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
Ingeniero en Computación	2009-2	12111	Base de Datos

PRÁCTICA No.	LABORATORIO DE	Base de Datos	DURACIÓN (HORAS)
4	NOMBRE DE LA PRÁCTICA	Consulta de INSERCION Y SELECCION	2

1. INTRODUCCIÓN

La importancia actual de las Bases de Datos se debe al auge del manejo de grandes volúmenes de información.

El SQL es el lenguaje estándar ANSI/ISO para la definición, manipulación y control de bases de datos relacionales. Es un lenguaje declarativo y, muy parecido al lenguaje natural; concretamente, se parece al inglés. Por estas razones, SQL es el lenguaje estándar usado en sistemas relacionales comerciales.

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, este lenguaje nos ofrece sentencias para la manipulación de datos como **INSERT** y **SELECT** que se estudian en esta práctica.

2. OBJETIVO (COMPETENCIA)

El alumno conocerá y utilizara instrucciones básicas de consulta del lenguaje SQL sobre una base de datos real.

3. FUNDAMENTO

Instrucción INSERT.

La instrucción **INSERT**, inserta nuevos registros en una tabla existente. Existen diferentes formas en que la instrucción puede ser utilizada, estas son:

- INSERT... VALUES
- INSERT... SELECT
- INSERT... SET

La primera y la última, insertan registros basados en valores explícitamente especificados. La segunda, inserta registros seleccionados de otra tabla o tablas.

En su forma **INSERT… VALUES**, es posible escribir la instrucción utilizando dos formas.

- La primera forma no especifica los nombres de columna en la que los datos se insertan, sólo sus valores. En este caso, se debe listar un valor para cada uno de los atributos que contiene la tabla por registro.

```
INSERT INTO nombre_Tabla VALUES (value1, value2, value3,...);
```

Ejemplo 1: Tenemos la siguiente tabla llamada Persona.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
3	López	Alicia	Independencia 14	Tijuana

Ejemplo 2: Para insertar un nuevo registro en Persona, utilizamos la siguiente instrucción:

```
INSERT INTO Persona VALUES  (4,'Navarro', 'Pablo', 'Cetys S/N', 'Mexicali');
```

Se afecta a la tabla Persona quedando como se muestra a continuación:

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
3	López	Alicia	Independencia 14	Tijuana
4	Navarro	Pablo	Cetys S/N	Mexicali

- La segunda forma especifica tanto los nombres de las columnas como los valores a insertar. En este caso, es necesario especificar los atributos para los cuales se está dando la lista de valores.

```
INSERT INTO nombre_Tabla (atributo1, atributo2, atributo3,...)
VALUES (value1, value2, value3,...);
```

Ejemplo 3: La siguiente sentencia SQL permite añadir una nueva fila o registro a la tabla Persona, pero sólo añadir datos en los campos "P_id", "Apellido" y "Nombre".

```
INSERT INTO Persona (P_Id, Apellido, Nombre)
VALUES (5, 'Martinez', 'Laura');
```

Se afecta a la tabla Persona de forma que ahora se verá como se muestra a continuación. Observe como los atributos Dirección y Ciudad se pusieron en NULL ya que en el INSERT no se especifico valor para ellos.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
3	López	Alicia	Independencia 14	Tijuana
4	Navarro	Pablo	Cetys S/N	Mexicali
5	Martínez	Laura	NULL	NULL

En su forma **INSERT… SELECT**, es posible insertar rápidamente varios registros en una tabla desde una o varias tablas.

```
INSERT INTO nombre_tabla (atributo1, atributo2, atributo3,...)
SELECT tbl_temp. atributo1, tbl_temp. atributo2, tbl_temp. atributo3
FROM tbl_temp WHERE tbl_temp.atributo = expr;
```

Ejemplo 4: Supongamos que además de la tabla Persona tenemos la tabla Empleado y que ambas comparten el mismo esquema como se muestra a continuación.

E_id	Apellido	Nombre	Dirección	Ciudad
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
12	Flores	Brenda	Revolución S/N	Tijuana
13	Herrera	Eva	Windows S/N	Caléxico
14	Rodríguez	Marcela	Arista S/N	Mexicali

Ejemplo 5: La siguiente sentencia SQL permite añadir simultáneamente a la tabla Persona, todos los registros de la tabla Empleado que viven en la ciudad de Mexicali, sin tener que hacer varias instrucciones INSERT para lograrlo.

```
INSERT INTO Persona (P_Id,Apellido,Nombre, Direccion,Ciudad)
SELECT Empleado.E_id, Empleado.Apellido, Empleado.Nombre, Empleado.Direccion, Empleado.Ciudad
FROM Empleado WHERE Empleado.Ciudad="Mexicali";
```

Después de ejecutar la instrucción anterior, la tabla Persona se afecta agregando todos los registros de la tabla Empleado que cumplieron con la condición como se muestra a continuación.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
3	López	Alicia	Independencia 14	Tijuana
4	Navarro	Pablo	Cetys S/N	Mexicali
5	Martínez	Laura	NULL	NULL
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
14	Rodríguez	Marcela	Arista S/N	Mexicali

En la forma **INSERT… SET**, es posible insertar nuevos registros o actualizar registros existentes en una tabla. Su sintaxis es:

```
INSERT INTO nombre_tabla
SET atributo=expr, . . .
[ON DUPLICATE KEY UPDATE atributo=expr, . . .];
```

Ejemplo 6: Supongamos que a la tabla Empleado queremos agregar el registro con E_id = 20 y con Direccion = “UABC”, los demás campos no se llenaran.

```
INSERT INTO Empleado SET E_id=20, Direccion="UABC"
ON DUPLICATE KEY UPDATE Direccion="YA EXISTE";
```

Al ejecutar la instrucción anterior, se agregara a la tabla Empleado un registro con E_id=20 y Direccion=”UABC” y con los campos faltantes en NULL ya que no fueron especificados. A continuación se muestra el resultado de la ejecución de la instrucción anterior.

E_id	Apellido	Nombre	Dirección	Ciudad
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
12	Flores	Brenda	Revolución S/N	Tijuana
13	Herrera	Eva	Windows S/N	Caléxico
14	Rodríguez	Marcela	Arista S/N	Mexicali
20	NULL	NULL	UABC	NULL

Ejemplo 7: Si ejecutamos la misma instrucción de nuevo, podemos observar que debido a que el E_id 20 ya existe, entonces las instrucción lo que haces es actualizar el registro escribiendo en el campo Direccion el texto “YA EXISTE” y eliminando el anterior como se muestra a continuación.

E_id	Apellido	Nombre	Dirección	Ciudad
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
12	Flores	Brenda	Revolución S/N	Tijuana
13	Herrera	Eva	Windows S/N	Caléxico
14	Rodríguez	Marcela	Arista S/N	Mexicali
20	NULL	NULL	YA EXISTE	NULL

Es importante mencionar que la parte de la instrucción **ON DUPLICATE KEY UPDATE campo=expr, . . .** es opcional y puede utilizarse en cualquiera de la formas de la instrucción INSERT.

Instrucción SELECT.

La instrucción **SELECT** se utiliza para seleccionar uno, varios o todos los registros de una o más tablas existentes. Existen diferentes clausulas que pueden utilizarse y que se muestran a continuación en la sintaxis de la instrucción.

```
SELECT  atributo1,atributo2, ...
[FROM  nombre_tabla1, nombre_tabla2, ...
[WHERE  condicion]
[GROUP BY {atributos de agrupacion}]
[HAVING condicion de agrupacion]
[ORDER BY {atributo de ordenacion} [ASC | DESC]]]
```

- Cada *atributo* indica una columna que se quiere obtener de la tabla.
- *nombre_tabla* indica la tabla o tablas desde la que se obtendrán los registros. Si nombra más de una tabla, está realizando un join.
- *condición* indica la condición o condiciones que deben satisfacer los registros para ser seleccionados.

Todas las cláusulas usadas deben darse exactamente en el orden mostrado en la descripción de la sintaxis. Por ejemplo, una cláusula **HAVING** debe ir tras cualquier cláusula **GROUP BY** y antes de cualquier cláusula **ORDER BY**.

Ejemplo 8: Supongamos que queremos obtener solo los atributos Nombre, Apellido y Ciudad de la tabla Persona, hacemos lo siguiente:

```
SELECT Nombre, Apellido, Ciudad FROM Persona;
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

Nombre	Apellido	Ciudad
Kenneth	Morgan	Caléxico
Gloria	Chávez	Mexicali
Alicia	López	Tijuana
Pablo	Navarro	Mexicali
Laura	Martínez	NULL
Cecilia	Curlango	Mexicali
Jorge	Ibarra	Mexicali
Marcela	Rodríguez	Mexicali

Ejemplo 9: Supongamos ahora que queremos obtener todos los atributos de la tabla Empleado:

```
SELECT * FROM Empleado;
```

Al ejecutarse la instrucción anterior sobre la tabla Empleado, obtenemos lo siguiente como resultado.

E_id	Apellido	Nombre	Dirección	Ciudad
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
12	Flores	Brenda	Revolución S/N	Tijuana
13	Herrera	Eva	Windows S/N	Caléxico
14	Rodríguez	Marcela	Arista S/N	Mexicali
20	NULL	NULL	YA EXISTE	NULL

Ejemplo 10: Supongamos que queremos obtener todas las ciudades de los registros de la tabla Empleado sin duplicados, hacemos lo siguiente:

```
SELECT DISTINCT Ciudad FROM Empleado;
```

Al ejecutarse la instrucción anterior sobre la tabla Empleado, obtenemos lo siguiente como resultado.

Ciudad
Mexicali
Tijuana
Caléxico
NULL

Ejemplo 11: Supongamos que queremos obtener todos los Nombres, Apellidos y Direcciones de los empleados que viven en Mexicali de la tabla Empleado, hacemos lo siguiente.

```
SELECT Nombre, Apellido, Direccion FROM Empleado
WHERE Ciudad=”Mexicali”;
```

Al ejecutarse la instrucción anterior sobre la tabla Empleado, obtenemos lo siguiente como resultado.

Nombre	Apellido	Dirección
Cecilia	Curlango	Independencia S/N
Jorge	Ibarra	Colon S/N
Marcela	Rodríguez	Arista S/N

Ejemplo 12: Supongamos que queremos obtener toda la información de la persona llamada Pablo Navarro de la tabla Persona, hacemos lo siguiente.

```
SELECT * FROM Persona
WHERE Nombre=”Pablo” AND Apellido=”Navarro”;
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
4	Navarro	Pablo	Cetys S/N	Mexicali

Ejemplo 13: Supongamos que queremos obtener toda la información de la persona llamada Cecilia o Kenneth de la tabla Persona, hacemos lo siguiente.

```
SELECT * FROM Persona
WHERE Nombre=”Cecilia” OR Nombre=”Kenneth”;
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
10	Curlango	Cecilia	Independencia S/N	Mexicali

Ejemplo 14: Supongamos que queremos obtener toda la información de la persona con apellido Morgan y que se llame Cecilia o Kenneth de la tabla Persona, hacemos lo siguiente.

```
SELECT * FROM Persona
WHERE Apellido=”Morgan”
AND (Nombre=”Cecilia” OR Nombre=”Kenneth”);
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico

El operador **LIKE** es usado en forma conjunta con la clausula **WHERE** para realizar la búsqueda de un patrón específico en una columna en particular y obtener los registros que coincidan con ese patrón. La sintaxis para este operador se muestra a continuación.

```
SELECT atributo1, atributo2, … FROM nombre_Tabla
WHERE atributo LIKE patron;
```

Ejemplo 15: Supongamos que queremos obtener de la tabla Persona, el Numero de Identificacion, Nombre y Ciudad de todas las personas que vivan en una ciudad que su nombre inicie con “Ca”, hacemos lo siguiente:

```
SELECT P_id, Nombre, Ciudad
FROM Persona WHERE Ciudad LIKE "Ca%";
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico

Se puede utilizar el signo **"%"** para definir comodines (las letras que faltan en el patrón), tanto antes como después del patrón.

Ejemplo 16: Supongamos que queremos obtener de la tabla Persona, toda la información de todas las Personas que vivan en una ciudad que en su nombre contenga “xi”, hacemos lo siguiente:

```
SELECT * FROM Persona WHERE Ciudad LIKE "%xi%";
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
1	Morgan	Kenneth	Timoteivn 10	Caléxico
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
4	Navarro	Pablo	Cetys S/N	Mexicali
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
14	Rodríguez	Marcela	Arista S/N	Mexicali

Ejemplo 17: Es posible también seleccionar a las personas que viven en una ciudad que no contiene el patrón "xi" de la tabla Persona, mediante el uso de la palabra clave **NOT** haciendo lo siguiente:

```
SELECT * FROM Persona WHERE Ciudad NOT LIKE "%xi%";
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
3	López	Alicia	Independencia 14	Tijuana

Podemos utiliza la palabra clave **ORDER BY** para realizar la ordenación ascendente o descendente de los registros seleccionados en base a un atributo o varios atributos. La sintaxis para este operador se muestra a continuación.

```
SELECT atributo1, atributo2, …
FROM nombre_Tabla ORDER BY atributo ASC | DESC;
```

Ejemplo 18: Supongamos que queremos obtener todos los registros de la tabla Persona ordenados ascendentemente por el Apellido, hacemos lo siguiente:

```
SELECT * FROM Persona ORDER BY Apellido;
```

Al ejecutarse la instrucción anterior sobre la tabla Persona, obtenemos lo siguiente como resultado.

P_id	Apellido	Nombre	Dirección	Ciudad
2	Chávez	Gloria	Blvd. B. Juárez	Mexicali
10	Curlango	Cecilia	Independencia S/N	Mexicali
11	Ibarra	Jorge	Colon S/N	Mexicali
3	López	Alicia	Independencia 14	Tijuana
5	Martínez	Laura	NULL	NULL
1	Morgan	Kenneth	Timoteivn 10	Caléxico
4	Navarro	Pablo	Cetys S/N	Mexicali
14	Rodríguez	Marcela	Arista S/N	Mexicali

Se puede dar a una tabla o una columna otro nombre mediante el uso de un **ALIAS**. Esto puede ser una buena opción si se tienen nombres de tablas o de columnas muy largas o complejas. Un nombre de alias puede ser cualquiera, aunque generalmente se utiliza un nombre corto.

Sintaxis para asignar un alias para una tabla.

```
SELECT atributo1, atributo2,…
FROM nombre_Tabla AS alias_Tabla;
```

Ejemplo 19: Supongamos que tenemos la tabla Ventas que se muestra a continuación.

V_id	OrdenNo	P_id
1	0001	1
2	0002	1
3	0003	1
4	0005	3
5	0010	4
6	0011	3
7	0051	1

Ahora queremos obtener todas las órdenes hechas por Kenneth Morgan. Como podemos observar, tenemos que hacer una consulta sobre dos tablas simultáneamente, esto lo haremos utilizando un alias para cada tabla, seleccionado los datos que nos interesa y estableciendo una condición para solo seleccionar las ordenes que pertenecen a Kenneth Morgan como se muestra a continuación:

```
SELECT ve.OrdenNo, p.Apellido, p.Nombre FROM Persona AS p, Ventas AS ve WHERE p.Apellido="Morgan" AND ve.P_id=p.P_id;
```

Al ejecutarse la instrucción anterior sobre las tablas Persona y Ventas, obtenemos lo siguiente como resultado.

OrdenNo	Apellido	Nombre
0001	Morgan	Kenneth
0002	Morgan	Kenneth
0003	Morgan	Kenneth
0051	Morgan	Kenneth

Sintaxis para asignar un alias para una columna.

```
SELECT atributo1 AS alias_Atributo
FROM nombre_Tabla;
```

Un *atributo* puede tener un alias usando **AS alias_name**. El alias se usa como el nombre de columna de la expresión y puede usarse en cláusulas **GROUP BY**, **ORDER BY**, o **HAVING**.

Ejemplo 20: Supongamos que queremos mostrar los nombre completos de todas las personas de la tabla Empleado utilizando el formato de Nombre y Apellido representados como un solo campo llamado NombreCompleto, hacemos lo siguiente.

```
SELECT CONCAT(Nombre, ', ', Apellido) AS NombreCompleto
FROM Empleado ORDER BY NombreCompleto;
```

Al ejecutarse la instrucción anterior sobre la tabla Empleado, obtenemos lo siguiente como resultado.

NombreCompleto
NULL

Brenda, Flores
Cecilia, Curlango
Eva, Herrera
Jorge, Ibarra
Marcela, Rodríguez

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL

Computadora con MySQL instalado.

B) DESARROLLO DE LA PRÁCTICA

1. Crear una Base de datos con el nombre practica4_XXXXX, donde XXXXX representan el numero de tu matricula.

2. Crea la tabla con el nombre **Persona** a la que se hace referencia en el **Ejemplo 1** de la fundamentación utilizando las características que se describen a continuación:

Atributo	Tipo	Observación
P_id	Entero	Llave Primaria
Apellido	Cadena	Longitud 20
Nombre	Cadena	Longitud 20
Direccion	Cadena	Longitud 30
Ciudad	Cadena	Longitud 20

3. Crea la tabla con el nombre **Empleado** a la que se hace referencia en el **Ejemplo 4** de la fundamentación utilizando las características que se describen a continuación:

Atributo	Tipo	Observación
E_id	Entero	Llave Primaria
Apellido	Cadena	Longitud 20
Nombre	Cadena	Longitud 20
Direccion	Cadena	Longitud 30
Ciudad	Cadena	Longitud 20

4. Crea la tabla con el nombre **Ventas** a la que se hace referencia en el **Ejemplo 19** de la fundamentación utilizando las características que se describen a continuación:

Atributo	Tipo	Observación
V_id	Entero	Llave Primaria
OrdenNo	Cadena	Longitud 20
P_id	Entero	

5. Realiza cada uno de los **Ejemplos** de la fundamentación excepto los que ya hiciste en los pasos anteriores y comprueba su funcionalidad. En cada ejemplo que vayas haciendo, consulta las tablas afectada para que compares los resultados con los que se muestran en los ejemplos de esta práctica.

6. En el **Ejemplo 18**, como le harías para obtener la misma información pero ordenada descendentemente por el Apellido?

7. Muestra todos los datos de la tabla Persona, ordenado descendentemente por la ciudad.

8. Muestra todos los datos de la tabla Persona, ordenado descendentemente por la ciudad y descendentemente por la llave.

9. Obtén un listado que contenga el Nombre, Apellido y Orden para cada una de las órdenes que tiene la tabla Ventas.

10. Investiga el uso y sintaxis del operador BETWEEN y el operador IN, encuentra una forma de aplicarlas sobre las tablas que tienes en la Base de Datos que creaste para esta práctica y anota las consultas que realizaste.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

[1] Manual de Referencia MySQL: http://dev.mysql.com/doc/refman/5.5/en/

[2] Sintaxis de Instrucciones SQL: http://dev.mysql.com/doc/refman/5.5/en/sql-syntax.html

[3] Tutorial de MySQL: http://dev.mysql.com/doc/refman/5.5/en/tutorial.html

M.C. Pablo M. Navarro			
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