Vytvorenie databáz, tabuliek a integrita dát - 2

A) Narušenie a zabezpečenie integrity dát
B) Kompozitný kľúč
C) Konfigurácia prístupu užívateľa k DB a jej objektom
D) Odobratie oprávnenia užívateľa

A) Narušenie a zabezpečenie integrity dát

1) Integrita dát, jeho porušenie a zabezpečenie
2) Obmedzenia
3) CHECK, ADD, DROP a NOCHECK/CHECK CONSTRAINT
4) Zistenie obmedzení programovo
5) UPDATE, DELETE

1) Integrita dát, jeho porušenie a zabezpečenie

integrita – celistvosť, **konzistentnosť** – dôslednosť a bez protirečení (chová sa rovnako); **validný** – platný

Integrita (úplnosť a neporušenosť, kompletné a bez porušenia) **dát** je platnosť (validity) dát. Modelovanie DB je v podstate modelovanie obmedzení na integritu, veď pravidlá reálneho sveta transformujeme na DB model ako obmedzenia na integritu dát.

Integrita dát sa zabezpečuje pomocou **obmedzení**.

Príčiny porušenia integrity dát:
- ľudská nepozornosť pri zadávaní údajov
- chyba počas prenosu dát cez sieť
- softvérové chyby
- hardvérová porucha
- infekcia vírusmi
- prírodná pohroma

Najdôležitejšie **kroky prevencie** straty dát
- systematické zálohovanie DB
- bezpečnostné opatrenia na všetkých úrovniach
- vhodné rozhranie s reštrikciami pre zadávanie dát
- integrita dát

http://msdn.microsoft.com/en-us/library/aa933058.aspx

Rozlišujeme nasledovné **typy** integrity dát:

- **integrita entít** (jedinečnosť) => **entita** - relačná schéma; predmet/objekt/riadok

Entita, ako abstrakcia modelovanej časti sveta pri navrhovaní databáz, je vec, ktorá je schopná nezávislej existencie a ktorá môže byť jednoznačne identifikovaná. Konkrétne tabuľky sú inštancie entity (relačnej schémy). Poznamenáme, že namiesto entity a jej inštancie sa požívajú aj definície entitný typ a entita.

- **doménová integrita** (typ)

- **referenčná integrita** (cudzí kľúč)
- **používateľom definovaná integrita**

Integrita entít definuje jedinečné riadky v rámci danej tabuľky **pomocou**
- id stĺpca, obmedzení UNIQUE alebo PRIMARY KEY , a IDENTITY

Doménová integrita znamená zabezpečenie platných hodnôt v danom stĺpci **pomocou**
- dátových typov a
- rozsahu hodnôt s použitím
    - definícií NOT NULL a DEFAULT
    - CHECK a FOREIGN KEY obmedzení

Referenčná integrita
- zabezpečuje prepojenosť tabuliek (ich vzťah) **pomocou**
    - FOREIGN a PRIMARY kľúčov
    - FOREIGN a UNIQUE kľúčov
- znamená referenciu iba na existujúce hodnoty a kaskádovitú zmenu (konzistentnosť) v referenciách v prípade zmeny hodnoty kľúča

2) Obmedzenia (Constraints)

	Int.entít	Doménova int.	Ref.int.
PRIMARY KEY	✓		✓
FOREIGN KEY		✓	✓
UNIQUE	✓		✓
CHECK		✓	
DEFAULT		✓	
NOT NULL		✓	

PRIMARY KEY
- zabezpečuje jedinečnosť riadkov
- každá tabuľka iba 1 PK
- PK sa skladá z jednej alebo viac stĺpcov

Cudzí kľúč
- každá tabuľka môže mať viac FK
- FK sa skladá z jednej alebo viac stĺpcov

Poznámky:
- každá tabuľka by mala mať jeden/alebo viac kľúčov
- kľúč nie je lokalitou dát/riadkov (LS: B-strom = level nodes ∪ leaf nodes: node=page/stránka 8kb; leaf nodes = data pages => data rows; level nodes = index pages => index rows)
- namiesto "nasledujúceho" a "predchádzajúceho riadku" by sme mali používať "ďalšieho". Treba rozlíšiť *fyzikálne* a *logické* (miesto).
- nie riadky, ale celé množiny riadkov vkladajme do, mažme z a zmeňme (updated) v tabuľky/e.

Default
- DEFAULT 'Polozka xyz'

Nedovolenie NULL hodnôt
- NOT NULL

UNIQUE - jedinečnosť

UNIQUE
- zabezpečuje jedinečnosť hodnôt v *nekľúčovom* stĺpci
- na rozdiel od PK v tabuľke môže byt viac UNIQUE obmedzení
- unique dovolí jednu null hodnotu
- cudzí kľuč sa môže odvolávať na UNIQUE stĺpec

3) CHECK, ADD, DROP a NOCHECK/CHECK CONSTRAINT

CHECK obmedzenie slúži na zabezpečenie **doménovej** integrity. Check môžeme použiť buď v CREATE TABLE alebo ALTER TABLE.

Pravidlá a výhody CHECK obmedzení:
- základné obmedzenia sú na jednom mieste
- tabuľky a stĺpce môžu obsahovať viac CHECK obmedzení
- ich nie je možné pomocou query alebo aplikačného programu (okrem ALTER) prepísať
- optimalizačné programy pre dopyt, INSERT, UPDATE, DELETE analyzujú obsah CHECK obmedzení
- CHECK obmedzenie je možné pomenovať
- názov pomenovaného obmedzenia sa objaví v chybových hláseniach

```
USE tempdb
GO

IF OBJECT_ID('T') IS NOT NULL DROP TABLE T;
GO

CREATE TABLE T -- 0
     (id CHAR(15) NOT NULL PRIMARY KEY,
      x INTEGER,
      y INTEGER NOT NULL);

INSERT T VALUES ('Kosice',NULL, 10)
INSERT T VALUES ('Roznava', 1, 11)
INSERT T VALUES ('Poprad',  3, 12);

SELECT id, x, y FROM T WHERE x < y; -- Poprad,3,12
```

Obmedzenia x<2 a x<y pomocou CHECK v CREATE TABLE

```
IF OBJECT_ID('T') IS NOT NULL DROP TABLE T;
GO

CREATE TABLE T -- 1, 2a, 2b
     (id CHAR(15) NOT NULL PRIMARY KEY,
      x INTEGER CHECK(x<2),
      y INTEGER NOT NULL,
                      --CHECK(x<y) -- v chybovom hlaseni nie je
                      CONSTRAINT CK_2b CHECK(x<y)
     );

INSERT T VALUES ('Kosice',NULL, 10)
INSERT T VALUES ('Roznava', 1, 0)   -- neprejde (conflict)
INSERT T VALUES ('Poprad',  3, 12); -- neprejde

SELECT id, x, y FROM T; -- Kosice,NULL,10
```

Obmedzenia x<2 a x<y pomocou CHECK v ALTER TABLE

```
IF OBJECT_ID('T') IS NOT NULL DROP TABLE T;
GO

CREATE TABLE T -- 3a, 3b
(id CHAR(15) NOT NULL PRIMARY KEY,
 x INTEGER,
 y INTEGER NOT NULL,
);

ALTER TABLE T  ADD CONSTRAINT CK_T_a CHECK (x<2)
ALTER TABLE T  ADD CONSTRAINT CK_T_b CHECK (x<y)

INSERT T VALUES ('Kosice',NULL, 10)
INSERT T VALUES ('Roznava', 1, 0)  -- neprejde
INSERT T VALUES ('Poprad',  3, 12); -- neprejde

SELECT id, x, y FROM T; -- Kosice,NULL,10
```

4) Zistenie obmedzení programovo

```
SELECT * FROM sys.check_constraints
--SELECT * FROM sys.default_constraints

SELECT * FROM INFORMATION_SCHEMA.Table_CONSTRAINTS
-- <==>:
SELECT Table_Schema + '.' + Table_Name, Constraint_Name
FROM INFORMATION_SCHEMA.Table_CONSTRAINTS
ORDER BY constraint_type , Table_Name
```

Odstránenie/DROP vs. vypnutie/NOCHECK obmedzení

DROP

```
IF EXISTS (SELECT Constraint_Name FROM INFORMATION_SCHEMA.Table_CONSTRAINTS
WHERE Constraint_Name = 'CK_T_a')
BEGIN
     ALTER TABLE T DROP CONSTRAINT CK_T_a;
     Print 'Obmedzenie CK_T_a bolo odstranene'
END
GO
INSERT T VALUES ('Poprad',    3, 12); -- uz prejde
```

NOCHECK and CHECK

```
ALTER TABLE T NOCHECK CONSTRAINT CK_T_b
INSERT T VALUES ('Roznava', 1, 0)  -- uz prejde
ALTER TABLE T CHECK CONSTRAINT CK_T_b
INSERT T VALUES ('Honolulu', 1, 0)  -- neprejde
SELECT * FROM T;
```

5) UPDATE a DELETE

UPDATE

```
UPDATE T SET x=11 WHERE id = 'KOSICE'  -- neprejde
UPDATE T SET x=9  WHERE id = 'KOSICE'  -- prejde

SELECT * FROM T;
UPDATE T SET x=-12 -- NEBEZPOCNE POZOR!
SELECT * FROM T;
```

DELETE

```
DELETE T WHERE y = 10
select * from T

DELETE T  -- NEBEZPOCNE - POZOR!
select * from T
```

B) Kompozitný kľúč

- Vytvorenie kompozitného kľúča z viacerých atribútov

```
USE tempdb
IF OBJECT_ID('T0') IS NOT NULL DROP TABLE T0;
IF OBJECT_ID('T1') IS NOT NULL DROP TABLE T1;
IF OBJECT_ID('T2') IS NOT NULL DROP TABLE T2;
-- Uz vieme, ze:
CREATE TABLE T0(i INT NOT NULL PRIMARY KEY, j INT);
GO
-- Kompozitny kluc:
CREATE TABLE T1(i INT, j INT, k INT, PRIMARY KEY(i,j)); -- NOT NULL nie je nutne
-- <=>:
--CREATE TABLE T1(i INT NOT NULL, j INT NOT NULL, k INT, PRIMARY KEY(i,j));
```

```
GO
-- <=>:
CREATE TABLE T2(i INT NOT NULL, j INT NOT NULL, k INT) -- NOT NULL je nutne!
ALTER TABLE T2 ADD CONSTRAINT pk_T2 PRIMARY KEY(i,j)

GO
INSERT T1 VALUES(1,1, NULL)
INSERT T1 VALUES(1,2, NULL)
INSERT T1 VALUES(2,1, 1)
```

C) Konfigurácia prístupu užívateľa k DB a jej objektom

Udelenie prístupu užívateľa k databáze zahŕňa tri kroky.

1) Najprv vytvoríme prihlásenie. Login umožňuje používateľovi pripojiť sa k SQL Server Database Engine.
2) Potom môžeme nakonfigurovať prihlásenie používateľa k danej databáze.
3) A konečne, povolíme používateľovi prístup k databázovým objektom (ako napr. k VIEW, SP).

1) Vytvorenie prístupu k SQL Server
V rámci "Local Users and Groups" Eva bude "New User" s plánovaným prístupom k databáze Studenti.

```
USE [Studenti];
GO
CREATE LOGIN [PC_meno\Eva]
    FROM WINDOWS
    WITH DEFAULT_DATABASE = [Studenti];
GO
```

Po vykonání príkazu Eva už má prístup k inštancii SQL Server, ale ešte nemá oprávnenie pre prístup k databáze.

2) Vytvorenie prístupu k DB Studenti (k jej plánovaným objektom)

```
USE [Studenti];
GO

CREATE USER [Eva] FOR LOGIN [PC_meno\Eva];
GO
```

3) Udelenie prístupu k objektom databázy
Uvažujme pohľad

```
CREATE VIEW menaDruhakov
   AS
   SELECT Priezvisko, Krstne FROM Druhaci;
GO
```

Prístup k VIEW menaDruhakov udeľujeme príkazom

```
GRANT SELECT ON menaDruhakov TO Eva
```

Potom

```
SELECT * FROM menaDruhakov
```

D) Odobratie oprávnenia užívateľa

Postupujeme v opčnom poradí.

3*) Príkaz REVOKE na odobratie práva vykonať VIEW http://msdn.microsoft.com/en-us/library/ms187719.aspx

```
USE Studenti;
GO

REVOKE EXECUTE ON menaDruhakov FROM Eva;
```

2*) Príkaz DROP na zrušenie prístupu Evy k DB Studenti:

```
DROP USER Eva;
```

1*) Príkaz DROP na zrušenie prístupu Evy k SQL Server:

```
DROP LOGIN [PC_meno\Eva];
```