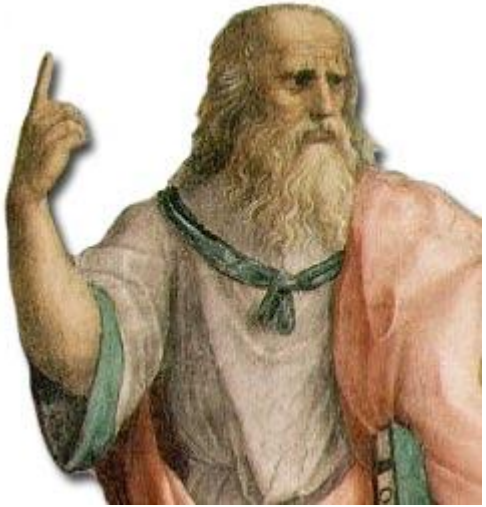
Transacciones, SQL Server

1º CFGS

Transacciones - definición



“Es un grupo de una o varias instrucciones,cuyo resultado final debe ser que se ejecuten todas o ninguna de ellas”

Transacciones - definición

Resultado

Si la transacción se realiza correctamente, se confirman todas las instrucciones que contiene. Si se produce un error en al menos una de ellas, se revierte todo el grupo.

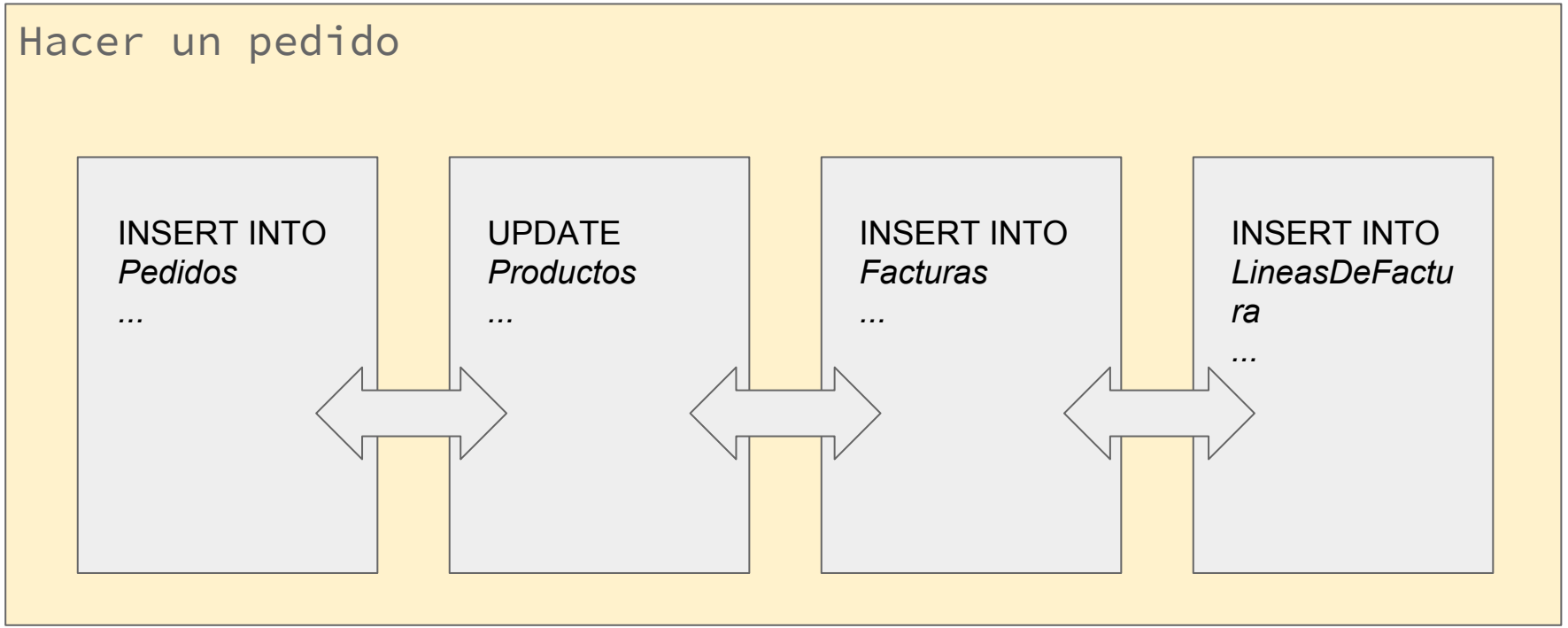
De esta forma se evita que **el sistema de datos quede en un estado incongruente.**

Transacciones - ejemplo

Comprar un producto en una página web

1. Comprobar que nuestra cuenta existe es válida y está operativa
2. Comprobar si hay saldo en nuestra cuenta
3. Comprobar los datos de la cuenta del vendedor (que existe, que tiene posibilidad de recibir dinero, etc...)
4. Retirar el dinero de nuestra cuenta
5. Ingresar el dinero en la cuenta del vendedor

Transacciones - ejemplo



Transacciones - propiedades

Para cumplir con su propósito las transacciones presentan características **ACID**

- **Atomicidad**, las operaciones de la transacción deben considerarse como una sola
- **Consistencia**, una operación nunca dejará datos inconsistentes
- **Aislamiento**, la información no confirmada no está disponible para el usuario
- **Durabilidad**, una vez completada la transacción los datos actualizados ya serán permanentes y confirmados.

Transacciones - uso

1. Cuando tengamos que realizar una operación que requiera actualizar más de una tabla:

   Transacción compra en página web...

2. Cuando tengamos que leer datos y hacer alguna actualización en función de los mismos:

   Gestión del pedido...

Transacciones - estructura general SQL server

```
BEGIN TRANSACTION -- O solo BEGIN TRAN

BEGIN TRY

/* Aquí irán las instrucciones que forman la transacción */

/* Confirmamos la transaccion*/

COMMIT TRANSACTION -- O solo COMMIT

END TRY

BEGIN CATCH

/* Hay un error, deshacemos los cambios*/

ROLLBACK TRANSACTION -- O solo ROLLBACK

END CATCH
```

Transacciones - consideraciones

- Si se produce un error se ejecutará ROLLBACK, si no, se confirman todas las operaciones.
- El COMMIT sólo se ejecutará si llegamos a él sin que ningún error haya detenido la ejecución del script.
- Estas operaciones pueden formar parte unas de otras, lo que daría lugar a transacciones anidadas.
- Una transacción sólo tiene sentido si vamos a hacer una actualización (INSERT, UPDATE, DELETE).

Transacciones - Excepciones sql server

Mecanismo de control de errores similar al control de excepciones de C# y C++

Si se produce un error en el bloque TRY el control se transfiere al grupo de instrucciones incluido en un bloque CATCH

https://docs.microsoft.com/es-es/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-2017

Transacciones - Excepciones sql server

En el ámbito de un bloque CATCH, se pueden utilizar las siguientes funciones del sistema para obtener información acerca del error que provocó la ejecución del bloque CATCH:

ERROR_NUMBER() devuelve el número del error.

ERROR_SEVERITY() devuelve la gravedad.

ERROR_STATE() devuelve el número de estado del error.

Transacciones - Excepciones sql server

ERROR_PROCEDURE() devuelve el nombre del procedimiento almacenado o desencadenador donde se produjo el error.

ERROR_LINE() devuelve el número de línea de la rutina que provocó el error.

ERROR_MESSAGE() devuelve el texto completo del mensaje de error. El texto incluye los valores proporcionados para los parámetros sustituibles, como las longitudes, nombres de objeto o tiempos.

Transacciones - Excepciones sql server

```
-- Create procedure to retrieve error information.
CREATE PROCEDURE usp_GetErrorInfo
AS
SELECT
    ERROR_NUMBER() AS ErrorNumber
    ,ERROR_SEVERITY() AS ErrorSeverity
    ,ERROR_STATE() AS ErrorState
    ,ERROR_PROCEDURE() AS ErrorProcedure
    ,ERROR_LINE() AS ErrorLine
    ,ERROR_MESSAGE() AS ErrorMessage;
GO

BEGIN TRY
    -- Generate divide-by-zero error.
    SELECT 1/0;
END TRY
BEGIN CATCH
    -- Execute error retrieval routine.
    EXECUTE usp_GetErrorInfo;
END CATCH;
```

Transacciones - raiserror

RAISERROR Genera un mensaje de error que se devuelve a un bloque **CATCH**. Usos:

1. Cadena de mensaje:

```
RAISERROR('Error insertando pedidos', 17, 15)
```

Los dos parámetros corresponden a:

severity = nivel de gravedad definido por usuario (0 - 18)

state = Entero entre 0 y 255.

Transacciones - raiserror

2.Guardar y rescatar desde la tabla de sistema, vista de catálogo sysmessages

Añadimos el registro con lang español e inglés.

```
EXEC  @msgnum = 50100, @severity = 7, @msgtext = 'Error in orders',@lang = 'us_english',@replace='replace'
```

```
EXEC sp_addmessage @msgnum = 50100, @severity = 7, @msgtext = 'Error en pedidos', @lang ='spanish',@replace='replace'
```

Transacciones - raiserror

Recuperamos el error

```
RAISERROR (50100, 7, 2)
```

También se pueden usar los mensajes del sistema con **RAISERROR** ID < 50000.

Algunos ejemplos

```
-- Todos los mensajes personalizados en español
select * from master.dbo.sysmessages where msglangid=3082 and error > 50000
-- Todos los mensajes personalizados
select * from master.dbo.sysmessages where error > 50000
select * from master.dbo.sysmessages where msglangid=3082 and description LIKE '%cadena%'
```

Transacciones - raiserror

```
-- Con parámetros
-- Primero en inglés. Se usan los descriptores de formato tipo C
EXEC sp_addmessage @msgnum = 50101, @severity = 7, @msgtext = 'This is brutaly crashed on
%d point.',@lang = 'us_english',@replace='replace'

-- En los demás idiomas se usan placeholders (1, 2, 3...)
EXEC sp_addmessage @msgnum = 50101, @severity = 7, @msgtext = 'Esto ha cascado brutalmente
en el punto %1.', @lang ='spanish',@replace='replace'
-- Y lo probamos
RAISERROR (50101,7,2,125)
```

→ A partir de SQL Server 2012 se utiliza **THROW**