DuocUC

SQL SERVER APLICADO (SSA010)

Ariel Alexis Fierro Sáez
afierrosaez@gmail.com

Vistas (View)

- Una vista es un resultado de una consulta SQL de una o varias tablas que suele ser considerada como una tabla virtual.
- Una vista no almacena datos de la consulta, sino solo la definición.
- La recuperación de datos desde una vista se realiza al igual que si se consultara una tabla física de la base de datos.

Aplicaciones Vistas

- Información derivada de consultas complejas que son accedidas frecuentemente.
- Obtención de información derivada de la relación entre varias tablas o estadística.
- Como mecanismo de seguridad para controlar solo la lectura de cierta información para algunos usuarios.
- Etc.

Sintaxis View

- Crear vista

```
CREATE VIEW  EMPLEADOS_MUJER AS
( SELECT * FROM EMPLEADO WHERE GENERO='M');
```

- Modificar vista

```
ALTER VIEW  EMPLEADOS_MUJER AS
( SELECT * FROM EMPLEADO WHERE GENERO='M' AND
EDAD>30);
```

- Eliminar vista

```
DROP VIEW EMPLEADOS_MUJER;
```

¿Que es una transacción?

- Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo que accede al contenido de una base de datos o modifica.

- La transacción representan eventos del mundo real como por ejemplo:

  - Realizar una transferencia electrónica
  - Registrar un nuevo cliente
  - Realizar un compra por internet
  - etc.


SITIO WEB DEL COMERCIO
web tech solutions
1
Botón de Pagos
Pedido
Pagar
Envió de información
Medio POST
2
Pague
3
Negativa
Evaluación Modulo Antifraude
Positiva
Carito de Compras
Página de respuesta
Gracias por comprar
Envió de la información de la transacción
4
Sistema Transaccional
Redes Tarjetas Crédito y/o Entidad Bancaria
Validación Manual
5
Página de confirmación
Actualización información del pedido
Envió del estado final de la transacción

¿Que es una transacción?

- La unidad lógica de trabajo describe las actividades necesarias ha realizar en la base de datos con el objetivo de hacer que la transacción sea el reflejo de la realidad.

- Dicha unidad lógica debe exhibir cuatro propiedades, conocidas como propiedades de atomicity, consistency, insalation y durability (ACID), para ser calificada como transacción.

Propiedad ACID

- Atomicidad

  Una transacción debe ser una unidad atómica de trabajo, tanto si se realizan todas sus modificaciones en los datos, como si no se realiza ninguna de ellas.

- Consistencia

  Cuando finaliza, una transacción debe dejar todos los datos en un estado coherente.

Propiedad ACID

- Aislamiento

  Las modificaciones realizadas por transacciones simultáneas se deben aislar de las modificaciones llevadas a cabo por otras transacciones simultáneas

- Permanencia

  Una vez concluida una transacción, sus efectos son permanentes en el sistema

Éxito y Fracaso Transacciones

- Si una transacción tiene éxito, todas las modificaciones de los datos realizadas durante la transacción se confirman y se convierten en una parte permanente de la base de datos.

- Si una transacción encuentra errores y debe cancelarse o revertirse, se borran todas las modificaciones de los datos.

Tipos de transacciones

- **Transacciones de confirmación automática**

  Es el modo de gestión de transacciones predeterminado de SQL Server.

- **Transacciones explicitas**

  Es una transacción que define el inicio y fin de la transacción. Donde cada transacción comienza con la sentencia BEGIN TRANSACCION y termina con la sentencia COMMIT o ROLLBACK.

- **Transacciones implícitas**

  Cuando una conexión funciona en modo de transacciones implícitas, se inicia automáticamente una nueva transacción después de confirmar o revertir la transacción actual.

Transacción de confirmación automática

```
CREATE TABLE TestBatch (Cola INT PRIMARY KEY, Colb CHAR(3));

INSERT INTO TestBatch VALUES (1, 'aaa');
INSERT INTO TestBatch VALUES (2, 'bbb');
INSERT INTO TestBatch VALUSE (3, 'ccc');  -- Syntax error.

SELECT * FROM TestBatch;  -- Returns no rows.
```

Transacción explicita

```
BEGIN TRANSACTION
BEGIN TRY
    /* Descontamos el importe de la cuenta origen */
    UPDATE CUENTAS
    SET SALDO = SALDO - 40000        -- Importa transferido
    WHERE NUMCUENTA = 1233456789     -- Cuenta origen
    /* Registramos el movimiento */
    INSERT INTO MOVIMIENTOS
    (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
    IMPORTE, FXMOVIMIENTO)
    SELECT IDCUENTA, SALDO + 40000, SALDO, 40000, getdate()
    FROM CUENTAS
    WHERE NUMCUENTA = 987654321 -- Cuenta destino
    /* Incrementamos el importe de la cuenta destino */
    UPDATE CUENTAS
    SET SALDO = SALDO + 40000
    WHERE NUMCUENTA = 987654321
    /*  Registramos el movimiento  */
    INSERT INTO MOVIMIENTOS
    (IDCUENTA, SALDO_ANTERIOR, SALDO_POSTERIOR,
    IMPORTE, FXMOVIMIENTO)
    SELECT IDCUENTA, SALDO - 40000, SALDO, 40000, getdate()
    FROM CUENTAS
    WHERE NUMCUENTA = 987654321
    /* Confirmamos la transaccion */
    COMMIT TRANSACTION  -- o solo COMMIT
END TRY
BEGIN CATCH
    /* Hay un error, deshacemos los cambios */
    ROLLBACK TRANSACTION -- O solo ROLLBACK
    PRINT 'Se ha producido un error!'
END CATCH
```

Transacción implícita

```
CREATE TABLE ImplicitTran
    (Cola int PRIMARY KEY,
    Colb char(3) NOT NULL);

SET IMPLICIT_TRANSACTIONS ON;

-- First implicit transaction started by an INSERT statement.
INSERT INTO ImplicitTran VALUES (1, 'aaa');
INSERT INTO ImplicitTran VALUES (2, 'bbb');

-- Commit first transaction.
COMMIT TRANSACTION;

-- Second implicit transaction started by a SELECT statement.
SELECT COUNT(*) FROM ImplicitTran;
INSERT INTO ImplicitTran VALUES (3, 'ccc');
SELECT * FROM ImplicitTran;

-- Commit second transaction.
COMMIT TRANSACTION;

SET IMPLICIT_TRANSACTIONS OFF;
```

Transacciones anidadas

```
BEGIN TRANSACTION

UPDATE EMPLEADOS
SET NOMBRE = 'Jose'
WHERE ID=101

    BEGIN TRANSACTION

    UPDATE EMPLEADOS
    SET APELLIDO1 = 'Saez'
    WHERE ID=106

    -- Este COMMIT solo afecta a la segunda transaccion.
    COMMIT

-- Este ROLLBACK afecta a las dos transacciones.
ROLLBACK
```

Puntos de recuperación (SavePoint).

- Los puntos de recuperación permiten manejar las transacciones pasos a paso, pudiendo hacer **ROLLBACK** hasta un punto marcado por el savepoint y no por toda la transacción.

Ejemplo SavePoint

```
BEGIN TRANSACTION
    BEGIN TRY
    UPDATE EMPLEADOS
    SET NOMBRE = 'Guillermo'
    WHERE ID=101


    UPDATE EMPLEADOS
    SET APELLIDO1 = 'Diaz'
    WHERE ID=101

    SAVE TRANSACTION P1 -- Guardamos la transaccion (Savepoint)

    UPDATE EMPLEADOS
    SET APELLIDO1 = 'Jose'
    WHERE ID=101

    COMMIT -- Confirma transaccion
    END TRY

    BEGIN CATCH
    /*  Este ROLLBACK afecta solo a las instrucciones
        posteriores al savepoint P1. */
    ROLLBACK TRANSACTION P1
    END CATCH
```

Referencias

- Transacciones (motor de la base de datos)
  http://msdn.microsoft.com/es-es/library/ms190612%28v=sql.90%29.aspx
- Transacciones (transct-sql)
  http://msdn.microsoft.com/es-es/library/ms174377%28v=sql.90%29.aspx
- SAVE TRANSACTION
  http://msdn.microsoft.com/es-ar/library/ms188378.aspx