	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)**

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE DE UNIDAD DE APRENDIZAJE	NOMBRE DE LA UNIDAD DE APRENDIZAJE
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORAS)
1	NOMBRE DE LA PRÁCTICA	Introducción al Laboratorio de Base de Datos	2

1. INTRODUCCIÓN

Las bases de datos forman una parte importante de los sistemas de información por lo que el saber cómo administrar, diseñar y consultar una base de datos es una parte fundamental de la formación de los Ingenieros en Computación.

Este laboratorio brindará la oportunidad de desarrollar competencia en el manejo de bases de datos y el desarrollo de programas que accedan a estas. Para asegurar un buen desempeño por parte de los alumnos, es importante conocer tanto la forma en que se trabajará durante el semestre como los reglamentos que rigen el uso de la infraestructura.

2. OBJETIVO (COMPETENCIA)

Conocer los reglamentos y forma de trabajar del laboratorio de base de datos.

3. FUNDAMENTO

El laboratorio de base de datos requerirá el acceso a la base de datos MySQL que está en el servidor con la dirección computación.mxl.uabc.mx por lo que cada alumno recibirá una cuenta para acceder a él. Será responsabilidad del alumno el bueno uso de la cuenta.

Formuló	Revisó	Aprobó	Autorizó
Cecilia Curlango Rosas	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Maximiliano de las Fuentes Lara
Nombre y Firma del Maestro	Nombre y Firma del Responsable de Programa Educativo	Nombre y Firma del Responsable de Gestión de Calidad	Nombre y Firma del Director de la Facultad

Así mismo cada alumno tendrá derechos para manipular las bases de datos del servidor. Debido a que la base de datos será compartida entre varios alumnos es importante evitar causar trastornos a los demás usuarios de la base de datos.
El laboratorio de computación opera bajo un reglamento que puede ser consultado en línea. La dirección de este se encuentra al final de la practica. Cada alumno deberá conocer el reglamento y seguirlo.

Durante este laboratorio, el maestro indicará cual será su forma de trabajar así como la forma en que se llevará a cabo la evaluación de la competencia de los estudiantes.
Finalmente, se entregarán las cuentas para trabajar en el servidor. Al recibir la cuenta, se deberá modificar el password de la misma utilizando el mando *yppasswd* desde una sesión local. Esto es, utilizar este mando sin estar en una sesión en el servidor. Así mismo, se deberá anotar el nombre completo y matricula en los datos de la cuenta.
Esto será empleando el mando *chfn* desde una sesión en el servidor. Por cuestión de seguridad, se deberá iniciar la sesión con *ssh* en lugar de utilizar *telnet*.

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL DE APOYO
Computadoras con Linux y acceso al servidor de trabajo	Cuenta de acceso al servidor.

B) DESARROLLO DE LA PRÁCTICA

1. Leer las reglas de uso de la cuenta en el servidor y después anotar su nombre, matricula y firma en la hoja de las cuentas. Al firmar esa hoja, usted indica que ha leído las reglas y que las acatará.

2. Ingresar a una máquina empleando su cuenta.

3. Abrir una sesión local y modificar su contraseña por medio del mando *yppasswd*.

4. Salga de todas sus sesiones y vuelva a ingresar para verificar que el cambio de contraseña se haya realizado. Si tiene algún problema consulte a su maestro.

5. Abra una sesión **ssh computacion.mxl.uabc.mx** empleando su nueva contraseña.

6. Modifique los datos de su cuenta empleando el mando *chfn*. Escriba su nombre completo como respuesta a la primer pregunta. Como respuesta a la segunda escriba su matrícula de la forma 01/XXXXX. Responda enter para las siguientes preguntas.

7. Verifique que los cambios hayan sido aceptados por medio del mando *cat /etc/passwd* | *grep XXXXX*. Donde XXXXX los sustituirá por su matrícula.
8. Lea el reglamento del laboratorio de Computación.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

Página de mysql http://dev.mysql.com/

7. REFERENCIAS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
2	NOMBRE DE LA PRÁCTICA	Algebra Relacional	2

1 INTRODUCCIÓN

El alumno se iniciará en el trabajar con un conjunto básico de operaciones para manipular los datos en el modelo relacional. Dada un base de datos con información se utilizaran las expresiones algebraicas para la utilización de SELECCIONAR, PROYECTAR y UNION.

2 OBJETIVO (COMPETENCIA)

El alumno aplicará el algebra relacional para la manipulacion de información de bases de datos.

Formuló	Revisó	Aprobó	Autorizó
M.C. Gloria E. Chavez Valenzuela	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

El algebra relacional esta formada por un conjunto básico de operaciones del modelo relacional. Estas operaciones van a permitir al usuario especificar las peticiones de recuperación básicas. El resultado de la recuperaciones da como resultado una nueva relación, que se forma apartir de una o mas relaciones. Por lo tanto las operaciones del algebra relacional producen nuevas relaciones que pueden manipularse en el futuro, utilizando operaciones de la misma algebra. Una secuencia de operaciones del algebra relacional forma una expresión del algebra relacional cuyo resultado será también una relación.

Los dos tipos de operaciones fundamentales son:
Unarias: operan sobre una sola relación (selección, proyección y renombramiento)
Binarias: operan sobre pares de relaciones (unión, diferencia de conjunto y productos cartesianos)

En este caso solo trabajaremos con las operaciones de Selección, Proyección y Unión.

Selección.

La selección nos va ha ser útil para seleccionar un subconjunto de renglones (tuplas) que satisfacen una condición de selección.

El símbolo de selección se representa por **σ**(sigma) y la condición se expresa en términos de atributos de la tabla a utilizar y R es una expresión del algebra relacional. La relación que resulta de la operación SELECCIONAR tiene los mismos atributos que R.

La forma general de denotar la operación de SELECCIONAR es la siguiente.

σ <condicion >(R)

condición
<nombre de atributo><operador de comparación><valor constante>
Ó
<nombre de atributo><operador de comparación><nombre de atributo>

Los operadores de comparaciones a utilizar son =, ≠, <, >, ≥,

Las condiciones llamadas también cláusulas, se puede dar, que sean varias al mismo tiempo y para conectar varias se utilizan los operadores boolanos Y(AND), O(OR) y NO(NOT).

De la siguiente tabla empleados tenemos los nombres de los campos
NOMBRE : Nombre de la persona
INIC: inicial
APELLIDO: Apellido de la persona
NSS: Número de Seguro Social.
FECHA_NCTO. Fecha de nacimiento.
DIRECCION: Dirección.
SEXO : Sexo del personal
SALARIO: Salario mensual
NSS_SUPERV : Número de seguro social de supervisor
ND : Numero de departamento.

Tabla 1

EMPLEADOS

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECION	SEXO	SALARIO	NSS_SUPERV	ND
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	H	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	H	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	M	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	M	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	H	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	M	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	H	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	H	55000	nulo	1

Ejemplo de una condición sencilla de SELECCIONAR

Ejemplo 1
Buscar a en la tabla EMPLEADOS (**Tabla 1**) todos aquellos empleados donde el número de departamento(ND) donde trabajan sea 4 y mostrarlo.

$\sigma_{ND=4}$(EMPLEADOS)

Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	M	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	M	43000	888665555	4
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	H	25000	987654321	4

Ejemplo de una condición con operadores booleanos
Buscar en la tabla EMPLEADOS todos aquellos donde el número de departamento(ND) donde trabajan sea 4 y salario mayor de 25000 ó el número de departamento(ND) donde trabajan sea 5 y salario mayor de 30000 y mostrarlo.

$\sigma_{(ND=4 \text{ y } SALARIO > 25000) \text{ O } (ND=5 \text{ Y } SALARIO>30000)}$(EMPLEADO)

Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	H	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	M	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	H	38000	333445555	5

Lo que se obtiene de la selección son todas las columnas de la tabla donde se cumpla la condición.

3 FUNDAMENTO

PROYECCION.

Proyeccion trabaja en la forma de que solo selecciona ciertas columnas de la tabla y desecha todas las demas a diferencia de la selección que trabaja con los renglones.

La forma general de denotar la operación de PROYECTAR es la siguiente.

π <lista de atributos >(R)

El resultado de la operación PROYECTAR contienen únicamente los atributos especificados en la lista de atributos.
Nota: se omiten los renglones con información repetida

De la tabla 1 de empleados realizamos solo la proyeccion de las columas que nosotros queramos vizualizar.

Ejemplo

Mostrar SEXO y EDAD de la tabla empelados
π <SEXO, SALARIO >(EMPLEADO)

SEXO	SALARIO
H	30000
H	40000
M	25000
M	43000
H	38000
H	25000
H	55000

De la tabla 1 de empleados ahora solo le decimos que nos muestre la columna SEXO y SALARIO
En el resultado omite las filas que se encuentren repetidas de información, si comparamos con la tabla original aquí no muestra la fila de sexo M y de salario 25000 que pertenece a JOYCE debido a que ya se encontraba en la tabla.
NOTA: no presentaran las filas repetidas, solo de las columnas que se le solicitaron.

Formato para prácticas de laboratorio

3 FUNDAMENTO

También es posible que ya que conozcamos varias operaciones del algebra relación, podamos anidarlas y hacer uso de las mismas. Por lo que en este caso ya podemos utilizar la operación de SELECCIONAR y PROYECTAR y obtener un resultado y poner la información que se genero en otra tabla.

Ejemplo

Mostrar nombre, apellido y salario de los empleados del que trabajan en el departamento 5. Para esto primero seleccionamos a los empleados que trabajan en el departamento numero 5 y después proyectamos solo las columnas deseadas en este caso es nombre, apellido y salario.

La expresión algebraica seria:

$\pi_{<APELLIDO, NOMBRE, SALARIO}(\sigma_{ND=5}(EMPLEADO))$

Esto realizar lo siguiente.

$EMPLEADO\_DEPTO5 \leftarrow \sigma_{ND=5}(EMPLEADO)$
$RESULTADO \leftarrow \pi_{<APELLIDO, NOMBRE, SALARIO}(EMPLEADO\_DEPTO5))$

John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

3 FUNDAMENTO

UNION
El resultado de la operación, R ∪ S, es una relación que incluye todas las columnas que están en R o en S o en ambas(en una tabla o en la otra o en ambas). Los renglones repetidos se eliminan y no se muestran.

La forma general de denotar la operación de UNION es la siguiente.

Tabla1 ∪ Tabla2

Operación de asignación.
Sirve para asignar temporalmente una relación. Se realiza mediante una flecha dirigida de izquierda a derecha

Obtener los números de seguro social de todos los empleados que trabajan en el departamento 5 o que supervisan directamente a un empleado que trabaja en ese mismo departamento.

1ro. Para esto primero creamos una tabla llamada EMPLEADOS _DEPTO5 donde tendremos a todos los empleados del departamento 5

EMPLEADOS _DEPTO5 ← $\sigma_{ND=5}$(EMPLEADOS)

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECION	SEXO	SALARIO	NSS_SUPERV	ND
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	H	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	H	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	H	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	M	25000	333445555	5

2do. Después obtendremos solo los números de seguro social(**NSS**) de la tabla EMPLEADOS _DEPTO5 y los asignaremos RESULTADO1

RESULTADO1 ←π_{NSS}(EMPLEADOS _DEPTO5)

NSS
123456789
333445555
666884444
453453453

3ro. Tomar los números de seguro social de los supervisores de la tabla EMPLEADOS_DEPTOS5 y los pondremos en RESULTADO2

RESULTADO2 ←π_{NSS_SUPERV}(EMPLEADOS _DEPTO5)

NSS_SUPERV
333445555
888665555

4to. Ahora realizamos la unión de la tabla RESULTADO1 y RESULTADO2 y la ponemos en RESULTADO
RESULTADO ←RESULTADO1 ∪ RESULTADO2

NSS
123456789
333445555
666884444
453453453
888665555

TABLA 2

DEPARTAMENTO

NOMBRED	NUMEROD	NSS_JEFE	FECHA_INIC_JEFE
Investigacion	5	333445555	1988-05-22
Administracion	4	987654321	1995-01-01
Direccion	1	888665555	1981-06-19

Tabla 3

DEPENDIENTES

NSSE	NOMBRE_DEPENDIENTE	SEXO	FECHA_NCTO	PARENTESCO
333445555	Alice	M	1986-04-05	HIJA
333445555	Theodore	H	1983-10-25	HIJO
333445555	Joy	M	1958-05-03	ESPOSA
987654321	Abner	H	1942-02-28	ESPOSA
123456789	Michael	H	1988-01-04	HIJO
123456789	Alice	M	1988-12-30	HIJA
123456789	Elizabeth	M	1967-05-05	ESPOSA

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con Linux y MySQL.	Práctica impresa y estudiada

B DESARROLLO DE LA PRÁCTICA

1.- Obtener todos los datos del empleado cuyo nombre es Jhon y apellido Smith. En la tabla empleados

2.- Obtener la fecha de nacimiento y dirrección del empleado cuyo nombre es Jhon y apellido Smith. En la tabla empleados

3.- De la tabla dependientes obtener toda la información dende el nombre del depenediente sea Alice y parentesco sea hija

4.- De la tabla dependientes obtener número de seguro social y nombre del dependiente.

5.- De la tabla1 llamado empledos liste a todos aquellos empleados que se apllido se SMITH.

6.- Exprese el resultado de la soguiente operación de algebra relacional.

$\pi_{<NOMBRE, APELLIDO, SALARIO}(\sigma_{NOMBRE, < 50000}(EMPLEADO))$

7.- De la tabla empleados muestre todos los salarios de los trabajadores

8.- De la tabla empleados muestre todos los datos, de los empelados que trabajan en el departamento 5 y cuyo salario esté entre 30000 y 40000

9.- De la tabla DEPENDIENTES mostrar todos los numeros de seguro social y nombre de los dependientes de todos cuyos sexo sea “M”

10.- Dentro del laboratorio investigar la diferencia antre UNION Y UNION EXTERNA

C	CÁLCULOS Y REPORTE

5 RESULTADOS Y CONCLUSIONES

6 ANEXOS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
3	NOMBRE DE LA PRÁCTICA	Modelo Entidad - Relación	4

1 INTRODUCCIÓN

El modelo **Entidad-Relación** también llamado modelo **E-R**, es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Fue introducido por Peter Chen en 1976.

El modelo entidad-relación está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas, es uno de los modelos lógicos basados en objetos y por lo tanto se enfoca primordialmente a los niveles conceptual y de visión.

Una de las características de este modelo es que permite representar con claridad las limitantes de los datos. El modelo Entidad-Relación es en esencia una herramienta para representar el mundo real por medio de simbologías y expresiones determinadas.

2 OBJETIVO (COMPETENCIA)

El alumno desarrollará modelos conceptuales de bases de datos utilizando el modelo Entidad–Relación para describir las entidades y las relaciones existente en una base de datos.

3 FUNDAMENTO

El modelo **E-R** describe los datos como entidades, relaciones (vínculos) y atributos y permite representar el esquema conceptual de una base de datos de forma gráfica mediante los diagramas **E-R.**

Entidades:

El objeto básico que se representa en el modelo E-R es la entidad, una entidad es "cualquier objeto del mundo real con existencia propia, sobre el cual queremos tener información en una base de datos". Una entidad puede ser un objeto con existencia física, por ejemplo: (una persona, una casa, un empleado, un coche,..) o un objeto con existencia conceptual (una empresa, un puesto de trabajo, un curso universitario,...).

En los diagramas E-R, las entidades se representan mediante un rectángulo y dentro del mismo se pone el nombre. Por ejemplo: CLIENTE, PROVEEDOR, ARTICULO, COCHE, etc. Debemos elegir nombres que comuniquen, hasta donde sea posible, el significado de cada entidad. Normalmente se utilizan nombres en singular y no en plural.

Tipos de entidades:

Hay dos tipos de entidades: **fuertes(o regulares) y débiles.**

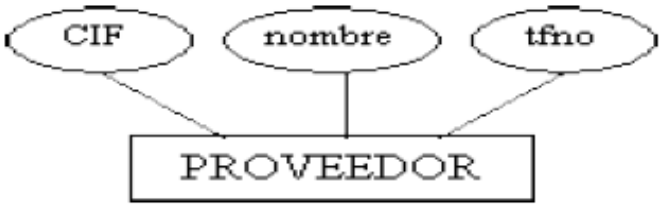
Fuertes (o regulares): son aquellas que tienen existencia por si mismas (Por ejemplo, EMPLEADO). Las entidades fuertes se representan como se ha dicho con un rectángulo con trazo simple.

Débiles: cuya existencia depende de otro tipo de entidad (Por ejemplo, FAMILIAR depende de EMPLEADO. La desaparición de un empleado de la base de datos hace que desaparezcan también todos los familiares del mismo). Estos tipos de entidades se representan normalmente con un rectángulo con líneas de doble trazo. Estas entidades normalmente no tienen suficientes atributos para formar una clave primaria.

Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil.

FUNDAMENTO

Cada entidad tiene propiedades específicas, llamadas atributos, que la describen. Por ejemplo, una entidad PROVEEDOR puede describirse por su número de proveedor (C.I.F.), su nombre, su teléfono, etc., una entidad EMPLEADO puede describirse por su nombre, edad, dirección,salario y puesto. Los atributos se representan por elipses que están conectadas a su entidad o relación mediante una línea recta.



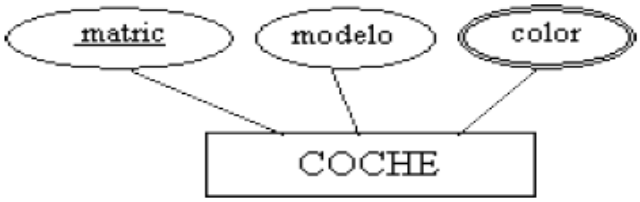
Al conjunto de valores que puede tomar cada atributo dentro de una entidad se le llama dominio del atributo.

Toda entidad debe tener al menos un atributo que permita diferenciar unas entidades particulares de otras, es decir que no toman nunca el mismo valor para dos entidades particulares diferentes. A estos atributos se les llaman claves. En el diagrama E-R los atributos clave deben aparecer destacados; por ejemplo, subrayando su nombre (por ejemplo, CIF de la entidad PROVEEDOR.

Tipos de atributos:

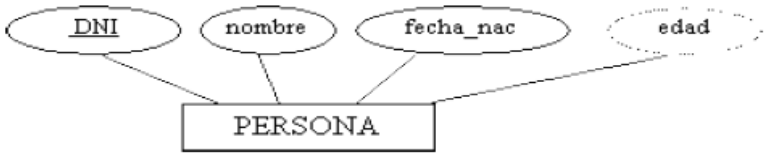
- Simples o compuestos: Los compuestos están formados por un conjunto de atributos, mientras que los simples no se pueden dividir.

- Monovaluados o multivaluados: Los monovaluados sólo pueden tener un valor para una entidad particular, mientras que los multivaluados pueden tener más de un valor. Los multivaluados se representan mediante una elipse con trazado doble. (Por ejemplo el atributo color de la entidad COCHE es un atributo multivaluado, pues un coche puede estar pintado de varios colores).



3 FUNDAMENTO

- **Almacenados o derivados:** Los derivados son atributos cuyo valor para una entidad particular puede obtenerse en función de los valores almacenados en otros atributos. Se representan mediante una elipse con trazo discontinuo. (Por ejemplo el atributo edad de la entidad PERSONA es un atributo derivado porque se puede obtener en función del valor del atributo fecha_nacimiento).
-



En 1979, Tardieu, propone tres reglas generales que debe cumplir una entidad:

* Tiene que tener existencia propia.
* Cada ocurrencia de un tipo de entidad debe poder distinguirse de las demás.
* Todas las ocurrencias de un tipo de entidad deben tener los mismo tipos de propiedades (atributos).

Vínculo o relación:

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. En los diagramas E-R se representa gráficamente como un rombo y sus nombres son verbos que aparecen dentro del rombo. Por ejemplo: VENDE, PERTENECE, etc.

Una relación puede tener atributos descriptivos. Por ejemplo, en la relación anterior, podría tener como atributo descriptivo fecha_venta (la fecha en que se hace la venta).

Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc. Se puede restringir el modelo **E-R** para incluir solo conjuntos de relaciones binarias, es decir de grado 2 (es aconsejable).

3 FUNDAMENTO

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

La cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. Según su cardinalidad, podemos clasificar las relaciones de los siguientes tipos:

Tipos de participación de las entidades en una relación:

- **Opcional (parcial)**: No todas las ocurrencias de una entidad tienen que estar relacionadas con alguna de la otra entidad. Se representa mediante una línea con trazo sencillo. (Por ejemplo, no toda persona posee animales, y no todo animal es posesión de alguna persona. En este caso ambas entidades participan parcialmente en la relación).

– **Obligatoria (total):** Todas las ocurrencias de una entidad deben estar relacionadas con alguna de la entidad con la que esta relacionada. Se dice también, que existen una participación total de ese conjunto de entidades en el conjunto de relaciones, y se representa mediante una línea con trazo doble. (Por ejemplo, todo proveedor tiene que vender algún artículo para serlo, y todo artículo es vendido por algún proveedor. En este caso ambas entidades participan de forma total en la relación).

REDUCCIÓN DE DIAGRAMAS E-R A TABLAS.

Con el objeto de observar instancias de las bases de datos, los diagramas E-R se convierten en tablas, Se obtiene una tabla por cada conjunto de entidades o de relaciones.

Existen reglas bien definidas para la conversión de los elementos de un diagrama E-R a tablas:

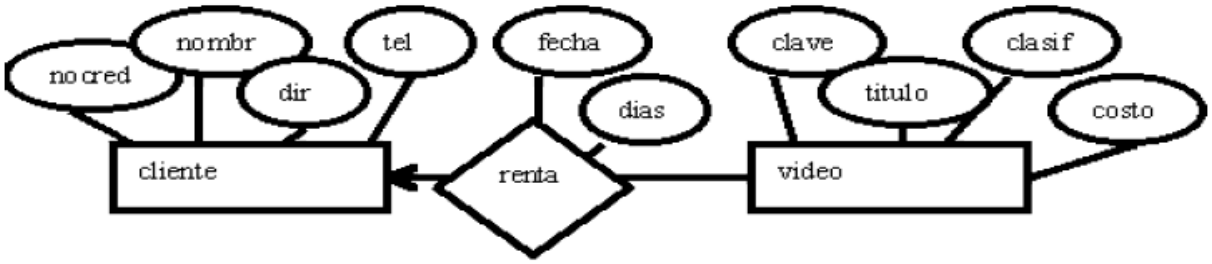
a) **ENTIDADES FUERTES**.- Se crea una tabla con una columna para cada atributo del conjunto de entidades.

3 FUNDAMENTO

b) **ENTIDADES DÉBILES.-** Se crea una tabla que contiene una columna para los atributos que forman la llave primaria de la entidad fuerte a la que se encuentra subordinada.

c) **RELACIÓN**.- se crea una tabla que contiene una columna para cada atributo descriptivo de la relación y para cada atributo que conforma la llave primaria de las entidades que están relacionadas.

En el siguiente ejemplo se convierte a tablas y se muestran instancias para observar la CARDINALIDAD en el diagrama E-R para el caso de un vídeo club.



TABLAS

CLIENTES

Nocred	Nombre	Dirección	teléfono
10	Pedro	Allende	2-10-40
15	Juan	Ramírez	2-13-45
20	María	16 de Sep.	5-67-89
25	Ana	5 mayo	2-34-32
30	Dora	5 de Feb.	1-23-12

RENTA

nocred	clave	Fecha	días
10	B10	10/09/98	
10	C10	10/09/98	
25	B10	12/09/98	

PELÍCULA

clave	titulo	clase	costo
A10	LA ROCA	B	10.00
A15	TORNADO	B	10.00
B10	BATMAN	A	10.00
B20	ROM.	B	10.00
B30	MAXV.	C	10.00

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con Linux instalado y Software para dibujo.	Práctica impresa.

B DESARROLLO DE LA PRÁCTICA

A continuación se describen ciertos requisitos sobre un problema en particular para el cual deberás realizar un análisis y generar un modelo conceptual utilizando el modelo Entidad-Relación, una vez generado y revisado el modelo que obtuviste como resultado, utilizarás alguna herramienta de diseño para digitalizarlo.

La bolsa de trabajo de **CANACINTRA** (BTC) coloca algunos trabajadores temporales en determinadas compañías durante períodos pico, esto es, durante espacios de tiempo donde es mucha la demanda de trabajadores debido a la cantidad de trabajo a desarrollar.

El gerente de BTC, describe el proceso de colocación de empleados temporales de la siguiente forma:

- BTC conserva un archivo de candidatos que desean trabajar.
- Si el candidato ha trabajado antes, ya cuenta con un historial de trabajo específico (obviamente, no existe ningún historial de trabajo si el candidato nunca ha trabajado. Cada vez que el candidato trabaje, se crea un historial de trabajo adicional).
- A cada candidato se le ha asignado un código de acuerdo a una tabla de asignaciones en función a su perfil académico o a la(s) especialidad(es) que tiene. Cada candidato puede ser asignado con varios códigos, así como cada código puede ser asignado a mas de un candidato (por ejemplo, es posible que mas de un candidato haya obtenido una licenciatura, una especialidad o una certificación de Microsoft).
- BTC también conserva una lista de compañías que solicitan empleados temporales.
- Cada vez que una compañía solicita un empleado temporal, BTC hace una entrada en el archivo Vacantes. Este archivo contiene un número de vacante, nombre de la compañía, códigos o especialidades requeridas, fecha de inicio, fecha de terminación y pago por hora.
- Cada vacante requiere solamente de un código o especialidad.
- Cuando un candidato tiene el código o especialidad requerida, él o ella obtiene el trabajo, y se hace una entrada en el archivo Registro de Colocaciones. Este archivo contiene número de vacante, numero de candidato, horas totales trabajadas, etc. Además se hace una entrada en el historial de trabajo del candidato.
- BTC utiliza códigos especiales para describir las especialidades de un candidato para una vacante. La lista de códigos incluye los siguientes:

4 PROCEDIMIENTO (DESCRIPCIÓN)

CODIGO	DESCRIPCION
SEC-45	Trabajo secretarial, por lo menos 45 palabras por minuto
SEC-60	Trabajo secretarial, por lo menos 60 palabras por minuto
OFIC	Trabajo de oficina general
PRG-VB	Programador, Visual Basic
PRG-CPP	Programador, C++
DBA-ORA	Administrador de Base de Datos, Oracle
DBA-MSQL	Administrador de Base de Datos, MySQL
SYS-1	Analista de Sistemas, nivel 1
SYS-2	Analista de Sistemas, nivel 2
NW-NOV	Administrador de Red, experiencia en Novell

La BTC desea dar seguimiento a las siguientes entidades:

- COMPAÑÍA
- VACANTE
- CALIFICACION
- CANDIDATO
- HISTORIAL_TRABAJO
- COLOCACION

Con la información anterior, haga lo siguiente:

- Identifique todas las entidades faltantes a la lista anterior.
- Identifique todos los atributos para cada una de las diferentes entidades identificadas.
- Identifique todas las relaciones posibles así como sus cardinalidades.
- Dibuje el diagrama Entidad-Relacion utilizando los elementos encontrados en los pasos anteriores.
- Resuelva todas las relaciones M:N

C CÁLCULOS Y REPORTE

El alumno entregará al maestro una copia hecha a mano del modelo conceptual propuesto como solución para el ejercicio antes descrito utilizando el modelo entidad-relación, además, entregará una copia del mismo modelo pero elaborado con la herramienta de dibujo que seleccionó.

5 RESULTADOS Y CONCLUSIONES

El alumno será capaz de modelar cualquier problema de la vida real, para el cual, se necesite conservar datos almacenándolos en una base de datos, el alumno utilizará el modelo entidad-relacion, y ofrecerá de esta forma la mejor solución posible.

6 ANEXOS

Para mayor informacion puede consultar ayuda en linea en las siguientes direcciones:

http://www.itlp.edu.mx/publica/tutoriales/basedat2/hdos2_1.htm
http://www3.uji.es/~mmarques/f47/apun/node83.html
http://www3.uji.es/~mmarques/f47/apun/node83.html

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
4	NOMBRE DE LA PRÁCTICA	Introducción a la Administración de MySQL	2

1. INTRODUCCIÓN

Uno de los manejadores de bases de datos que se utilizará durante este curso es el manejador MySQL. Este manejador aun cuando esta disponible de manera gratuita, resulta ser poderoso y de existen actualmente muchas empresas que lo utilizan para almacenar su información.

2. OBJETIVO (COMPETENCIA)

Al finalizar esta practica se tendran los conocimientos para la administración demostrativa, se conocerá la forma de instalar MySQL tanto en el sistema operativo Windows como en Linux. Adicionalmente se conocerá la forma de trabajar con el MySQL desde la consola de administración para hacer consultas. Finalmente se visitarán algunos sitios a los que se puede recurrir cuando se tienen dudas sobre MySQL.

Formuló	Revisó	Aprobó	Autorizó
Nayely Amaro Ortega	M.C. Gloria E. Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3. FUNDAMENTO

El manejador de bases de datos MySQL puede instalarse tanto en la plataforma de Windows como en la de Linux y otros sistemas operativos. En esta practica se hará una demostración del proceso de instalación en ambos sistemas operativos.

Instalación en Windows

Para realizar la instalación de MySQL en Windows, es necesario contar con el archivo de instalacion de MySQL que puede descargarse del sitio de MySQL. El archivo para esta demostración es: **`mysql-essential-4.1.14-win32.msi`** y el proceso de instalación se describe a continuación.

Proceso de instalación

1. Descargar el archivo **`mysql-essential-4.1.14-win32.msi`** (Este se puede descargar del sitio de MySQL http://dev.mysql.com/downloads/mysql/4.1.html).
2. Haga doble-click en el archivo que se descargó.
3. Siga las indicaciones de instalación.
4. La ultima pantalla de instalación tendrá un checkbox donde pregunta si quiere configurar en ese momento el servidor, seleccionelo.
5. Iniciará el asistente para la configuración del servidor.
6. Acepte la configuración detallada y presione *next*.
7. Indique que el tipo de maquina será *developer machine* y presione *next*.
8. Acepte la opción Multifunctional Database y presione *next*.
9. Acepte los valores por default para la ubicación de la base de datos y presione *next*.
10. Acepte 20 conexiones simultaneas por default y presione *next*.
11. Habilite TCP/IP Networking aceptando el puerto 3306 y presione *next*.

12. Acepte el conjunto de caracteres default y presione *next*.
13. Acepte que se instale MySQL como un servicio de windows y que se lance automaticamente al cargar Windows. Si no lo quiere como servicio, lo deberá levantar el servicio manualmente cada vez que quiera trabajar con la base de datos.
14. Tambien seleccione la opción Include bin Directory in windows PATH para que pueda correr los ejecutables desde la línea de mandos. Presione *next*.
15. Ahora debe escribir el password de root. Anotelo y no lo olvide.
16. Cree tambien una cuenta anonima ya que su servidor será para aprendizaje.
17. Presione *Execute* e iniciará la configuración. Cuando termine, presione Finish.

Instalación en Linux

El proceso de instalación de MySQL en el sistema operativo Linux puede variar dependiendo del tipo de archivo que descargue. El mas fácil de instalar es el tipo RPM (originalmente RPM significaba Red Hat Package Manger). En el sitio de MySQL podemos encontrar un RPM para el servidor (**`MySQL-server-4.0.26-0.i386.rpm`**), otro para el cliente (**`MySQL-client-4.0.26-0.i386.rpm`**) y algunos otros para el desarrollo de aplicaciones.

El otro tipo de archivo que se puede descargar es aquel con terminación **`tar.gz`** por ejemplo (**`mysql-standard-4.0.26-pc-linux-gnu-i686.tar.gz`**). Este tipo de archivo primero debe descomprimirse y desempaquetarse antes de poderse instalar. En algunos casos contiene el código fuente de la aplicación por lo que deberá configurarse y después compilarse. En otros casos no será necesario compilar y bastará con mover el directorio que se creo al desempaquetarse al directorio final de MySQL. Usualmente los archivos tar.gz contienen algún tipo de archivo de ayuda o README que indica el procedimiento que deberá seguirse para la instalación del software deseado.

Por otra parte, la mayoría de las distribuciones de Linux ya incluyen MySQL como un de los paquetes standard. En este caso, la instalación de MySQL dependerá de la distribución que se este manejando. A continuación se describe el proceso que se seguiría con la distribución Mandriva (antes Mandrake).

Proceso de instalación

1. Abrir el instalador de paquetes seleccionando del menu principal System-Configuration-Packaging-Install Software.
2. Escriba la contraseña de root cuando le sea solicitada.
3. Cuando aparezca la ventana del instalador, escriba mysql en el área de texto para que búsque todos los paquetes relacionados. Seleccione todos los paquetes como se indica en la Figura 1. En ocaciones, al seleccionar el primer paquete, se detectará que la instalación de este dependerá de la instalación de otros paquetes y aparecerá una ventana indicando esto. En esta ventana se puede indicar que se desea que se instalen todos los paquetes necesarios para resolver las dependencias.

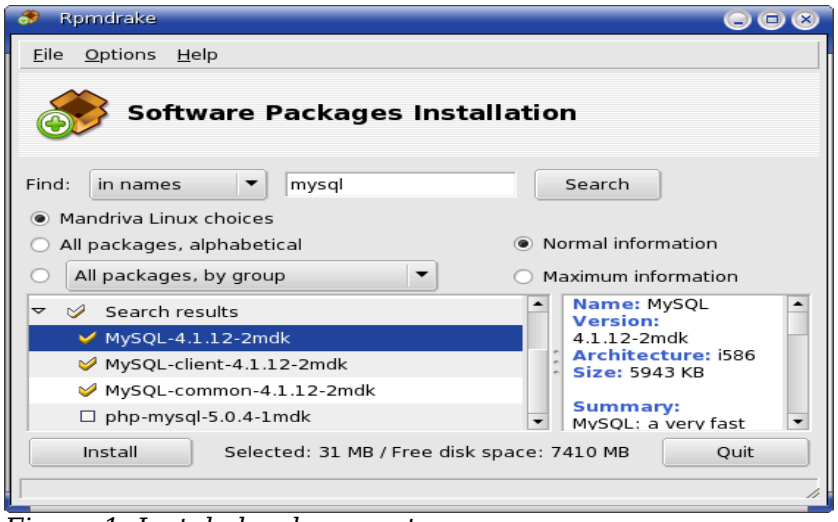


Figura 1: Instalador de paquetes.

4. Una vez seleccionados todos los paquetes que se desean instalar, presione el botón Install y procederá la instalación. Durante la instalación, el programa solicitará que inserte el CD que contenga los paquete que va requiriendo.
5. Una vez terminada la instalación, cierre la ventana del instalador.
6. Para verificar que el servidor MySQL este levantado y que se va a levantar cada vez que se encienda la máquina, se debe consultar el listado de servicios. Esto se pude acceder desde el menu principal seleccionando System-Configuration-Configure your computer. Una vez que se escribe la contraseña de root, seleccionar System en el Control Center.
7. Seleccionar el icono correspondiente a servicios Figura 2.

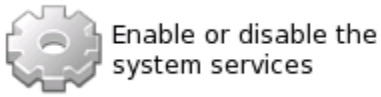


Figura 2: Configuración de Servicios

8. Aparecerá un listado con todos los servicios disponibles en el sistema. Ahi deberá buscar el servicio del servidor MySQL y activarlo y si lo desea indicar que este debe levantarse cada vez que se reinicie el servidor. La Figura 3 muestra un listado parcial de servicios.

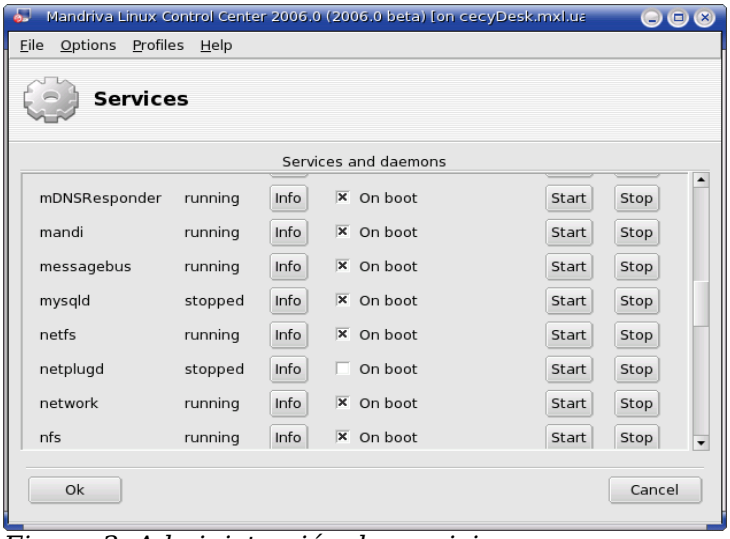


Figura 3: Administración de servicios

Administración de MySQL

Dentro de la administracion de MySQL las operaciones que podemos realizar sobre los usuarios son:

- Crear usuario
- `Asignar Password`
- Renombar usuario
- Borrar usuario
- Otorgar y revocar permisos a usuario

Crear Usuario

Este comando crea nuevas cuentas MySQL. Para usarlas, debe tener el permiso global CREATE USER o el permiso INSERT para la base de datos mysql . Para cada cuenta, CREATE USER crea un nuevo registro en la tabla mysql.user que no tiene permisos. Un error ocurre si la cuenta ya existe

```
mysql> CREATE USER XYZ3@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Asignar Password

El comando SET PASSWORD asigna una contraseña a una cuenta de usuario MySQL existente.

```
mysql> SET PASSWORD FOR XYZ3@'localhost'=PASSWORD('XYZ3');
Query OK, 0 rows affected (0.00 sec)
```

Renombrar Usuario

El comando RENAME USER renombra cuentas de usuario MySQL existentes. Para usarlo, debe tener el permiso CREATE USER global o el permiso UPDATE para la base de datos mysql . Ocurre un error si cualquier de las antiguas cuentas no existe o cualquiera de las nuevas ya existe.

```
mysql> RENAME USER XYZ3@'localhost' TO XYZ33@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

Borrar Usuario

El comando DROP USER borra una o más cuentas MySQL . Para usarlo, debe tener el permiso global CREATE USER o el permiso DELETE para la base de datos mysql.

```
mysql> DROP USER CALIS2;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Otorgar o Revocar Permisos a Usuario

Los comandos GRANT y REVOKE permiten a los adminitradores de sistemas crear cuentas de usuario MySQL, darles permisos o quitarlos de las cuentas. Los permisos pueden darse en varios niveles:

- *Nivel global*

  Los permisos globales se aplican a todas las bases de datos de un servidor dado. Estos permisos se almacenan en la tabla `mysql.user`. `GRANT ALL ON *.*` y `REVOKE ALL ON *.*` otorgan y quitan sólo permisos globales.

```
mysql> grant all on *.* to xyz2@'localhost' identified by 'xyz2';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> revoke all on *.* from xyz2@'localhost' identified by 'xyz2';
Query OK, 0 rows affected (0.00 sec)
```

- *Nivel de base de datos*

  Los permisos de base de datos se aplican a todos los objetos en una base de datos dada. Estos permisos se almacenan en las tablas `mysql.db` y `mysql.host` . `GRANT ALL ON db_name.*` y `REVOKE ALL ON db_name.*` otorgan y quitan sólo permisos de bases de datos.

```
mysql> grant create,insert,select on XYZ2.* to xyz2@'localhost' identified by '
xyz2';
Query OK, 0 rows affected (0.02 sec)
```

- *Nivel de tabla*

  Los permisos de tabla se aplican a todas las columnas en una tabla dada. Estos permisos se almacenan en la tabla `mysql.tables_priv` . `GRANT ALL ON` *db_name.tbl_name* y `REVOKE ALL ON db_name.tbl_name` otorgan y quian permisos sólo de tabla.

- *Nivel de columna*

  Los permisos de columna se aplican a columnas en una tabla dada. Estos permisos se almacenan en la tabla `mysql.columns_priv` . Usando REVOKE, debe especificar las mismas columnas que se otorgaron los permisos.

- *Nivel de rutina*

  Los permisos `CREATE ROUTINE`, `ALTER ROUTINE`, `EXECUTE`, y `GRANT` se aplican a rutinas almacenadas. Pueden darse a niveles global y de base de datos. Además, excepto para `CREATE ROUTINE`, estos permisos pueden darse en nivel de rutinas para rutinas individuales y se almacenan en la tabla `mysql.procs_priv` .

Los permisos que se pueden otorgar o revocar son los siguientes:

Permiso	Significado
ALL [PRIVILEGES]	Da todos los permisos simples excepto GRANT OPTION
ALTER	Permite el uso de ALTER TABLE
ALTER ROUTINE	Modifica o borra rutinas almacenadas
CREATE	Permite el uso de CREATE TABLE
CREATE ROUTINE	Crea rutinas almacenadas
CREATE TEMPORARY TABLES	Permite el uso de CREATE TEMPORARY TABLE
CREATE USER	Permite el uso de CREATE USER, DROP USER, RENAME USER, y REVOKE ALL PRIVILEGES.
CREATE VIEW	Permite el uso de CREATE VIEW
DELETE	Permite el uso de DELETE
DROP	Permite el uso de DROP TABLE
EXECUTE	Permite al usuario ejecutar rutinas almacenadas
FILE	Permite el uso de SELECT ... INTO OUTFILE y LOAD DATA INFILE
INDEX	Permite el uso de CREATE INDEX y DROP INDEX
INSERT	Permite el uso de INSERT
LOCK TABLES	Permite el uso de LOCK TABLES en tablas para las que tenga el permiso SELECT
PROCESS	Permite el uso de SHOW FULL PROCESSLIST
REFERENCES	No implementado
RELOAD	Permite el uso de FLUSH
REPLICATION CLIENT	Permite al usuario preguntar dónde están los servidores maestro o esclavo
REPLICATION SLAVE	Necesario para los esclavos de replicación (para leer eventos del log binario desde el maestro)
SELECT	Permite el uso de SELECT
SHOW DATABASES	SHOW DATABASES muestra todas las bases de datos
SHOW VIEW	Permite el uso de SHOW CREATE VIEW
SHUTDOWN	Permite el uso de **mysqladmin shutdown**
SUPER	Permite el uso de comandos CHANGE MASTER, KILL, PURGE MASTER LOGS, and SET GLOBAL , el comando **mysqladmin debug** le permite conectar (una vez) incluso si se llega a max_connections
UPDATE	Permite el uso de UPDATE
USAGE	Sinónimo de "no privileges"
GRANT OPTION	Permite dar permisos

Use el comando SHOW GRANTS para determinar qué permisos tiene la cuenta

mysql> SHOW GRANTS FOR 'root'@'localhost';

```
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
```

Utilización de MySQL

Una vez instalado, podra ejecutar el cliente desde la línea de mandos, mediante el mando mysql. Al entrar de esta manera, será el usuario *anonimo* y no le pedirá ningún password. La Figura 4 muestra la interfaz que presenta MySQL al ingresar como el usuario anonimo.

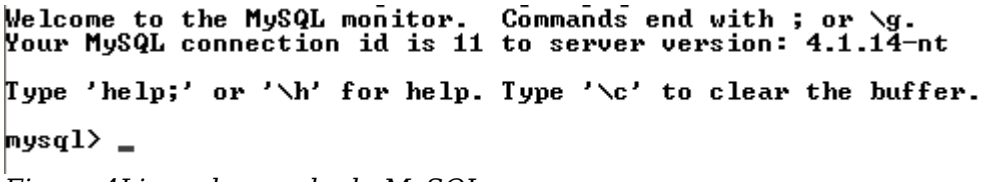


Figura 4Linea de mando de MySQL

Aqui se podrán dar mandos al manejador de base de datos, para trabajar con bases de datos. Por ejemplo, el mando show databases; nos mostrará las bases de datos que se tienen actualmente. Cuando recien se instala MySQL, son dos las bases de datos que se tienen que son *mysql* y *test*. Es importante conocer como se administra una base de datos a partir de la consola de trabajo ya que en ocaciones no se cuenta con una interfaz grafica para ello.

Creación de bases de datos

Durante el curso, se trabajará con MySQL que se encuentra en el servidor tiburon.mxl.uabc.mx por lo que será importante recordar en todo momento que todos los alumnos tendrán acceso a manejador. Se deberá tener cuidado en todo momento de no destruir la información que tengan los demás compañeros en la base de datos. En un ambiente de producción, esta no sería la situación y existiría un administrador que sería responsable de la base de datos.

Para crear una base de datos, ingresará al cliente de mysql como root y seguirá los siguientes pasos. (El instructor, le proporcionará la contraseña para root.)

1. Escriba en la línea de mandos mysql -u root -p y después la contraseña.
2. Creará una base de datos con nombre único que será db_XXXXX donde XXXXX serán los últimos 5 digitos de su matrícula. Escribir create database db_12345;
3. Verifique que se creo la base de datos con el mando: show databases;
4. Ahora usará la base de datos que acaba de crear: use db_12345;
5. Ahora agregará una tabla a la base de datos: create table clientes (ID int not null primary key auto_increment, Nombre varchar(60), Edad int);
6. Para verificar que si creo la tabla utilice: show tables;
7. Para agregar datos a la tabla utilice: insert into clientes (ID, Nombre, edad) values (NULL, 'Carlos Vega', 25);
8. Para verificar que se agregaron los datos utilice: select * from clientes; note que el campo ID tiene un valor que le fue asignado automaticamente.
9. Para borrar una base de datos utilice: drop database db_12345;

En los pasos anteriores se demostraron algunos mandos básicos de MySQL para trabajar con bases de datos. Durante el curso se profundizará mas en el uso de estos y ademas se estudiarán otros. Es importante tener presente que será imposible que se cubrán de manera exhaustiva todos los mandos disponibles bajo MySQL por lo que es de gran importancia darse a la tarea de buscar sitios en los que se tengan manuales de referencia a los que se pueda recurrir en caso de dudas. En la sección 7 de esta practica se encuentra una lista de algunos sitos que pueden ser de gran ayuda.

4. PROCEDIMIENTO (DESCRIPCIÓN)	
A) EQUIPO NECESARIO	**MATERIAL DE APOYO**

El maestro requerirá una computadora con Linux y Windows instalados así como conocer la contraseña de Root y del administrador en Windows para poder realizar la instalación de paquetes.

Para agilizar la demostración, se recomienda que en la partición de Windows ya se encuentre descargado el archivo: **`mysql-essential-4.1.14-win32.msi`**

Para la demostración bajo Linux, se requerirán los discos de instalación de la versión de Linux que tenga la máquina.

B) DESARROLLO DE LA PRÁCTICA

1. Consultar el sitio de MySQL para determinar el tipo de licencias que manejan. En que casos puede utilizarse sin costo? Cúando debe pagarse por utilizar el manejador?
2. Ubique el manual de referencia de MySQL en el sitio para desarrolladores.
3. Ingrese a la línea de mandos de MySQL como usuario anonimo y trate de crear una base de datos. Fue posible? Explique los resultados.
4. Ingrese a la línea de mandos de MySQL como root.
5. Creará una base de datos con nombre único que será db_XXXXX donde XXXXX serán los últimos 5 digitos de su matrícula.
6. Verifique que se haya creado.
7. Crear 2 usuarios (AXXXXX,BXXXXX) y asignarles password.
8. Otorgarle todos los derechos a los usuarios y entrar a mysql con las cuentas creadas.
9. Revocar todos los derechos a los usuarios.
10. Al usuario BXXXXX permitir insertar, seleccionar y borrar pero no permitir crear tablas en la base de datos, al usuario AXXXXX no permitir insertar datos pero permitirle crear tablas, seleccionar y borrar. Estos derechos seran sobre la base de datos que acaba de crear (db_XXXXX).

11. Intente agregar las dos tablas que se muestran en Figura 5 y Figura 6 a su base de datos con ambos usuarios que acaba de crear y tome nota de los mensajes. Haga que el campo *student_id* sea un campo cuyo valor se incremente automáticamente.

Field	Type
name	varchar(50)
sex	char(1)
student_id	int(10) unsigned

Figura 5: Tabla student.

Field	Type
student_id	int(10) unsigned
event_id	int(10) unsigned
score	int(11)

Figura 6: Tabla score.

12. Verifique que se hayan agregado las tablas.
13. Intente insertar datos en cada tabla con ambos usuarios y tome nota de los mensajes.
14. Intente insertar datos erroneos y tome nota de los mensajes de error que envía MySQL.
15. Que sucede si envía un valor determinado para el campo *student_id*?
16. Verifique que se hayan insertado los datos.
17. Crear otra base de datos con nombre único que será db2_XXXXX donde XXXXX serán los últimos 5 digitos de su matrícula.
18. Verifique que se haya creado.
19. Borrar la base de datos db2_XXXXX.
20. Verifique la estructura de la base de datos mysql. Cuántas tablas tiene? que campos tienen las tablas?
21. Busque algunos otros sitios en los que se tenga información sobre MySQL y la forma de acceder a una base de datos desde Java.

22. Busque algún sitio en el que indique como se cambia la contraseña de root en MySQL. **NO CAMBIE LA CONTRASEÑA DE MYSQL EN tiburon.mxl.uabc.mx**.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

Sitio principal de MySQL http://www.mysql.com/

Sitio para desarrolladores de MySQL http://dev.mysql.com/

Sitio de descarga MySQL http://dev.mysql.com/downloads/mysql/4.1.html

Información sobre el empaquetado RPM http://www.rpm.org/

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
5	NOMBRE DE LA PRÁCTICA	Administración básica de base de datos en Mysql	2

1 INTRODUCCIÓN

Después de conectarnos de manera satisfactoria al servidor de Mysql, el siguiente paso es crear la base de datos que contendrá la o las tablas compuestas por una o mas columnas. En MySql se pueden realizar diferentes operaciones con tablas y columnas así como con la misma base de datos.

2 OBJETIVO (COMPETENCIA)

Practicar las diferentes sentencias que son utilizadas en la creación y manipulación de una base de datos en MySql, mediante la realización de una serie de ejercicios en el laboratorio. Además se efectuarán también diferentes operaciones con tablas y columnas.

Formuló	Revisó	Aprobó	Autorizó
	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

Creando y usando una base de datos

Supongamos que tenemos diversas mascotas en casa (nuestro pequeño zoológico) y deseamos tener registros de los datos acerca de ellas. Podemos hacer esto al crear tablas que guarden esta información, para que posteriormente la consulta de estos datos sea bastante fácil y de manera muy práctica. Esta sección muestra como crear una base de datos y una tabla.

La base de datos "zoológico" será muy simple (deliveradamente), pero no es difícil pensar en situaciones del mundo real, en las cuales una base de datos similar puede ser usada.

Primeramente usaremos la sentencia SHOW para ver cuáles son las bases de datos existentes en el servidor al que estamos conectados:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

Es probable que la lista de bases de datos que veamos sea diferente en nuestro caso, pero seguramente las bases de datos "mysql" y "test" estarán entre ellas. En particular, la base de datos "mysql" es requerida, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos "test" es creada durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Se debe anotar también que es posible que no veamos todas las bases de datos si no se tiene el privilegio SHOW DATABASES. Si este es tu caso comentaselo a tu tutor de laboratorio.

Si la base de datos "test" existe, hay que intentar accesar a ella:

```
mysql> USE test
Database changed
mysql>
```

Observar que USE, al igual que QUIT, no requieren el uso del punto y coma, aunque si se usa éste, no hay ningún problema. El comando USE es especial también de otra manera: éste debe ser usado en una sóla línea.

3 FUNDAMENTO

Podríamos usar la base de datos "test" (si tenemos acceso a ella) para los ejemplos que vienen a continuación, pero cualquier cosa que hagamos puede ser eliminada por cualquier otro usuario que tenga acceso a esta base de datos. Por esta razón, es recomendable que preguntemos al administrador MySQL acerca de la base de datos que podemos usar. Supongamos que deseamos tener una base de datos llamada "zoologico" (nótese que no se está acentuando la palabra) a la cual sólo nosotros tengamos acceso, para ello el administrador necesita ejecutar un comando como el siguiente:

```
mysql> GRANT ALL on zoologico.* TO MiNombreUsuario@MiComputadora
    -> IDENTIFIED BY 'MiContraseña';
```

en donde MiNombreUsuario es el nombre de usuario asignado dentro del contexto de MySQL, MiComputadora es el nombre o la dirección IP de la computadora desde la que nos conectamos al servidor MySQL, y MiContraseña es la contraseña que se nos ha asignado, igualmente, dentro del ambiente de MySQL exclusivamente. Ambos, nombre de usuario y contraseña no tienen nada que ver con el nombre de usuario y contraseña manejados por el sistema operativo (si es el caso).

Si el administrador creó la base de datos al momento de asignar los permisos, podemos hacer uso de ella. De otro modo, nosotros debemos crearla:

```
mysql> USE zoologico
ERROR 1049: Unknown database 'zoologico'
mysql>
```

El mensaje anterior indica que la base de datos no ha sido creada, por lo tanto necesitamos crearla.

```
mysql> CREATE DATABASE zoologico;
Query OK, 1 row affected (0.00 sec)

mysql> USE zoologico
Database changed
mysql>
```

Bajo el sistema operativo Unix, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto debemos de tener cuidado de escribir correctamente el nombre de la base de datos. Esto es cierto también para los nombres de las tablas.

Al crear una base de datos no se selecciona ésta de manera autómatica; debemos hacerlo de manera explícita, por ello usamos el comando USE en el ejemplo anterior.

3 FUNDAMENTO

Borrando una base de datos

Para borrar la base de datos utilizamos la sentencia

```
mysql>Drop database  nombre_de_la_base_de_datos;
```

Creando y visualizando tablas

Crear la base de datos es la parte más fácil, pero en este momento la base de datos está vacía, como lo indica el comando SHOW TABLES FROM:

```
mysql> SHOW TABLES FROM nombre_de_la_base_de_datos;
Empty set (0.00 sec)
```

La parte un tanto complicada es decidir la estructura que debe tener nuestra base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla.En principio, necesitamos una tabla que contenga un registro para cada una de nuestras mascotas. Ésta puede ser una tabla llamada mascotas, y debe contener por lo menos el nombre de cada uno de nuestros animalitos. Ya que el nombre en sí no es muy interesante, la tabla debe contener alguna otra información. Por ejemplo, si más de una persona en nuestra familia tiene una mascota, es probable que tengamos que guardar la información acerca de quien es el dueño de cada mascota. Así mismo, también sería interesante contar con alguna información más descriptiva tal como la especie, y el sexo de cada mascota.

¿Y que sucede con la edad?. Esto puede ser también de interés, pero no es una buena idea almacenar este dato en la base de datos. La edad cambia conforme pasa el tiempo, lo cual significa que debemos de actualizar los registros frecuentemente. En vez de esto, es una mejor idea guardar un valor fijo, tal como la fecha de nacimiento. Entonces, cuando necesitemos la edad, la podemos calcular como la diferencia entre la fecha actual y la fecha de nacimiento. MySQL proporciona funciones para hacer operaciones entre fechas, así que no hay ningún problema por que dichas funciones se estudiarán en la práctica 9 .

Al almacenar la fecha de nacimiento en lugar de la edad tenemos algunas otras ventajas:

Podemos usar la base de datos para tareas tales como generar recordatorios para cada cumpleaños próximo de nuestras mascotas. Podemos calcular la edad en relación a otras fechas que la fecha actual. Por ejemplo, si almacenamos la fecha en que murió nuestra mascota en la base de datos, es fácil calcular que edad tenía nuestro animalito cuando falleció. Es probable que estemos pensando en otro tipo de información que sería igualmente útil en la tabla "mascotas", pero para nosotros será suficiente por ahora contar con información de nombre, propietario, especie, nacimiento y fallecimiento.

Usaremos la sentencia CREATE TABLE para indicar como estarán conformados los registros de nuestras mascotas.

```
mysql> CREATE TABLE mascotas(
    -> nombre VARCHAR(20), propietario VARCHAR(20),
    -> especie VARCHAR(20), sexo CHAR(1), nacimiento DATE,
    -> fallecimento DATE);
Query OK, 0 rows affected (0.02 sec)
mysql>
```

3 FUNDAMENTO

Para verificar que la tabla fué creada como nosotros esperabamos, usaremos la sentencia DESCRIBE:

```
mysql> DESCRIBE mascotas;
+-------------+-------------+-----------+-------+-----------+---------+
| Field       |   Type      |   Null    |  Key  | Default |  Extra  |
+-------------+-------------+-----------+-------+-----------+---------+
| nombre      | varchar(20) | YES   |         | NULL    |         |
| propietario | varchar(20) | YES   |         | NULL    |         |
| especie     | varchar(20) | YES   |         | NULL    |         |
| sexo        | char(1)     | YES   |         | NULL    |         |
| nacimiento  | date        | YES   |         | NULL    |         |
| fallecimento| date        | YES   |         | NULL    |         |
+-------------+-------------+------ ---+--------+-----------+-------+
6 rows in set (0.01 sec)

mysql>
```

Podemos hacer uso de la sentencia DESCRIBE en cualquier momento, por ejempo, si olvidamos los nombres ó el tipo de las columnas en la tabla.

Alterando tablas

MySQL nos provee del comando ALTER TABLE para realizar modificaciones a la estructura de una tabla.Con este comando nosotros podemos:

- add (agregar) o bien delete (borrar) columnas "ADD y DROP"
- change (cambiar) la definición de tablas existentes "ALTER, CHANGE, MODIFY".
- rename (renombrar) columnas "CHANGE" o aun en si "RENAME AS".

Para agregar una columna a una tabla utilizamos la sentencia

ALTER TABLE nombre_de_la_tabla ADD nombre_de_la_columna tipo;

Ejemplo 1:

En el sig. ejemplo se agrega a la tabla Tablita un index en la columna d y la columna a se agrega como llave primaria.

```
mysql>AlLTER TABLE Tablita ADD index(d), ADD PRIMARY KEY (a);
```

Ejemplo 2:

En el sig. ejemplo se agrega una columna llamada C de tipo entero que se autoincrementa

```
mysql>AlLTER TABLE Tablita ADD  C  INT UNSIGNED NOT NULL AUTO_INCREMENT, ADD INDEX(c);
```

Ejemplo 3:

En el sig. ejemplo agregarmos en la tabla user un campo llamado yahoo de tipo varchar con longitud de 50

```
mysql>ALTER TABLE user ADD yahoo VARCHAR(50) NOT NULL;
```

Se puede utilizar la palabra AFTER para especificar la posición en donde se desea colocar el nuevo campo

3 FUNDAMENTO

Ejemplo 4:
En el sig ejemplo para la tabla user agregamos el campo yahoo exactamente después del campo userid.

mysql>ALTER TABLE user ADD yahoo VARCHAR(50) NOT NULL AFTER userid;

Para cambiar el tipo de dato de una columna utilizamos la sentencia
ALTER TABLE nombre_de_la_tabla MODIFY nombre_de_la_columna nuevo_tipo;
Para cambiar una columna de nombre utilizamos la sentencia
ALTER TABLE nombre_de_la_tabla CHANGE nombre_actual_de_la_columna nombre_nuevo tipo:
Parra borrar una columna de una tabla utilizamos la sentencia
ALTER TABLE nombre_de_la_tabla DROP nombre_de_la_columna;
Para renombrar una tabla utilizamos la sentencia
ALTER TABLE nombre_de_la_tabla RENAME nombre_nuevo;
o bien
RENAME Table nombre_de_la_tabla TO nuevo_nombre;
Para borrar una tabla utilizamos la siguiente sentencia
DROP TABLE nombre_de_la_tabla;

4 PROCEDIMIENTO (DESCRIPCIÓN)

A	EQUIPO NECESARIO	MATERIAL DE APOYO

Formato para prácticas de laboratorio

4 PROCEDIMIENTO (DESCRIPCIÓN)

Computadora con acceso al servidor Mysql	Práctica impresa

B DESARROLLO DE LA PRÁCTICA

1.-Introduce tu login y password para acceder al servidor de MySQL.
2.-Crear una base de datos llamada ZooXXXX donde XXXX es el número de tu matrícula.
3.-Nuestra base de datos contendrá las sig. Tablas

Tabla Animales	Tabla Zoologicos	Tabla Veterinarios
Id_Animal (llave)	Id_Zoologico(llave)	Id_Veterinario
Especie	Id_Animal	Id_Zoologico
Hábitat	Nombre_Zoo	Nombre_Vete
Nombre	Tel_Zoo	Cedula
FechaNaci		Dire_Vete
Paísdeorigen		Tel_Vete
Id_Veterinario		

4.- Lista las BD existentes
5.- Ahora Lista las tablas que contiene la BD ZooXXXX
6.- Muestre en pantalla las columnas que contiene la Tabla Animales
7.- Liste las columnas que contiene Zoologicos y por último las de Veterinarios
8.- Sustituya el nombre de la tabla animales por el de Mascotas
9.-Para la tabla Mascotas, cambie el nombre de la columna Paisdeorigen por Origen
10.-Agregar una columna nueva a la tabla Mascotas y pongale como nombre Alimentacion
11.-Liste las columnas de la Tabla Mascotas
12.-Borre la Tabla Veterinarios
13.-Liste las tablas existentes en la BD ZooXXXX
14.-Cambie la longitud del tipo de dato de la columna Nombre de la tabla Mascotas. Es decir aumentelo tres caracteres
15.-Borre la columna Origen de la tabla Mascotas
16.-Usted se ha equivocado y realmente no deseaba borrar la columna Origen, por lo tanto agregela de nuevo
17.-Para la tabla Mascotas, coloque la columna Id_Veterinario como índice
18.-Agregar en la tabla Zoologicos la columna Pais justo después de la columna Nombre_Zoo
19.- Utilizando las sentencias necesarias de Mysql, elabore una BD que tenga mínimo 3 tablas y cada tabla tendrá como mínimo 3 columnas.
20.-Liste las bases de datos existemtes
21.- Muestre en pantalla las tablas de la BD creada en el punto 19.
22.- Borre la base de datos creada en el punto 19
23.- Liste las bases de datos existentes
24.- Fin

C	CÁLCULOS Y REPORTE

5 RESULTADOS Y CONCLUSIONES

Al finalizar la práctica el alumno será capaz de manejar las principales ordenes que se utilizan en MySql para la manipulacion de tablas y columnas en una base de datos sencilla.

6 ANEXOS

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	BASES DE DATOS

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
6	**NOMBRE DE LA PRÁCTICA**	**Consulta de INSERCIÓN Y SELECCION**	2

1. INTRODUCCIÓN

Dada la importancia y el auge de manejo de grandes cantidades de información. Las bases de datos vienen a dar una solución rápida al manejo de la misma información en grandes volúmenes. El lenguaje SQL se puede considerar como una de las razones más importantes del éxito de las bases de datos relacionales en el mundo para el manejo de información haciendo uso de varias sentencias como INSERT y SELECT.

2. OBJETIVO (COMPETENCIA)

El alumno conocerá y utilizara el lenguaje SQL en la aplicación real de una base de datos. Donde aplicará las consultas básicas de SQL de INSERT Y SELECT.

3. FUNDAMENTO

Insert

En su forma mas simple insert sirve para añadir un renglón(tupla) en una tabla. Debemos especificar el nombre de la relación y una lista de valores para el renglón. Los valores deben insertarse en el mismo orden en que se especificaron los atributos en la tabla.

La forma general es:
mysql> INSERT INTO tbl_name (col1,col2) VALUES(15,2);

mysql> INSERT INTO tbl_name () VALUES();

Ejemplo.
mysql> Insert into empleado values
('Richard', 'k','Marin','653298653','1962-12-30','98 Oaks Forest, Katy TX', 37000,'987',4);

mysql> Insert into empleado (nombre,apellido,nss,nd)
values ('Richard', 'Marin','653298653',4);

mysql> INSERT INTO table (a,b,c) VALUES (1,2,3)
-> ON DUPLICATE KEY UPDATE c=c+1;

mysql> UPDATE table SET c=c+1 WHERE a=1 OR b=2 LIMIT 1;

mysql> INSERT INTO tbl_temp2 (fld_id)
SELECT tbl_temp1.fld_order_id
FROM tbl_temp1 WHERE tbl_temp1.fld_order_id > 100;

mysql> INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6)
-> ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);

SELECT

Una consulta SQL puede constar de un máximo de seis cláusulas, pero solo son obligatorias las dos primeras, SELECT y FROM. Las cláusulas se especifican en el siguiente orden y las que estén entre corchetes son opcionales:
SELECT <lista de atributos>
FROM <lista de tablas>
[WHERE <condición>]
[GROUP BY <atributos de agrupación>]
[HAVING <condicion de agrupación>]
[ORDER BY <lista de atributos>]
Ejemplos.
No especificación de la clausula WHERE
A) SELECT nss
FROM empleado.

B) SELECT *
FROM empleado.

C) SELECT *
FROM empleado, departamento

Uso de DISTINCT

D) SELECT **distinct** salario
FROM empleado;
Recuperar la fecha de nacimiento y dirección del empleado JUAN B Martinez usando la tabla empleado

E) SELECT Fecha_nac, direccion
FROM empleado
WHERE nombre=´JUAN´ and inc=´b´and apellido='Martinez'
Uso de los alias.

F) De cada empleado, recuperar su nombre de pila y apellido y los de sus supervisor inmediato.
SELECT e.nombre, e.apellido, s.nombre, s.apellido
FROM empleado as E. empleado as s
WHERE e.nss_superv=s.nss

De cada proyecto ubicado en Stanford, haga una lista con el numero de proyecto, el numero de departamento controlador y el apellido, dirección y fecha de nacimiento del jefe de departamento(es este caso se involucran 3 tablas).

G) SELECT numerop, numd, apellido, direccion, fecha_nac
FROM proyecto, departamento, empleado
WHERE numd=numerod and nss_jefe=nss and localizacion=´Satanfford';
Comparación de subcadenas, operadores aritméticos y ordenaciones, caundo no solo conosca una parte del texto a buscar.

H) SELECT nombre, apellido
FROM empleado;
WHERE direccion like "%houston Tx%"

Obtener una lista de empleados y de los proyectos en los que trabajan, ordenados por departamento y, dentro de cada departamento, **alfabéticamente** por apellido y nombre

I) SELECT nombred, apellido, nombre, nombrep
FROM departamento, empleado, trabaja_en, proyecto
WHERE numerod=nd and nss=nsse and np=numerop
ORDER BY nombred **desc**, apellido, nombre

Funciones agregadas predefinidas
count, sum, max, min, avg

Ejemplo
Obtener la suma de los salarios de todos los empleados, el salario máximo, el salario mínimo y el salario medio
J) SELECT **sum**(salario), **max**(salario), **min**(SALARIO), **avg**(SALARIO))
FROM empleado;

K) Obtener el número de empleados de la empresa
Select **count**(*)
From empleado;

Obtener el número de proyecto, nombre del proyecto, y contar el número de empleados que trabaja en cada uno de los proyectos

L) SELECT numerop, nombrep,count(*)
FROM proyecto, trabaja_en
WHERE numerop=np
GROUP BY numerop, nombrep;

4. PROCEDIMIENTO (DESCRIPCIÓN)	
A) EQUIPO NECESARIO	**MATERIAL DE APOYO**

B)	**DESARROLLO DE LA PRÁCTICA**

1.- Crear una base de datos llamada empresa_matricula con los siguientes atributos.

2.- Crear las siguientes tablas y atributos.

```
-- Estructura de la tabla `departamento`

CREATE TABLE departamento (
    nombred varchar(15) NOT NULL default '',
    numerod int(11) NOT NULL default '0',
    nss_jefe char(9) NOT NULL default '',
    fecha_inic_jefe_dept date default NULL,
    PRIMARY KEY  (nombred)
) TYPE=InnoDB;
```

```
-- Estructura de la tabla `dependientes`

CREATE TABLE dependientes (
    nsse char(9) NOT NULL default '0',
    nombre_dependiente varchar(30) NOT NULL default '',
    sexo enum('M','H') default NULL,
    fecha_nto date default NULL,
    parentesco enum('HIJO','HIJO','ESPOSA') default NULL,
    PRIMARY KEY  (nsse)
) TYPE=InnoDB;

-- Estructura de la tabla `empleado`

CREATE TABLE empleado (
    nombre varchar(20) NOT NULL default '',
    inic char(2) default NULL,
    apellido varchar(20) NOT NULL default '',
    nss char(9) NOT NULL default '0',
    fecha_ncto date default NULL,
    direccion varchar(30) default NULL,
    sexo enum('H','M') default NULL,
    salario float(6,2) default NULL,
    nss_superv  char(9) default NULL,
    nd tinyint(4) default NULL,
    PRIMARY KEY  (nss)
) TYPE=InnoDB;

-- Estructura de la tabla `localizaciones_dept`

CREATE TABLE localizaciones_dept (
    numerod tinyint(4) NOT NULL default '0',
    localizacion varchar(25) NOT NULL default '',
    PRIMARY KEY  (numerod,localizacion)
) TYPE=InnoDB;

-- Estructura de la tabla `proyecto`

CREATE TABLE proyecto (
    nombrep varchar(30) NOT NULL default '',
    numerop tinyint(4) NOT NULL default '0',
    localizacion varchar(25) default NULL,
    numd tinyint(4) NOT NULL default '0',
    PRIMARY KEY  (numerop)
) TYPE=InnoDB;

-- Estructura de la tabla `trabaja_en`
CREATE TABLE trabaja_en (
    nsse char(9) NOT NULL default '0',
    np tinyint(4) NOT NULL default '0',
```

```
    horas decimal(3,1) NOT NULL default '0.0',
    PRIMARY KEY  (nsse)
) TYPE=InnoDB;
```

3.- Insertar el siguiente conjunto de datos en la siguiente tabla empelado.

NOMBRE	INIC	APELLIDO	NSS	FECHA_NCTO	DIRECION	SEXO	SALARIO	NSS_SUPERV	ND
JUAN	B	Martinez	123	1965-01-09	731 Fondren, Houston, TX	H	300	333	5
Francisco	T	Wong	333	1955-12-08	638 Voss, Houston, TX	H	400	888	5
Alicia	J	Zelaya	999	1968-07-19	3321 Castle, Spring, TX	M	250	987	4
Jennifer	S	Wallace	987	1941-06-20	291 Berry, Bellaire, TX	M	430	888	4
Ramesh	K	Narayan	666	1962-09-15	975 Fire Oak, Humble, TX	H	380	333	5
Joyce	A	English	453	1972-07-31	5631 Rice, Houston, TX	M	250	333	5
Aime	V	Jabbar	879	1969-03-29	980 Dallas, Houston, TX	H	250	987	4
Jaime	E	Borg	888	1937-11-10	450 Stone, Houston, TX	H	550	nulo	1

4.- Insertar el siguiente conjunto de datos en la siguiente tabla:

LOCALIZACIONES_DEPT

NUMEROD	LOCALIZACIOND
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

5.- Insertar el siguiente conjunto de datos en la siguiente tabla:

DEPARTAMENTO

NOMBRED	NUMEROD	NSS_JEFE	FECHA_INIC_JEFE
Investigacion	5	333	1988-05-22
Administracion	4	987	1995-01-01
Direccion	1	888	1981-06-19

6.- Insertar el siguiente conjunto de datos en la siguiente tablaÑ

TRABAJA_EN

NSSE	***NP***	***HORAS***
123	1	32,5
123	2	7,5
666	3	40,0
453	1	20,0
453	2	20,0
333	2	10,0
333	3	10,0
333	10	10,0
333	20	10,0
999	30	30,0
999	10	10,0
879	10	35,0
879	30	5,0
987	30	20,0
987	20	15,0
888	20	nulo

7.- Insertar el siguiente conjunto de datos en la siguiente tabla

PROYECTO

NOMBREP	NUMEROP	LOCALIZACIONP	ND
ProductoX	1	Bellaire	5
ProductoY	2	Sugarland	5
ProductoZ	3	Houston	5
Automatizacion	10	Stafford	4
Reorganizacion	20	Houston	1
Nuevos beneficios	30	Stafford	4

8.- Insertar el siguiente conjunto de datos en la siguiente tabla

DEPENDIENTE

NSSE	***NOMBRE_DEPENDIENTE***	***SEXO***	***FECHA_NCTO***	***PARENTESCO***
333	Alice	M	1986-04-05	HIJA
333	Theodore	H	1983-10-25	HIJO
333	Joy	M	1958-05-03	ESPOSA
987	Abner	H	1942-02-28	ESPOSA
123	Michael	H	1988-01-04	HIJO
123	Alice	M	1988-12-30	HIJA
123	Elizabeth	M	1967-05-05	ESPOSA

9.- Realizar los siguientes los ejemplos de la práctica utilizando SELECT del inciso A al K

10.- Realizar los ejercicios adicionales que crea conveniente el docente.

C)	CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
7	NOMBRE DE LA PRÁCTICA	Modificación y Eliminación de información en MySQL	2

1 INTRODUCCIÓN

Al inicio o arranque de un sistema de información construido alrederor de una base de datos, es muy común que la primera vez que se almacenan datos contenga mucha información erronea y por lo tanto ocupe de mecanismos o instrucciones que permitan corregir, modificar, editar o eliminar esta información.

La modificación y eliminación son dos de los procesos más comunes que se realizan con tablas en bases de datos, se le conoce como actualizacion, edicion o modificacion de los datos o registros o renglones contenidos en una tabla.

Las instrucciones UPDATE y DELETE son las instrucciones en SQL especializadas en esta area de procesos comunes con tablas, y son estas instrucciones las que se desarrollarán en esta práctica.

2 OBJETIVO (COMPETENCIA)

El alumno aplicará el Lenguaje de consulta estructurado (SQL) para realizar operaciones de modificación (UPDATE) y eliminación (DELETE) de información dentro de las tablas que componen un base de datos.

Formuló	Revisó	Aprobó	Autorizó
Ing. Pablo M. Navarro Álvarez	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

Modificación de Información dentro de una tabla (UPDATE)

Hasta ahora ya conocemos la forma de introducir datos en las tablas que componen nuestra base de datos, pero que pasa cuando echamos un vistazo, y nos damos cuenta de que hemos cometido algún error en alguno de los datos, ¿qué hacemos para arreglar este tipo de problemas?.

La instrucción UPDATE, nos permite realizar cambios en los datos que ya tenemos dentro de una tabla de nuestra base de datos. La sintáxis de esta orden es:

UPDATE nombretabla **SET** nomcolumna=expresion **WHERE** condicion ;

La instruccion UPDATE actualiza o modifica los renglones de una tabla, SET le indica a MYSQL cuales son las columnas a modificar y WHERE se usa para seleccionar un renglon determinado o un conjunto de renglones, los casos mas comunes son:

1.- Actualizar una columna o varias columnas a todos los renglones de la tabla. En el siguiente ejemplo se pone el valor de 150.00 en la columna preciopelicula de la tabla peliculas.

UPDATE peliculas SET preciopelicula = 150.00;

A continuación se muestra la pantalla que aparecera al ejecutarse la instrucción SQL anterior:

```
mysql> use video;
Database changed
mysql> update peliculas set preciopelicula=150.00;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 7  Changed: 0  Warnings: 0

mysql> select * from peliculas;
+---------------+---------------------+---------------+-----------------+
| clavepelicula | nombrepelicula      | tipopelicula  | preciopelicula  |
+---------------+---------------------+---------------+-----------------+
|            21 | terminator          | accion        |             150 |
|            22 | becool              | comedia       |             150 |
|            23 | the pacifier        | ninos         |             150 |
|            24 | la masacre de texas | accion        |             150 |
|            25 | birth               | horror        |             150 |
|            26 | star wars III       | accion        |             150 |
|            27 | legalmente rubia    | comedia       |             150 |
+---------------+---------------------+---------------+-----------------+
7 rows in set (0.44 sec)
```

2.- También se pueden utilizar expresiones algebraicas. En el siguiente ejemplo se multiplican los valores 19.99 por 11.28 y el resultado es puesto en la columna preciopelicula de la tabla peliculas.

UPDATE peliculas SET preciopelicula = 19.99 * 11.28 ;

3 FUNDAMENTO

A continuación se muestra la pantalla que aparecera al ejecutarse la instrucción SQL anterior:

```
mysql>
mysql> update peliculas set preciopelicula=19.99 * 11.28;
Query OK, 7 rows affected (0.02 sec)
Rows matched: 7  Changed: 7  Warnings: 0

mysql> select * from peliculas;
+---------------+---------------------+--------------+-----------------+
| clavepelicula | nombrepelicula      | tipopelicula | preciopelicula  |
+---------------+---------------------+--------------+-----------------+
|            21 | terminator          | accion       |         225.487 |
|            22 | becool              | comedia      |         225.487 |
|            23 | the pacifier        | ninos        |         225.487 |
|            24 | la masacre de texas | accion       |         225.487 |
|            25 | birth               | horror       |         225.487 |
|            26 | star wars III       | accion       |         225.487 |
|            27 | legalmente rubia    | comedia      |         225.487 |
+---------------+---------------------+--------------+-----------------+
7 rows in set (0.00 sec)

mysql>
```

3.- El caso más común es solo actualizar un renglón o registro, para esta situación solo usar la claúsula WHERE. En el siguiente ejemplo se pone el valor de 500 a la columna preciopelicula de la tabla peliculas pero solo en aquel renglón en el que el valor de la columna nombrepelicula sea igual a “becool”.

Update peliculas set preciopelicula = 500 where nombrepelicula = 'becool';

A continuación se muestra la pantalla que aparecera al ejecutarse la instrucción SQL anterior:

```
mysql> update peliculas set preciopelicula=500
    -> where nombrepelicula='becool';
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from peliculas;
+---------------+---------------------+--------------+-----------------+
| clavepelicula | nombrepelicula      | tipopelicula | preciopelicula  |
+---------------+---------------------+--------------+-----------------+
|            21 | terminator          | accion       |         225.487 |
|            22 | becool              | comedia      |             500 |
|            23 | the pacifier        | ninos        |         225.487 |
|            24 | la masacre de texas | accion       |         225.487 |
|            25 | birth               | horror       |         225.487 |
|            26 | star wars III       | accion       |         225.487 |
|            27 | legalmente rubia    | comedia      |         225.487 |
+---------------+---------------------+--------------+-----------------+
```

4.- SQL UPDATE en MYSQL también puede actualizar un subconjunto de renglones de la tabla. En el siguiente ejemplo se pone el valor de 333 a la columna preciopelicula de la tabla peliculas pero solo en aquel o aquellos renglones en los que el valor de la columna tipopelicula sea igual a 'accion'.

Update peliculas set preciopelicula = 333 where tipopelicula = 'accion';

A continuación se muestra la pantalla que aparecera al ejecutarse la instrucción SQL anterior:

3 FUNDAMENTO

```
mysql> update peliculas set preciopelicula=333
    -> where tipopelicula='accion';
Query OK, 3 rows affected (0.45 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from peliculas;
+---------------+----------------------+---------------+-----------------+
| clavepelicula | nombrepelicula       | tipopelicula  | preciopelicula  |
+---------------+----------------------+---------------+-----------------+
|            21 | terminator           | accion        |             333 |
|            22 | becool               | comedia       |             500 |
|            23 | the pacifier         | ninos         |         225.487 |
|            24 | la masacre de texas  | accion        |             333 |
|            25 | birth                | horror        |         225.487 |
|            26 | star wars III        | accion        |             333 |
|            27 | legalmente rubia     | comedia       |         225.487 |
+---------------+----------------------+---------------+-----------------+
7 rows in set (0.00 sec)
```

Ejemplos adicionales con la instrucción UPDATE

Supongamos que tenemos una base de datos la cual tiene una tabla llamada **refranero** y sobre ella se realisan las siguientes instrucciones SQL:

UPDATE refranero SET fecha="2003-06-01" WHERE ID=1;

Lo que aparecerá en la pantalla es algo similar a la siguiente información para mostrar el cambio hecho sobre la tabla refranero.

```
+---------+-----------------------------------------------------------------+-----------------+
| ID | refran | fecha |
+---------+-----------------------------------------------------------------+-----------------+
| 1 | Más vale párajo en mano que ciento volando | 2003-06-01 |
+---------+-----------------------------------------------------------------+-----------------+
  1 row in set (0.00 sec)
```

Supongamos ahora que queremos cambiar párajo por pájaro y para ello utilizamos el LIKE como hacíamos antes para ver los registros, así que le decimos a mysql que debe cambiar la columna refran de la tabla refranero.

UPDATE refranero SET refran="Más vale pájaro en mano que ciento volando" WHERE refran LIKE "%párajo%";

Aparecerá en la pantalla algo parecido a lo siguiente, para mostrar el cambio hecho sobre la tabla refranero.

```
+---------+-----------------------------------------------------------------+-----------------+
| ID | refran | fecha |
+---------+-----------------------------------------------------------------+-----------------+
| 1 | Más vale pájaro en mano que ciento volando | 2003-06-01 |
+---------+-----------------------------------------------------------------+-----------------+
1 row in set (0.00 sec)
```

3 FUNDAMENTO

Eliminación de Información dentro de una tabla (DELETE)

Una de las operaciones de proceso o manipulación de tablas en MySQL es DELETE que permite borrar o eliminar algún registro o renglón de la tabla, o un subconjunto de renglones de la tabla o si es necesario eliminar todos los renglones de la tabla. La sintáxis de esta orden es:

DELETE FROM nombretabla WHERE condicion;

La instruccion DELETE borra o elimina los renglones de una tabla, FROM le indica a MYSQL la tabla sobre la cual se hará la eliminación, y WHERE cual o cuales son los renglones que se desean eliminar, los casos más comunes son:

1.- EL caso más simple es eliminar un renglón cualquiera. En el siguiente ejemplo, se borra el registro cuya columna clavepelicula sea igual a 27 de la tabla peliculas

DELETE from peliculas where clavepelicula = 27;

Lo que aparecerá en la pantalla después de realizar una instrucción SELECT para verificar que se borraron los registros con clavepelicula=27, es algo similar a la siguiente información donde se muestra el cambio hecho sobre la tabla peliculas.

```
Query OK, 1 row affected (0.44 sec)

mysql> select * from peliculas;
+---------------+---------------------+--------------+-----------------+
| clavepelicula | nombrepelicula      | tipopelicula | preciopelicula  |
+---------------+---------------------+--------------+-----------------+
|            21 | terminator          | accion       |             333 |
|            22 | becool              | comedia      |             500 |
|            23 | the pacifier        | ninos        |         225.487 |
|            24 | la masacre de texas | accion       |             333 |
|            25 | birth               | horror       |         225.487 |
|            26 | star wars III       | accion       |             333 |
+---------------+---------------------+--------------+-----------------+
6 rows in set (0.00 sec)
```

2.- También puden eliminarse un subconjunto de renglones usando la clausula WHERE con un filtro o condición de manera apropiada y bien construido. En el siguiente ejemplo se borra el registro o los registros de la tabla peliculas en los cuales su columna preciopelicula sea menor o igual a 300.

Delete from peliculas where preciopelicula <= 300;

3 FUNDAMENTO

Lo que aparecerá en la pantalla después de realizar la instrucción anterior y ejecutar una instrucción SELECT para verificar que se borraron los registros con preciopelicula <= 300, es algo similar a la siguiente información donde se muestra el cambio hecho sobre la tabla peliculas.

```
mysql> delete from peliculas where preciopelicula<=300;
Query OK, 2 rows affected (0.00 sec)

mysql> select * from peliculas;
+---------------+---------------------+---------------+-----------------+
| clavepelicula | nombrepelicula      | tipopelicula  | preciopelicula  |
+---------------+---------------------+---------------+-----------------+
|            21 | terminator          | accion        |             333 |
|            22 | becool              | comedia       |             500 |
|            24 | la masacre de texas | accion        |             333 |
|            26 | star wars III       | accion        |             333 |
+---------------+---------------------+---------------+-----------------+
4 rows in set (0.00 sec)

mysql>
```

3.- Se puede eliminar todos los registro o renglones de una tabla, como se muestra en el siguiente ejemplo.

DELETE FROM PELICULAS;

No es recomdable utilizar la instrucción anterior porque se eliminan todos los renglones, aunque si la prueban y necesitan recargar de datos la tabla pueden hacerlo como lo vieron en las practicas anteriores.

Ejemplos adicionales con la instrucción DELETE

Supongamos que seguimos trabajando con la base de datos que tiene la tabla llamada **refranero** y sobre ella se realisan las siguientes instrucciones SQL:

DELETE FROM refranero where id =1;

Ó si queremos borrar todos los refranes que contengan la palabra "amor", entonces hacemos la siguiente instrucción SQL.

DELETE FROM refranero WHERE LIKE "%amor%";

Y para borrar todos los refranes de nuestra tabla?

DELETE FROM refranero;

Un consejo, cuando se quiera borrar registros de una tabla, puedes probar primero con un select si el select, funciona correctamente y te selecciona exactamente los registros que quieras borrar; entonces cambias SELECT por DELETE y listo.

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con Linux y MySQL instalado.	Práctica impresa.

Formato para prácticas de laboratorio

4 PROCEDIMIENTO (DESCRIPCIÓN)

B DESARROLLO DE LA PRÁCTICA

1.-Introduce tu login y password para acceder al servidor de MySQL.
2.-Borra las bases de datos utilizadas en la práctica anterior si es que ya fueron revisadas por tu tutor.
3.-Crear una Base de Datos, para resolver el ejercicio que se propone a continuación.

La Dirección General de Casas y Gentes desea digitalizar la elaboración del padrón de Habitantes de los municipios en el estado de Baja California.

Cada habitante registrado en el estado sólo puede habitar en una única casa y estar empadronado en único municipio del estado.

Cualquier persona puede ser propietaria de varias viviendas.

Taqmbien interesa registrar los datos de las personas que dependen del Cabeza de Familia en cada casa.

4.-Utilizando las intrucciones que conoces de MySQL, realiza el llenado de las tablas que creaste en el paso anterior, por lo menos deberán ser 3 tablas (PERSONA, CASA, MUNICIPIO), La tabla PERSONA deberá por lo menos tener 7 personas con todos sus datos, la tabla Casa deberá por lo menos tener información de 10 casas y la tabla MUNICIPIO deberá tener información de 4 municipios.

5.-Modifica el nombre de una persona dentro de la tabla PERSONA y pon el nombre “Alicia del Refugio Lopez Aguirre”
6.-Modifica el campo municipio y pon el valor de 50, a todos aquellos registros que tengan como municipio el valor de 1.
7.-Modifica el campo costo de la tabla de viviendas e incremeta el 10% a todos los valores actuales en esa columna.
8.-Modifica el campo costo de la tabla de viviendas y decrementa 1000.00 a todas aquellas viviendas que tengas un costo mayor a 50000.00 y que pertenezcan al municipio 50.
9.-Invierte todos los cambios hechos a partir del paso 5 y deja las tablas como estaban inicialmente.
10.-Elimina todas las personas que pertenecen al municipio numero 1
11.-Elimina todas las casas que pertenecen al municipio 2 y que tienen un costo menor a los 20000.00
12.-Elimina todos los registros de la tabla PERSONA.
13.-Elimina todos los registros de la tabla VIVIENDA.
14.-Elimina todos los registros de la tabla MUNICIPIO.
15.-Borra la tabla PERSONA.
16.-Borra la tabla VIVIENDA.
17.-Borra la tabla MUNICIPIO.
15.-Borra la Base de Datos.

C	CÁLCULOS Y REPORTE

El alumno entregará al maestro la práctica realizada en el laboratorio contestada, de tal forma que el maestro pueda verificar y validar las soluciones dadas por el alumnos a cada uno de los puntos descritos en la práctica.

5 RESULTADOS Y CONCLUSIONES

El alumno será capaz de realizar modificaciones de información sobre los registros de una tabla en una base de datos, así como eliminación de registros, utilizando instrucciones SQL como UPDATE y DELETE utilizando el DBMS de MySQL

6 ANEXOS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
8	NOMBRE DE LA PRÁCTICA	MySQL Uniones	2

1. INTRODUCCIÓN

Los sistemas de software organizan sus datos en bases de datos. Éstas a su vez están conformadas por múltiples tablas. En ocasiones, recuperar la información deseada requiere buscar en múltiples tablas simultáneamente. Para facilitar la selección de información en dos tablas, SQL provee una función conocida como join o en español unión. El uso de la instrucción join reduce el número de querys de SQL que se tienen que escribir para acceder la información deseada.

2. OBJETIVO (COMPETENCIA)

Recuperar información de múltiples tablas empleando la instrucción join de una manera creativa y lógica.

Formuló	Revisó	Aprobó	Autorizó
Cecilia M. Curlango Rosas	Gloria Etelbina Chávez Valenzuela		Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3. FUNDAMENTO

Las tablas de una base de datos se pueden relacionar entre si por medio de campos llave. Una llave primaria es una columna que contiene un valor único que no se repite en ninguno de los

renglones. El propósito es unir datos entre tablas de tal manera que no se tengan que estar repitiendo los mismos datos en todas las tablas.

La Tabla 1 Empleados tiene como llave primaria el campo `empleadoID`, esto significa que no cada clave corresponde a un solo empleado. Este identificador es independiente del nombre de los empleados. La Tabla 2 Ordenes tiene como llave primaria a la columna `productoID`. En esta misma tabla, el campo `empleadoID` es una referencia a la tabla Empleados.

empleadoID	***Nombre***
101	Carmona, Juan
102	Alvarez, Daniela
103	Sanchez, Julieta
456	Torres, Esperanza

Tabla 1Empleados

productoID	***Producto***	***empleadoID***
340	copiadora	102
854	sumadora	103
971	lampara	103

Tabla 2Ordenes

Para seleccionar datos de ambas tablas se puede usar el siguiente query (quien ordenó un producto y qué fue ese producto) el cual nos arroja como resultado los registros mostrados en la Tabla 3.

```
SELECT Empleados.Nombre, Ordenes.Producto
FROM Empleados, Ordenes
WHERE
Empleados.empleadoID=Ordenes.empleadoID
```

Query 1

Nombre	Producto
Alvarez,Daniela	copiadora
Sanchez,Julieta	sumadora
Sanchez,Julieta	lampara

Tabla 3Resultado de query 1.

Si quisiéramos saber quien ordenó una lampara, el query sería:

```
SELECT Empleados.Nombre
FROM Empleados, Ordenes
WHERE
Empleados.empleadoID=Ordenes.empleadoID
AND Ordenes.Producto='lampara'
```

Query 2

Nombre
Sanchez,Julieta

Tabla 4Resultado de query 2.

Otra forma de hacer el Query 1 es utilizando un join específicamente un Inner Join. La sintaxis de un inner join genérico sería:

```
SELECT campo1, campo2, campo3
FROM primera_tabla
INNER JOIN segunda_tabla
ON primera_tabla.campo_llave = segunda_tabla.llave_foranea
```

Específicamente para obtener la misma información que en el Query 1 el nuevo query sería como el que se muestra en el Query 3.

```
SELECT Empleados.nombre,
Ordenes.Producto
FROM Empleados
INNER JOIN Ordenes
ON
Empleados.empleadoID=Ordenes.EmpleadoID
```

Query 3

Un inner join regresa todos los renglones en los cuales hay concordancia. Si hay renglones en Empleados que no concuerdan con Ordenes, entonces estos renglones no se mostrarán.

Otro tipo de join que existe es el Left Join. Este join regresa todos los renglones de la primer tabla aun cuando no concuerde con la segunda tabla. En este caso en el que los reglones no concuerden, tambien se muestran. La sintaxis de este join es:

```
SELECT campo1, campo2, campo3
FROM primera_tabla
LEFT JOIN segunda_tabla
ON primera_tabla.campo_llave = segunda_tabla.llave_foranea
```

El Query 4 se utiliza para obtener un listado de todos los empleados y sus todas las ordenes que puedan tener. El resultado de este query puede verse en Tabla 5.

```
SELECT Empleados.Nombre, Ordenes.Producto
FROM Empleados
LEFT JOIN Ordenes
ON Empleados.empleadoID=Ordenes.empleadoID
```

Query 4

Nombre	Producto
Carmona,Juan	NULL
Alvarez,Daniela	copiadora
Sanchez,Julieta	sumadora
Sanchez,Julieta	lampara
Torres,Esperanza	NULL

Tabla 5Resultado del query 4

Otro tipo de join es el Right Join. Un Right Join regresa todos los renglones de la segunda tabla, aun cuando no exista concordancia con la primer tabla. Si existen renglones en la segunda tabla que no concuerdan con algo de la primer tabla, entonces estos renglones también se muestra. A continuación se muestra la sintaxis de este join:

```
SELECT campo1, campo2, campo3
FROM primera_tabla
RIGHT JOIN segunda_tabla
ON primera_tabla.campo_llave = segunda_tabla.llave_foranea
```

El siguiente query lista todas las ordenes y los nombres que los que hicieron esas ordenes:

```
SELECT Empleados.Nombre, Ordenes.Producto
FROM Empleados
RIGHT JOIN Ordenes
ON Empleados.empleadoID=Ordenes.EmpleadoID
```

Query 5

Nombre	***Producto***
Alvarez,Daniela	copiadora
Sanchez,Julieta	sumadora
Sanchez,Julieta	lampara

Tabla 6Resultado del Query 5

Para saber quien ordenó una lampara, el query se escribiría como sigue:

```
SELECT Empleados.Nombre
FROM Empleados
INNER JOIN Ordenes
ON Empleados.empleadoID=Ordenes.empleadoID
WHERE Ordenes.Producto='lampara'
```

Query 6

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL DE APOYO
Computadoras con el sistema operativo Linux Acceso y permisos para trabajar con el manejador de bases de datos MySQL en un servidor	Manual de MySQL

B) DESARROLLO DE LA PRÁCTICA

Utilice las tablas de la base de datos descritas en la practica 6 para realizar y probar los siguiente queries empleando joins.

1. Muestre los nombres de todos los jefes de departamento.
2. Muestre los nombres de los participantes en proyectos.
3. Muestre los nombres de las personas que trabajan en el departamento de investigación.
4. Muestre el nombre de cada persona que ha participado en proyectos y el numero total de horas que ha dedicado a proyectos.
5. Muestre la ubicación de los nombres de cada departamento y las cuidades en las que se encuentran.
6. Muestre los nombres de los dependientes de aquellos empleados cuyo ingreso es mayor a 300.
7. Ejercicios adicionales planteados por el maestro.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

Manual de Referencia MySQL Sintaxis `join`: http://dev.mysql.com/doc/refman/5.0/en/join.html

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
9	NOMBRE DE LA PRÁCTICA	Funciones agregadas de Mysql	2

1 INTRODUCCIÓN

Las bases de datos son usadas a menudo para responder a preguntas como las siguientes: ¿Cuantas veces aparece un determinado dato en una tabla?, ¿Cuál es la edad máxima o mínima de un conjunto de personas? . Un gran número de pregunta como estas pueden ser contestadas gracias a las funciones que MySql agrega, para manipular la información contenida en las tablas. Por ejemplo podemos utilizar la función count para contar un conjunto de filas y la función max para determinar el máximo valor contenido en una columna. En esta práctica estudiaremos algunas de las funciones agregadas mas importantes que MySql soporta.

2 OBJETIVO (COMPETENCIA)

Utilizar las funciones adecuadas de Mysql, para dar respuesta a cuestionamientos que se presentan de manera práctica en la sección de desarrollo.

Formuló	Revisó	Aprobó	Autorizó
Ing. Alicia López Aguirre	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

Funciones de agregados. (Aggregate functions)

A menudo queremos realizar operaciones sobre un conjunto de filas de una tabla. Por ejemplo, queremos sumar los contenidos de una columna Importe en toda la tabla, o sólo en las filas que cumplen cierta condición. Para eso existen ciertas funciones llamadas funciones de agregados (Aggregate functions). Veremos algunas de estas funciones.

Cálculos de fecha

MySQL ofrece muchas funciones que puedes usar para realizar cálculos con fechas, por ejemplo, para calcular edades o extraer partes de fechas. Por ejemplo para determinar cuantos años tiene una de tus mascotas, puedes calcular la edad como la diferencia entre la fecha de nacimiento y la fecha actual. Puedes hacerlo convirtiendo las dos fechas a dias, coge la diferencia, y divídela por 365 (el número de dias en un año) como se muestra en el sig. Ejemplo:

```
mysql> SELECT nombre, (TO_DAYS(NOW())-TO_DAYS(nacimiento))/365 FROM mascota;
+--------------+-----------------------------------------------------------------+
| nombre     | (TO_DAYS(NOW())-TO_DAYS(nacimiento))/365     |
+---------------+-----------------------------------------------------------------+
| Bluffy       |                                                            6.94 |
| Claws        |                                                            5.83 |
| Buffy        |                                                           10.68 |
| Fang         |                                                            9.39 |
| Bowser       |                                                           10.38 |
| Chirpy       |                                                            1.34 |
| Whistler     |                                                            2.10 |
| Slim         |                                                            3.71 |
| Puffball     |                                                            0.79 |
+--------------+-----------------------------------------------------------------+
```

Aunque la consulta funcione, existen algunos puntos que podrían ser mejorados. Primero, el resultado podría ser revisado más fácilmente si las filas se presentaran ordenadas de alguna manera. Segundo, la cabecera de la columna edad no es muy significativa. El primer problema puede ser solucionado añadiendo una cláusula ORDER BY nombre para ordenar la salida por nombre. Para arreglar el tema del encabezamiento de columna, puedes darle un nombre a dicha columna de tal forma que aparezca una etiqueta diferente en la salida (esto es lo que se llama un alias de columna):

```
mysql> select nombre, (TO_DAYS(NOW())-TO_DAYS(nacimiento))/365 AS edad FROM mascota ORDER BY nombre;
```

Para ordenar la salida por edad en lugar de por nombre, puedes hacerlo usando símplemente una cláusula ORDER BY diferente:

```
mysql> select nombre, (TO_DAYS(NOW())-TO_DAYS(nacimiento))/365 AS edad  FROM mascota ORDER BY edad;
```

3 FUNDAMENTO

Puede usarse una consulta similar para determinar la edad de la muerte para los animales que hayan muerto. Puedes determinar qué animales son estos comprobando si el valor muerte es NULL o no. Después, para aquellos que no tengan un valor NULL, calcular la diferencia entre los valores muerte y nacimiento:

```
mysql> select nombre, nacimiento, muerte,
   -> (TO_DAYS(NOW())-TO_DAYS(nacimiento))/365 AS edad
   -> FROM mascota WHERE muerte IS NOT NULL ORDER BY edad;

+------------+-----------------+----------------+-------+
| nombre    | nacimiento  | muerte       | edad |
+------------+-----------------+----------------+-------+
| Bowser     | 1989-08-31 | 1995-07-29 | 10.38|
+------------+-----------------+----------------+-------+
```

La consulta usa muerte IS NOT NULL en lugar de muerte != NULL dado que NULL es un valor especial. Esto se explica más adelante. [Puedes consultar la sección [Working with NULL] del manual de MySQL.

¿Qué harías si quisieras saber qué animales cumplen años el mes que viene? Para este tipo de cálculos, año y día son irrelevantes, simplemente querrás extraer la parte mes de la columna nacimiento. MySQL ofrece muchas funciones de extracción de parte-de-fecha, como YEAR(),MONTH() y DAY(). La función apropiada para nuestro problema es MONTH(). Para ver cómo funciona, ejecuta una consulta rápida que muestre el valor de la fecha de nacimiento y el mes de nacimiento (MONTH(nacimiento)):

```
mysql> SELECT nombre, nacimiento, MONTH(nacimiento) FROM mascota;
```

Buscar animales que hayan nacido en el mes próximo es también sencillo de realizar. Suponte que Abril es el mes actual. Entonces el valor del mes es 4 y lo que buscas son animales nacidos en Mayo (mes 5):

```
mysql> SELECT nombre, nacimiento FROM mascota WHERE MONTH(nacimiento) = 5;
```

Existe una pequeña complicación si el mes actual es Diciembre, por supuesto. No puedes añadir simplemente uno al número de mes (12) y buscar animales nacidos en el mes 13, dado que no existe tal mes. En lugar de eso, debes buscar animales nacidos en Enero (mes 1).

Puedes escribir la consulta de tal forma que funcione independientemente del mes en el que estemos. De esa forma no tendrás que usar un número de mes en particular en la consulta. DATE_ADD() te permite añadir un intervalo de tiempo a una fecha dada. Si añades un mes al valor de NOW(), y después extraes la parte del mes con MONTH(), el resultado produce el mes del cumpleaños que buscamos:

3 FUNDAMENTO

```
mysql> select NOW();
+---------------------------+
| NOW()                     |
+---------------------------+
| 2000-01-13 18:13:09 |
+---------------------------+

mysql> SELECT nombre, nacimiento FROM mascota
   -> WHERE MONTH(nacimiento) = MONTH(DATE_ADD(NOW(),INTERVAL 1 MONTH));

+-------------+---------------+
| nombre    | nacimiento |
+-------------+---------------+
| Bluffy       | 1993-02-04 |
+-------------+---------------+
```

Una manera difente de conseguir los mismos resultados es añadir 1 al mes actual para conseguir el mes siguiente (tras usar la función módulo (MOD) para convertir el valor de mes actual en 0 si estamos en Diciembre (mes 12)):

```
mysql> SELECT nombre, nacimiento FROM mascota
   -> WHERE MONTH(nacimiento) = MOD(MONTH(NOW()),12) +1;
+----------+-----------------+
| nombre | nacimiento    |
+----------+-----------------+
| Bluffy    | 1993-02-04   |
+----------+-----------------+
```

Contando filas

Las bases de datos son usadas a menudo para responder a la pregunta, ¿cuantas veces aparece un determinado tipo de datos en una tabla?". Por ejemplo, podrías querer saber cuántas mascotas tienes, o cuántas mascotas tiene cada propietario, o podrías querer realizar varios tipos de censos respecto a tus animales. Contar el número total de animales que tienes es lo mismo que preguntar ¿cuántas filas hay en la tabla mascota?, dado que hay sólo una fila por mascota. La función COUNT() cuenta el número de resultados no-NULL (N.T.: más adelante se indica una consulta SQL en la que se cuenta el número de animales por sexo, y aparece el símbolo NULL como parte de esa cuenta... lo cual invalida la afirmación que precede a este comentario!), así pues, la consulta a realizar para contar el número de animales tiene la siguiente forma:

```
mysql> SELECT COUNT(*) FROM mascota;
+----------------+
| COUNT(*)     |
+----------------+
|       9          |
+----------------+
```

3 FUNDAMENTO

Antes, conseguiste los nombres de las personas que poseen una mascota. Puedes usar COUNT() para averiguar cuántas mascotas tiene cada propietario:

mysql> SELECT propietario, COUNT(*) FROM mascota GROUP BY propietario;

Observa el uso de GROUP BY para agrupar todos los registros de cada propietario. Si no lo hubiéramos puesto, todo lo que conseguirias sería un mensaje de error:

mysql> SELECT propietario, COUNT(propietario) FROM mascota;

ERROR 1140: Mixing of GROUP columns (MIN(),MAX(),COUNT()...) with no GROUP columns is illegal if there is no GROUP BY clause COUNT() y GROUP BY son útiles para la caracterización de tus datos de varias formas. Los siguientes ejemplos muestran difentes maneras para realizar operaciones de censo animal.

Número de animales por especies:

mysql> SELECT especie, COUNT(*) FROM mascota GROUP BY especie;

Número de animales por sexo:

mysql> SELECT sexo , COUNT(*) FROM mascota GROUP BY sexo;

```
+-------------+---------------+
| sexo        | COUNT(*) |
+-------------+---------------+
| NULL        |             1 |
| f           |             4 |
| m           |             4 |
+------------+---------------+
```

(En este resultado, NULL indica "sexo desconocido")

El número de animales por combinación de especies y sexo:

mysql> SELECT especie , sexo, COUNT(*) FROM mascota GROUP BY especie, sexo;

3 FUNDAMENTO

No necesitas recuperar una tabla completa cuando uses COUNT(). Por ejemplo, la consulta anterior, cuando se realiza sólo sobre perros y gatos, se escribe así:

```
mysql> SELECT especie , sexo, COUNT(*) FROM mascota
    -> WHERE especie = "perro" OR especie = "gato"
    -> GROUP BY especie, sexo;

+--------------+-------------+--------------+
| especie      | sexo        | COUNT(*) |
+--------------+-------------+--------------+
| gato         | f           |            1 |
| gato         | m           |            1 |
| perro        | f           |            1 |
| perro        | m           |            2 |
+--------------+-------------+--------------+
```

O, si quieres conocer el número de animales por sexo sólo para animales de sexo conocido:

```
mysql> SELECT especie , sexo, COUNT(*) FROM mascota  WHERE sexo IS NOT NULL GROUP BY especie, sexo;
```

COUNT (DISTINCT Nombre_de_columna). Cuenta cuántos valores no NULL diferentes hay de la columna especificada, en las filas especificadas. El siguiente ejemplo produce el número de clientes distintos a los cuales se les han registrado ventas.

SUM(). Esta función totaliza una columna, dentro de las filas que cumplen una condición, o de todas las filas si no se especifica ninguna condición. En lo que sigue abreviaremos esta expresión y diremos simplemente “de las filas especificadas'”. Ejemplos: Sumar todos los importes de Ventas.

```
mysql> SELECT SUM(importe) FROM ventas;
+------------------+
| SUM(importe) |
+------------------+
|     1010.00     |
+------------------+
```

3 FUNDAMENTO

Sumar todos los importes de Ventas, con fecha del año 2003.

```
mysql> SELECT SUM(importe) FROM ventas WHERE YEAR(fecha)=2003;
+-------------------+
| SUM(importe) |
+-------------------+
|        NULL      |
+-------------------+
```

De manera similar podemos utilizar las siguientes funciones:

MAX(). Produce el máximo valor de una columna dentro de las filas especificadas.
MIN(). Produce el mínimo valor de una columna en las filas especificadas.
AVG(). Produce el promedio de los valores de una columna numérica, contando sólo las filas especificadas.

Problemas resueltos

1.- Obtener la suma de los salarios mayores a 11000 de un grupo de Empleados.

Solución:

Debemos obtener la suma de la columna Salario de aquellas filas de la tabla Empleados que tienen Salario mayor a 11000.

```
mysql> SELECT SUM(Salario) FROM Empleados WHERE Salario > 11000;
```

2.- Contar cuántas direcciones de clientes se tienen.

Solución:

Debemos obtener el número de filas de Clientes cuya columna Direccion no es NULL.

```
mysql>SELECT COUNT(Direccion) FROM Clientes;
```

Observar que como se especificó COUNT(Direccion) las direcciones NULL no se cuentan. En cambio SELECT COUNT(*) FROM Clientes cuenta todas las filas, independientemente que tengan o no dirección (o algúna otra columna) NULL.

3.- ¿Cuál es el salario promedio de la empresa?
Solución:

```
mysql>SELECT AVG(Salario) FROM Empleados;
```

4.-¿Cuál es el salario máximo pago por la empresa?
Solución:

```
mysql>SELECT MAX(Salario) FROM Empleados;
```

3 FUNDAMENTO

5.- ¿Cuáles son los empleados más antiguos de la empresa?
Solución:

Vamos a seleccionar todas las filas de la tabla Empleados con Fecha de ingreso igual a la fecha de ingreso mínima.

```
SELECT * FROM Empleados WHERE Fecha_ingreso = (SELECT MIN(Fecha_ingreso) FROM Empleados);
```

6.- Hallar los nombres de los empleados con salario máximo en la empresa.
Solución:

```
SELECT Nombre FROM Empleado  WHERE Salario =(SELECT MAX(Salario) FROM Empleados);
```

7.-¿Cuántos salarios distintos se pagan en la empresa?
Solución:

```
SELECT COUNT(DISTINCT Salario) FROM Empleados;
```

La cláusula HAVING.

Después de un GROUP BY, todavía se pueden volver a seleccionar filas con una cláusula HAVING. Por ejemplo, podemos estar interesados, hipotéticamente, en las facturas con más de una línea. Entonces deberíamos usar

```
mysql>SELECT Factura, COUNT(*) FROM Lineas_Factura GROUP BY Factura HAVING COUNT(*) > 1;
```

Si no nos interesa ver el número de líneas podemos usar

```
mysql>SELECT Factura  FROM Lineas_Factura GROUP BY Factura HAVING COUNT(*) > 1;
```

Otra variante más legible

```
mysql>SELECT factura, COUNT(*) AS cantidad FROM lineas_factura GROUP BY factura HAVING cantidad > 1;
```

El orden de las cláusulas vistas hasta ahora es SELECT.....FROM...., WHERE, GROUP BY, HAVING, ORDER BY

Recuerda que existen dos momentos en que se seleccionan filas, una con el WHERE antes de la agrupación y otro con el HAVING, después de la misma.

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con acceso al servidor Mysql	Práctica impresa

B DESARROLLO DE LA PRÁCTICA

1.- Introduce tu login y password para acceder al servidor de MySQL.
2.- Borra las bases de datos utilizadas en la práctica anterior si es que ya fueron revisadas por tu tutor.
2.- Crear la Base de datos llamada FACKBOOKXXX donde XXX es tu matrícula
3.- La base de datos contendra la siguiente tabla llamada CIA:

name	region	area	population	gdp
Yemen	Middle East	527970	14728474	23400000000
Zaire	Africa	2345410	44060636	18800000000
Zambia	Africa	752610	9445723	7900000000
Zimbabwe	Africa	390580	11139961	17400000000
Mexico	America	1346232	80000000	167340000000
EEUU	America	1568731	65000000	34647540000000

Donde:

Area: Es una medida en metros cuadrados
GDP: Es el producto interno bruto. Es una medida de la abundancia total generada por el país en Dolares en un año.

Esta tabla contiene en total 265 datos, por comodidad trabajaremos solo con el fragmento presentado arriba suponiendo que dicho fragmento representa a todos los paises del mundo. Si desea obtener la tabla completa puede consultar la sig. dirección http:www.cia.gov que es la dirreción de la agencia central de inteligencia conocida por sus siglas en ingles como CIA.

4.-Realice una consulta que muestre la población total del mundo.
5.-Enumere todas las regiones distintas que existen.
6.-Enumere los países que tienen un GDP mayor que el conjunto de Africa.
7.-Enumere cada región y los países que pertenecen a dicha región.
8.-Enumere las regiones con una población superior a 14 millones de habitantes.

B DESARROLLO DE LA PRÁCTICA

9.-Para cada región, enumere región y número de paises que contienen una población inferior a los 10 millones de habitantes.
10.-Obtenga el promedio del GPD mundial.
11.-Muestre el nombre de los paises que tienen una superficie cuadrada mayor al promedio la superficie mundial.
12.-Muestre la región en la que se encuentran los paises que tienen una superficie cuadrada menor al promedio de la superficie mundial.
13.-Muestre el nombre del país con mayor población.
14.-Muestre el nombre del país con menor población.
15.-Enumere para las diferentes regiones del mundo el promedio de población.
16.-Enumere los paises que contienen un GDP mayor al conjunto de America.
17.-Obtenga el promedio de población para America.
18.-Obtenga el promedio de población para Africa.
19.-Muestre el nombre de la región con mayor población.
20.-Muestre el nombre de la región con menor población.

C CÁLCULOS Y REPORTE

El alumno deberá entregar un reporte impreso que contenga las respuestas de cada uno de los puntos propuestos en la parte del desarrollo de esta práctica.

5 RESULTADOS Y CONCLUSIONES

Al finalizar la práctica el alumno será capaz de manejar las principales funciones agregadas que soporta Mysql.

6 ANEXOS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA)
10	NOMBRE DE LA PRÁCTICA	Conexión Mysql-Java	2

1 INTRODUCCIÓN

JDBC es un API de Java para acceder a sistemas de bases de datos, y prácticamente a cualquier tipo de dato tabular. El API JDBC consiste de un conjunto de clases e interfaces que permiten a cualquier programa Java acceder a sistemas de bases de datos de forma homogénea. En otras palabras, con el API JDBC no es necesario escribir un programa para accesar a Sybase, otro programa para accesar a Oracle, y otro programa para accesar a MySQL; con esta API, se puede crear un sólo programa que sea capaz de enviar sentencias SQL a la base de datos apropiada. Una aplicación de Java debe tener acceso a un controlador (driver) JDBC adecuado.

2 OBJETIVO (COMPETENCIA)

Elaborar una aplicación gráfica en Java utilizando el conjunto de clases e interfaces necesarias, que permitan al alumno conocer la forma correcta de realizar transacciones en una base de datos de Mysql.

Formuló	Revisó	Aprobó	Autorizó
Ing. Alicia del R. López Aguirre	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

JDBC

Este controlador es el que implementa la funcionalidad de todas las clases de acceso a datos y proporciona la comunicación entre el API JDBC y la base de datos real.

De una manera muy simple, al usar JDBC se pueden hacer tres cosas:
1.Establecer una conexión a una fuente de datos (ej. una base de datos).
2.Mandar consultas y sentencias a la fuente de datos.
3.Procesar los resultados.

Los distribuidores de bases de datos suministran los controladores que implementan el API JDBC y que permiten acceder a sus propias implementaciones de bases de datos. De esta forma JDBC proporciona a los programadores de Java una interfaz de alto nivel y les evita el tener que tratar con detalles de bajo nivel para acceder a bases de datos.

En el caso del manejador de bases de datos MySQL, **Connector/J** es el driver JDBC oficial. En el momento de escribir esta práctica, se pueden encontrar dos versiones de este driver, la versión estable (Connector/J 2), y la versión en desarrollo (Connector/J 3). Para los ejemplos que se mostrarán a continuación se hará referencia a la versión 2 del driver, y en particular a la versión 2.0.14. Los procedimientos descritos aquí deben de ser practicamente los mismos si se utiliza alguna otra versión del driver, incluso, si se usa alguna de las versiones en desarrollo. Por supuesto, se recomienda siempre utilizar la versión estable más reciente que se pueda obtener.

Cabe señalar que actualmente JDBC es el nombre de una marca registrada, y ya no más un acrónimo; es decir, JDBC ya no debe entenderse como "Java Database Conectivity".

Herramientas necesarias

1.Un ambiente de desarrollo para Java, tal como el Java 2 SDK, el cual está disponible en java.sun.com. La versión estándar del SDK 1.4 ya incluye el API JDBC.
2.Un servidor de bases de datos MySQL al que se tenga acceso con un nombre de usuario y contraseña.
3.El driver JDBC para MySQL, Connector/J

Cargar el controlador JDBC

Para trabajar con el API JDBC se tiene que importar el paquete java.sql, tal y como se indica a continuación:

import java.sql.;*

En este paquete se definen los objetos que proporcionan toda la funcionalidad que se requiere para el acceso a bases de datos. El siguiente paso después de importar el paquete java.sql consiste en cargar el controlador JDBC, es decir un objeto Driver específico para una base de datos que define cómo se ejecutan las instrucciones para esa base de datos en particular.

3 FUNDAMENTO

Hay varias formas de hacerlo, pero la más sencilla es utilizar el método forName() de la clase Class:

Class.forName("Controlador JDBC");

para el caso particular del controlador para MySQL, Connector/J, se tiene lo siguiente:

Class.forName("com.mysql.jdbc.Driver");

Debe tenerse en cuenta que el método estático forName() definido por la clase Class genera un objeto de la clase especificada. Cualquier controlador JDBC tiene que incluir una parte de iniciación estática que se ejecuta cuando se carga la clase. En cuanto el cargador de clases carga dicha clase, se ejecuta la iniciación estática, que pasa a registrarse como un controlador JDBC en el DriverManager. Es decir, el siguiente código:

Class.forName("Controlador JDBC");

es equivalente a:

```
Class c = Class.forName("Controlador JDBC");
Driver driver = (Driver)c.newInstance();
DriverManager.registerDriver(driver);
```

Algunos controladores no crean automáticamente una instancia cuando se carga la clase. Si forName() no crea por sí solo una instancia del controlador, se tiene que hacer esto de manera explícita:

Class.forName("Controlador JDBC").newInstance();

De nuevo, para el Connector/J:

Class.forName("com.mysql.jdbc.Driver").newInstance();

Establecer la conexión

Una vez registrado el controlador con el DriverManager, se debe especificar la fuente de datos a la que se desea acceder. En JDBC, una fuente de datos se especifica por medio de un URL con el prefijo de protocolo jdbc: , la sintaxis y la estructura del protocolo es la siguiente:

jdbc:{subprotocolo}:{subnombre}

El {subprotocolo} expresa el tipo de controlador, normalmente es el nombre del sistema de base de datos, como db2, oracle o mysql.

El contenido y la sintaxis de {subnombre} dependen del {subprotocolo}, pero en general indican el nombre y la ubicación de la fuente de datos.

3 FUNDAMENTO

El formato general para conectarse a MySQL es:

jdbc:mysql:///[servidor][:puerto]/[base_de_datos][?param1=valor1][param2=valor2]...

Si queremos acceder de manera local a la base de datos, el URL sería :

String url = "jdbc:mysql://localhost/Nombre_de_la_base_de_datos";

Una vez que se ha determinado el URL, se puede establecer una conexión a una base de datos. El objeto Connection es el principal objeto utilizado para proporcionar un vínculo entre las bases de datos y una aplicación Java. Connection proporciona métodos para manejar el procesamiento de transacciones, para crear objetos y ejecutar instrucciones SQL y para crear objetos para la ejecución de procedimientos almacenados.

Se puede emplear tanto el objeto Driver como el objeto DriverManager para crear un objeto Connection. Se utiliza el método connect() para el objeto Driver, y el método getConnection() para el objeto DriverManager.

El objeto Connection proporciona una conexión estática a la base de datos. Esto significa que hasta que se llame en forma explícita a su método close() para cerrar la conexión o se destruya el objeto Connection, la conexión a la base de datos permanecerá activa.

La manera más usual de establecer una conexión a una base de datos es invocando el método getConnection() de la clase DriverManager. A menudo, las bases de datos están protegidas con nombres de usuario (login) y contraseñas (password) para restringir el acceso a las mismas. El método getConnection() permite que el nombre de usuario y la contraseña se pasen también como parámetros.

```
String login = "Alicia";
String password = "reprobaralumnos";
    Connection conn = DriverManager.getConnection(url,login,password);
```

En toda aplicación de bases de datos con MySQL es indispensable poder establecer la conexión al servidor para posteriormente enviarle las consultas. Los programas en Java no son la excepción. El siguiente código nos servirá para verificar que podemos establecer una conexión a una base de datos.

```
import java.sql.*;

public class TestConnection
{
  static String bd = "UABC";
  static String login = "Alicia";
  static String password = "reprobaralumnos";
  static String url = "jdbc:mysql://localhost/"+bd;
```

3 FUNDAMENTO

```
  public static void main(String[] args) throws Exception
  {
    Connection conn = null;
   try
    {
      Class.forName("com.mysql.jdbc.Driver").newInstance();
       conn = DriverManager.getConnection(url,login,password);
      if (conn != null)
      {
        System.out.println("Conexión a base de datos "+url+" ... Ok");
        conn.close();
      }
    }
    catch(SQLException ex)
    {
      System.out.println(ex);
    }
    catch(ClassNotFoundException ex)
    {
      System.out.println(ex);
    }

  }
}
```

Creación de sentencias

Como se mencionó en la sección anterior, el objeto Connection permite establecer una conexión a una base de datos. Para ejecutar instrucciones SQL y procesar los resultados de las mismas, debemos hacer uso de un objeto Statement.

Los objetos Statement envían comandos SQL a la base de datos, que pueden ser de cualquiera de los tipos siguientes:

1.Un comando de definición de datos como CREATE TABLE o CREATE INDEX.
2.Un comando de manipulación de datos como INSERT, DELETE o UPDATE.
3.Un sentencia SELECT para consulta de datos.

Un comando de manipulación de datos devuelve un contador con el número de filas (registros) afectados, o modificados, mientras una instrucción SELECT devuelve un conjunto de registros denominado conjunto de resultados (result set). La interfaz Statement no tiene un constructor , sin embargo, podemos obtener un objeto Statement al invocar el método createStatement() de un objeto Connection.

3 FUNDAMENTO

conn = DriverManager.getConnection(url,login,pasword);
Statement stmt = conn.createStatement();

Una vez creado el objeto Statement, se puede emplear para enviar consultas a la base de datos usando los métodos execute(), executeUpdate() o executeQuery(). La elección del método depende del tipo de consulta que se va a enviar al servidor de bases de datos:

execute( )
Se usa principalmente cuando una sentencia SQL regresa varios conjuntos de resultados. Esto ocurre principalmente cuando se está haciendo uso de procedimientos almacenados.
executeUpdate( )
Este método se utiliza con instrucciones SQL de manipulación de datos tales como INSERT, DELETE o UPDATE.
ExecuteQuery( )
Se usa en las instrucciones del tipo SELECT.

Es recomendable que se cierren los objetos Connection y Statement que se hayan creado cuando ya no se necesiten. Lo que sucede es que cuando en una aplicación en Java se están usando recursos externos, como es el caso del acceso a bases de datos con el API JDBC, el recolector de basura de Java (garbage collector) no tiene manera de conocer cuál es el estado de esos recursos, y por lo tanto, no es capaz de liberarlos en el caso de que ya no sean útiles. Lo que sucede en estos casos es que pueden quedar almacenados en memoria grandes cantidades de recursos relacionados con la aplicación de bases de datos que se está ejecutando. Es por esto que se recomienda que se cierren de manera explícita los objetos Connection y Statement.

De manera similar a Connection, la interfaz Statement tiene un método close() que permite cerrar de manera explícita un objeto Statement. Al cerrar un objeto Statement se liberan los recursos que están en uso tanto en la aplicación Java como en el servidor de bases de datos.

```
Statement stmt = conn.createStatement();
    ....
stmt.close();
```

Ejecución de consultas

Cuando se ejecutan sentencias SELECT usando el método executeQuery(), se obtiene como respuesta un conjunto de resultados, que en Java es representado por un objeto ResultSet.

```
Statement stmt = conn.createStatement();
ResultSet res  = stmt.executeQuery("SELECT * FROM nombre_de_la_tabla");
```

La información del conjunto de resultados se puede obtener usando el método next() y los diversos métodos getXXX() del objeto ResultSet. El método next() permite moverse fila por fila a través del ResultSet, mientras que los diversos métodos getXXX() permiten accesar a los datos de una fila en particular.

3 FUNDAMENTO

Los métodos getXXX() toman como argumento el índice o nombre de una columna, y regresan un valor con el tipo de datos especificado en el método. Así por ejemplo, getString() regresará una cadena, getBoolean() regresará un booleano y getInt() regresará un entero. Cabe mencionar que estos métodos deben tener una correspondencia con los tipos de datos que se tienen en el ResultSet, y que son a las vez los tipos de datos provenientes de la consulta SELECT en la base de datos, sin embargo, si únicamente se desean mostrar los datos se puede usar getString() sin importar el tipo de dato de la columna.

Por otra parte, si en estos métodos se utiliza la versión que toma el índice de la columna, se debe considerar que los índices empiezan a partir de 1, y no en 0 (cero) como en los arreglos, los vectores, y algunas otras estructuras de datos de Java.

Existe un objeto ResultSetMetaData que proporciona varios métodos para obtener información sobre los datos que están dentro de un objeto ResultSet. Estos métodos permiten entre otras cosas obtener de manera dinámica el número de columnas en el conjunto de resultados, así como el nombre y el tipo de cada columna.

```
ResultSet res  = stmt.executeQuery("SELECT * FROM nombre_de_la_tabla");
ResultSetMetaData metadata = res.getMetaData();
```

Un ejemplo completo

El siguiente programa realiza la inserción de renglones en la tabla tablaPrueba de la base de datos test utilizando una interfaz gráfica para captura de datos.Este programa ejecuta la inserción (Insert) a través de una instrucción de SQL precompilada que permite ejecutar sentencias de SQL de una manera más eficiente y fácil de codificar. De una conexión se obtiene, utilizando el método preparedStatement(), el llamado PreparedStatment ó SQL precompilado. Un PreparedStatement es más eficiente ya que se prepara una vez y se puede utilizar en múltiples ocaciones. Además resulta más fácil de codificar con éste, ya que libera al programador de escribir complejas concatenaciones de Strings y de usar métodos de formateo de datos propietarios para formular sentencias del SQL. En vez, se utiliza el signo de interrogación (?) para reservar un lugar para cada parámetro de entrada y programáticamente se especifican los valores para cada lugar reservado previo a la ejecución de la sentencia de SQL.

```
import java.awt.*; import java.awt.event.*; import java.sql.*;
public class JDBCInserta extends Frame implements ActionListener{
TextField texto; Button b1; TextArea data;  Label etiqueta;
 int rows = 0;
public JDBCInserta()
{  super("JDBC Inserta"); setSize(240, 280); setLayout(new FlowLayout());
   TextField("jdbc:mysql://127.0.0.1/test? user=Ala&password=", 20);
   etiqueta = new Label("Escriba el texto a insertar:"); add(etiqueta);
   texto=new TextField("algun texto", 18); add(texto);
   b1=new Button("Inserta Datos"); b1.addActionListener(this); add(b1);
   data=new TextArea(10, 30); add(data);
   addWindowListener(new WindowAdapter(){ public void windowClosing(WindowEvent e)
      { dispose(); System.exit(0); }}); show();
}
```

3 FUNDAMENTO

```
public void insertaDatos() throws SQLException
{ String usr, pass;
   String f1, f2;
   String url="jdbc:mysql://127.0.0.1/test? user=Ala&password=";
   String text1 = texto.getText();
    PreparedStatement pstmt = null;
    try{ // inicializar y cargar Driver de Mysql.
            Class.forName("com.mysql.jdbc.Driver");
       }catch(ClassNotFoundException e){System.out.println("Could not find driver!"); System.exit(1); }
    Connection con=DriverManager.getConnection(url);//, usr, pass);
    try{ pstmt = con.prepareStatement(  "INSERT into tablaPrueba ( "+ " identif,"+ "campoTexto)"+ " values ( "+ " ","+
          "   ?) ");
         }
          catch (SQLException e) {  System.err.println(e); }
          pstmt.setString(1,text1);
         rows = 0;
         rows=pstmt.executeUpdate();
         pstmt.close();
     if( rows == 1 ) { data.append("\nRegistro Insertado"); }
}

public void actionPerformed(ActionEvent e)
{
   data.replaceRange("Field 1\t\tField 2",0,59);
   data.append("\n-----\t\t-----");
   try{ insertaDatos(); }
   catch(SQLException ex)
   { data.append("\nError inserting in database:"+rows+"inserted.");
      data.append("\n"+ex.getMessage());
   }
}

public static void main(String args[])
{   new JDBCInserta(); }

}
```

3 FUNDAMENTO

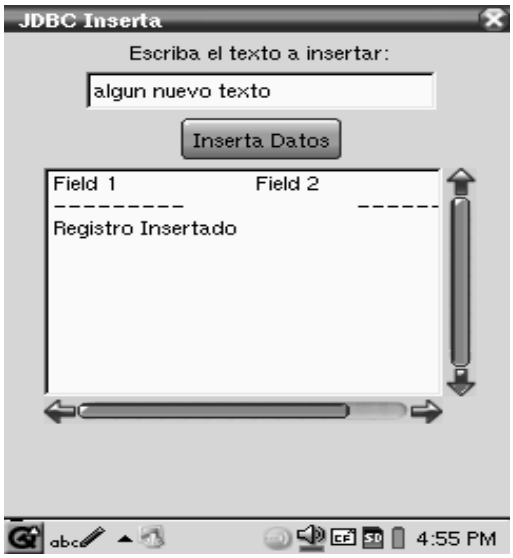


Figura del programa en ejecución

4 PROCEDIMIENTO (DESCRIPCIÓN)	
A EQUIPO NECESARIO	**MATERIAL DE APOYO**
Computadora con acceso al servidor Mysql	Práctica impresa
B DESARROLLO DE LA PRÁCTICA	

4 PROCEDIMIENTO (DESCRIPCIÓN)

Sistema de infomación ecológica para áreas protegidas (FORIS)

En el a?o 2000 la FAO elaboró nuevos mapas mundiales de los bosques y de las zonas ecológicas, los cuales definen espacialmente las estadísticas del área, que a su vez fueron producto del estudio realizado en cada país y en cada una de las regiones, proporcionando así un cuadro sinóptico de la **cubierta forestal en todo el mundo.** El mapa mundial de zonas ecológicas proporciona un medio importante para agregar la información mundial sobre los bosques u otros recursos naturales de acuerdo a su carácter ecológico. El mapa de la cubierta forestal fue desarrollado mediante la utilización de imágenes satelitares de resolución gruesa. En las evaluaciones mundiales previas, no existían los medios ni la tecnología para producir un mapa mundial basado en imágenes satelitares. Actualmente otro organismo forestal llamado FRA se dio a la tarea de dividir el mapa mundial en zonas ecológicas y recopilar cierta información sobre ellas. Por ejemplo entre otras cosas se recopilaron los datos de áreas protegidas tanto nacionales como internacionales. La infomación se encuentra en la sig. tabla

Datos internacionales y nacionales para las áreas protegidas

	POLIGONOS			PUNTOS		
Región	**Nacional**	**Internacional**	**Total**	**Nacional**	**Internacional**	**Total**
África	1 926	293	2 219	2 088	74	2 162
Asia	3 907	288	4 195	2 384	107	2 491
Europa	2 1468	1 587	23 055	19 478	1 915	21 393
Norte y Centro América	10 119	352	10 471	4 722	92	4 814
Oceanía	816	427	1 243	2 739	53	2 792
América del Sur	2 436	158	2 594	1 413	48	1 461
Antártica	0	28	28			
Otros	25	12	37	3 156	517	3 673

Nota: Se manejan dos tipos de area para cada región POLIGONOS y PUNTOS.

Elabore una aplicación gráfica en Java que permita realizar las opciones:

a) Almacenar toda la información anterior en una base de datos.
b) Modificaciones para los datos de región, superficie nacional y internacional (tanto para poligonos como para puntos).
c) Eliminar alguna región específica.
d) Opciones varias:
   1.- Total de áreas POLIGONOS protegidas tanto nacional como internacional para cada región
   2.- Nombre de la región con mayor área protegida de PUNTOS nacionalmente
   3.- Total de áreas protegidas tanto nacional e internacionalmente en todo el planeta.

C	CÁLCULOS Y REPORTE
El alumno deberá entregar impreso el código de la aplicación elaborado en esta práctica.	

5 RESULTADOS Y CONCLUSIONES

Al finalizar la práctica el alumno será capaz de ralizar mediante aplicaciones gráficas en Java conexiones y transacciones con el servidor de base de datos Mysql .

6 ANEXOS

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA
11	NOMBRE DE LA PRÁCTICA	Introduccion a Microsoft SQL Server	2 horas

1. INTRODUCCIÓN

Microsoft SQL Server es un sistema administrador de bases de datos relacionales muy poderoso e importante en el mercado, dado que es multi-usuario, ademas todas sus herramientas tienen interfaces graficas amigables y es el competidor principal de Oracle. Posee las siguientes caracteristicas:

Variedad de interfaces de usuario
Independencia logica y fisica de datos
Optimizacion de busquedas o querys
Integridad de datos
Control de concurrencia
Respaldo y recuperacion
Seguridad y Autorizacion

2. OBJETIVO (COMPETENCIA)

El alumno utilizara el manejador de bases de datos relacional Microsoft SQL Server para crear una base de datos, agregara informacion a la misma y ejecutara procedimientos almacenados.

Formuló	Revisó	Aprobó	Autorizó
M.C. Monceni Anabel Perez	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero

Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3. FUNDAMENTO

SQL Server posee tres formas para crear bases de datos.

1.- En Enterprise Manager.- Seleccione el folder **Databases**, despues **Action** -> **New Database**

2.- Usando el Asistente.- Simplemente de click derecho en el **Enterprise Manager o Tools** -> **Wizards** -> **Create Database Wizards**, una vez que aparezca la ventana unicamente siga los pasos del asistente y al terminar habra creado una base de datos.

3.- Lenguaje Transact-SQL.- Para lograrlo SQL Server posee **SQL Server Query Analizer**, el cual es un procesador de consultas que permitiran realizar cualquier operacion mediante lineas de codigo en el lenguaje estructurado de consultas(SQL).

Stored Procedure (Procedimiento almacenado).- Es un tipo especial de batch usando el lenguaje SQL y sus extensiones. Son una coleccion precompilada de declaraciones SQL almacenadas bajo un nombre y procesadas como una unidad.

Estan almacenados dentro de una base de datos, pueden ser ejecutados desde un llamado de una apliacacion, permitiendo al usuario declarar variables, condicionar ejecuciones, etc.

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL DE APOYO

Computadoras con Microsoft SQL Server Instalado

Permisos para crear bases de datos y ejecutar procedimientos almacenados

B) DESARROLLO DE LA PRÁCTICA

1.- Una vez que haya ingresado a SQL Server, presione el boton de **Query Analizer** para entrar al procesador de consultas. Anote lo siguiente y al terminar no se olvide presionar el boton "**Execute**"

```
CREATE DATABASE VENTAS;
```

2.- Aparecera en la ventana "**Messages**"que su base de datos ha sido creada exitosamente. Para poder verla en el folder "**Databases**" debera presionar el boton "**Refresh**". Una vez que haya verificado que su base de datos se creo, es preciso anotar las tablas que contendra. Entre de nuevo al Query Analizer y escriba lo siguiente:

```
USE VENTAS;
CREATE TABLE Producto(
ClaveP      int,
Nombre      varchar(30),
AA          int,
Tipo        varchar(3),
Edicion     varchar(3),
Precio      money,
Constraint uno Primary key(ClaveP) );
GO
CREATE TABLE Ordenes(
Norden      int not null,
ClaveP      int not null,
Forden      DATETIME not null,
Fenvio AS DATEADD(Day,7, Forden),
Cant        int not null,
Constraint dos Primary Key(Norden),
Constraint tres Foreign Key(ClaveP) References Producto(ClaveP) );
```

3.- Ejecute, una vez que haya verificado que las tablas se crearon, agregue a la tabla Ordenes el atributo Total que debera ser el resultado de la multiplicacion del precio por Cantidad.

ALTER TABLE Ordenes ADD Total AS Producto.Precio * Ordenes.Cant;

4.- Ya que tiene todo lo anterior, ingrese los siguientes valores a la tabla de Producto:

INSERT INTO Producto(ClaveP, Nombre, AA, Tipo, Edicion, Precio) VALUES (1, 'Blond Ambition Concert', '1990', 'VHS', 'FS', 40), (2, 'Lord of the Rings Ext Edit', '2003', 'DVD', 'LBX', 20), (3, 'Drowned World Tour', '2001', 'DVD', 'FS', 20), (4, 'Alien Vs. Predator', '2005', 'DVD', 'FS', 15), (5, 'Confessions on a dance floor', '2006', 'CD', 'LP', 19), (6, 'Mexico en la Piel', '2005', 'CD', 'LP', 18);

5.- Haga combinaciones con estos productos al insertar datos en la tabla ordenes.

6.- Practique las instrucciones SELECT, UPDATE y DELETE en varios tuples de ambas tablas.

7.- Dentro del Query Analizer anote el siguiente codigo:

USE VENTAS;

GO

CREATE PROCEDURE incrementar-precio(@percent int=6)

AS UPDATE Producto SET Precio=(Precio+(Precio*@percent))/100;

8.- Para ejecutar un stored procedure entre de nuevo al Query Analizer y escriba:

EXECUTE incrementar_precio 10;

9.- Para modificar informacion de un stored procedure utilize la instruccion ALTER PROCEDURE nombre_procedimiento .... y para eliminarlo DROP PROCEDURE Nombre_procedimiento

Realize los puntos de la practica del 1 al 9 y tome notas en cada uno de ellos en caso de ser necesario.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

SQL Server en su folder “Security” tiene varios procedimientos almacenados, los cuales podra identificar porque inician con la palabra reservada **sp**_nombreprocedimiento.

sp_addlogin 'UsuarioBD', 'LabBD';
sp_grantdbaccess 'UsuarioBD';
sp_revokedbaccess 'UsuarioBD';
sp_droplogin 'UsuarioBD';

Estos procedimientos almacenados solo puede utilizarlos el administrador de SQL Server (sa).

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC	2003-1	5046	Bases de Datos

PRÁCTICA No.	LABORATORIO DE	Bases de Datos	DURACIÓN (HORA
12	NOMBRE DE LA PRÁCTICA	Aplicaciones Web con SQL Server	2 horas

1. INTRODUCCIÓN

Java ha emergido como uno de los lenguajes mas populares gracias a su independencia de plataforma. Puede usar Java para crear applets que se ejecuten en páginas Webs, escribir JSPs(Java Server Pages) o desarrollar aplicaciones solas. Java siempre utiliza JDBC(Java Database Connectivity) para conectarse a una base de datos. Si escribe el código de manera adecuada y se adhiere al estandar SQL puede ser fácil pasarlo de una plataforma a otra.

Un driver JDBC es requerido para que los programas en Java o JSP puedan accesar a informacioón de las bases de datos.

Para descargar el driver vaya a la dirección:
http://download.microsoft.com/download/d/2/e/d2e1ffb6-2cfa-4a62-a22d-a413cce93118/Download_SQL_JDBC_Driver.htm

2. OBJETIVO (COMPETENCIA)

Realizar una conexion al servidor de bases de datos SQL Server con Java o JSP

Formuló	Revisó	Aprobó	Autorizó
M.C. Monceni Anabel Perez	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3. FUNDAMENTO

El Driver JDBC no es parte de Java SDK y debe establecerse en el classpath e incluir el archivo sqljdbc.jar si quiere usarlo. De lo contrario su aplicación dirá "Class not found" exception. El sqljdbc.jar esta instalado en la siguiente ubicación:
<*installation directory*>\sqljdbc_1.0\<loc>\sqljdbc.jar

El siguiente es un ejemplo de CLASSPATH usado en una aplicación Windows:
CLASSPATH =.;C:\Program Files\Microsoft SQL Server 2005 JDBC Driver\sqljdbc_1.0\enu\sqljdbc.jar

Si nota que los espacios le entre los folders le causan problemas, mueva el archivo .jar a un directorio mas sencillo.

Para conectarse a una base de datos usando el Driver Manager class, primero debe registrar el driver de la siguiente manera:

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

Una vez que el driver haya sido cargado, puede establecer una conexión usando el URL:
Connection con DriverManager.getConnection("jdbc:sqlserver://localhost;user=

MyUserName;password=***");**

En el ejemplo inferior definimos primero el paquete que contendrá la clase, después se importan dos clases estandar de Tomcat. Después se define la clase como pública con el nombre *ConnectionCreator*, luego definimos el único método de la clase *getSqlServerConnection* que recibirá como parámetro una lista de valores, para devolver luego el objeto *java.sql.connection.*

Los parámetros que recibirá la clase serán database(nombre de la base de datos), servername(nombre o ip del servidor de base de datos), port(puerto de conexión, por lo general en SQL Server, 1433), username y password.

Un ejemplo completo sería:

```
package notas;

import java.sql.DriverManager;
import java.sql.Connection;

public class ConnectionCreator {

  public static java.sql.Connection  getSqlServerConnection (String database,
    String servername, int port, String username, String password)  {
    try {
      Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
      String url  = "jdbc:microsoft:sqlserver://" + servername + ":" + port +
      ";DatabaseName=" + database + ";user=" +username + ";password=" + password;
    Connection conn = DriverManager.getConnection(url);
    if (conn != null)
      System.out.println(" --> CONECTANDO AL SERVIDOR : "+servername);
    else
       System.out.println("--> NO ES POSIBLE CONECTARSE AL SERVIDOR : "+servername);

    return conn;
}
catch (Exception e) {
        System.out.println( " ERROR = " +e);
        return null;
}
```

// Para usarlo dentro de una página JSP, anote lo siguiente:

```
<%@ page language="java" %>
<%@ page import = "notas.ConnectionCreator"%>
<%@ page import = "java.sql.Connection"%>

<%
Connection miConexion = ConnectionCreator.getSqlServerConnection("BD", "127.0.0.1",1433,"sa",
"");

if (!miConexion.isClosed())
out.print("FUNCIONA !");

/* CUERPO DE LA PAGINA */

miConexion.close(); /* no olvidarse de cerrar las conexiones. */

%>
```

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL DE APOYO

Computadoras con Java y SQL Server Instalados

Driver de conexión jdbc SQL Server

B) DESARROLLO DE LA PRÁCTICA

Haga una conexión entre Java o JSP con SQL Server a la base de datos “Ventas” creada en la práctica pasada.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS