TRANSAKCIA

1) Databázová transakcia
Vlastnosti ACID
2) Formy transakcie
3) Explicitná transakcia
4) Príklad

1) Databázová transakcia je samostatný celok príkazov - práce. http://msdn.microsoft.com/en-us/library/ms174377.aspx

Ak transakcia je

- úspešná, potom všetky operácie v rámci transakcie s dátami sú vykonané a uložené do DB.
- neúspešná, potom operácie v rámci transakcie sú anulované a zmeny v dátach sú odstránené

Transakcia spĺňa **vlastnosti** ACID:

- **A – Atomicity -** transakcia je atomická operácia: previdie sa buď ako **celok** alebo vôbec nie
- **C – Consistency -** transakciou sa nenaruší žiadne **integritné** obmedzenie v rámci DB
- **I – Isolation –** vrátenie transakcie **nevyvolá** ďalšiu transakciu
- **D – Durability – trvalosť -** úspešné transakcie sú **uložené** do databázy

2-3) Formy SQL Server transacie a explicitná transakcia:

- Autocommit transactions – každý individualný príkaz je transakcia.
- Explicit transactions – každá transakcia je explicitne počatá s BEGIN TRANSACTION a explicitne ukončená s COMMIT alebo ROLLBACK príkazom
  - **BEGIN TRANSACTION**
  - **COMMIT or ROOLBACK**
    Commit označuje koniec úspešnej implicitnej alebo explicitnej transakcie.
    Ak **@ @ TRANCOUNT** je 1, COMMIT TRANSACTION zabezpečí, aby všetky zmeny dát vykonané od začiatku transakcie stali natrvalo súčasťou databázy, uvoľní prostriedky potrebné pre transakcie, a @ @ TRANCOUNT nastaví na 0.
    Ak @ @ TRANCOUNT je väčšie ako 1, zníži sa @ @ TRANCOUNT iba o 1 a transakcia zostane aktívna.
    ROLLBACK TRANSACTION sa využíva na vymazanie všetkých zmien dát vykonané od začiatku transakcie alebo uloženého bodu. Príkaz tiež uvoľní prostriedky držané transakciou.
- implicit transactions - nová transakcia je implicitne zahájená po dokončení predchádzajúcej transakcie pomocou COMMIT alebo ROLLBACK.
- Batch-scoped transactions – platí len pre multiple active result sets (MARS),

4) Príklad

V rámci transakcie s využitím uložených procedúr budeme vykonať dve operácie
- vkladanie údajov (insert)
- vymazanie údajov (delete)

Pri vytvorení databázy TransakciaDB ukážeme ako je možné nastaviť niektoré parametre, ako jej počiatočná veľkosť a uvedieme dve úspešné transakcie a tri neúspešné.

```
Skontroluj obsah adresára
      D:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA

USE Master
GO
IF DB_ID ('TransakciaDB') IS NOT NULL DROP DATABASE TransakciaDB
GO

CREATE DATABASE TransakciaDB ON PRIMARY(
      NAME = TransakciaData,
      FILENAME = 'D:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\TransakciaDB.mdf',
      -- SIZE = 3MB,  -- = 5MB
      MAXSIZE = 10MB,
      FILEGROWTH = 20%
)
LOG ON(
      NAME = TransakciaLog,
      FILENAME = 'D:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\TransakciaDB.ldf',
      SIZE = 1MB,
      MAXSIZE = 5MB,
      FILEGROWTH = 1MB
)
GO

USE TransakciaDB
GO
create table Osoba(
      idOsoba int primary key not null,
      meno nvarchar(10) not null,
      firma nvarchar(15)
)
create table OsobaUdaje(
      idOsoba int foreign key references dbo.Osoba(idOsoba),
      adresa nvarchar(30)
)
Insert into Osoba values
(1,'Bobo Bibi', 'IBM ABC'),
(2,'Tata Tutu', 'Micro XYZ')

Insert into OsobaUdaje values
(1, 'Roznava'),
(2, 'Kosice')
GO
```

```
IF OBJECT_ID ( 'sp_InsertDelete', 'P' ) IS NOT NULL
    DROP PROCEDURE sp_InsertDelete;
GO
create procedure sp_InsertDelete
      @idOs int,
      @meno nvarchar(10),
      @firma nvarchar(15),
      @idOsOld int
as
      declare @erIns int
      declare @erDel int
      declare @er int

      begin transaction
      -- Pridaj osobu
      insert into Osoba (idOsoba, meno, firma) values(@idOs, @meno, @firma)

      -- Error po Insert
      set @erIns = @@error
      if @erIns > 0 set  @er  = @erIns

      -- Maz osobu
      delete from Osoba where idOsoba = @idOsOld

      -- Error po Delete
      set @erDel = @@error
      if @erDel > 0 set  @er  = @erDel

      -- If error, roll back
      if @er <> 0
        begin
          rollback
          print  'Transaction  rolled  back'
        end
      else
        begin
          commit
          print  'Transaction  committed'
        end
      print 'INSERT error number:' + cast(@erIns as nvarchar(8))
      print 'DELETE error number:' + cast(@erDel as nvarchar(8))
return @er
GO
```

Teraz vložíme 2 osoby s novými id a vymažeme osobu s id 33. Operácie s danými hodnotami nenarúšajú integritu dát, preto systém ich vykoná a nehlási chybu.

```
--In 2008 DELETE returns error number 0 even though it has not deleted any rows
select * from osoba
select * from OsobaUdaje

exec sp_InsertDelete 3, 'Fero', null, 33
exec sp_InsertDelete 4, 'Jano', null, 33

select * from osoba
select * from OsobaUdaje
```

Ani jeden z nasledujúcich troch príkazov sa nevykoná a systém hlási rôzne chyby.

Tu chceme vkladať osobu s id, ktoré ž existuje a vymazať osobu s existujúcim id, ale neexistujúcim cudzím kľúčom.

```
exec sp_InsertDelete 4, 'Stevo', null, 3 -- Err Primary Key
```

Tu by sme chceli vložiť osobu s novým id a mazať inú osobu s existujúcim primárnym a cudzím kľúčom.

```
exec sp_InsertDelete 5, 'Stevo', null, 2 -- Err Delete Reference error
```

Tu ide o snahu vložiť osobu s id, ktoré už existuje a mazať inú osobu s existujúcim primárnym a cudzím kľúčom.

```
exec sp_InsertDelete 4, 'Stevo', null, 2 -- Err PK + Err Delete
```