ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

Bases de datos:

Transaction
Procedure
Trigger
Vistas

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Transacciones

Las transacciones, como los batches, son unidades de trabajo que se compone de una o más instrucciones SQL que realizan un conjunto de acciones relacionadas.

Son admitidas si se ejecutan todas (COMMITED TRANSACTION)

o rechazadas si falla alguna de ellas (ROLLBACK TRANSACTION).

Tipos de Transacciones

- **Explícitas:** hay que indicar cuando se inician
  (BEGIN TRANSACTION)
  y cuando finalizan
  (COMMIT TRANSACTION),
  y pueden albergar un conjunto de instrucciones dentro de la misma transacción.

- **Implícitas:** son de confirmación automática y es el comportamiento predeterminado de SQL SERVER, donde ejecuta por separado cada sentencia Transact-SQL justo después de que se termine dicha sentencia.

Sintaxis

BEGIN TRANSACTION <nombre de la transacción>;
<órdenes de sql>;

COMMIT o ROLLBACK;

Ejemplo

```
BEGIN TRANSACTION;
DELETE FROM HumanResources.JobCandidate
    WHERE JobCandidateID = 13;
COMMIT;
```

Ejemplo

```
CREATE TABLE ValueTable (id int);
BEGIN TRANSACTION;
    INSERT INTO ValueTable VALUES(1);
    INSERT INTO ValueTable VALUES(2);
ROLLBACK;
```

Ejemplo

```
DECLARE @TranName VARCHAR(20);
SELECT @TranName = 'MyTransaction';

BEGIN TRANSACTION @TranName;
USE AdventureWorks2012;
DELETE FROM
   AdventureWorks2012.HumanResources.JobCandidate
    WHERE JobCandidateID = 13;

COMMIT TRANSACTION @TranName;
GO
```

Ejemplo

```
BEGIN TRANSACTION [Tran1]
 BEGIN TRY
    INSERT INTO [Test].[dbo].[T1] ([Title], [AVG])
    VALUES ('Tidd130', 130), ('Tidd230', 230)
    UPDATE [Test].[dbo].[T1]
    SET [Title] = N'az2' ,[AVG] = 1
    WHERE [dbo].[T1].[Title] = N'az'
    COMMIT TRANSACTION [Tran1]
 END TRY
 BEGIN CATCH
    ROLLBACK TRANSACTION [Tran1]
 END CATCH
```

Transacciones

Al realizar una transacción SQL hay que tener en cuenta que apenas se realice un INSERT, UPDATE o DELETE se genera un bloqueo sobre la tabla y que otros clientes no pueden acceder para escribir esta tabla. Otros clientes podrán realizar SELECTs sobre la tabla, pero no podrán ver los datos del primer cliente hasta que los mismos sean confirmados.-

Consideraciones:

- Un grupo de órdenes puede incrementar la performance de las operaciones, ya que todas de una vez realizan la recuperación. Pero si es una lista larga en una misma transacción podría causar problemas de concurrencia.
- No se necesitan permisos para definir transacciones

Procedimientos almacenados

Es un conjunto de comandos SQL que pueden almacenarse en el servidor.

Es ejecutado directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado.

Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes.

Pueden hacer referencia a tablas vistas o otros procedimientos almacenados.

Ventajas

Los procedimientos almacenados pueden mejorar el rendimiento ya que se necesita enviar menos información entre el servidor y el cliente.

- Mejora de la calidad.
- Transparencia y posibilidad de reutilización.
- Reducen el tráfico a través de la red.

Pueden capturar errores antes que ellos puedan entrar a la base de datos. Establece consistencia porque ejecuta las tareas de la misma forma.

Desventajas

- Aumentan el uso de memoria
- Restringidos para una lógica de negocios compleja
- Difíciles de mantener

Implementación

- Estos procedimientos, se usan a menudo, pero no siempre, para realizar consultas SQL sobre los objetos de la base de datos de una manera abstracta, desde el punto de vista del cliente de la aplicación.

  Un **procedimiento almacenado** permite agrupar en forma exclusiva parte de algo específico que se desee realizar o, mejor dicho, el SQL apropiado para dicha acción.

Tipificación

Procedimientos almacenados definidos por el usuario: son:

- Triggers: Son procedimientos definidos por el usuario que se ejecutan automáticamente cuando se modifica un dato en un tabla.
- Procedimientos del sistema: Procedimientos suministrados por el sistema.
- Procedimientos Extendidos: Procedimientos que se hacen llamadas al sistema operativo y ejecutan tareas a ese nivel.

Ventajas en el rendimiento

Un procedimiento almacenado se ejecuta más rápido que un batch porque:

- El procedimiento almacenado ya ha sido analizado.
- Ya se han resuelto las referencias a los objetos referenciados en el procedimiento almacenado.
- No se necesita construir el árbol de búsqueda, el usa el que se hace en el momento de compilarlo.
- No se necesita crear un plan de búsqueda, porque ya el procedimiento tiene uno.

SINTAXIS

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
EXEC procedure_name;
```

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Ejemplo

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
EXEC SelectAllCustomers;
```

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Ejemplo con parámetro

```
CREATE PROCEDURE SelectAllCustomers @City
nvarchar(30)
AS
SELECT * FROM Customers WHERE City =
@City
GO;

EXEC SelectAllCustomers @City = 'London';
```

Ejemplo con varios parámetro

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30),
@PostalCode nvarchar(10)
AS
SELECT * FROM Customers WHERE City = @City AND
PostalCode = @PostalCode
GO;

EXEC SelectAllCustomers @City = 'London', @PostalCode =
'WA1 1DP';
```

VARIABLES

Los procedimientos almacenados pueden crear y usar variables locales, las cuales solo existen mientras exista el procedimiento. Las variables no las puede usar otro proceso.

SENTENCIAS VÁLIDAS

Un procedimiento almacenado puede:

- Seleccionar y modificar datos;
- Crear tablas temporales y permanentes.
- Llamar a otros procedimientos almacenados.
- Referenciar objetos de base de datos.

SENTENCIA INVALIDAS

Un procedimiento no puede ejecutar: - Use database. - Create view. - Create default. - Create rule. - Create procedure. - Create trigger

Trigger

Es un procedimiento almacenado asociado con una tabla, el cual se ejecuta automáticamente cuando se modifica un dato de esa tabla.

Un trigger se define asociado con una tabla para una o más sentencias de manipulación de datos.

Se puede definir para insert, update, o delete o cualquier combinación de ellos.

APLICACIONES TÍPICAS

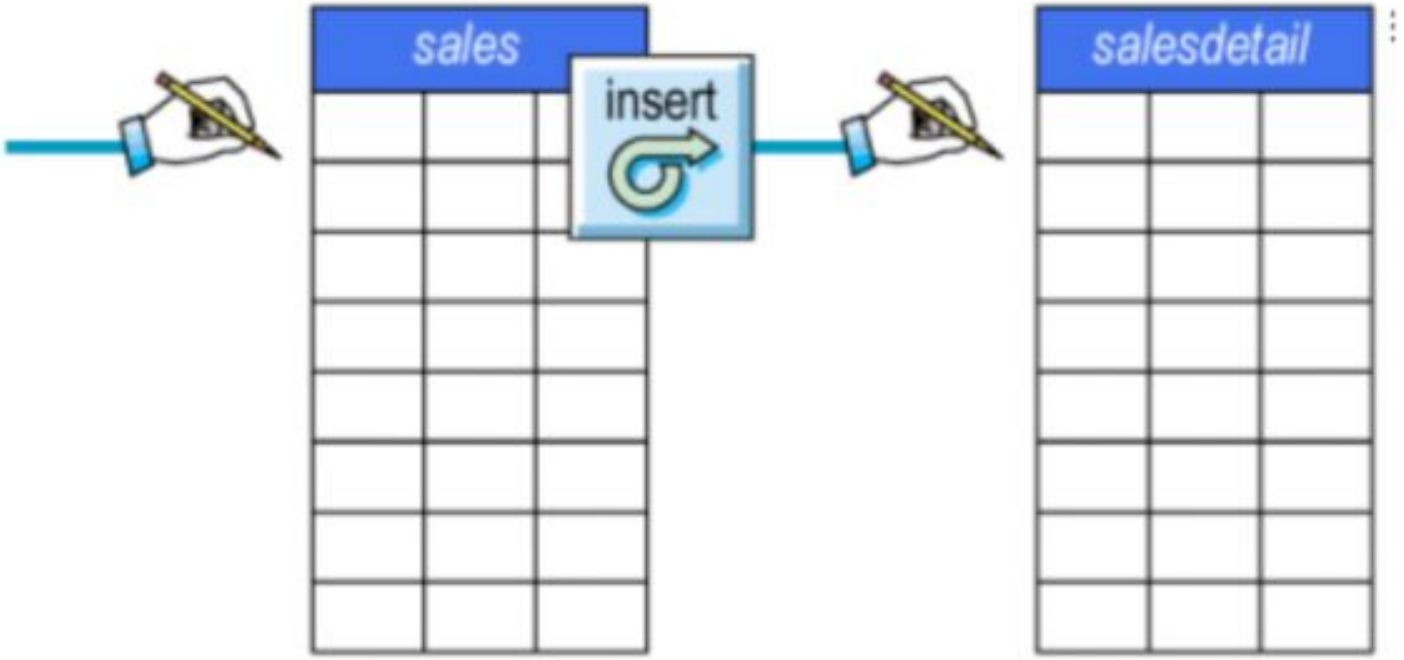
- Hace modificaciones en cascada sobre tablas relacionadas.
- Deshacer cambios que violan la integridad de los datos.
- Forzar restricciones que son muy complejas para reglas y restricciones.
- Mantener datos duplicados.
- Mantener columnas con datos derivados. - Hacer ajustes de registros.

Modificación Trigger

Cuando se modifica un dato en una tabla que tiene declarado un trigger para esa sentencia, el trigger se “dispara”.

El trigger se dispara una vez, independiente del número de filas afectadas.

Trigger



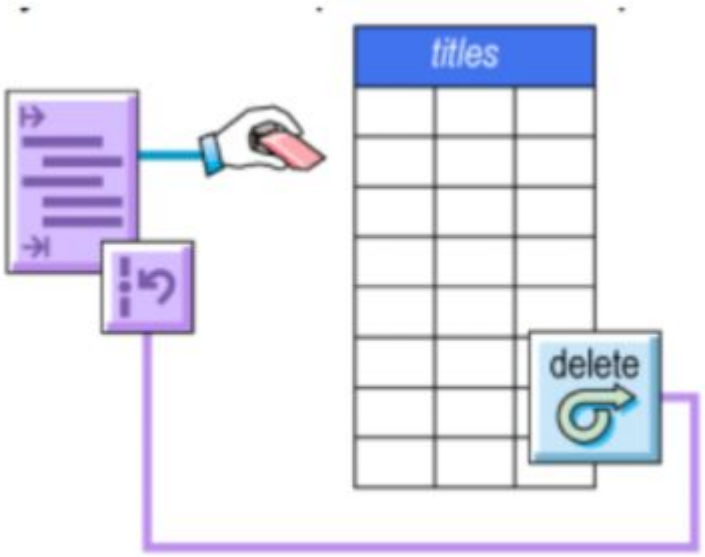
Trigger

Un trigger es parte de la transacción que causa el disparo.

El trigger puede deshacer:

- Así mismo solamente.
- Así mismo y la sentencia que causa el disparo
- La transacción total.

Trigger



Trigger

Un trigger pueden:

- Declarar variables locales.
- Invocar procedimientos almacenados.

Un trigger no puede:

- Llamarse directamente.
- Usar parámetros.
- Definirse sobre tablas temporales o vistas.
- Crear objetos permanentes de la base de datos.

Las operaciones con registro mínimo (como select into) no disparan los triggers.

Consideraciones

Los triggers se pueden ejecutar antes (BEFORE) y/o después (AFTER) de que sean modificados los datos.

Los triggers tienen dos palabras clave, OLD y NEW que se refieren a los valores que tienen las columnas antes y después de la modificación.

Los INSERT permiten NEW, los DELETE solo OLD y los UPDATE ambas.

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

```
-- SQL Server Syntax
-- Trigger on an INSERT, UPDATE, or DELETE statement to a table or view (DML
   Trigger)

CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name
```

```
-- SQL Server Syntax
-- Trigger on an INSERT, UPDATE, or DELETE statement to a
-- table (DML Trigger on memory-optimized tables)

CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS { sql_statement  [ ; ] [ ,...n ] }

<dml_trigger_option> ::=
    [ NATIVE_COMPILATION ]
    [ SCHEMABINDING ]
    [ EXECUTE AS Clause ]
```

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

```
-- Trigger on a CREATE, ALTER, DROP, GRANT, DENY,
-- REVOKE or UPDATE statement (DDL Trigger)

CREATE [ OR ALTER ] TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n
  ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <
  method specifier >  [ ; ] }

<ddl_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
```

```
-- Trigger on a LOGON event (Logon Trigger)

CREATE [ OR ALTER ] TRIGGER trigger_name
ON ALL SERVER
[ WITH <logon_trigger_option> [ ,...n ] ]
{ FOR| AFTER } LOGON
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL
  NAME < method specifier >  [ ; ] }

<logon_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
```

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

SENTENCIA SQL	OLD	NEW
Insert	No definido; todos los campos toman valor NULL	Valores que serán insertados cuando se complete la orden.
Update	Valores originales de la fila, antes de la actualización	Nuevos valores que serán escritos cuando se complete la orden.
Delete	Valores antes del borrado de la fila.	No definido; todos los campos toman el valor NULL.

INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Ejemplos

```
CREATE TRIGGER reminder1
ON Sales.Customer
AFTER INSERT, UPDATE
AS RAISERROR ('Notify Customer Relations',
  16, 10);
GO
```

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Ejemplos

```
CREATE TRIGGER reminder2
ON Sales.Customer
AFTER INSERT, UPDATE, DELETE
AS
   EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'AdventureWorks2012 Administrator',
        @recipients = 'danw@Adventure-Works.com',
        @body = 'Don''t forget to print a report for the
  sales force.',
        @subject = 'Reminder';
GO
```

Funciones de una vista

Una vista es un SELECT almacenado.

El motor distingue dos tipos de tabla, las base y las derivadas.

Una tabla base es una tabla existente en el motor de almacenamiento.

Una tabla derivada, es la tabla que surge de cualquier combinación de tablas base, literales y/o funciones.

Funciones de una vista

Proporcionar un nivel de seguridad, ya que permiten excluir datos para que ciertos usuarios no los vean.-

- Proporcionan un mecanismo para que los usuarios vean los datos en el formato que deseen.-

- Representan una imagen consistente y permanente de la base de datos, incluso si la base de datos cambia su estructura.

Sintaxis

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Sintaxis

Ejemplo:

```
CREATE VIEW [Brazil Customers] AS
SELECT CustomerName, ContactName
FROM Customers
WHERE Country = 'Brazil';

SELECT * FROM [Brazil Customers];
```

Sintaxis

```
CREATE VIEW [Products Above Average Price] AS
SELECT ProductName, Price
FROM Products
WHERE Price > (SELECT AVG(Price) FROM
  Products);
```

y se consulta la vista anterior así:

```
SELECT * FROM [Products Above Average Price];
```

ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

https://docs.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver15

https://www.w3schools.com/sql/sql_stored_procedures.asp

ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

Preguntas