Facultad: **Ingeniería**
Escuela: **Computación**
Asignatura: **Modelamiento y diseño de base de datos**

Guía N° 5

Tema: Consultas de manipulación de datos

I. Objetivos

Que el estudiante sea capaz de:

1. Agregar información a una o varias tablas almacenadas en una Base de datos.
2. Actualizar datos almacenados en una Base de datos tomando ciertos criterios
3. Eliminar datos o información innecesaria almacenada en una Base de datos

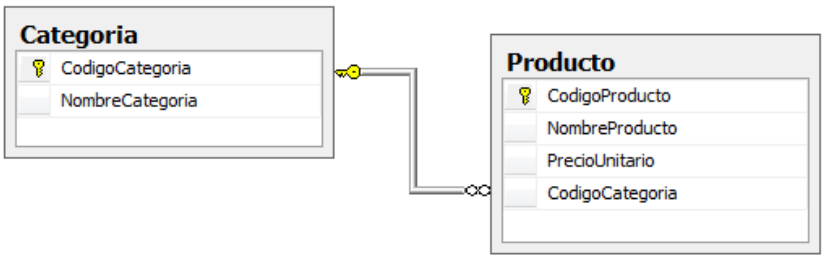
II. Introducción Teórica

Consultas de acción

Las consultas de acción son aquellas consultas que no devuelven ningún registro, sino que se encargan de acciones como:

- Agregar registros
- Actualizar registros
- Eliminar registros

Para comprender este tipo de consultas vamos a utilizar como ejemplo el siguiente diagrama de base de datos:



Creando la base de datos relacional:

Tabla: Categoria

```
CREATE TABLE Categoria(
CodigoCategoria int NOT NULL,
NombreCategoria varchar(50)
CONSTRAINT pk_categoria PRIMARY KEY(CodigoCategoria)
```

```
)
```

Tabla: Producto

```
CREATE TABLE Producto
(CodigoProducto int NOT NULL,
NombreProducto varchar(50),
PrecioUnitario decimal(18,2),
CodigoCategoria int
CONSTRAINT pk_producto PRIMARY KEY (CodigoProducto)
CONSTRAINT fk_categoria FOREIGN KEY (CodigoCategoria)
REFERENCES Categoria(CodigoCategoria)
)
```

Sentencia INSERT

INSERT. Permite agregar, adicionar o insertar uno o más registros a una (y solo una) tabla en una base de datos relacional.

Debe proporcionar en una sentencia INSERT igual número de campos o columnas.

Ejemplo 1:

Insertando sin colocar los nombres de los campos esto indica que se debe agregar datos a todas las columnas NOT NULL y se debe tomar en cuenta el orden de los campos de la tabla

```
INSERT INTO Categoria VALUES
(1,'Bebidas')
```

Si se van a proporcionar valores para todos los campos de una tabla pueden omitirse los nombres de dichos campos en la instrucción.

Ejemplo 2:

Colocando los nombres de los campos, no importa el orden de como estén los campos en la tabla

```
INSERT INTO Categoria (CodigoCategoria,NombreCategoria)VALUES
(2,'Carnes rojas')
```

```
INSERT INTO Categoria (NombreCategoria,CodigoCategoria)VALUES
('Harinas',3)
```

Ejemplo 3:

Agregando varios registros al mismo tiempo en la tabla

```
 INSERT INTO Categoria VALUES
 (4,'Vegetales'),
 (5,'Frutas'),
 (6,'Mariscos')
```

Ejemplo 4:

En el siguiente ejemplo se agrega datos a la tabla Producto en donde se respeta la relación entre las tablas, quiere decir que no se puede agregar un código de categoría si este no ha sido ingresado previamente en la tabla donde se encuentre la clave primaria (en este caso en la tabla Categoria)

```
INSERT INTO Producto VALUES
 (1, 'Soda Coca Cola', 1.25,1),
 (2, 'Carne bistec',3.50,2),
 (3, 'Camarones pequeños',1.15,6),
 (4, 'Harina blanca',0.75,3),
 (5, 'Te verde',1.0,1),
 (6, 'Lomo de aguja',4.50,2),
 (7, 'Soda de naranja',1.25,1),
 (8, 'Chiles verdes',0.25,4),
 (9, 'Tomates',0.2,4),
 (10, 'Manzana verde',0.25,5)
```

Este dato indica el código de la categoría, siguiendo el ejemplo, aquí solo se pueden agregar datos del 1 al 6, los cuales son los datos almacenados en el campo CodigoCategoria de la tabla Categoria

Sentencia SELECT - INTO

SELECT - INTO se utiliza para crear una tabla a partir de los valores de otra tabla existente en la base de datos.

Ejemplo:

Se desea crear una tabla con los datos de la tabla producto que pertenezcan a la categoría bebidas

Al hacer un SELECT a la tabla Producto se tienen los siguientes resultados:

```
SELECT * FROM Producto
```

Resultado:

CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	Soda Coca Cola	1.25	1
2	Carne bistec	3.50	2
3	Camarones pequeños	1.15	6
4	Harina blanca	0.75	3
5	Te verde	1.00	1
6	Lomo de aguja	4.50	2
7	Soda de naranja	1.25	1
8	Chiles verdes	0.25	4
9	Tomates	0.20	4
10	Manzana verde	0.25	5

El código de la categoría Bebidas tiene el valor de 1

Ahora ejecutamos la siguiente consulta SELECT – INTO

```
SELECT * INTO [Producto CategoriaBebidas]
FROM Producto
WHERE CodigoCategoria=1
```

Después de ejecutar la sentencia SELECT INTO y al hacer un SELECT a la tabla Producto CategoriaBebidas, se obtienen los siguientes resultados

```
SELECT * FROM [Producto CategoriaBebidas]
```

Resultado:

CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	Soda Coca Cola	1.25	1
5	Te verde	1.00	1
7	Soda de naranja	1.25	1

Sentencia INSERT INTO – SELECT

La consulta SELECT de la instrucción INSERT se puede utilizar para agregar valores a una tabla de la base de datos.

Ejemplo:

Crear la tabla Producto CategoriaVegetales, con las mismas propiedades de la tabla Producto

```
CREATE TABLE [Producto CategoriaVegetales]
(CodigoProducto int NOT NULL,
NombreProducto varchar(50),
PrecioUnitario decimal(18,2),
CodigoCategoria int
CONSTRAINT pk_producto1 PRIMARY KEY (CodigoProducto)
CONSTRAINT fk_categoria1 FOREIGN KEY (CodigoCategoria)
REFERENCES Categoria(CodigoCategoria)
)
```

Al hacer un SELECT a la tabla Producto CategoriaVegetales esta no tiene datos

```
SELECT * FROM [Producto CategoriaVegetales]
```

En el siguiente ejemplo, la instrucción INSERT agregar en la tabla Producto CategoriaVegetales, los datos de la tabla Producto donde el valor del campo CodigoCategoria sea igual a 4

```
INSERT INTO [Producto CategoriaVegetales]
 SELECT CodigoProducto,NombreProducto,PrecioUnitario,CodigoCategoria
 FROM Producto
 WHERE CodigoCategoria=4
```

Al hacer un SELECT a la tabla esta debe tener los datos de los productos que pertenecen a la categoría con código igual a 1

```
SELECT * FROM [Producto CategoriaVegetales]
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	8	Chiles verdes	0.25	4
2	9	Tomates	0.20	4

Nota: a diferencia con la sentencia SELECT - INTO, es que aquí debe de crearse la tabla previamente

Sentencia UPDATE

UPDATE. Permite la actualización o modificación de uno o varios registros de una única tabla.

Se debe utilizar en conjunto con la cláusula **SET** con la cual se indicará(n) el(los) campo(s) a actualizar con el valor indicado.

Una segunda cláusula WHERE, opcional, permite indicar qué registros deben ser actualizados.

Si se omite la cláusula WHERE la ejecución de la consulta modificará todos los registros de la tabla.

Ejemplo 1. Actualizando datos a varios registros

Al hacer un SELECT a la tabla se obtiene los siguientes resultados:

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	5	Te verde	1.00	1
3	7	Soda de naranja	1.25	1

Con este ejemplo se actualiza el dato almacenado en el campo PrecioUnitario de cada registro de la tabla Producto CategoriaBebidas

```
UPDATE [Producto CategoriaBebidas] SET PrecioUnitario=1.50
```

Ahora hacer un SELECT a la tabla se obtiene los siguientes resultados:

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.50	1
2	5	Te verde	1.50	1
3	7	Soda de naranja	1.50	1

Ejemplo 2. Actualizando datos donde se cumpla una o varias condiciones

Una condición:

Actualiza el precio del producto donde el Codigo del producto sea igual 1

```
UPDATE [Producto CategoriaBebidas] SET PrecioUnitario=1.25
WHERE CodigoProducto=1
```

Ahora hacer un SELECT a la tabla se obtiene los siguientes resultados:

```
SELECT * FROM [Producto CategoriaBebidas]
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	5	Te verde	1.50	1
3	7	Soda de naranja	1.50	1

Varias condiciones:

Actualiza el precio del producto donde el Codigo del producto sea igual 1 y el codigo de la categoria sea igual 1

```
UPDATE [Producto CategoriaBebidas] SET PrecioUnitario=1.75
WHERE CodigoProducto=1 AND CodigoCategoria=1
```

Ahora hacer un SELECT a la tabla se obtiene los siguientes resultados:

```
SELECT * FROM [Producto CategoriaBebidas]
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.75	1
2	5	Te verde	1.50	1
3	7	Soda de naranja	1.50	1

Sentencia DELETE

DELETE. Permite eliminar o borrar todos los registros de una tabla.

La sentencia DELETE no borra la estructura física de la tabla únicamente elimina los datos.

Ejemplo 1:

Elimina todos los registros de la tabla Producto CategoriaBebidas

```
DELETE FROM [Producto CategoriaBebidas]
```

Ahora hacer un SELECT a la tabla se obtiene los siguientes resultados:

```
SELECT * FROM [Producto CategoriaBebidas]
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria

Ejemplo 2:

La tabla productos tiene los siguientes datos:

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	2	Carne bistec	3.50	2
3	3	Camarones pequeños	1.15	6
4	4	Harina blanca	0.75	3
5	5	Te verde	1.00	1
6	6	Lomo de aguja	4.50	2
7	7	Soda de naranja	1.25	1
8	8	Chiles verdes	0.25	4
9	9	Tomates	0.20	4
10	10	Manzana verde	0.25	5

Eliminar los registros de la tabla Producto donde el código de la categoría sea igual a 4

```
DELETE FROM Producto
WHERE CodigoCategoria=4
```

Ahora hacer un SELECT a la tabla se obtiene los siguientes resultados:

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	2	Carne bistec	3.50	2
3	3	Camarones pequeños	1.15	6
4	4	Harina blanca	0.75	3
5	5	Te verde	1.00	1
6	6	Lomo de aguja	4.50	2
7	7	Soda de naranja	1.25	1
8	10	Manzana verde	0.25	5

Ya no se encuentran los productos con código 8 y 9 en los registros obtenidos

ON DELETE CASDADE y ON UPDATE CASCADE

Debido a que el sistema de gestión de base de datos hace cumplir las restricciones de referencia, se debe garantizar la integridad de los datos, si las filas de la tabla de la clave principal se van a eliminar o van a ser actualizadas, el gestor verifica si todavía existen filas dependientes en tablas de claves foráneas, esas referencias tienen que ser consideradas.

ON DELETE CASCADE

Específica que si se intenta eliminar una fila con una clave primaria a la que hacen referencia claves foráneas de filas existentes en otras tablas, todas las filas que contienen dichas claves foráneas también se eliminan.

ON UPDATE CASCADE

Específica que si se intenta actualizar un valor de clave primaria de una fila a cuyo valor de clave hacen referencia claves foráneas de filas existentes en otras tablas, también se actualizan todos los valores que conforman la clave foránea al nuevo valor especificado para la clave primaria.

Ejemplo 1:

Hacer un SELECT a la tabla Categoria para verificar que registro se quiere eliminar

```
SELECT * FROM Categoria
```

Se obtienen los siguientes resultados:

	CodigoCategoria	NombreCategoria
1	1	Bebidas
2	2	Carnes rojas
3	3	Harinas
4	4	Vegetales
5	5	Frutas
6	6	Mariscos

Se quiere eliminar de la tabla la categoría donde el código sea igual a 6

```
DELETE FROM Categoria WHERE CodigoCategoria =6
```

Pero se obtiene el siguiente error:

```
Msg 547, Level 16, State 0, Line 1
The DELETE statement conflicted with the REFERENCE constraint "fk_categoria". The conflict occurred
in database "Guia7", table "dbo.Producto", column 'CodigoCategoria'.
The statement has been terminated.
```

Dicho error indica que existe una referencia externa con ese dato que queremos eliminar, para verificar hacemos un SELECT a la tabla Producto

```
SELECT * FROM Producto
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	2	Carne bistec	3.50	2
3	3	Camarones pequeños	1.15	6
4	4	Harina blanca	0.75	3
5	5	Te verde	1.00	1
6	6	Lomo de aguja	4.50	2
7	7	Soda de naranja	1.25	1
8	10	Manzana verde	0.25	5

Y exactamente tenemos un registro en la tabla Producto que hace referencia al dato con clave principal que queremos eliminar

Para no tener ese problema se debe agregar en la restricción de la clave foránea las sentencias ON DELETE CASCADE y ON UPDATE CASCADE

Eliminando la restricción **fk_categoria**

```
ALTER TABLE Producto
DROP CONSTRAINT fk_categoria
```

Agregando la restricción nuevamente pero ahora se adiciona al final de la restricción las sentencias **ON DELETE CASCADE y ON UPDATE CASCADE**

```
ALTER TABLE Producto
ADD CONSTRAINT fk_categoria
FOREIGN KEY (CodigoCategoria) REFERENCES Categoria (CodigoCategoria)
ON DELETE CASCADE
ON UPDATE CASCADE
```

Intentando nuevamente de eliminar el registro

```
DELETE FROM Categoria WHERE CodigoCategoria =6
```

Ahora ya no se debe tener problemas a eliminar el registro

Hacer un SELECT a la tabla Categoria, y verificamos que ya no existe el registro

```
SELECT * FROM Categoria
```

	CodigoCategoria	NombreCategoria
1	1	Bebidas
2	2	Carnes rojas
3	3	Harinas
4	4	Vegetales
5	5	Frutas

Ahora verificamos que también ya no existe el registro en la clave foránea

```
SELECT * FROM Producto
```

	CodigoProducto	NombreProducto	PrecioUnitario	CodigoCategoria
1	1	Soda Coca Cola	1.25	1
2	2	Carne bistec	3.50	2
3	4	Harina blanca	0.75	3
4	5	Te verde	1.00	1
5	6	Lomo de aguja	4.50	2
6	7	Soda de naranja	1.25	1
7	10	Manzana verde	0.25	5

Ejemplo 2:

También se puede agregar las sentencias ON DELETE CASCADE y ON UPDATE CASCADE a nivel de tabla, o sea cuando esta se está creando.

```
CREATE TABLE [Producto CategoriaFrutas]
(CodigoProducto int NOT NULL,
NombreProducto varchar(50),
PrecioUnitario decimal(18,2),
CodigoCategoria int
CONSTRAINT pk_producto2 PRIMARY KEY (CodigoProducto)
CONSTRAINT fk_categoria2 FOREIGN KEY (CodigoCategoria)
```

```
REFERENCES Categoria(CodigoCategoria)
ON DELETE CASCADE
ON UPDATE CASCADE
)
```

Recuerde que las sentencias ON DELETE CASCADE y ON UPDATE CASCADE se deben agregar en la restricción de la clave foránea

III. Requerimientos

- Máquina con SQL Server 2012, 2014 0 2016
- Guía Número 5 de Modelamiento y diseño de base de datos

IV. Procedimiento

Uso de las diferentes tipos de consultas de acción

Ejercicio 1. Creación de la base de datos

1. Crear la siguiente base de datos

```
CREATE DATABASE ControlAlumnoSuCarnet
GO
```

2. Hacer uso de la base de datos

```
USE ControlAlumnoSuCarnet
GO
```

Nota: cambiar la palabra **SuCarnet por su número de carnet**

Ejercicio 2. Creación de las tablas y las restricciones de campo

Tomando el siguiente diccionario de datos:

Tabla Alumno:

Nombre del campo	Tipo de dato	Tamaño	Permite valores nulos	Tipo de restricción
Carnet	Char	8	No	Llave primaria
NombreCompleto	Varchar	50	Si	

Tabla Materia:

Nombre del campo	Tipo de dato	Tamaño	Permite valores nulos	Tipo de restricción
Codigo	Char	5	No	Llave primaria
Nombre	Varchar	30	Si	Valor único, el nombre de la materia no se puede repetir
UV	Int		Si	Ckeck, en donde solo se aceptan valores entre 2 y 5 unidades

Tabla Inscripcion:

Nombre del campo	Tipo de dato	Tamaño	Permite valores nulos	Tipo de restricción	
Carnet	Char	8	No	La unión de los tres campos se crea una llave única	Llave Foránea la cual hace referencia a la tabla Alumno
CodigoMateria	Char	5	No		Llave Foránea la cual hace referencia a la tabla Materia
Ciclo	Char	5	No		

1. Crear las tablas de la base de datos

Crear la tabla Alumno

```
CREATE TABLE Alumno(
Carnet char(8) NOT NULL,
NombreCompleto varchar(50)
--creando la llave primaria de la tabla
CONSTRAINT pk_alumno PRIMARY KEY(Carnet)
)
```

Crear la tabla Materia

```
CREATE TABLE Materia(
Codigo char(5) NOT NULL,
Nombre varchar(30),
UV int
--Creando la llave primaria de la tabla
CONSTRAINT pk_materia PRIMARY KEY(Codigo),
--Creando una restriccion UNIQUE en el campo Nombre de la materia
CONSTRAINT u_nombre UNIQUE(Nombre),
--Creando una restriccion Check en donde las unidades valorativas
--se encuentren entre 2 y 5 unidades
CONSTRAINT ck_uv CHECK (UV>=2 AND UV<=5)
)
```

Crear la tabla Inscripcion

```
CREATE TABLE Inscripcion(
Carnet char(8),
CodigoMateria char(5),
Ciclo char(5),
--se esta creando una clave primaria compuesta
--en donde la union de los tres datos es un valor unico
CONSTRAINT pk_inscripcion PRIMARY KEY(Carnet,CodigoMateria,Ciclo)
)
```

La llave primaria está compuesta por medio de tres campos, los cuales individualmente pueden ser identificados como claves foráneas.
¿Por qué se ha creado una llave compuesta? por qué un alumno no puede inscribir la misma materia más de una vez en un mismo ciclo

2. Crear las relaciones entre las tablas

Inscripcion y Alumno:

```
ALTER TABLE Inscripcion
ADD
--Creando la relacion entre la tabla inscripcion y la tabla alumno
CONSTRAINT fk_alumno_ins FOREIGN KEY (Carnet) REFERENCES Alumno(Carnet)
ON UPDATE CASCADE
ON DELETE CASCADE
```

Inscripcion y Materia:

```
ALTER TABLE Inscripcion
ADD
--Creando la relacion entre la tabla inscripcion y la tabla materia
CONSTRAINT fk_materia_ins FOREIGN KEY (CodigoMateria) REFERENCES Materia(Codigo)
ON UPDATE CASCADE
ON DELETE CASCADE
```

En la **tabla Inscripcion** se está creando las relaciones entre tablas (creando claves foráneas) por lo tanto si se intenta eliminar un registro de la **tabla Alumno** cuyo valor de clave primaria existe referenciada en la **tabla Inscripcion**, la acción no se ejecuta y aparece un mensaje de error. Esto sucede porque, por defecto, para eliminaciones, la opción de la restricción FOREIGN KEY es "NO ACTION" (ninguna acción).
El mismo error se obtendría si se intenta actualizar un valor del campo Carnet de la **tabla Alumno** si esta referenciada por una FOREIGN KEY en este caso sería el campo Carnet de la **tabla Inscripcion**.

La restricción FOREIGN KEY de la **tabla Inscripcion** tiene las cláusulas ON DELETE CASCADE y ON UPDATE CASCADE las cuales estas cláusulas son opcionales.

Estas cláusulas especifican cómo debe actuar SQL Server frente a eliminaciones y modificaciones de las tablas referenciadas (tablas con la clave primaria) en la restricción del ejemplo la acción la

determina la opción CASCADE, la cual indica que si eliminamos o actualizamos un valor del campo Carnet de la tabla Alumno los registros coincidentes en la tabla foránea en este caso los datos del campo Carnet de la tabla Inscripcion, también se eliminan o se modifican a esto se le conoce como **integridad referencial en cascada**

Ejercicio 3. Creación del diagrama de la base de datos

1. Crear el diagrama de la base de datos, el cual queda de la siguiente manera

2. Guardar el diagrama de la base de datos con el nombre: **DiagramaBD_ControlAlumno**

Ejercicio 4. Uso de la instrucción INSERT

1. Agregar los siguientes datos a la tabla Alumno

```
--Agregando multiples registros

INSERT INTO Alumno VALUES
('GH121214','Gerardo Hierro'),
('VN121415','Veronica Nuñez'),
('CD121515','Cesar Deras'),
('HL130334','Helen Lara'),
('GM119056','Gricelda Martinez')
```

2. Realice un SELECT a la tabla y observara que esta ya tiene registros

```
SELECT  * FROM  Alumno
```

	Carnet	NombreCompleto
1	CD121515	Cesar Deras
2	GH121214	Gerardo Hierro
3	GM119056	Gricelda Martinez
4	HL130334	Helen Lara
5	VN121415	Veronica Nuñez

3. Agregar los siguientes datos a la tabla Materia

```
--Agregando datos a la tabla Materia, registro por registro

INSERT INTO Materia VALUES('BD01','Base de datos I',4)
INSERT INTO Materia VALUES('IP01','Introduccion a la programacion',4)
INSERT INTO Materia VALUES('AL01','Algebra Lineal',3)
INSERT INTO Materia VALUES('RD02','Redes de area amplia',5)
INSERT INTO Materia VALUES('GE01','Gestion Empresarial',2)
INSERT INTO Materia VALUES('HM02','Humanistica II',3)
```

4. Realice un SELECT a la tabla para verificar la inserción de los registros

5. Agregar los siguientes datos a la tabla Inscripcion

```
--Agregando datos especificando el orden de los campos de la tabla

INSERT INTO Inscripcion (Carnet,Codigomateria,Ciclo)
VALUES ('GH121214','BD01','C1-15')
INSERT INTO Inscripcion (Codigomateria,Carnet,Ciclo)
VALUES('GE01','GH121214','C1-15')
INSERT INTO Inscripcion (Ciclo,Carnet,Codigomateria)
VALUES('C1-15','GH121214','HM02')
```

6. Realice un SELECT a la tabla para verificar la inserción de los registros

Ejercicio 5. Uso de la instrucción UPDATE

1. El alumno que tiene el carnet GH121214 se debe cambiar el nombre de Gerardo Hierro a Gerardo Hernández, digitar la siguiente consulta:

```
UPDATE Alumno SET NombreCompleto='Gerardo Hernández'
WHERE Carnet='GH121214'
```

2. Realice un SELECT a la tabla para verificar, para verificar la actualización del campo en la tabla

```
SELECT * FROM Alumno
```

	Carnet	NombreCompleto
1	CD121515	Cesar Deras
2	GH121214	Gerardo Hernández
3	GM119056	Gricelda Martinez
4	HL130334	Helen Lara
5	VN121415	Veronica Nuñez

3. En el siguiente ejemplo se actualizará el carnet del alumno Gerardo Hernández y se verificará el funcionamiento de la instrucción ON UPDATE CASDADE

4. Primero se realizará un SELECT a las dos tablas para verificar la información

```
SELECT * FROM Alumno

SELECT * FROM Inscripcion
```

	Carnet	NombreCompleto
1	CD121515	Cesar Deras
2	GH121214	Gerardo Hernández
3	GM119056	Gricelda Martinez
4	HL130334	Helen Lara
5	VN121415	Veronica Nuñez

	Carnet	CodigoMateria	Ciclo
1	GH121214	BD01	C1-15
2	GH121214	GE01	C1-15
3	GH121214	HM02	C1-15

5. Ahora digitamos la consulta de actualización

```
UPDATE Alumno SET Carnet='GH111214'
WHERE NombreCompleto='Gerardo Hernández'
```

6. Realizar un SELECT a las tablas y verificar la actualización de los datos

	Carnet	NombreCompleto
1	CD121515	Cesar Deras
2	GH111214	Gerardo Hernández
3	GM119056	Gricelda Martinez
4	HL130334	Helen Lara
5	VN121415	Veronica Nuñez

	Carnet	CodigoMateria	Ciclo
1	GH111214	BD01	C1-15
2	GH111214	GE01	C1-15
3	GH111214	HM02	C1-15

Como se observa en los resultados se realizó la actualización de los datos en el campo carnet de la tabla Inscripcion al mismo tiempo que se ejecutó la consulta UPDATE en la tabla Alumno

Ejercicio 6. Uso de la instrucción DELETE

1. Crear la siguiente consulta de eliminación de datos

   Se quiere eliminar el registro del alumno donde el carnet GH111214 almacenado en la tabla Alumno

```
DELETE FROM Alumno WHERE Carnet='GH111214'
```

2. Verifique los datos de las tablas Alumno e Inscripcion por medio de una consulta SELECT

```
SELECT * FROM Alumno
SELECT * FROM Inscripcion
```

	Carnet	NombreCompleto
1	CD121515	Cesar Deras
2	GM119056	Gricelda Martinez
3	HL130334	Helen Lara
4	VN121415	Veronica Nuñez

	Carnet	CodigoMateria	Ciclo

3. Y observará que los datos del alumno con carnet GH111214 se han eliminado tanto de la tabla Alumno y de la tabla Inscripcion

Guardar el Script con el nombre: **EjercicioProcedimiento_Guia5.sql**

V. Ejercicio complementario

Con la base de datos creada en el procedimiento realice las siguientes consultas

a. Agregar los siguientes registros a la tabla alumno

Carnet	NombreCompleto
MC129854	Mauricio Campos
IP110943	Ignacio Pérez
MU127895	Mikel Urrutia
CH132390	Carlos Hernández
HL139032	Herson López

b. Agregar los siguientes registros a la tabla Inscripcion

Carnet	CodigoMateria	Ciclo
CD121515	AL01	C1-14
CD121515	GE01	C1-14
CD121515	HM02	C1-15
GM119056	IP01	C2-14
GM119056	RD02	C2-14
HL130334	BD01	C1-15
VN121415	BD01	C1-15
VN121415	RD02	C1-15
MC129854	AL01	C1-14
MC129854	GE01	C1-14
IP110943	GE01	C1-15
IP110943	HM02	C1-15

c. Con la instrucción SELECT INTO, crear una tabla con el nombre MateriaUV que tenga los datos de la materia donde las unidades valorativas sean mayores o iguales a 4

d. Con la instrucción INSERT INTO – SELECT, crear una tabla con el nombre Alumno2012 en donde se almacenen aquellos alumnos que tengan el carnet del año 2012

e. Crear las siguientes consultas de actualización de datos
   i. Modificar el apellido del alumno con carnet GM119056 a Márquez
   ii. Cambiar el carnet del alumno Gricelda Márquez a GM119156
   iii. Modificar el ciclo de la inscripción de C1-14 a C2-15
   iv. Modificar el código de la materia HM02 a HM01 y el nombre de la materia a Humanística I
   v. Modificar el apellido del alumno con carnet IP110943 a Pereira

f. Crear las siguientes consultas de eliminación de datos
   I. Eliminar el alumno con el carnet IP110943
   II. Eliminar los alumnos en donde el carnet comience con letra M
   III. Eliminar la materia Introducción a la Programación
   IV. Eliminar el alumno Oscar Hernández
   V. Eliminar la inscripción donde el código de la materia es igual RD02 y el ciclo es igual C1-15

Guardar el Script con el nombre: **EjercicioComplementario_Guia5.sql**

V. Fuente de consulta

1. La Biblia de SQL Server 2005
Madrid, España: Anaya, 2006
Autor: Mike Gundelerloy y Joseph L. Jorden
Biblioteca UDB – Clasificación: 005.361 G975 2006

2. Microsoft SQL Server 2008: Guía del Administrador
Madrid, España: ANAYA, 2009
Autor: William Stanek
Biblioteca UDB – Clasificación: 005.361 S784 2009