

***SQL DDL***

***Triggers***

**UTN - FRBA**

**Ing. en Sistemas de Información**

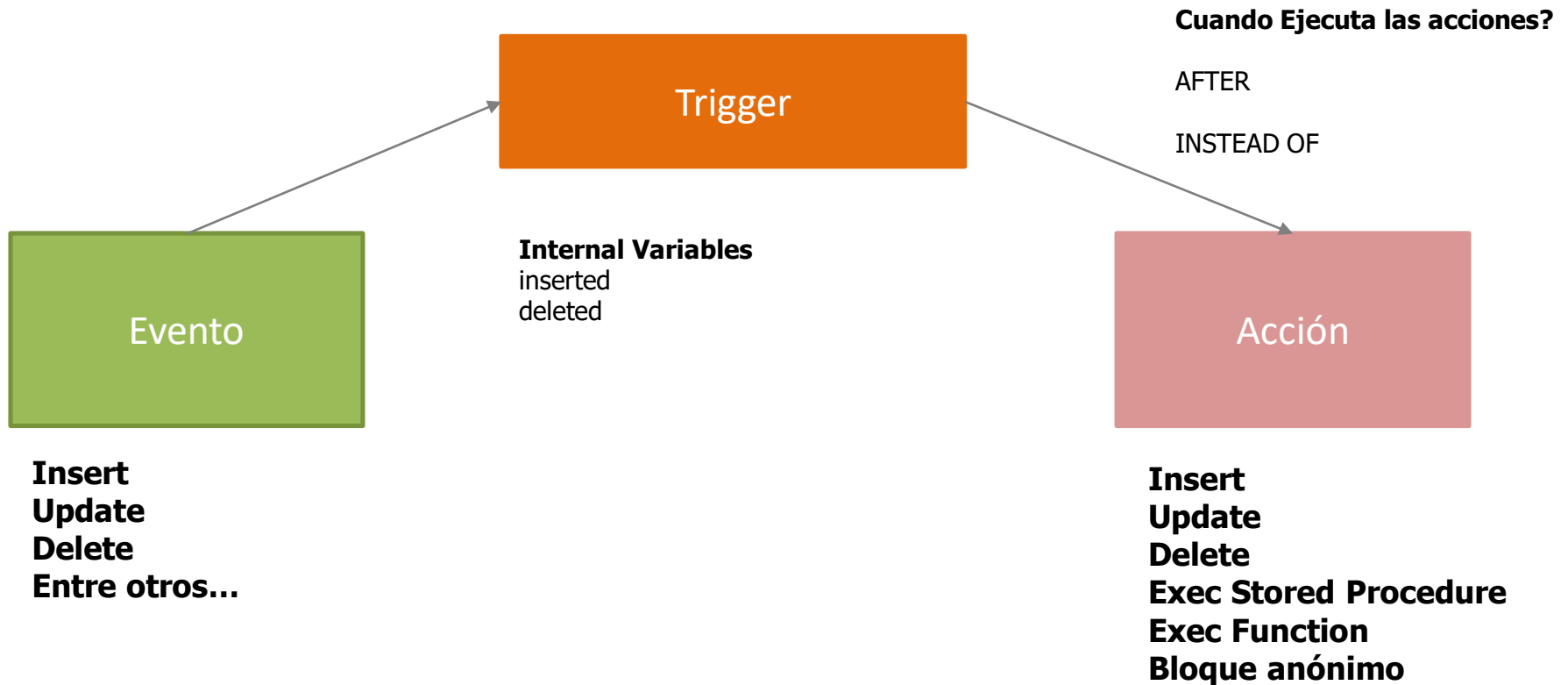
***Gestión de Datos***

**Prof.: Ing. Juan Zaffaroni**

# Triggers

Qué es un Trigger?

Es un mecanismo que ejecuta una o varias sentencias de SQL automáticamente cuando cierto evento ocurre. Generalmente está asociado a uno o varios eventos sobre una determinada tabla.



# Triggers

## PORQUE USAR TRIGGERS?

- Se pueden aplicar las reglas de negocio  
Ej.: Si el inventario de una columna pasa x valor, entonces insertar un pedido en la tabla de compras
- Valores de columnas derivadas  
Ej.: Si modifico la cantidad o precio de un Item de una Orden de Compra, se actualiza el total de la cabecera automáticamente.
- En algunos casos es necesario que ante tal acción se actualicen datos en base a otros.
- Replicación automática de tablas  
Ej. Implementar en mi sistema un esquema de replicación de tablas clave a otro Site en forma controlada por la misma aplicación.

# Triggers

## PORQUE USAR TRIGGERS?

- Implementación de Foreign Keys sobre Tablas de otras BD.  
Ej. Los motores de BD Relacionales no permiten implementar FK contra PKs de tablas de otras BD. Un Triggers nos lo permitiría.
- Logs. De Auditoría  
Ej. Implementar en mi sistema un esquema de auditoria de cambios en tablas clave de nuestro sistema, en forma controlada por la misma aplicación.
- Delete en Cascada (No recomendado)  
Ej.: Implementar un esquema de borrado en cascada, reemplazaria al ON DELETE CASCADE de las Foreign key.
- Autorización de Seguridad  
Ej. Ante el acceso a la tabla de Cierre de Caja, realizar un chequeo en la tabla de horarios por cajero para ver si el usuario que la realiza esta habilitado en dicho horario para dicha operación.

# Triggers

## Eventos Posibles

### Instrucciones DML sobre Tablas o Views

- INSERT ON tab\_name
- DELETE FROM tab\_name
- UPDATE tab\_name
- UPDATE of col\_name ON tab\_name
- SELECT tab\_name (Informix)
- SELECT of col\_name ON tab\_name (Informix)

### Instrucciones DDL sobre Base de Datos (Oracle/Sql Server)

- CREATE
- ALTER
- DROP

# Triggers

## Eventos Posibles (Cont.)

### Operaciones de Base de Datos (Oracle/Sql Server)

- SERVERERROR
- LOGON
- LOGOFF
- STARTUP
- SHUTDOWN

En algunos motores se permiten múltiples triggers sobre una tabla, pero sólo 1 por tipo. Para el caso de UPDATE las columnas deben ser mutuamente exclusivas.

La tabla especificada por el trigger debe estar en modo local, en general los motores no aceptan tablas en servers remotos.

# Triggers

## Transaccionabilidad entre Evento y Acción

Tanto el evento cómo las acciones que ejecuta el trigger conforman una transacción, si alguno de los dos falláse, se realiza un roll back automático.

# Triggers

## Momentos de Ejecución de Acciones

Los momentos de ejecución de las con acciones ANTES, DURANTE y DESPUES del evento asociado al Trigger.

Existe una opción de INSTEAD OF que se reemplaza el EVENTO por la ACCION ejecutada.

- BEFORE
- FOR EACH ROW
- AFTER
- INSTEAD OF



# Triggers

## Momentos de Ejecución de Acciones (Cont.)

- BEFORE (execute procedure xyz()) → Se ejecuta antes de que el evento de trigger ocurra (Oracle / Informix)
- AFTER (execute procedure xyz()) → Se ejecuta después de que el evento de trigger ocurra
- FOR EACH ROW (execute procedure xyz()) → Se ejecuta para cada una de las filas del evento. (existen diferencias de uso entre motores) (Oracle / Informix)
- INSTEAD OF (Sql Server / Oracle/Informix) → Se ejecuta las acciones en lugar del evento de trigger recibido. Reemplaza el evento que disparó el trigger por la/s acción/es del trigger.

En Oracle/Informix sólo pueden ser utilizados en Triggers sobre VIEWS.

En SQLServer pueden ser utilizados tanto en Triggers sobre TABLAS como sobre VIEWS.

# Triggers

Tablas/Vistas del Diccionario de Datos Asociadas

`sys.triggers`

`sys.objects`

`sys.all_sql_modules`

Estas son algunas de las tablas que contienen información de la Metadata de todos los DB Objects creados en nuestra BD. En este caso puntual, de los triggers.

# Triggers

Veamos el apunte de Transact-SQL  
para Stored Procedures

# Triggers

Pasemos a los ejemplos...

# Triggers

## Trigger de Auditoria sobre Tabla

```
CREATE TABLE state_upd(  
    id_auditoria INT IDENTITY(1,1),  
    code char(2) NOT NULL,  
    sname varchar(15) NULL,  
    accion char,  
    fechaYHora datetime  
)
```

Previamente Creamos tabla de auditoria para tabla state.

```
CREATE TRIGGER stateUpdateAudit  
ON state  
AFTER update  
AS  
BEGIN  
    INSERT INTO state_upd  
    SELECT code, sname, 'A', getdate() from deleted;  
  
    INSERT INTO state_upd  
    SELECT code, sname, 'N', getdate() from inserted;  
END
```

# Triggers

## Trigger de Auditoria sobre Tabla

### Ejecución de Prueba

```
update state SET sname='AZ...'
WHERE state='AZ'
```

```
SELECT * FROM STATE_UPD
SELECT * FROM state WHERE state='AZ';
```

100 %

Results Messages

	id_auditoria	state	sname	accion	fechaYHora
1	1	AZ	Arizona	A	2020-05-06 09:04:43.503
2	2	AZ	AZ...	N	2020-05-06 09:04:43.507

	state	sname
1	AZ	AZ...

Luego de crear el trigger se observa que el update de ciertos datos fue auditado en la tabla state\_upd.

Otro ejemplo.

```
update state SET sname='Arizona'
WHERE state='AZ'
SELECT * FROM STATE_UPD
SELECT * FROM state WHERE state='AZ';
```

100 %

Results Messages

	id_auditoria	state	sname	accion	fechaYHora
1	1	AZ	Arizona	A	2020-05-06 09:04:43.503
2	2	AZ	AZ...	N	2020-05-06 09:04:43.507
3	3	AZ	AZ...	A	2020-05-06 09:06:29.453
4	4	AZ	Arizona	N	2020-05-06 09:06:29.453

	state	sname
1	AZ	Arizona

# Triggers

## Tablas de Diccionario de Datos

### Ejecución de Prueba

```
SELECT * FROM SYS.TRIGGERS
```

100 %

Results Messages

Datos particulares de cada trigger

	name	object_id	parent_class	parent_class_desc	parent_id	type	type_desc	create_date	modify_date
1	upd_items_ordenes	1525580473	1	OBJECT_OR_COLUMN	437576597	TR	SQL_TRIGGER	2020-05-06 08:36:41.930	2020-05-06 08:36:41.930
2	stateUpdateAudit	1637580872	1	OBJECT_OR_COLUMN	565577053	TR	SQL_TRIGGER	2020-05-06 09:04:19.223	2020-05-06 09:04:19.223

```
select * from sys.objects where type='TR'
```

100 %

Results Messages

Tabla que contiene data de todos los objetos

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped
1	upd_items_ordenes	1525580473	NULL	1	437576597	TR	SQL_TRIGGER	2020-05-06 08:36:41.930	2020-05-06 08:36:41.930	0
2	stateUpdateAudit	1637580872	NULL	1	565577053	TR	SQL_TRIGGER	2020-05-06 09:04:19.223	2020-05-06 09:04:19.223	0

```
SELECT * FROM SYS.all_sql_modules  
WHERE object_id=1525580473
```

100 %

Results Messages

Tabla que contiene la definición de cada objeto.

	object_id	definition	uses_ansi_nulls	uses_quoted_identifier	is_schema_bound	uses_database_collation	is_dml_trigger
1	1525580473	CREATE TRIGGER upd_items_ordenes ON items AFTE...	1	1	0	0	0

# Triggers

## Transacciones- evento-accion implícitas en Triggers

### Ejecución de Prueba

Forzaremos un error en el trigger queriendo insertar un nulo en la tabla de auditoria, no contemplado.

```
DROP TRIGGER stateUpdateAudit
CREATE TRIGGER stateUpdateAudit
ON state
AFTER update
AS
BEGIN
INSERT INTO state_upd
SELECT state,sname,'A',getdate() from deleted;

INSERT INTO state_upd
SELECT null,sname,'N',getdate() from inserted;
-- INSERTO UN NULO EN LA TABLA STATE_UPD FORZANDO ERROR

END
```



# Triggers

## Transacciones- evento-accion implícitas en Triggers

### Ejecución de Prueba

```
update state SET sname='AZ...'
WHERE state='AZ'
```

100 %

Messages

(1 row(s) affected)  
Msg 515, Level 16, State 2, Procedure stateUpdateAudit, Line 9 [Batch Start Line 44]  
Cannot insert the value NULL into column 'state', table 'stores7ParaRomper.dbo.state\_upd'; column does not allow nulls.  
The statement has been terminated.

```
SELECT * FROM STATE_UPD
SELECT * FROM state WHERE state='AZ';
```

100 %

Results Messages

	id_auditoria	state	sname	accion	fechaYHora
1	1	AZ	Arizona	A	2020-05-06 09:04:43.503
2	2	AZ	AZ...	N	2020-05-06 09:04:43.507
3	3	AZ	AZ...	A	2020-05-06 09:06:29.453
4	4	AZ	Arizona	N	2020-05-06 09:06:29.453

Misma cantidad de filas ejemplo anterior.

	state	sname
1	AZ	Arizona

Luego de crear el trigger se observa que el error en el insert de state\_upd hizo que nada se actualice. En definitiva hubo rollback de acción y evento.

# Triggers

## Trigger de INSTEAD OF sobre Vista

```
CREATE VIEW ordenes_por_cliente
    (cod_cliente, nombre, apellido, nro_orden, fecha_orden)
AS
SELECT c.customer_num, fname, lname, order_num, order_date
FROM customer c JOIN orders o
    ON (c.customer_num = o.customer_num)
```

```
CREATE TRIGGER InsertaClienteyOrdenEnVista
ON ordenes_por_cliente
INSTEAD OF INSERT
AS
BEGIN

    INSERT INTO customer (customer_num, fname, lname)
    SELECT cod_cliente,nombre,apellido FROM inserted;

    INSERT INTO orders (order_num, order_date, customer_num)
    SELECT nro_orden, fecha_orden, cod_cliente FROM inserted;
END
```

# Triggers

## Trigger de INSTEAD OF sobre Vista

### Ejecución de Prueba

```
INSERT INTO ordenes_por_cliente
VALUES (1666, 'Filomeno', 'Mas', 2666, '2020-04-23');
SELECT customer_num, fname, lname
FROM customer WHERE customer_num=1666;
SELECT order_num, customer_num, order_date
FROM orders WHERE order_num=2666;
```

100 %

Messages

Msg 4405, Level 16, State 1, Line 23  
View or function 'ordenes\_por\_cliente' is not updatable because the modification affects multiple base tables.

```
INSERT INTO ordenes_por_cliente
VALUES (1666, 'Filomeno', 'Mas', 2666, '2020-04-23');
SELECT customer_num, fname, lname
FROM customer WHERE customer_num=1666;
SELECT order_num, customer_num, order_date
FROM orders WHERE order_num=2666;
```

100 %

Results Messages

	customer_num	fname	lname
1	1666	Filomeno	Mas

	order_num	customer_num	order_date
1	2666	1666	2020-04-23 00:00:00.000

Luego de crear el trigger se observa que el insert funciona insertando los datos correctos en cada tabla.

# Triggers

## Trigger de AFTER Update con ejecución de Bloque anónimo Modificación automática de columnas derivadas

```
CREATE TRIGGER upd_items_ordenes
ON items
AFTER UPDATE
AS
BEGIN
    DECLARE @i_prec_del  dec(8,2), @n_orden int, @i_prec_ins  dec(8,2) ,
            @quantity_del int, @quantity_ins int;

    SELECT @i_prec_del=unit_price, @quantity_del=quantity
    FROM deleted;

    SELECT @n_orden= order_num, @i_prec_ins=unit_price, @quantity_ins=quantity
    FROM inserted;

    IF UPDATE (unit_price) OR UPDATE(quantity)
    BEGIN
        UPDATE orders
        SET total= total
            -(@quantity_del*@i_prec_del)
            +(@quantity_ins*@i_prec_ins)
        WHERE order_num = @n_orden;
    END
END
```

# Triggers

## Trigger de AFTER Update con ejecución de Bloque anónimo Modificación automática de columnas derivadas

Para implementar este ejemplo previamente creamos un total en la tabla Orders y actualizamos dicho campo.

```
ALTER TABLE orders ADD total decimal(12,2)
```

```
UPDATE orders SET total=  
(SELECT SUM(unit_price*quantity) FROM items i  
WHERE i.order_num= orders.order_num)
```

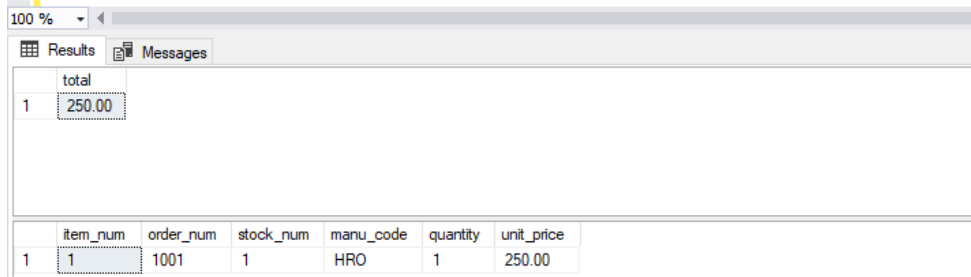
```
CREATE TRIGGER...
```

# Triggers

Trigger de AFTER Update con ejecución de Bloque anónimo  
Modificación automática de columnas derivadas

## Ejecución de Prueba

```
select total from orders where order_num=1001  
select * from items where order_num=1001
```



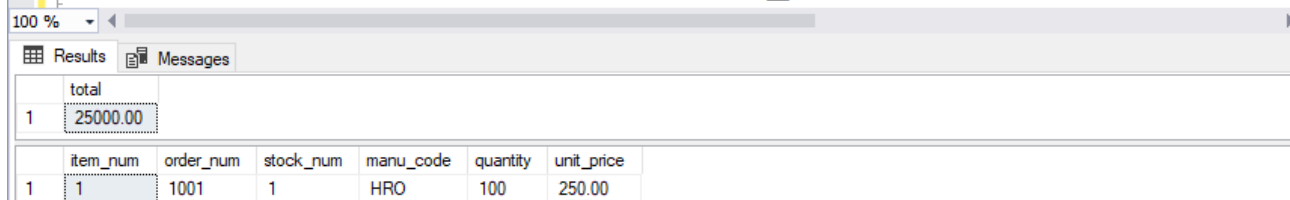
total
250.00

item_num	order_num	stock_num	manu_code	quantity	unit_price
1	1001	1	HRO	1	250.00

Luego de actualizar la cantidad a 100, se observa aplicado el cambio en el campo total de la tabla orders.

```
update items set quantity=100 where order_num=1001  
select total, order_num from orders where order_num=1001  
select * from items where order_num=1001
```



total
25000.00

item_num	order_num	stock_num	manu_code	quantity	unit_price
1	1001	1	HRO	100	250.00

**Vamos a la Practica!!!**

