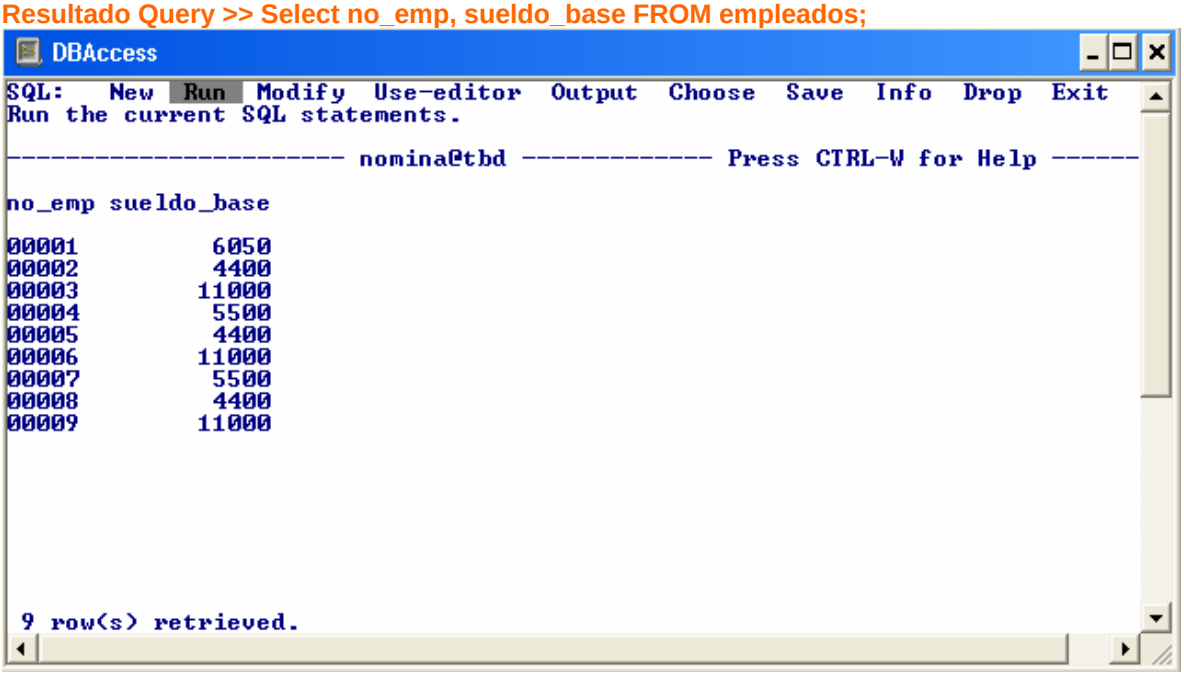
En este ejercicio se muestran los componentes de un procedimiento almacenado. Debe constar de un nombre, parámetros de ejecución (opcional), SQL, variables, valor regresado (opcional), separador de queries. La complejidad de estos procedimientos dependerá de las

necesidades del sistema. Sin embargo un buen diseño de base de datos inicial se reflejará en una buena programación de un procedimiento.

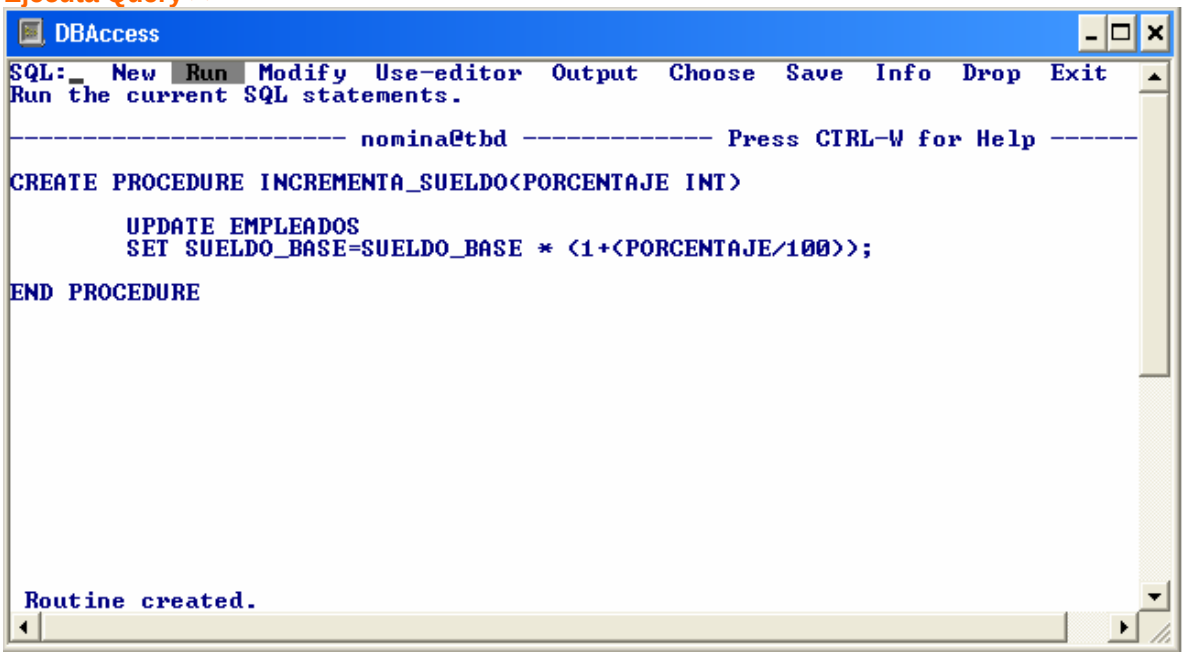
Ejercicio 2. Realizar un procedimiento almacenado que incremente de forma variable el sueldo_base de todos los empleados. (Se hará la prueba con 10% de incremento).

Para entender mejor el funcionamiento de un procedimiento, se muestran todos los pasos. Primero se muestra la información de sueldo antes de ser modificada.

Resultado Query >> Select no_emp, sueldo_base FROM empleados;

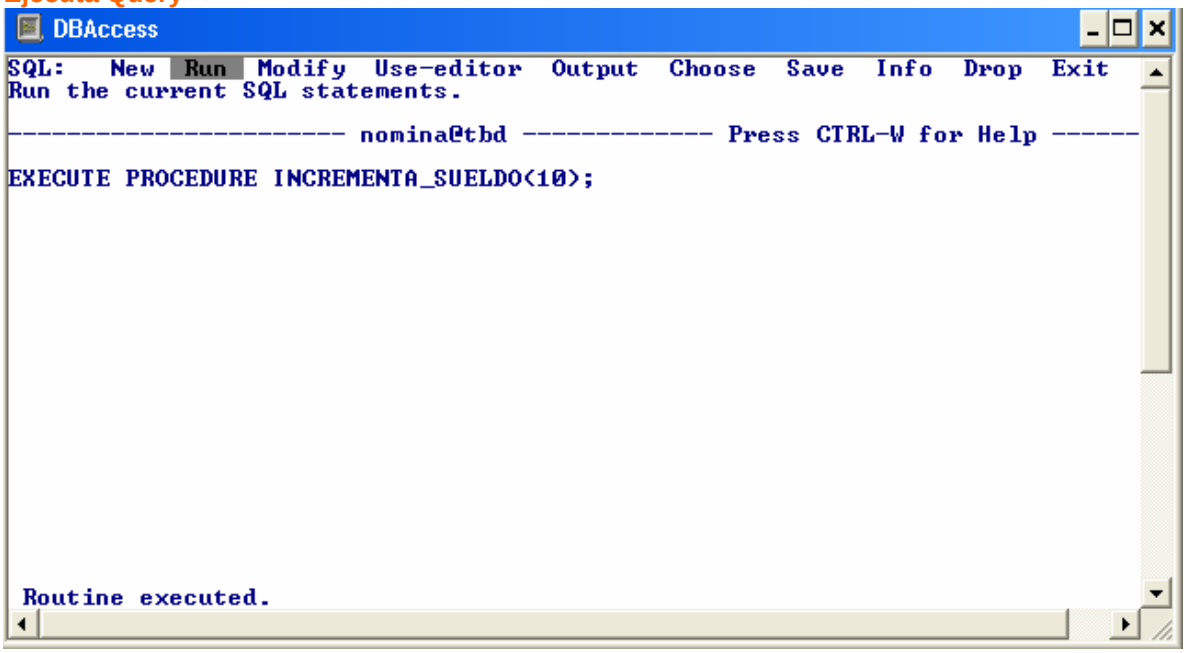


Ejecuta Query >>



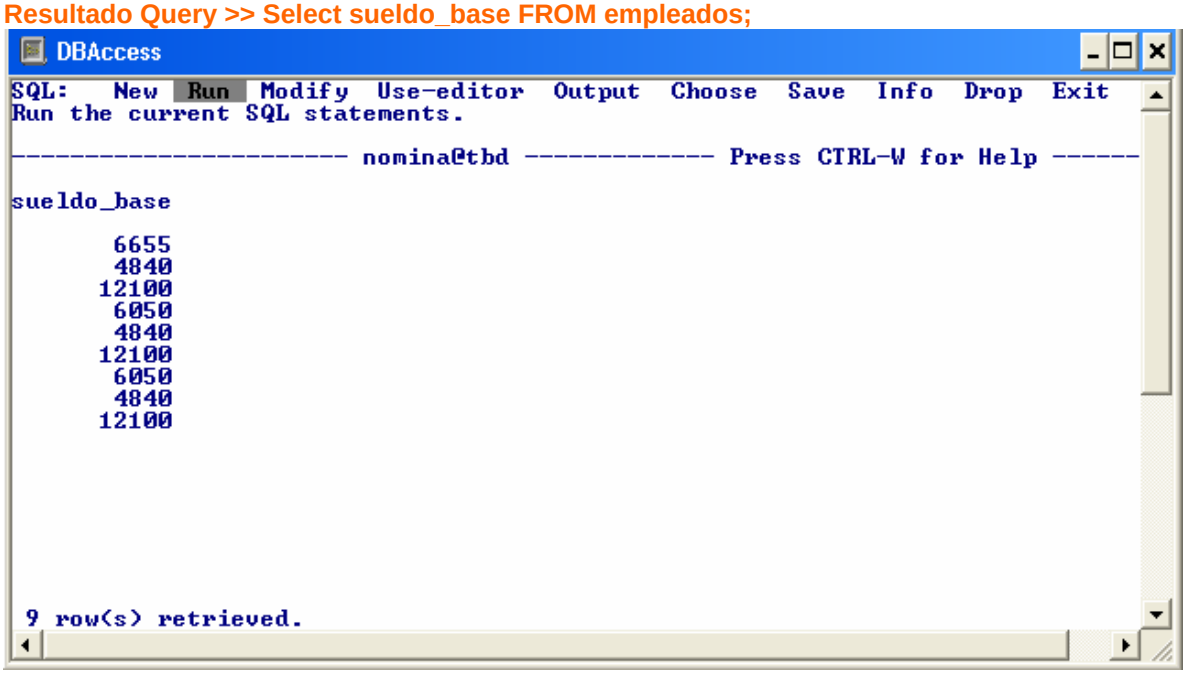
Recuerda que el procedimiento debe ser creado y después ejecutado. Se puede observar que el incremento de sueldo se define como variable y aplica a todos los empleados. Este procedimiento se crea una sola vez y se puede ejecutar tantas veces se necesite por cualquier usuario.

Ejecuta Query >>



Se ejecuta el procedimiento como se muestra, se manejo para el ejercicio un 10% de incremento.

Resultado Query >> Select sueldo_base FROM empleados;



Se observa el incremento que generó el procedimiento.

Ejercicio 3. Ver todos los procedimientos creados para la base de datos NOMINA.

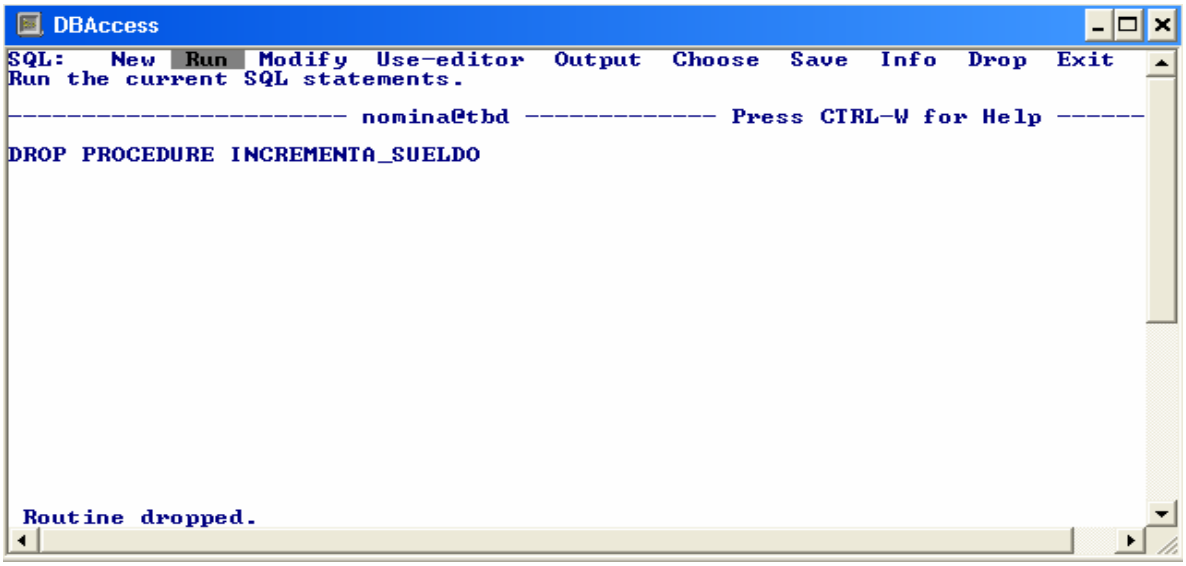
Esto debe hacerse desde la línea de comando con la siguiente instrucción:

Dbschema –d NOMINA –f all

Resultado >>

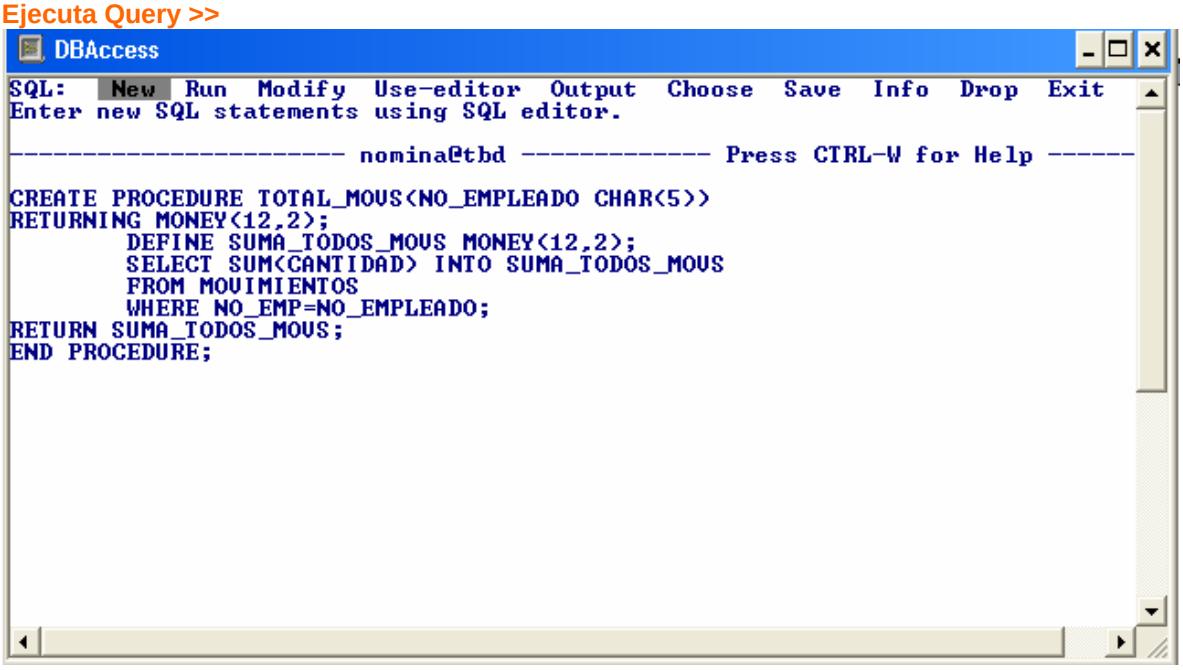
```
DBSCHEMA Schema Utility       INFORMIX-SQL Version 10.00.TC3
Copyright IBM Corporation 1996, 2004 All rights reserved
Software Serial Number AAA#B000000

create procedure "Tita".descuenta2(sueldob int)
returning float;
define porcen float;
  select porcentaje_descto  into porcen from descuentos
  where (lim_superior<=sueldob) and (lim_superior>=sueldob);
return porcen;
end procedure;

CREATE PROCEDURE "Tita".incrementa_sueldo(PORCENTAJE INT)
        UPDATE EMPLEADOS
        SET SUELDO_BASE=SUELDO_BASE * (1+(PORCENTAJE/100));
END PROCEDURE;
```

Ejercicio 4. Eliminar el procedimiento incrementa_sueldo.



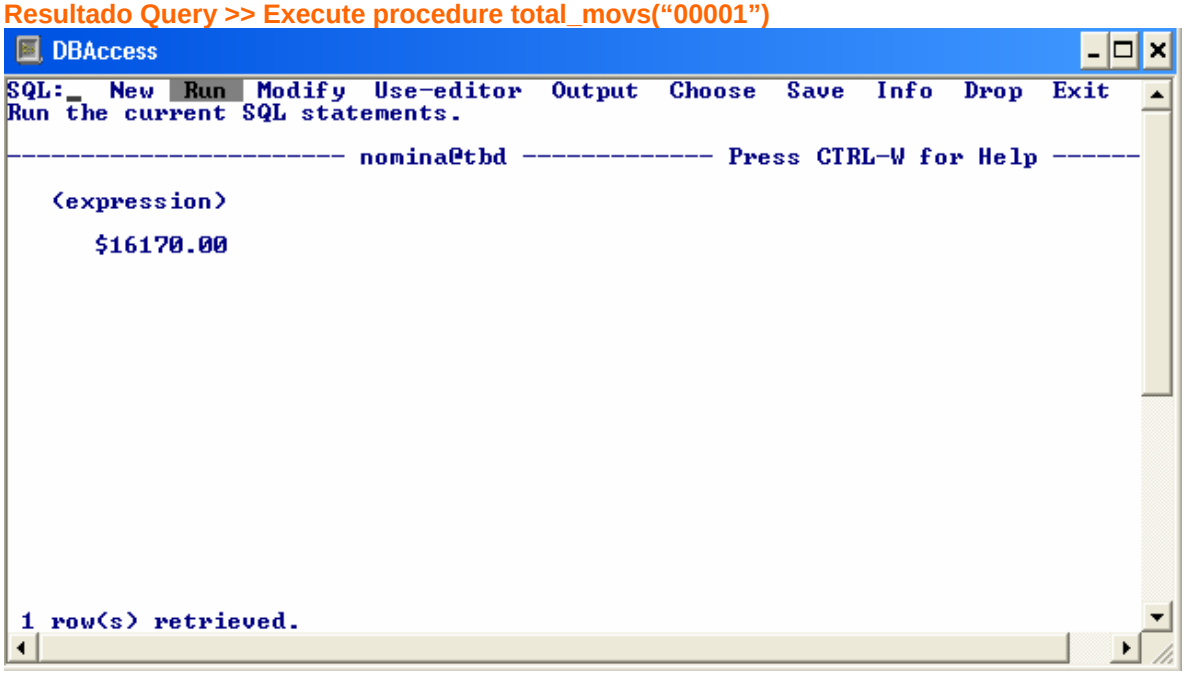
Ejercicio 5. Crear un procedimiento que totalice todos los movimientos de un empleado específico, y regrese el valor del total.

Ejecuta Query >>



Se observa que en este procedimiento si interesa el regreso del valor total, por lo que aplica el Returning para indicar que va a regresar un valor tipo Money; y en el RETURN envía el total con ese tipo de dato.

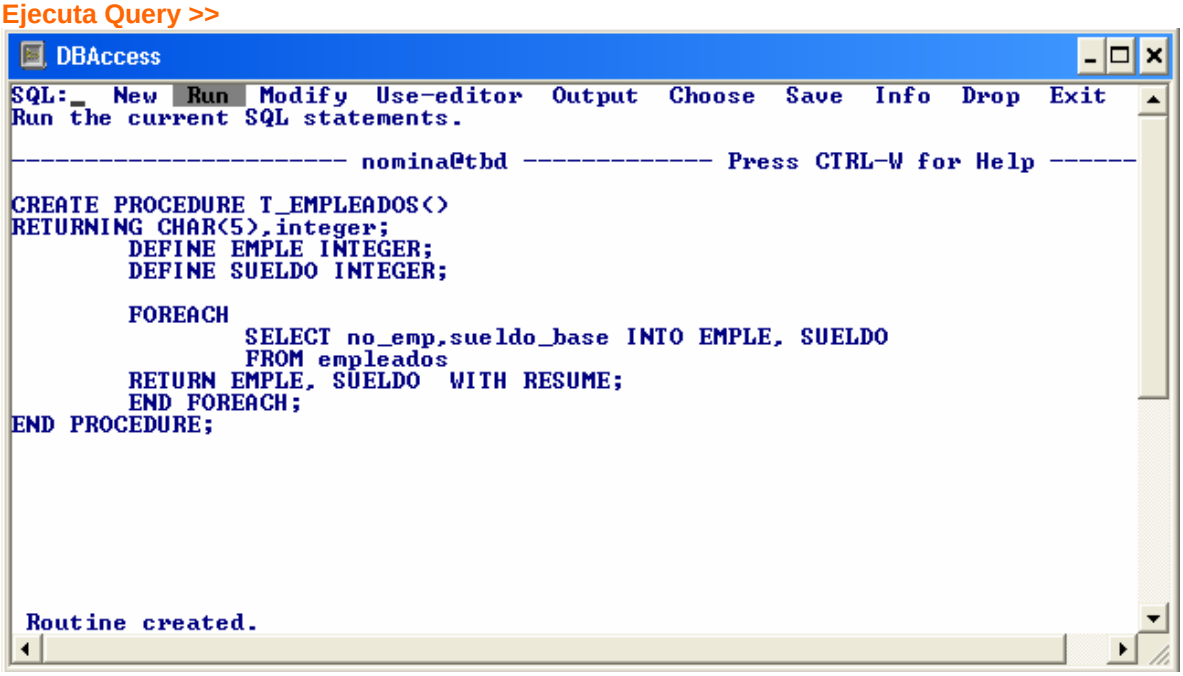
Resultado Query >> Execute procedure total_movs("00001")



Aquí se despliega el resultado del query.

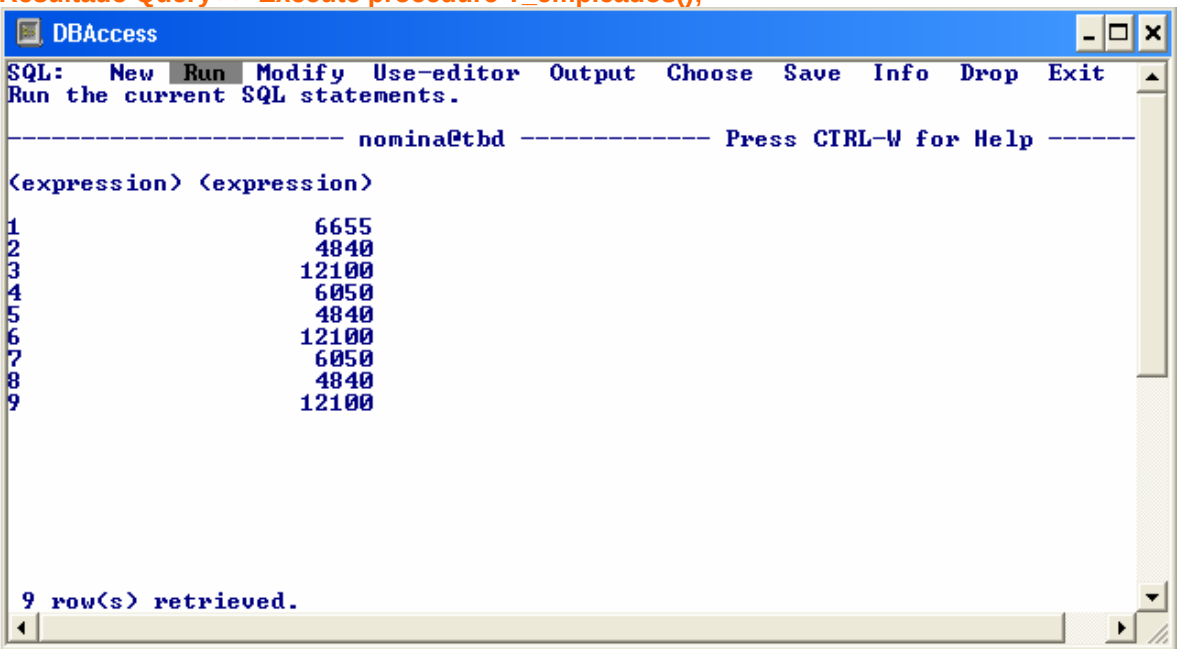
Ejercicio 6. Crear un procedimiento que despliegue los datos generales de cada empleado.

Ejecuta Query >>



Este procedimiento muestra el uso del foreach, útil para accesar todos los renglones de un query, este debe ir acompañado del WITH RESUME. La función INTO genera que los valores del Select pasen a las variables y así poderlas manipular o desplegar.

Resultado Query >> Execute procedure T_empleados();



Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Se debe realizar siempre un respaldo previo a esta práctica, ya que implica la actualización de información.

Se deben conocer los Stored Procedures almacenados para saber las acciones que se pueden realizar de cualquier Evento ejecutado.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. ¿Qué es un Stored Procedure, cual es su uso y cómo es almacenado?.
2. Genera un procedimiento que obtenga para cada empleado y su sueldo_base el porcentaje de descuento que se aplicará para el concepto de ISR(001).
3. Genera un procedimiento que una vez generado el porcentaje de descuento del ejercicio 2. Entonces inserte en la tabla de MOVIMIENTOS un registro con los datos completos para ese empleado en la quincena 3.
4. Genera un procedimiento que genere automáticamente los registros de la quincena 3 para cada empleado. (Procedimiento de NOMINA).

NOTA: Si requieres insertar más movimientos, se puede hacer usando INSERT, sin utilizar procedimientos.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

Práctica 24
Disparadores o Triggers

Objetivos

- Entender el concepto y ventajas del uso de Triggers o Disparadores
- Implementar bajo SQL Disparadores y comprobar su funcionamiento y ventajas contra queries normales
- Implementar Disparadores y Procedimientos almacenados relacionados

Introducción

Un Trigger o Disparador es un mecanismo que automáticamente se activa cuando ocurre un evento específico en una tabla de base de datos. Por ejemplo, cada que se inserta un registro en la tabla de EMPLEADOS le asignará siempre un 10% extra en el sueldo base, y esta acción siempre se ejecutará después de cada inserción.

Básicamente un disparador consiste de un Evento y una Acción a aplicar. Un Evento de un Trigger puede ser un DELETE, INSERT o UPDATE. Una Acción de un Trigger es una conjunto de estatutos de SQL ejecutadas cuando el evento ocurre. Una acción de un Trigger puede ser un DELETE, UPDATE, INSERT y EXECUTE PROCEDURE.

Un Trigger es almacenado como un objeto de la Base de Datos y tiene las mismas ventajas que un Stored Procedure. Para una tabla no pueden existir más de un trigger con el mismo evento. Para un Trigger con INSERT no puede tomar una acción de actualizar o insertar los mismos datos insertados.

SINTAXIS

```
CREATE TRIGGER trigger-name
    [INSTEAD OF]
     {
     INSERT
     |
     DELETE
     |
     UPDATE [ OF (column-list)]
     }
     ON table-name | view-name [REFERENCING
                    {
                    OLD [AS] correlation-name [ NEW [AS] correlation-name]
                    |
                    NEW [AS] correlation-name [ OLD [AS] correlation-name]
                    }
                ]
     {
     BEFORE trigger-action-list [ FOR EACH ROW trigger-action-list ]
          [ AFTER trigger-action-list ]
     |
     FOR EACH ROW trigger-action-list [ AFTER trigger-action-list]
     |
     AFTER trigger-action-list
    }
```

trigger-action-list
trigger-action [, trigger-action ...]

trigger-action
[WHEN search-condition]
{ DELETE statement | UPDATE statement | INSERT statement |
EXECUTE PROCEDURE statement }

Tema y subtema relacionado a cubrir

Tema: **Introducción al SQL procedural**
Subtema: **Disparadores (Triggers)**

Material y equipo necesario

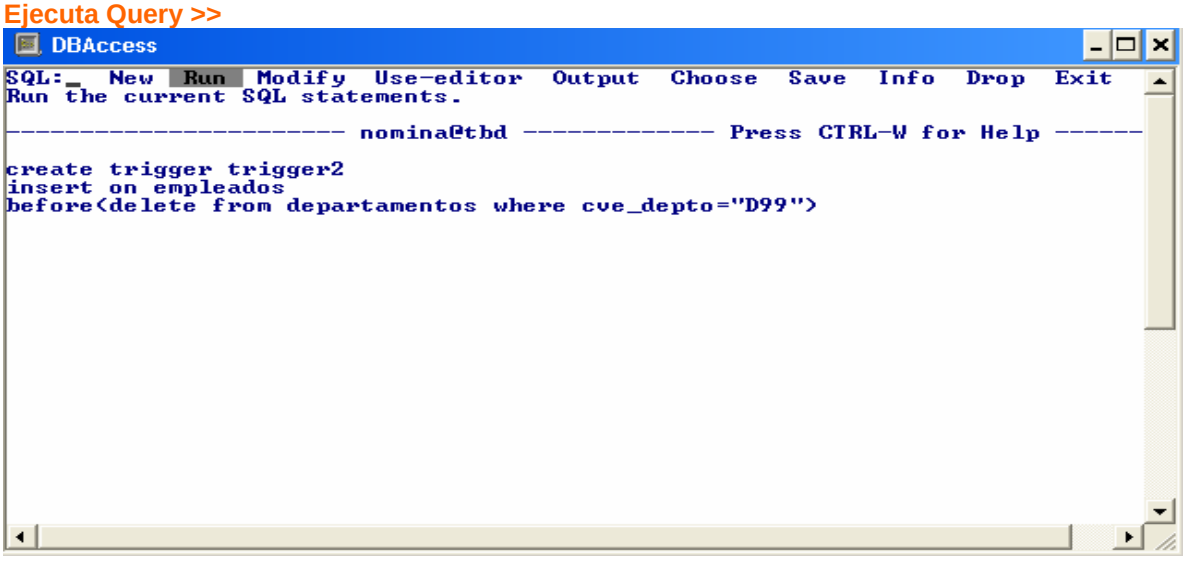
1. Material teórico visto en clase sobre el Manejador de Base de Datos y SQL.
2. Software del Manejador de Base de Datos instalado. EL SQL será estándar para cualquier manejador, pero en este caso se utilizará Informix On-Line Versión 7.30 en adelante o cualquier otro manejador de Base de Datos.
3. Preferentemente trabajar bajo Sistema Operativo Unix (Cualquier Arquitectura)
4. Se trabajará con dbaccess para el acceso de SQL o del monitor de SQL correspondiente al DBMS.
5. ***Siempre se trabajará con la Base de datos EJEMPLO que se encuentra en el capítulo introductorio.***
6. Debe existir información en las tablas de la bases de datos, esto ayuda a una mejor comprensión de los queries que se va a ir desarrollando en esta práctica.

Metodología

Se debe realizar un respaldo previo para seguridad de los datos. Puede utilizarse dbexport/dbimport.

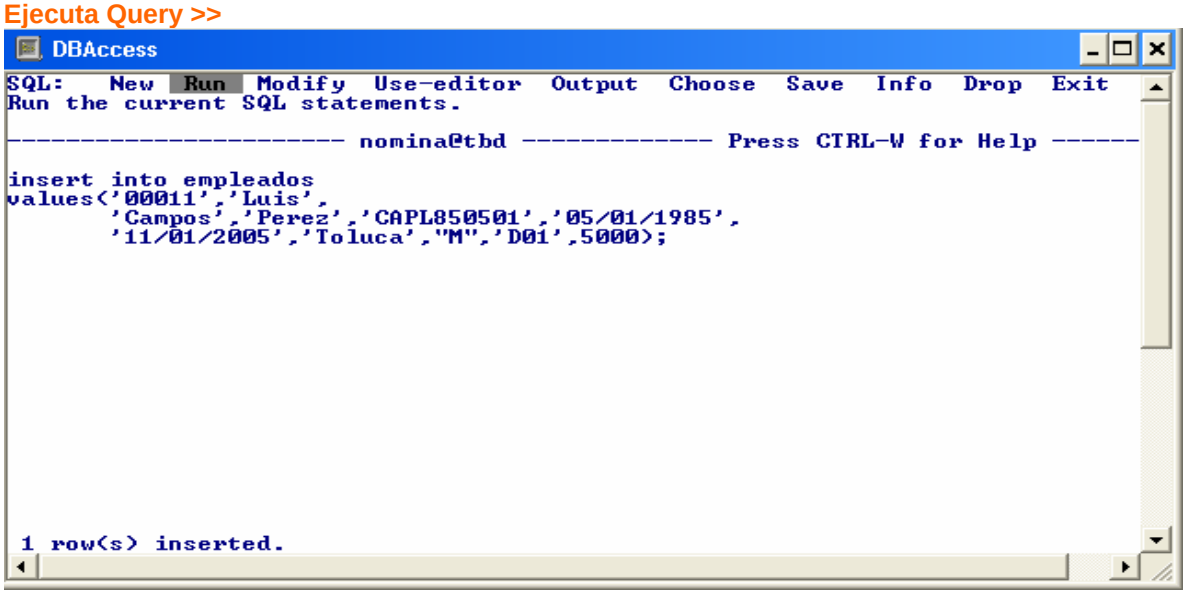
Ejercicio 1. **TRIGGER CON EVENTO INSERT . Crear un Trigger que cada que se dé de Alta un empleado, antes de insertarlo borre los datos correspondientes al departamento D99.**

Ejecuta Query >>



Se observa que el trigger2 se ejecuta cada que hay una inserción en EMPLEADOS y después (AFTER) de la inserción eliminará al depto. D99.

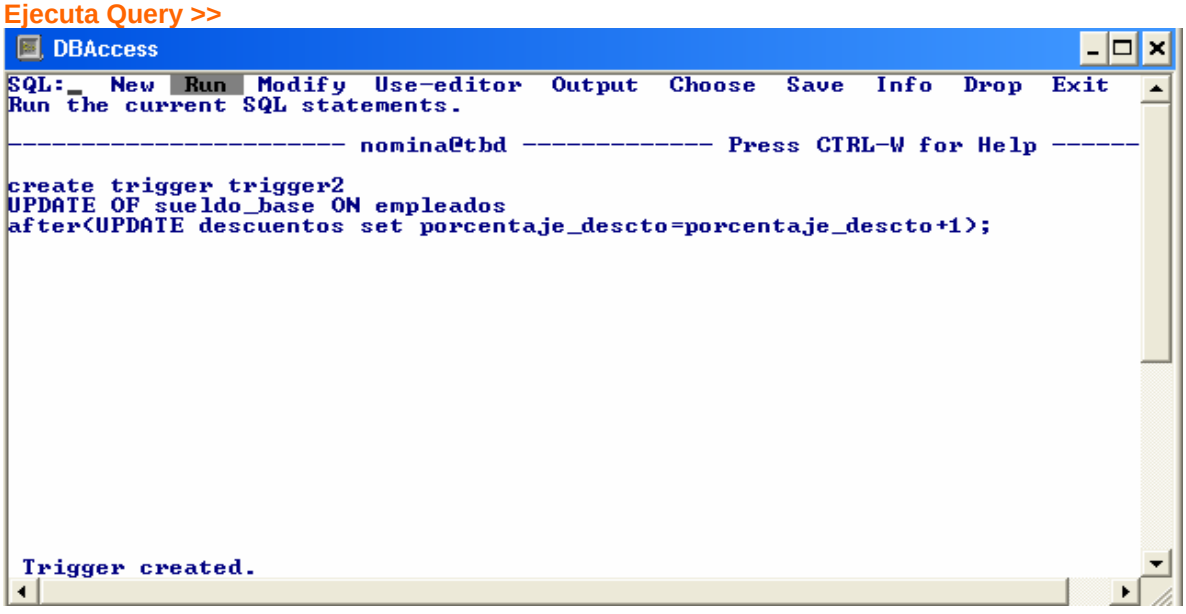
Ejecuta Query >>



La inserción se realiza sin problema y error eliminando al Depto. D99, exista o no.

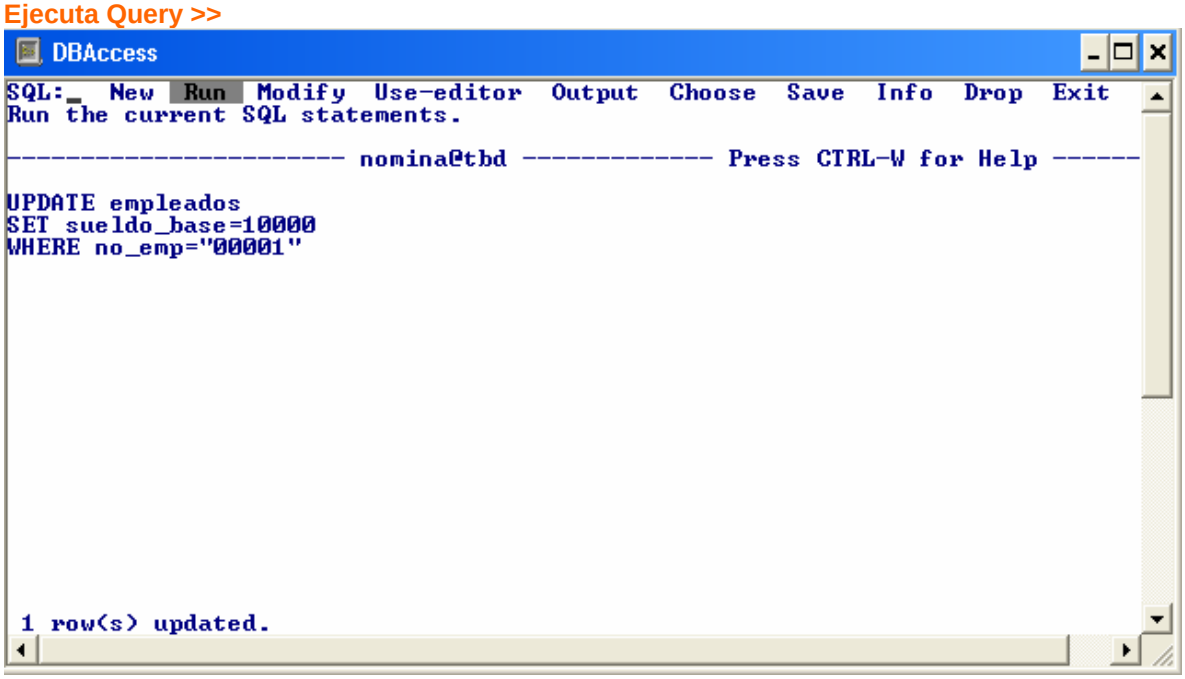
Ejercicio 2. TRIGGER CON EVENTO UPDATE. Crear un Trigger que cada que se actualice el campo de sueldo_base, incremente el porcentaje de descuento en 1 de la tabla DESCUENTOS.

Ejecuta Query >>



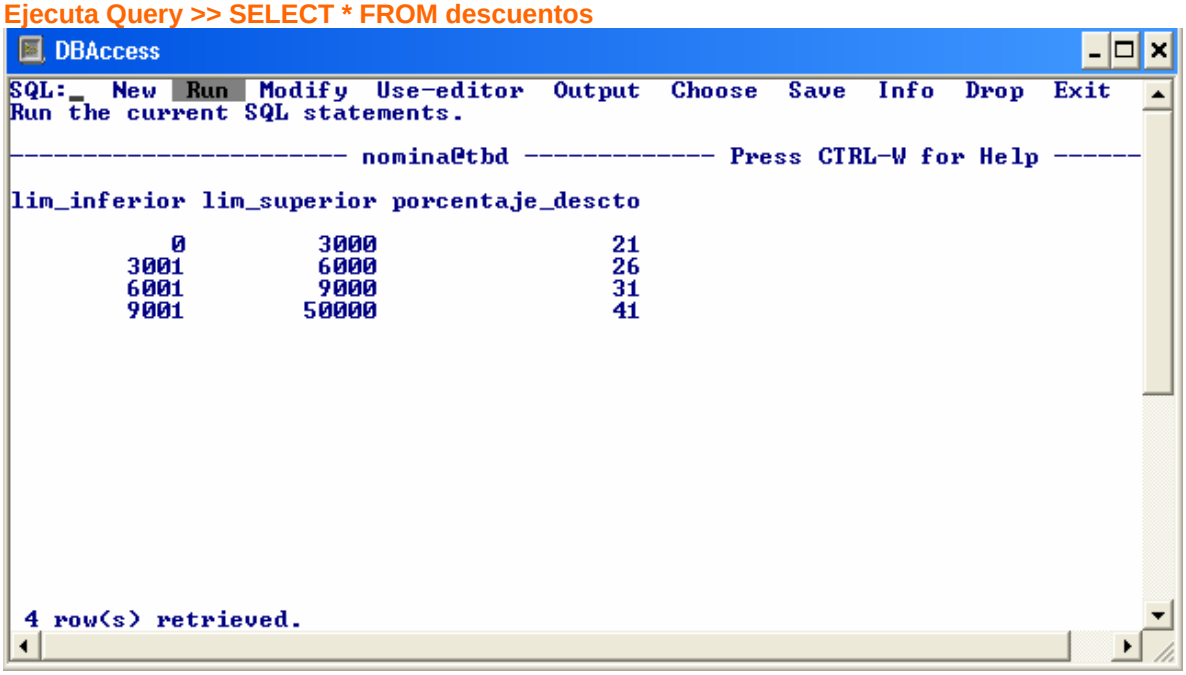
Conociendo los valores de la tabla de Descuento, se crea este trigger2, que se ejecutará automáticamente al actualizar únicamente el campo sueldo_base de la tabla EMPLEADOS.

Ejecuta Query >>



Se actualiza sueldo_base para el empleado 00001. Se ejecuta sin problemas actualizando la tabla de DESCUENTOS que se muestra.

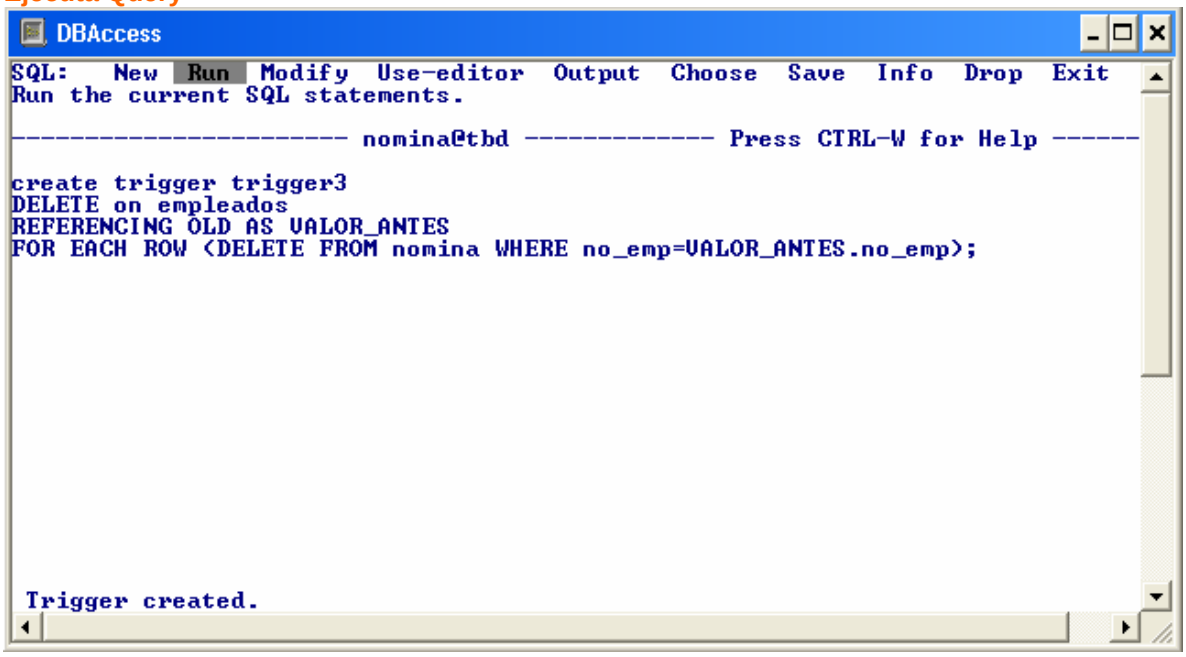
Ejecuta Query >> SELECT * FROM descuentos



Se observa el incremento en 1 punto en el porcentaje de descuento. Esta acción se dará cada que haya una actualización del Suelo_base.

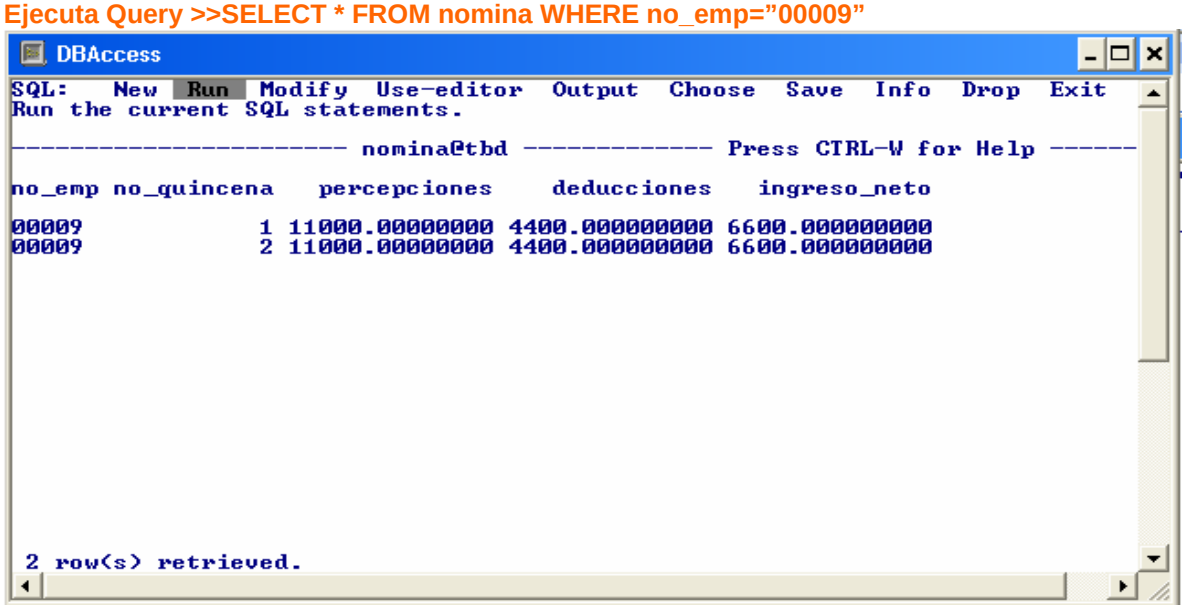
Ejercicio 3. TRIGGER CON EVENTO DELETE. Crear un Trigger que cada que se borre un empleado se borre el registro correspondiente de la tabla de NOMINA.

Ejecuta Query >>



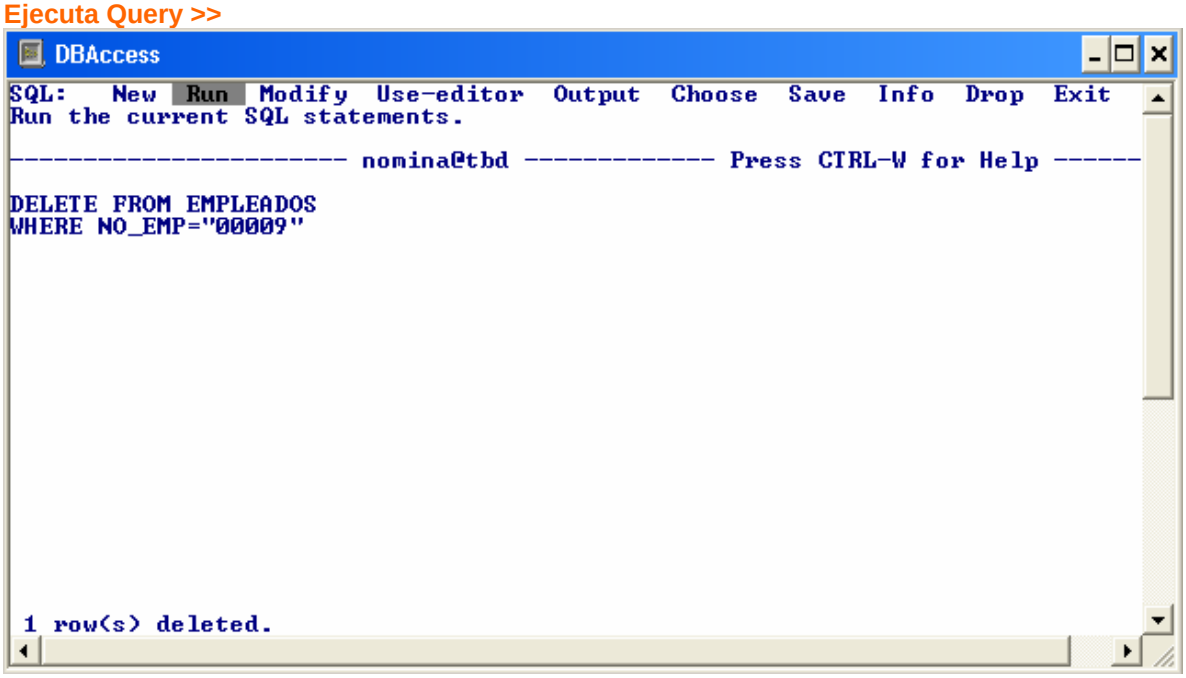
Se realiza la creación del Trigger3 sin error. La función del REFERENCING es la de tomar el valor del empleado que se borró y este lo aplicará para el borrado de ese empleado en la tabla de NOMINA. El FOR EACH ROW eliminará uno a uno los registros encontrados en NOMINA, dependiendo cuantas quincenas este procesado.

Ejecuta Query >>SELECT * FROM nomina WHERE no_emp="00009"



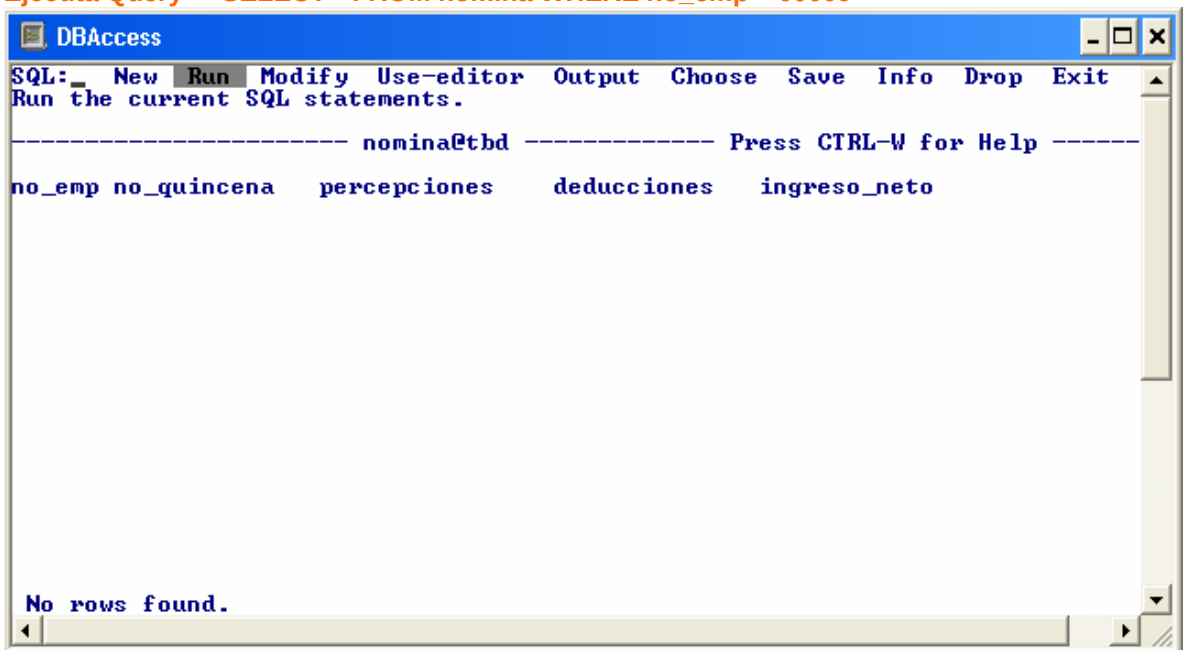
Se muestra que este empleado tiene dos registros en la tabla de NOMINA, antes de ser ejecutado el DELETE.

Ejecuta Query >>



Al eliminarse al empleado 00009, también automáticamente se borran todos los registros de la tabla de NOMINA.

Ejecuta Query >>SELECT * FROM nomina WHERE no_emp="00009"



Quedando por completo eliminados para el empleado 00009.

Sugerencias Didácticas

Esta práctica debe realizarse en equipos para apoyarse y resolver dudas y posteriormente cada uno de los estudiantes debe realizarla de manera individual. Usar comando en línea FINDERR no.error en cualquier problema de SQL.

Iniciar realizando el Diagrama por cada tabla para identificar las características generales de definición.

Trabajar directamente con el editor de SQL, creando directamente sin ayuda de menús los queries.

Tener presentes las definiciones de integridad referencial (Primary y Foreign Keys) y consistencia de datos (On Delete Cascade), para las diferentes consultas a realizar.

Se debe realizar siempre un respaldo previo a esta práctica, ya que implica la actualización de información.

Se deben conocer los Triggers almacenados para saber las acciones que se pueden realizar de cualquier Evento ejecutado.

Reporte del estudiante

EL estudiante entregará los queries de la práctica además de los que a continuación se piden como complemento, el resultado de cada uno de ellos y la información solicitada.

1. Crear un trigger/Stored Procedure que cuando se inserte un movimiento de nomina en la tabla MOVIMIENTOS, genere el registro correspondiente de Nómina en la tabla NOMINA.
2. Crear un Trigger/ Stored Procedure que genere para cada empleado a partir de sus datos generales su RFC y lo actualice en la tabla.
3. Crear un Trigger/ Stored Procedure que cuando se de de alta un empleado, indique cuantos existen en total en la empresa.
4. Crear un Trigger/ Stored Procedure que cuando se de de baja un empleado, indique cuantos quedan en total en la empresa.

Bibliografía o consultas sugeridas

http://hera.ittoluca.edu.mx
http://www.ibm.com.mx
http://hera.ittoluca.edu.mx/mescamilla

ANEXO I
Contenido del Curso

1. Introducción al Sistema Manejador de Base de Datos (DBMS)
    - 1.1 Conceptos.
    - 1.2 Características del DBMS
2. Lenguaje de Definición de Datos (DDL)
    - 2.1 Creación de base de datos
    - 2.2 Creación de tablas
        - 2.2.1 Integridad
        - 2.2.2 Integridad referencial declarativa
    - 2.3 Creación de índices
3. Consultas y Lenguaje de Manipulación de Datos (DML)
    - 3.1 Instrucciones INSERT, UPDATE, DELETE
    - 3.2 Consultas Básicas SELECT, WHERE y funciones a nivel de registro
    - 3.3 Consultas sobre múltiples tablas
        - 3.3.1 Subconsultas.
        - 3.3.2 Operadores JOIN
    - 3.4 Agregación GROUP BY, HAVING
    - 3.5 Funciones de conjunto de registros COUNT, SUM, AVG, MAX, MIN
4. Control de Transacciones
    - 4.1 Propiedades de la transacción
    - 4.2 Grados de consistencia
    - 4.3 Niveles de aislamiento
    - 4.4 Instrucciones COMMIT y ROLLBACK
5. Vistas
    - 5.1 Definición y objetivo de las vistas
    - 5.2 Instrucciones para la administración de vistas.
6. Seguridad
    - 6.1 Esquemas de autorización
    - 6.2 Instrucciones GRANT y REVOKE
7. Introducción al SQL Procedural
    - 7.1 Procedimientos almacenados
    - 7.2 Disparadores (Triggers)