UNIDAD 5 CONSTRUCCIÓN

Introducción a la unidad

Una vez que hemos realizado el diseño de la base de datos y obtenido el modelo relacional (unidad 4), ha llegado el momento de implementarlo (programarlo) en un manejador de base de datos específico. Para realizar esto será necesario determinar cuál nos conviene. Hoy en día existe una gran variedad de ellos, desde los que cuentan con licencia de uso comercial hasta los basados en la postura del software libre.

Algunos de los aspectos a considerar para la selección son: el volumen de información que pueden almacenar, generalmente medido en *megabytes* o *terabytes*; el número de usuarios que pueden acceder al mismo tiempo; sus ventajas en el manejo de transacciones; su velocidad de respuesta a un número considerable de transacciones, y sus características para imponer seguridad a los datos.

Como recordarás, la programación en una base de datos relacional se realiza con el lenguaje SQL, y a lo largo de esta unidad conocerás de manera general el uso de este lenguaje. El objetivo de la materia de bases de datos no consiste en que aprendas a programar con SQL, por lo que sólo haremos mención de los comandos más importantes y necesarios. Te explicaremos para qué sirve cada uno de los objetos de una base de datos, cuál es su objetivo y sus principales características.

Afortunadamente el lenguaje SQL es un estándar y, si bien hay diferencias de programación entre los distintos manejadores de bases de datos, lo que revisemos en esta unidad aplicará para cualquiera de ellos. En otras palabras, desarrollaremos la unidad sin pensar en un software específico. En caso de que sea necesario, haremos alguna anotación adicional. Si te interesa conocer al detalle aspectos de los principales manejadores de bases de datos puedes revisar los capítulos 26 al 29 del libro de Silberschatz (2006: 807-921), en los que presentan cuatro manejadores: PostgreSQL, Oracle, DB2 UDB y SQL Server.

Revisemos entonces los objetos de base de datos que utilizamos para implementar una base de datos útil en un sistema de información.

Objetivo particular de la unidad

Identificar las principales actividades que realiza un programador de bases de datos, así como enumerar los objetos que son creados en una base de datos relacional y diferenciarlos mediante sus principales características.

Lo que sé:

Ve los siguientes videos que se encuentran en las siguientes direcciones:

- http://www.youtube.com/watch?v=KHHmRhFRM20 (referente al software SQL Server 2005 Managment Studio Express para administrar bases de datos)

Posteriormente, elabora un resumen de los mismos y entrega tu trabajo en formato pdf, resaltando las actividades del Administrador de Bases de Datos y las etapas automatizadas de la construcción de las bases de Datos.

Realiza tu actividad en un procesador de textos, imprímela en formato pdf y guárdala en tu computadora; una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

Temas de la unidad V

1. Roles del implementador
2. Tablas
3. Integridad
4. Índices
5. Vistas
6. Triggers
7. Stored Procedures
8. Manejo de Transacciones
9. Recuperación

Resumen de la unidad

En esta Unidad se vieron las actividades del Diseñador de la Base de Datos, en qué consisten y su impacto en la entrega del diseño de la Base de Datos; la aplicación de comandos o instrucciones con las cuales se crean tablas o Bases de Datos Complejas como son CREATE TABLE y los valores por DEFAULT así como sus restricciones CONSTRAINT. También se definen las Llaves Primarias y Foráneas con las cuales se vinculan las tablas sobre criterios establecidos.

Se estudiaron los valores de NOT NULL, CHECK, UNIQUE, PRIMARY KEY, FOREING KEY, y su aplicación en los atributos específicos de las Tablas para conservar la integridad de las Tablas en la Base de Datos; de igual forma de vieron los Comandos SELECT con WHERE, CREATE INDEX y la creación de archivos temporales para realizar los movimientos y actualizaciones de las Tablas.

Se crearon Vistas para guardar consultas efectuadas con anterioridad, como una forma de no emplear más de un archivo y ahorrar el empleo de memoria interna (con el empleo del comando CREATE WIEW y el nombre de la vista). Se abordó la forma de Almacenamiento de los Procedimientos para poder guardar procedimientos empleando comandos de SQL como son BEGIN TRANSATION, COMMIT TRANSACTION Y ROOLBACK TRANSACTION para confirmar, deshacer o confirmar una transacción en las Tablas dentro de la Base de Datos.

Tema 1. Roles del implementador

Objetivo del tema

Reconocer la importancia del papel que desempeña el diseñador de bases de datos.

Desarrollo

El diseñador de bases de datos entrega el modelo lógico de tablas al implementador o programador de bases datos para que construya la base de datos en el sistema manejador. Sus principales roles o actividades son:

1. Programar las estructuras de almacenamiento (tablas).
2. Implementar las reglas de integridad mediante restricciones o triggers.
3. Agilizar las consultas mediante la creación de índices.
4. Encapsular consultas en vistas.
5. Programar procedimientos almacenados para implementar la lógica de procesamiento de datos dentro de la base de datos.

Para que conozcas a fondo estas actividades, en los siguientes temas revisaremos cada uno de los objetos programables en una base de datos.

ACTIVIDAD 1

Investiga y documenta las actividades del diseñador que construye la Base de Datos en general.

Realiza tu actividad en un procesador de textos, imprímela en formato pdf y guárdala en tu computadora; una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Ve los siguientes videos, referentes al rendimiento de la Base de Datos SQL:

http://www.youtube.com/watch?v=8Ukrz7P_MMI
http://www.youtube.com/watch?v=FlnVFiHrg3k
http://www.youtube.com/watch?v=PUR7u7Y20T0

Elabora un resumen de los mismos y entrega tu resumen por escrito en formato pdf .

Haz énfasis en las Actividades del implementador de BD.

Realiza tu actividad en un procesador de textos, imprímela en formato pdf y guárdala en tu computadora; una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Responde las siguientes preguntas.

1. Anota el concepto de Diseñador.
2. ¿Cuáles son las actividades que realiza un implementador de bases de datos?
3. Anota el concepto de Diseñador de Bases de Datos.
4. Anota las actividades del Programador.
5. ¿Qué es un rol?

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

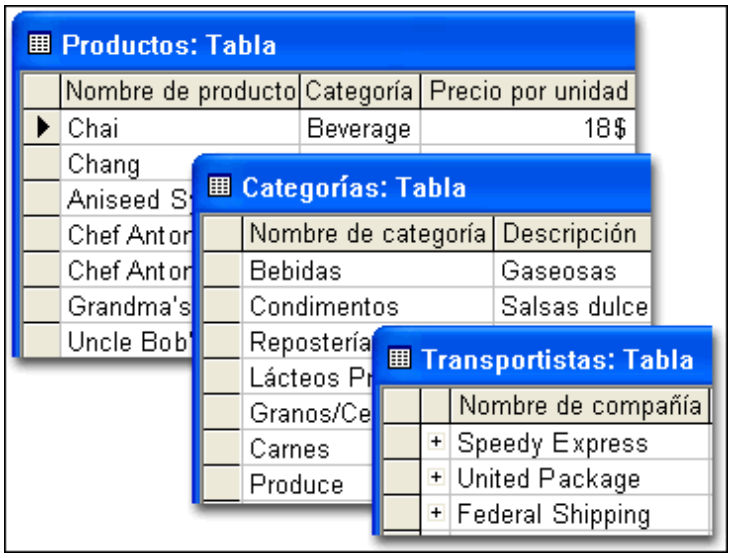
Tema 2. Tablas

Objetivo del tema

Identificar los comandos DML en la creación de una tabla.

Desarrollo

Las tablas son un conjunto de filas y columnas que nos permiten almacenar los datos bajo el enfoque de un modelo relacional. Como sabes, el término tabla se conoce de manera formal como relación, al renglón como tupla y a la columna como atributo. Son en éstas donde almacenamos los datos mediante instrucciones en DML (Lenguaje de Manipulación de Datos).



Para crear una tabla utilizamos el comando CREATE TABLE, con la siguiente sintaxis general[1]

[1] La sintaxis que presentamos está simplificada en comparación con la que puedes encontrar en los manuales de programación de SQL de los sistemas de base de datos. Ya que nuestro objetivo no es presentar la manera de programar sino el entendimiento de los objetos programables de una base de datos, creemos que es necesario ser más específicos. Para entenderla mejor, recuerda que los [] encierran elementos que pueden o no incluirse en la programación y que el uso del carácter | es para proponer distintas opciones agrupadas entre llaves {}.

```
CREATE  TABLE nombre_tabla
(
nombre_columna tipo_dato
[DEFAULT valor_default]
[CONSTRAINT nombre_constraint TIPO (condición)],
nombre_columna tipo_dato...,
nombre_columna tipo_dato...
)
```

Para ejemplificar la manera de crear una tabla veamos un ejemplo. Pensemos que deseamos crear la siguiente tabla:

Autor		
Idautor	Nombres	Apellidos
1	Ignacio Manuel	Altamirano
2	Manuel	Payno
3	Jorge Luis	Borges
4	Sor Juana Inés	De la Cruz
5	Julio	Cortázar

Siguiendo la plantilla de sintaxis expuesta arriba, podemos crear la tabla de la siguiente manera:

```
CREATE TABLE autor
(
idautor integer CONSTRAINT pkautor PRIMARY KEY,
nombres varchar(40) NOT NULL,
apellidos varchar(40) NOT NULL
);
```

Creamos una tabla de nombre auto con tres columnas. Cada una de ellas se declara por separado y termina su declaración con una coma. Observa que la última columna no tiene una coma al final ya que no le sigue ninguna columna más. Cada definición de columna se compone de nombre, por ejemplo **idautor**, su tipo de dato, en este caso entero **integer,** y la declaración de una restricción de integridad o **constraint.**

El *constraint* se forma por la palabra reservada **CONSTRAINT**, seguida de un nombre para esa restricción, pkauto, y el tipo de restricción, en este caso de clave primaria, **PRIMARY KEY**. Las otras columnas se definen de la misma forma y sólo cambian de tipo de dato y de restricción por una de **NOT NULL.**

Otras operaciones que podemos realizar con una tabla son:

- Renombrarla.
- Renombrar una columna.
- Agregar o eliminar columnas.
- Cambiar el tipo de dato de una columna.
- Agregar o eliminar una restricción (*constraint*).
- Agregar o eliminar un DEFAULT.

Estas operaciones se realizan con el comando ALTER TABLE.

Un DEFAULT es un valor predefinido para una columna que se inserta automáticamente si no definimos un valor específico. Otra operación es la de eliminar una tabla con todo y sus datos que ejecutamos con el comando DROP TABLE.

ACTIVIDAD 1

Investiga el significado de las siguientes palabras:

a) Constraint
b) Default

Determina la sintaxis de los comandos CONSTRAINT Y DEFAULT de lenguaje DML.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

ACTIVIDAD 2

Escribe los comandos para crear una tabla de nombre Artículo, con los atributos código, descripción, precio y stock y su llave primaria es código.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

ACTIVIDAD 3

Define qué es:

1) Una Tabla
2) Una Tupla
3) Una Columna
4) El Comando Create
5) El Comando Constraint

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

Autoevaluación

Conteste falso o verdadero a las siguientes afirmaciones.

	Verdadera	Falsa
1. Un conjunto de tuplas forman una Tabla.	( )	( )
2. El comando CREATE TABLE nombre_tabla es correcto.	( )	( )
3. Es necesario anotar el tipo de dato junto con el nombre de columna de campo de la tabla.	( )	( )
4. Las comas sirven para terminar la declaración una columna.	( )	( )
5. Primary Key da el atributo de llave única a un campo.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 3. Integridad

Objetivo del tema

Emplear las diferentes restricciones (constraints) en una base de datos para mantener su integridad.

Desarrollo

La integridad de una base de datos se establece con restricciones, en inglés *constraints*. Ponemos restricciones en las columnas de una tabla para que acepten o rechacen ciertos valores. Con ello evitamos que los datos ingresados a la base sean incorrectos o inapropiados. A continuación te ofrezco una lista con posibles restricciones de una columna y la descripción de los valores que acepta o rechaza.

RESTRICCIÓN	VALORES QUE ACEPTA	VALORES QUE RECHAZA
NOT NULL	Distintos a Null	Valor Null
CHECK	Los que cumplen con la condición establecida por la restricción	Los que no cumplen con la condición establecida por la restricción.
UNIQUE	Valores únicos en la columna	Valores repetidos en la columna.
PRIMARY KEY	Valores únicos distintos a Null	Valores repetidos y valor Null.
FOREIGN KEY	Valores que existan previamente en la clave primaria	Valores que no existan previamente en la clave primaria.

Tabla de restricciones de una columna

Las restricciones se programan dentro de la definición de cada columna en un CREATE TABLE. En la siguiente tabla puedes ver ejemplos de la programación de cada tipo de *constraint*.

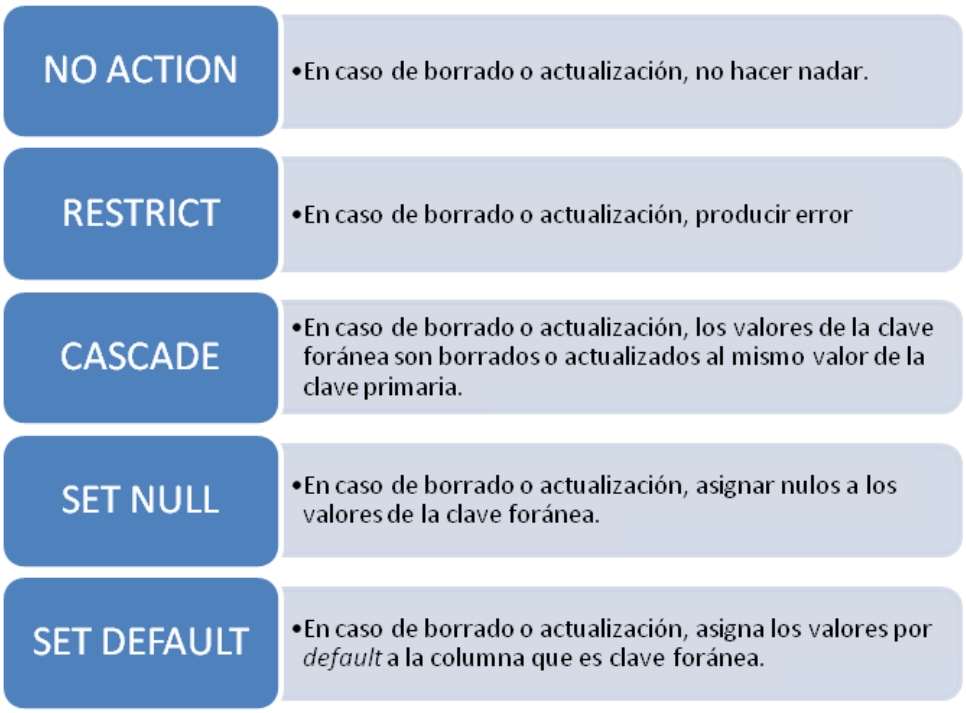
Restricción	Ejemplo
NOT NULL	titulo text NOT NULL
CHECK	sexo char(1) CONSTRAINT chksexo CHECK (sexo IN ('F', 'M'))
UNIQUE	rfc char(13) CONSTRAINT unirfc UNIQUE
PRIMARY KEY	idlibro integer CONSTRAINT pklibro PRIMARY KEY
FOREIGN KEY	REFERENCES autor (idautor) ON DELETE CASCADE ON UPDATE CASCADE

Ejemplos de programación de restricciones

El *constraint* de FOREIGN KEY es particularmente más elaborado. La parte de REFERENCES hace referencia a la tabla padre de la relación, es decir, aquella donde está la clave primaria a la que hace referencia la clave foránea que se está declarando. La restricción incluye dos cláusulas:

1. ON DELETE action
2. ON UPDATE action

Éstas indican la acción a ejecutar con los valores de la clave foránea, en caso de que los valores de la clave primaria sean borrados o actualizados. El parámetro *action dependiente de cláusulas,* se refiere a las siguientes posibles acciones:



En seguida puedes ver el código completo para crear una tabla con todos los tipos de *constraints*.

```
CREATE TABLE empleado
(
idempleado integer CONSTRAINT pkempleado PRIMARY KEY,
nombres varchar(40) NOT NULL,
apellidos varchar(40) NOT NULL,
rfc char(13) CONSTRAINT unirfc UNIQUE,
idarea integer REFERENCES area(idarea)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

ACTIVIDAD 1

Construye un cuadro sinóptico con los tipos de restricciones que existen.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Escribe el código necesario para crear la tabla Músico donde su llave primaria es idmusico de tipo integer, los campos nombre, apellidos serán capturados siempre, el campo obra será de 30 caracteres.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

ACTIVIDAD 3

Basándote en la Estructura de la Tabla Músico, agrega la llave foránea idorquesta no nula y el campo fecha_obra de tipo fecha de 8 caracteres no nulo.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

ACTIVIDAD 4

Responde las siguientes preguntas.

1. ¿Qué valores rechaza la restricción CHECK?
2. ¿Qué valores rechaza Unique?
3. ¿Cómo se aplican las restricciones?
4. ¿Qué significa el valor único en una tabla?
5. ¿Qué significa la cláusula ON DELETE?

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

Autoevaluación

Escriba en los espacios en blanco la (s) palabra (s) que complete las siguientes frases:

nulos *repetidos en la columna* *valores únicos distintos a NULL* *campo 1 es llave primaria* *más elaborada*	**1.**. La Restricción NOT NULL rechaza valores _____ **2.**. La Restricción UNIQUE rechaza valores _____ **3.**. La Restricción FOREING KEY acepta _____ **4.**. El siguiente comando CONSTRINT Pkcampo1 PRIMARY KET, significa _____ **5.**. La Restricción FOREING KEY debe de ser: _____

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 4. Índices

Objetivo del tema

Emplear los índices en los campos de las tablas para incrementar su rendimiento en las consultas.

Desarrollo

Los índices permiten incrementar el rendimiento de la base de datos haciendo más rápida la ejecución de consultas. En otras palabras, una cláusula SELECT con WHERE encontraría más rápido los registros que cumplen la condición.

Se hacen sobre una o más columnas de una tabla. Es importante resaltar que debemos poner índices sólo en columnas usadas frecuentemente en nuestras expresiones de comparación con la cláusula WHERE.

Para crear un índice utilizamos la cláusula CREATE INDEX y para borrarlo se hace con el comando DROP INDEX, por ejemplo:

```
CREATE INDEX idx_nombres ON empleado(nombres);
DROP INDEX idx_nombres;
```

Los Índices de Tablas asociados a campos de tipo numérico y carácter pueden ser compuestos en virtud de que se pueden vincular dos campos para formar un solo índice como por ejemplo en la Tabla “Producto” y los Atributos Cod_Prod y Suc_Prod se pueden combinar para formar el índice compuesto Pedido. El comando sería el siguiente:

```
CREATE INDEX Pedido ON Producto(Cod-Prod, Suc_Prod);
```

Para ahondar más en el tema, favor de consultar las siguientes direcciones electrónicas:

http://www.aspfacil.com/articulos/2808001.asp	ASP FACIL, Consulta 31 de marzo del 2009
http://www.youtube.com/watch?v=FlnVFiHrg3k	
http://www.youtube.com/watch?v=2Uy-bvTppbc	
http://www.youtube.com/watch?v=8Ukrz7P_MMI	
http://www.youtube.com/watch?v=BNR-yJsTsLc	
http://www.youtube.com/watch?v=G0tcnKHFZPA	Rendimiento de Bases de Datos, Consulta 31 de marzo del 2009
http://www.youtube.com/watch?v=iXMyISZS_f4	
http://www.youtube.com/watch?v=7l_tZlZW1wk	
http://www.youtube.com/watch?v=hXju_gHOUnE	Consultas a Bases de Datos, Consulta 31 de marzo del 2009

ACTIVIDAD 1

Crea la Tabla Ventas con los atributos Fecha de Venta, Folio de Venta, Importe, Cliente y Código de Producto.

El campo indexado será Folio.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Aplica a la Tabla anterior un campo compuesto indexado constituido por los campos Folio y Código de Producto.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Contesta las siguientes preguntas:

1. ¿Qué función tiene un Índice?

2. ¿Qué te permiten hacer la cláusula Create Index, el comando Drop Index y Select y Where?

3. Los índices se deben aplicar a columnas que:

4. ¿Qué es un índice?

5. ¿Cómo crear un índice compuesto?

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Autoevaluación

Conteste falso o verdadero a las siguientes afirmaciones.

	Verdadera	Falsa
1. Un índice solo se aplica a campos numéricos.	( )	( )
2. Un índice es más eficiente si es de tipo numérico.	( )	( )
3. Create Index over es un comando correcto.	( )	( )
4. El Comando Drop campo_id es un campo correcto.	( )	( )
5. Un índice ayuda a ordenar los campos de la tabla.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 5. Vistas

Objetivo del tema

Crear vistas generadas a partir de consultas provenientes de Tablas.

Desarrollo

Una vista es un objeto de la base de datos que almacena una consulta. Funciona como una tabla, pero no existe físicamente en la base de datos, se genera de forma dinámica. Una vista nos permite encapsular consultas que utilizamos de forma recurrente, nos evita escribirlas de nuevo. También nos ayuda a manipular consultas muy complejas de una forma más sencilla.

La instrucción SQL para consultar datos es SELECT. La sintaxis básica general de esta instrucción es:

```
SELECT nombre_columna, nombre_columna,...
FROM nombre_tabla
WHERE condición
```

A partir de este tipo de consultas se crean las vistas con el comando CREATE VIEW, de la siguiente forma:

```
CREATE VIEW nombre_vista AS
SELECT ...
```

Veamos algunos ejemplos de vistas creadas a partir de diversas consultas:

Seleccionar todos los renglones y todas las columnas:	Seleccionar algunas columnas:	Seleccionar renglones a partir de una condición expresada en la cláusula WHERE:
CREATE VIEW vempleados AS SELECT * FROM empleado;	CREATE VIEW vemp AS SELECT idempleado, nombres, apellidos FROM empleado;	CREATE VIEW vempleado2 AS SELECT nombres, apellidos FROM empleado WHERE idempleado = 2;

Recuperar registros de una sola tabla es muy inusual. En la realidad siempre obtenemos datos de diferentes tablas, a veces de muchas. Es importante entonces conocer la manera de "unirlas" para poder obtener valores almacenados en columnas de unas y de otras. Esto lo realizamos con la operación relacional llamada junta o *join*. Existen varios tipos de *join*, entre ellos podemos mencionar:

1. Cross join

El resultado es un producto cartesiano, es decir, una combinación de todos los valores de una tabla contra todos los valores de otra tabla.

2. Inner join

El resultado es un conjunto de registros que resultan de la combinación de dos o más tablas, siempre y cuando existan columnas en común y los valores de dichas columnas coincidan. Este *join* necesita de una cláusula ON que iguale las columnas en común.

3. Outer join

El resultado es un *inner join* que además incluye aquellos valores donde no hay coincidencia en el origen del lado izquierdo (LEFT OUTER JOIN) o del lado derecho (RIGHT OUTER JOIN) o aquellos que no coinciden en ningún lado (FULL OUTER JOIN).

A continuación un ejemplo de consulta de dos tablas.

```
CREATE VIEW vlibroautor AS
SELECT libro.titulo, autor.nombres, autor.apellidos
FROM libro INNER JOIN autor
ON (libro.idautor = autor.idautor);
```

El resultado de este *join* no incluiría los libros sin autor en caso de que no haya coincidencia entre las claves idautor de las dos tablas. Esto sucede porque el *inner join* rescata sólo aquellos que cumplen la condición de igualdad entre columnas en común. Si quisiéramos rescatar los libros que sí tienen autor y los que no tienen, deberíamos utilizar un *outer join*.

```
CREATE VIEW vlibroautor AS
SELECT libro.titulo, autor.nombres, autor.apellidos
FROM libro LEFT OUTER JOIN autor
ON (libro.idautor = autor.idautor);
```

Se trata de un *left outer join* porque la tabla libro está a la izquierda del *join*. Si lo que quisiéramos fueran todos los autores con y sin libro asociado, entonces tendríamos que programar un *right outer join*.

```
CREATE VIEW vlibroautor AS
SELECT libro.titulo, autor.nombres, autor.apellidos
FROM libro RIGHT OUTER JOIN autor
ON (libro.idautor = autor.idautor);
```

ACTIVIDAD 1

Crea la siguiente Tabla.

ARTICULO				
CODIGO	**DESCRIPCION**	**CANTIDAD**	**PRECIO**	**SUCURSAL**
4500	PANTALLA 15"	30	4500	CENTRO
56700	REGULADOR	100	1200	SUR
4554667	NO_BREAKER	150	800	CENTRO
56788	KIT MTTO.	200	450	NORTE
4500	PANTALLA 15"	30	4500	CENTRO
56700	REGULADOR	100	1200	SUR
4554667	NO_BREAKER	150	800	CENTRO
56788	KIT MTTO.	200	450	NORTE
4500	PANTALLA 15"	30	4500	CENTRO
56700	REGULADOR	100	1200	SUR
4554667	NO_BREAKER	150	800	CENTRO
56788	KIT MTTO.	200	450	NORTE

Crea la vista con el nombre de Almacén, donde se seleccionen las columnas código y cantidad para cantidades superiores a 149 artículos.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

De la Tabla creada en la actividad anterior, crea una vista con el nombre de Inventario donde se seleccionen únicamente los reguladores.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Responde las siguientes preguntas.

1. Una selección proveniente de una Tabla es:
2. ¿Qué significa la Clausula From y Where?
3. ¿Qué significa Cross join e Inner join?
4. ¿Qué es una vista?
5. Enliste los diferentes tipos de Join.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Autoevaluación

Conteste falso o verdadero a las siguientes afirmaciones. Al finalizar tendrás tu calificación de manera automática.

	Verdadera	Falsa
1. Una vista se integra a la Tabla consultada.	( )	( )
2. Una vista genera Archivos Temporales.	( )	( )
3. Las vistas se generan a partir de consultas.	( )	( )
4. Las vistas se aplican a los atributos de las Tablas.	( )	( )
5. El Comando Create on View Screen es correcto.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 6. Triggers

Objetivo del tema

Emplear la función triggers para actualizar tablas.

Desarrollo

Un *trigger* ejecuta un determinado procedimiento almacenado en la base de datos cuando se realiza cualquier modificación en una tabla. Un *trigger* es una función que se "dispara" antes o después de la instrucción que actualiza la tabla a la que está asociado. Son usados para implementar reglas de integridad de datos, auditoría de tablas importantes y actividades de mantenimiento de datos.

Para crear un *trigger* debemos usar la siguiente sintaxis general:

```
CREATE TRIGGER nombre_trigger
{ BEFORE | AFTER } { DELETE OR UPDATE OR INSERT }
ON nombre_tabla FOR EACH { ROW | STATEMENT }
EXECUTE PROCEDURE nombre_procedimiento;
```

Las opciones de BEFORE o AFTER permiten definir si el trigger se ejecutará antes o después de un evento INSERT, DELETE o UPDATE. La cláusula FOR EACH ROW indica que el procedimiento que dispara el *trigger* será ejecutado por cada renglón actualizado por el evento INSERT, DELETE o UPDATE. Si por el contrario se indica FOR EACH STATEMENT, el procedimiento será disparado una sola vez.

Para borrar un *trigger* contamos con la sentencia:

```
DROP TRIGGER nombre_trigger ON nombre_tabla;
```

Siendo redundantes, un *trigger* dispara un procedimiento almacenado, el cuál puede realizar cualquier acción en la base de datos. De esta manera, podemos crear un *trigger* para los siguientes casos:

- Después de borrar o actualizar en la tabla ventas, actualizar en la tabla venta_total el total de venta.
- Después de modificar, eliminar o actualizar la tabla venta, registrar en la tabla venta_bitacora el usuario que modificó la venta y la hora de modificación.
- Antes de actualizar o eliminar en la tabla cliente, revisar si el cliente no tiene registros en la tabla clientes_morosos, en caso de tener registros detener la actualización.
- Después de eliminar un registro de la tabla préstamo, insertar en la tabla prestamo_historico el registro eliminado.

ACTIVIDAD 1

Escribe tres operaciones que podrían implementarse con un trigger.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

ACTIVIDAD 2

De la Tabla creada en el tema anterior, crea un trigger con el nombre de cant_stok a ejecutarse después de actualizar las cantidades del atributo cantidad por cada renglón.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Sobre la misma tabla que has modificado en la actividad 1, crea un trigger con el nombre de code_art a ejecutarse después de actualizar los códigos de los artículos por cada renglón de la Tabla después insertar dos artículos más con el código del Artículo pantalla.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 4

Responde las siguientes preguntas.

1. ¿Qué es un Trigger?
2. ¿Qué ejecuta un Trigger?
3. La Opción After cómo afecta a un Trigger.
4. La Clausula For Each Procedure sirve para:
5. La Sentencia Drop Trigger sirve para:

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Autoevaluación

Conteste falso o verdadero a las siguientes afirmaciones. Cuando concluyas obtendrás tu calificación de manera inmediata.

	Verdadera	Falsa
1. Before y After son clausulas.	( )	( )
2. Insert y Delete son opciones.	( )	( )
3. Drop Trigger es una sentencia.	( )	( )
4. Se puede borrar e insertar un elemento en una tabla con una serie de instrucciones.	( )	( )
5. Con solo escribir Execute Procedure basta para ejecutar un procedimiento.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 7. Stored Procedures

Objetivo del tema

Identificar la importancia de almacenar procedimientos para acelerar la ejecución de instrucciones aplicadas a las tablas e implementarlos.

Desarrollo

Un procedimiento almacenado (*stored procedure*) es un conjunto de instrucciones en un lenguaje de programación que se almacenan como un objeto de la base de datos y puede ser ejecutado en cualquier momento. Por lo general estos procedimientos almacenados están escritos en un lenguaje peculiar que combina un lenguaje *procedural* (o de procedimiento) con el SQL.

Este lenguaje se vuelve bastante poderoso en tanto que une declaración de variables, manejo de excepciones, expresión de ciclos y condicionales con las capacidades del SQL. Todos los manejadores de bases de datos incluyen un lenguaje de este tipo, por ejemplo: Oracle tiene el pl/sql, PostgreSQL el pl/pgsql, y SQL Server cuenta con el Transact-sql.

Todo procedimiento almacenado sigue una estructura basada en tres secciones:

1. Sección declarativa.
   Permite declarar variables y cursores.

2. Sección ejecutiva.
   Es donde se expresan las instrucciones que procesan los datos.

3. Sección de excepciones.
   Aquí podemos manejar excepciones producidas por el procedimiento.

Hay **tres tipos de procedimientos almacenados**. Los procedimientos como tales, que no regresan valor; las funciones, que sí regresan algún valor; y los *triggers*, que como ya vimos, se ejecutan automáticamente cuando se actualiza la tabla a la que están vinculados. Tanto las funciones como los procedimientos reciben parámetros que utilizan dentro de la sección ejecutiva. Dependiendo del lenguaje que utilicemos podremos contar con parámetros de entrada, salida o entrada-salida.

Es justo decir que la programación de procedimientos almacenados varía de manejador en manejador. Además, es muy importante, ya que nos permite encapsular la lógica del procesamiento de datos al interior de la base de datos. Este hecho es una ventaja en comparación con programar el procesamiento de datos en los programas de aplicación, ya que los *stored procedures* se ejecutan más rápido y están pre-compilados y probados desde antes.

Es posible programar dentro de la base de datos, desde cálculos completos de impuestos y procesamientos estadísticos hasta modificaciones complejas de datos. Desafortunadamente, por el desconocimiento de estos objetos de bases de datos, en muchos equipos de desarrollo se continúa programando fuera de la base de datos.

ACTIVIDAD 1

Codifica en Lenguaje SQL las siguientes indicaciones:

a) Crea la tabla almacén.
b) Introduce los atributos idart, descrip, stock, precio.
c) Introduce datos a los atributos.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

De las características de la Tabla anterior, indéxala sobre el atributo idart y dale el nombre de almacen1.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Describe lo siguiente:

a) El concepto de procedimiento
b) El procedimiento almacenado
c) Las clases de procedimientos.
d) 4 ejemplos de lenguajes de manejadores de bases de datos.
e) La estructura de todo procedimiento.

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

Autoevaluación

Completa las siguientes frases escribiendo en el espacio en blanco la (s) palabras (s) que hagan falta.

1. La sección _____ sirve para expresar las instrucciones que procesan los datos.

2. La sección _____ permite declarar variables.

3. La sección de _____ almacena excepciones a instrucciones.

4. Los procedimientos reciben _____.

5. Las _____ pueden tener parámetros de entrada.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 8. Manejo de transacciones

Objetivo del tema

Aplicar transacciones a las bases de datos.

Desarrollo

Una transacción de base de datos es un conjunto de dos o más instrucciones que modifican la información de la base (UPDATE, DELETE o INSERT), las cuales son tratadas como una unidad, de tal forma que se realizan todas o no se realiza ninguna. Una transacción puede terminar en una actualización de los datos (*commit*) o puede terminar sin actualización regresando la base de datos al estado consistente con el cuál empezó (*rollback*).

Una transacción comienza con la palabra BEGIN TRANSACTION, todas las instrucciones siguientes a ella forman parte de esa transacción. Para que los cambios sean permanentes debe terminar con el comando COMMIT TRANSACTION, en caso de querer deshacer los cambios utilizaríamos ROLLBACK TRANSACTION.

Las transacciones son utilizadas en los casos en los que las empresas requieren que las operaciones se completen como una unidad. La falta de actualización completa implicaría una falta de consistencia en la información. El ejemplo más claro es el de un cargo y su respectivo abono, sabemos que de no completarse las dos operaciones sufriríamos de un problema en los resultados financieros. Toda base de datos es susceptible de sufrir un error, originado por diversas fuentes, que impida que las operaciones se completen en conjunto.

El peligro de realizar sólo parte de la transacción es que algunos registros queden actualizados y otros no, dando como resultado información errónea. Para evitar

estos problemas de consistencia, los manejadores de bases de datos implementan automáticamente un mecanismo de recuperación que revisaremos en la siguiente sección.

La siguiente información y comandos son de Librerías basadas en SQL SERVER 2005 del sitio de Msdn

Una transacción explícita es aquella en que se define explícitamente el inicio y el final de la transacción.

Las aplicaciones de DB-Library y las scripts Transact-SQL utilizan las instrucciones BEGIN TRANSACTION, COMMIT TRANSACTION, COMMIT WORK, ROLLBACK TRANSACTION o ROLLBACK WORK de Transact-SQL para definir transacciones explícitas.

BEGIN TRANSACTION. Marca el punto de inicio de una transacción explícita para una conexión.

COMMIT TRANSACTION o COMMIT WORK. Se utiliza para finalizar una transacción correctamente si no hubo errores. Todas las modificaciones de datos realizadas en la transacción se convierten en parte permanentes de la base de datos. Se liberan los recursos ocupados por la transacción.

ROLLBACK TRANSACTION o ROLLBACK WORK. Se utiliza para eliminar una transacción en la que se encontraron errores. Todos los datos modificados por la transacción vuelven al estado en el que estaban al inicio de la transacción. Se liberan los recursos ocupados por la transacción.

También puede utilizar transacciones explícitas en OLE DB. Llame al método **ITransactionLocal::StartTransaction** para iniciar una transacción. Llame al método **ITransaction::Commit o ITransaction::Abort** con *fRetaining* establecido en FALSE para finalizar la transacción sin iniciar otra automáticamente. En ADO, utilice el método **BeginTrans** en un objeto **Connection** para iniciar una transacción explícita. Para finalizar la transacción, llame a los métodos **CommitTrans** o **RollbackTrans** del objeto **Connection**.

En el proveedor administrado de **SqlCliente** de ADO.NET, utilice el método **BeginTransaction** en un objeto **SqlConnection** para iniciar una transacción explícita. Para finalizar la transacción, llame a los métodos **Commit() o Rollback()** del objeto **SqlTransaction.**

La API de ODBC no acepta transacciones explícitas, sólo acepta transacciones de confirmación automática y transacciones implícitas.

El modo de transacciones explícitas se mantiene solamente durante la transacción. Cuando la transacción termina, la conexión vuelve al modo de transacción en que estaba antes de iniciar la transacción explícita, es decir, el modo implícito o el modo de confirmación automática."[2]

[2] http://msdn.microsoft.com/es-es/library/ms175127.aspx Microsoft © Msdn, Consulta 31 de marzo del 2009.

ACTIVIDAD 1

De la Tabla de almacén1 creada en las actividades del tema anterior, introduce dos registros más y aplica COMMIT TRANSACTION para terminar de procesar los cambios.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Empleando la tabla de la actividad anterior, introduce otros tres registros.

Aplica BEGIN TRANSACTION y después el ROLLBACK TRANSACTION para deshacer los cambios.

Guarda en imágenes cada una de estas funciones y envíalas a tu asesor a través del sitio.

Realiza tu actividad, captura las imágenes y guárdalas en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Describe lo siguiente:

a) Concepto de Transacción

b) Para qué sirve BEGIN TRANSACTION,
COMMIT TRANSACTION y ROLLBACK TRANSACTION.

c) Un ejemplo de transacción.

Explica las dos maneras de término de una transacción.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Autoevaluación

Seleccione falso (F) o verdadero (V) a las siguientes afirmaciones. Al concluir tendrás tu calificación de manera automática.

	Verdadera	Falsa
1. Una transacción es cualquier modificación de una base de datos.	( )	( )
2. Una transacción es trata como una unidad.	( )	( )
3. BEGIN TRANSACTION se anota al final de una declaración.	( )	( )
4. COMMIT TRANSACTION actualiza movimientos.	( )	( )
5. Toda base de Datos es sujeta al riesgo de riesgos.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 9. Recuperación

Objetivo del tema

Utilizar COMMIT y ROLLBACK en transacciones de operaciones de bases de datos.

Desarrollo

Como se estudió en el tema anterior, la recuperación es un mecanismo automático de un sistema de bases de datos que entra en acción cuando ocurre un error en medio de una transacción y ésta no se ha completado. El principio fundamental de la recuperación es que se hacen todas las operaciones o ninguna; si ya se habían registrado algunas operaciones, éstas deben deshacerse. Es mejor que los datos no se actualicen a que se actualicen de manera incompleta y, por tanto, errónea.

En seguida verás dos esquemas que muestran los dos comportamientos de una transacción, uno termina con un *commit* y el otro con *rollback*. En el segundo caso, los datos no se actualizan ya que ocurrió un error antes de terminar la transacción.

Estado consistente inicial			Transacción	Estado consistente final		
Id	Nombre	Saldo	1) Update saldo = 3000 Where id =1; 2) Update saldo = 2500 Where id =2;	Id	Nombre	Saldo
1	Alberto	2000		1	Alberto	3000
2	Alma	5000		2	Alma	2500

Cuadro. Transacción sin error que termina en *commit*

Estado consistente inicial			Transacción	Estado consistente final		
Id	Nombre	Saldo	1) Update saldo = 3000 Where id =1; ERROR 2) Update saldo = 2500 Where id =2;	Id	Nombre	Saldo
1	Alberto	2000		1	Alberto	2000
2	Alma	5000		2	Alma	5000

Cuadro. Transacción con error que termina en *rollback*

En el primer esquema (Cuadro de Transacción sin error que termina en *commit*), dado que no existió error en la transacción, ésta finaliza con un *commit* y los datos se actualizan. Pero en el segundo caso (Cuadro Transacción con error que termina en *rollback*), observa que ningún registro es actualizado a pesar de que la operación (1) sí se realizó; al presentarse el error ésta es desecha y la tabla queda como en el estado inicial.

Hemos descrito una serie de objetos de bases de datos que son programables y que permiten implementar el sistema relacional de almacenamiento de datos junto con sus restricciones. Con estos, el programador construye una base de datos funcional para un sistema de información. En la siguiente unidad revisaremos cómo administrar la base de datos una vez que está construida.

ACTIVIDAD 1

Crea la Tabla “Almacén” y realiza las siguientes transacciones:

Id_Video	Tit_Video	Dist_Video	Dir_Video
345667	“El Amazonas”	Discovery Channel	Samuel Podosky
34567	“Los Volcanes de Sudamérica”	“National Geographic”	R. J. Mendez

1. Añadir dos registros más.

2. Eliminar el registro cuyo id_Video es 34567

3. Recuperar el registro anteriormente eliminado.

4. Eliminar de forma definitiva el registro cuyo TiT_Video es igual a “Los Volcanes de Sudamérica”.

5. Cerrar la base de Datos.

Emplear para las transacciones anteriores comandos de SQL.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Realiza un cuadro de tres columnas que resuma los comandos de SQL presentados en el material, colocando en la primera columna el comando, en la segunda el objetivo y en la tercera un ejemplo.

Comando	Objetivo	Ejemplo

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Contesta las siguientes preguntas:

1. ¿A que está asociado Commit?
2. ¿A que está asociado Rollback?
3. ¿En qué consiste la importancia de la recuperación de transacciones?
4. ¿Qué son las recuperaciones?
5. ¿Cuál es el principio fundamental de la recuperación?

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

Autoevaluación

Seleccione falso (F) o verdadero (V) a las siguientes afirmaciones. Al concluir tendrás tu calificación de manera automática.

	Verdadera	Falsa
1. La recuperación es un mecanismo al cual que tiene que activar el programador.	( )	( )
2. Su principio fundamental es cerrar cada operación como una unidad.	( )	( )
3. Es mejor guardar algún dato de una transacción aunque los demás estén incorrectos.	( )	( )
4. Commit se asocia con una transacción errónea.	( )	( )
5. Rollback se asocia con una transacción errónea.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Lo que aprendí

Crea las siguientes Tablas.

Artículo				
idArt	**DescArt**	**Precio**	**Cantidad**	**Sucursal**
A3450	Chasis pantallas	$450.00	50	Norte
A6789	Protector de Pantallas	$670.00	100	Norte
A4500	Fundas para Computado	$370.00	200	Sur

Clientes			
idCliente	**Nombrecte**	**Rfccte**	**Saldo**
567889	Carlos Duran Alfaro	Duac800923	$5670
799755	Mario Canseco Avilés	Caam781002	$6340
234574	Irma Domínguez Vera	Dovi790924	$5670

Relaciona ambas Tablas para que formen la Base de Datos Administración.

Procesa la operación donde el cliente Carlos Duran Alfaro compre el artículo Chasis para Pantallas por una cantidad de tres artículos. Confirma la Transacción.

Procesa la operación donde el Cliente Mario Canseco Avilés compre Protectores de Pantallas por una cantidad de 5 artículos. Cancela la operación pero confirma sólo por 3 artículos en virtud de que regresaron dos artículos.

Selecciona con comandos Select aquellos clientes que han comprado más de dos artículos.

Selecciona con un comando Select aquellos clientes que no han comprado ningún artículo.

Realiza tu actividad en un procesador de textos, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza el archivo, ya seleccionado, presiona **Subir este archivo** para guardarlo en la plataforma.

GLOSARIO

Cross join

El resultado es un producto cartesiano, es decir, una combinación de todos los valores de una tabla contra todos los valores de otra tabla.

Foreign key

Es una restricción en una tabla donde enlaza con la llave principal de otra tabla.

Inner join

El resultado es un conjunto de registros que resultan de la combinación de dos o más tablas, siempre y cuando existan columnas en común y los valores de dichas columnas coincidan. Este *join* necesita de una cláusula ON que iguale las columnas en común.

Integridad

La integridad de una base de datos se establece con restricciones, en inglés *constraints*.

Outer join

El resultado es un *inner join* que además incluye aquellos valores donde no hay coincidencia en el origen del lado izquierdo (LEFT OUTER JOIN) o del lado derecho (RIGHT OUTER JOIN) o aquellos que no coinciden en ningún lado (FULL OUTER JOIN).

Primary Key

Es la llave primaria en una tabla con la cual se pueden buscar de forma eficiente sus datos.

Procedimiento almacenado (*stored procedure*)

Es un conjunto de instrucciones en un lenguaje de programación que se almacenan como un objeto de la base de datos y puede ser ejecutado en cualquier momento.

Tablas

Son un conjunto de filas y columnas que nos permiten almacenar los datos bajo el enfoque de un modelo relacional.

Transacción de base de datos

Es un conjunto de dos o más instrucciones que modifican la información de la base (UPDATE, DELETE o INSERT), las cuales son tratadas como una unidad, de tal forma que se realizan todas o no se realiza ninguna.

Trigger

Es una función que se “dispara” antes o después de la instrucción que actualiza la tabla a la que está asociado.

Vista

Es un objeto de la base de datos que almacena una consulta. Funciona como una tabla, pero no existe físicamente en la base de datos, se genera de forma dinámica.

MESOGRAFÍA

Bibliografía básica

Bibliografía complementaria

Sitios electrónicos