LENGUAJE ESTRUCTURADO DE CONSULTAS SQL (2)

NOTAS DE LA MATERIA BASES DE DATOS I

LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE SONORA

MAESTRA: DRA. MARÍA DE GUADALUPE COTA ORTIZ

EJERCICIO 1

```
Create database `inform`;

CREATE TABLE `t1` (
  `id` int(11) NOT NULL auto_increment,
  `campo1` int(11) default NULL,
  PRIMARY KEY  (`id`)
)

CREATE TABLE `t2` (
  `id` int(11) NOT NULL auto_increment,
  `campo2` int(11) default NULL,
  PRIMARY KEY  (`id`)
)
```

INSERCIONES

```
INSERT INTO `t1` VALUES (1,10);
INSERT INTO `t1` VALUES (2,20);
INSERT INTO `t1` VALUES (3,30);
INSERT INTO `t1` VALUES (4,40);
INSERT INTO `t1` VALUES (5,50);
INSERT INTO `t1` VALUES (6,80);
INSERT INTO `t1` VALUES (7,120);

INSERT INTO `t2` VALUES (1,10);
INSERT INTO `t2` VALUES (2,2);
INSERT INTO `t2` VALUES (3,10);
INSERT INTO `t2` VALUES (4,20);
```

RESULTADO

```
1  select * from t1
2
3
4
5
```

id	campo1
1	10
2	20
3	30
4	40
5	50
6	80
7	120

```
1  select * from t2
2
3
4
5
```

id	campo2
1	10
2	2
3	10
4	20

ACTUALIZAR DATOS

update t1 inner join t2 on t1.campo1 = t2.campo2 set t1.campo1 = 0 where t1.id = t2.id

Valores originales:

```
1 select * from t1
2
3
4
5
```

id	campo1
1	10
2	20
3	30
4	40
5	50
6	80
7	120

Valores modificados:

```
1 select * from t1
2
3
4
5
6
```

id	campo1
1	0
2	20
3	30
4	40
5	50
6	80
7	120

Instrucción alternativa:

update t1 set campo1 = 0
where campo1 in(select campo2 from t2 where t1.id = t2.id)

EJERCICIO 2

a) Seleccionar la base de datos 'nwind': Use 'nwind';

b) Crear una réplica de la estructura de la tabla 'Clientes' y nombrarla como 'temp_clientes':

```
CREATE TABLE `temp_clientes` (
    `IdCliente` varchar(5) default NULL,
    `NombreCompania` varchar(40) default NULL,
    `NombreContacto` varchar(30) default NULL,
    `CargoContacto` varchar(30)  default NULL,
    `Direccion` varchar(60) collate latin1_general_ci default NULL,
    `Ciudad` varchar(15) collate latin1_general_ci default NULL,
    `Region` varchar(15) collate latin1_general_ci default NULL,
    `CodPostal` varchar(10) collate latin1_general_ci default NULL,
    `Pais` varchar(15) collate latin1_general_ci default NULL,
    `Telefono` varchar(24) collate latin1_general_ci default NULL,
    `Fax` varchar(24) collate latin1_general_ci default NULL,
    UNIQUE KEY `PrimaryKey` (`IdCliente`),
    KEY `Ciudad` (`Ciudad`),
    KEY `CódPostal` (`CodPostal`),
     KEY `NombreCompañía` (`NombreCompania`),
    KEY `Region` (`Region`)  );
```

c) Insertar valores en la tabla:

```
insert into temp_clientes select * from clientes where isnull(region);
```

Crear vista de la tabla Clientes

a) Crear vista ‘vista_clientes1’:

```
CREATE VIEW vista_clientes1 AS SELECT * FROM
temp_clientes;
```

b) Realizar consulta para verificar contenido de la vista creada:

```
select * from vista_clientes1;
```

c) Actualizar el campo region con valor NULL con valor ‘DESCONOCIDA’:

```
UPDATE vista_clientes1 set region = 'DESCONOCIDA';
```

d) Realizar consulta para verificar contenido de la vista:

```
select * from vista_clientes1;
```

(Notar que el campo region que antes tenia valores NULL ahora tiene el valor 'DESCONOCIDA')

Modificar estructura lógica de la vista 'temp_clientes'

a) Modificar estructura eligiendo solamente dos campos:

```
ALTER VIEW vista_clientes1 as
select IdCliente, Region from clientes;
```

b) Realizar consulta para verificar cambios:

```
Select * from vista_clientes1;
```

(Notar que la tabla temp_clientes solamente tiene dos campos y el campo region no perdió el valor después de modificar la vista)

c) Modificar la estructura de la vista nuevamente:

```
alter VIEW vista_clientes1 AS SELECT * FROM temp_clientes;
```

d) Eliminar 'vista_clientes1':

```
drop view vista_clientes1;
```

Crear Procedimiento Almacenado

a) Crear Procedimiento Almacenado ‘muestra_clientes) que recibe de entrada un valor como argumento para realizar la consulta:

```
CREATE PROCEDURE muestra_clientes(regionE varchar(20))
BEGIN
  SELECT NombreContacto FROM clientes where region = regionE;
END;
CREATE PROCEDURE `muestra_clientes`(regionE varchar(20))
BEGIN
  SELECT NombreContacto collate latin1_general_ci  FROM clientes
  where region collate latin1_general_ci = regionE collate
  latin1_general_ci;
END;
```

b) Invocar el Procedimiento Almacenado:

```
call muestra_clientes('BC');
```

c) Eliminar el procedimiento almacenado:

```
drop procedure if exists muestra_clientes;
```

Crear Procedimiento Almacenado

a) Crear procedimiento:

```
CREATE PROCEDURE num_clientes(regionE
        varchar(50))
BEGIN
        select count(*) into cuantos_son from clientes
        where region collate latin1_general_ci =
        regionE  collate latin1_general_ci;
END
```

b) Invocar el procedimiento:

```
CALL num_clientes('BC');
```

c) Eliminar procedimiento validando que exista:

```
drop procedure if exists num_clientes;
```

Crear procedimiento

a) Crear procedimiento:

```
CREATE PROCEDURE contar(OUT param1 INT)
BEGIN
        SELECT COUNT(*) INTO param1 FROM
        clientes where region = 'BC';
END;
```

b) Invocar procedimiento:

```
CALL contar(@param1);
```

c) Consultar valor almacenado en variable en memoria;

```
select @param1 as 'total';
```

Ejercicio Procedimiento Almacenado

a) Crear tabla 'tempo2':

```
CREATE TABLE `tempo2` (
  `id` int(11) NOT NULL auto_increment,
  `nombre` varchar(50) collate latin1_general_ci default NULL,
  `apellido_p` varchar(50) collate latin1_general_ci default NULL,
  `apellido_m` varchar(50) collate latin1_general_ci default NULL,
  `salario` double(8,2) default NULL,
  PRIMARY KEY  (`id`)
);
```

b) Crear procedimiento almacenado:

```
CREATE PROCEDURE `proc`(OUT param VARCHAR(50), OUT param1
    VARCHAR(50),OUT param1b VARCHAR(50),IN param2 INT)
BEGIN
   INSERT INTO tempo2 VALUES(param2,'Alejandro','Gomez','Durazo',20000.00);
   SELECT nombre, apellido_p, apellido_m INTO param, param1, param1b FROM
    tempo2 WHERE id = param2;
END;
```

Ejercicio Procedimiento Almacenado

c) Invocar procedimiento:

```
CALL proc(@Nombre,@Apellido_p,@Apellido_m, 1);
```

d) Revisar resultados:

```
select @Nombre, @Apellido_p, @Apellido_m
```

e) Revisar resultados concatenando:

```
select @Nombre as 'Nombre', concat(@Apellido_p, ' ',
    @Apellido_m) as 'Apellido'
```

Uso de ciclos en procedimientos almacenados

a)Crear procedimiento:

```
CREATE PROCEDURE repetir(p1 INT)
BEGIN
 SET @x = 0;
  REPEAT
        SET @x = @x + 1;
        UNTIL @x > p1
  END REPEAT;
END
```

b) Invocar procedimiento:

```
CALL repetir(1000);
SELECT @x;
```

Ejercicio 3 (Revisar resultados)

```
CREATE TABLE t (qty INT, price INT);
 INSERT INTO t VALUES(3, 50), (5, 60);
 CREATE VIEW v AS SELECT qty, price, qty*price AS
   value FROM t;
 SELECT * FROM v;
 UPDATE v set qty = 8 where qty = 3;
 SELECT * FROM v;
 SELECT * FROM v WHERE qty = 5;
 ALTER VIEW v as select qty, price from t;
 UPDATE v set qty = 3 where qty = 8;
 DROP VIEW v;
 DROP VIEW B;
```

Ejercicio 4 (Revisar resultados)

```
CREATE PROCEDURE simple (OUT param1 INT)
BEGIN
    SELECT COUNT(*) INTO param1 FROM t;
END;

CALL simple(@param1);
SELECT @param1;
SELECT @param1 as 'Resultado';
```

SALIDA: 3

Ejercicio 5 (Revisar resultados)

```
CREATE FUNCTION hola (s CHAR(20))
   RETURNS CHAR(50) DETERMINISTIC
        RETURN CONCAT('Hola ', s ,'!');

SELECT hola('mundo');
```

SALIDA: Hola mundo!

Ejercicio 6 (Revisar resultados)

```
CREATE TABLE test1(a1 INT);
CREATE TABLE test2(a2 INT);
CREATE TABLE test3(a3 INT NOT NULL
   AUTO_INCREMENT PRIMARY KEY);
CREATE TABLE test4(
  a4 INT NOT NULL AUTO_INCREMENT
   PRIMARY KEY,
  b4 INT DEFAULT 0);
```

Ejercicio 6 (CREAR TRIGGER)

a) Crear disparador (TRIGGER):

```
CREATE TRIGGER testref BEFORE INSERT ON test1
  FOR EACH ROW BEGIN
    INSERT INTO test2 SET a2 = NEW.a1;
    DELETE FROM test3 WHERE a3 = NEW.a1;
    UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
  END;
```

b) Insertar valores para probar el disparador:

```
  INSERT INTO test3 (a3) VALUES
    (NULL), (NULL), (NULL), (NULL), (NULL),
    (NULL), (NULL), (NULL), (NULL), (NULL);
 INSERT INTO test4 (a4) VALUES
    (0), (0), (0), (0), (0), (0), (0), (0), (0), (0);
 INSERT INTO test1 VALUES (1), (3), (1), (7), (1), (8), (4), (4);
```

Ejercicio 6 (Revisar resultados)

SELECT * FROM test1;

SALIDA:
1
3
1
7
1
8
4
4

SELECT * FROM test2;

SALIDA:
1
3
1
7
1
8
4
4

Ejercicio 6 (Revisar resultados)

SELECT * FROM test3;

SALIDA:

2
5
6
9
10

SELECT * FROM test4;

SALIDA:

1	3
2	0
3	1
4	2
5	0
6	0
7	1
8	1
9	0
10	0

Otras instrucciones de Sql

```
select user(), current_date()
SHOW PROCESSLIST;   #VISUALIZA USUARIOS CONECTADOS
SHOW DATABASES; #VISUALIZA LISTA DE BASES DE DATOS
show tables;              #VISUALIZA LISTA DE TABLAS
describe clientes;        #VISUALIZA ESTRUCTURA LOGICA
explain clientes;         #VISUALIZA ESTRUCTURA LOGICA
```

INSTRUCCION PARA CARGAR DATOS EN UNA TABLA DESDE UN ARCHIVO DE TEXTO QUE UTILIZA COMO SEPARADOR EL TABULADOR Y ESTA ORGANIZADO POR LINEAS:

```
LOAD DATA LOCAL INFILE '/datos_prueba.txt' INTO TABLE t1
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
```

Exportar datos de MySql a un archivo externo (Ejercicio 7)

a) Crear tabla temporal en ‘nwind’:

```
CREATE TABLE `temporal` (
  `idProveedor` int(11) default NULL,
  `compañia` varchar(50) collate latin1_general_ci default NULL,
  `contacto` varchar(50) collate latin1_general_ci default NULL,
  `idProducto` int(11) default NULL,
  `nombreProducto` varchar(50) collate latin1_general_ci default
   NULL,
  `unidadExistencia` int(11) default NULL,
  `UnidadPedido` int(11) default NULL,
  `costo` int(11) default NULL
)
```

Nombre	Tipo	NULL	Por ...
idProveedor	int(11)	Sí	<NU...
compañia	varchar(50)	Sí	<NU...
contacto	varchar(50)	Sí	<NU...
idProducto	int(11)	Sí	<NU...
nombreProducto	varchar(50)	Sí	<NU...
unidadExistencia	int(11)	Sí	<NU...
UnidadPedido	int(11)	Sí	<NU...
costo	int(11)	Sí	<NU...

Exportar datos de MySql a un archivo externo (Ejercicio 7)

b) Insertar los valores regresados en la siguiente consulta en la tabla temporal creada en el inciso a).

```
insert into temporal
select r1.idproveedor, r1.nombrecompania,
r1.cargocontacto, r2.idproducto, r2.nombreproducto,
sum(r2.unidadesenexistencia) AS 'U en Existencia',
sum(r2.unidadesenpedido) as 'U en Pedido',
sum(format(r2.PrecioUnidad * unidadesenpedido, 2))
as 'Costo'
from proveedores as r1 left join productos as r2
on r1.idproveedor = r2.idproveedor
group by r2.IdProveedor, r2.idproducto with rollup
```

Exportar datos de MySql a un archivo externo (Ejercicio 7)

c) Insertar los valores de la tabla temporal en el archivo y ruta especificada

```
SELECT * INTO OUTFILE '/alumnobd/valores_consulta.txt'
FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n'
FROM temporal
```

Los valores se exportan separados por tabulador y en registros por línea.