Adatvezérelt rendszerek - 2. előadás

Adatbázis

- Def: Logikailag összefüggő adatok rendezett gyűjteménye
  - adat → mérhető és rögzíthető, sokféle lehet
  - rendezett → könnyű tárolás, módosítás lekérés
  - összefüggő → szükséges adatok kellenek
- **Metaadat**
  - adatdefiníció, adatstruktúra, szabályok és korlátozások
  - adatszótár (data repositiry / data dictionary)

Relációs adatmodell

- matematikai alap
- alap komponensek
  - **tábla** vagy **reláció**, adatok sorokban és oszlopokban
    - 2d reprezentáció, megnevezett oszlopok vannak, sorok száma korlátlan
    - sor és oszlop kereszteződése → cella, ebben max 1 érték van
    - nincs két egyforma sor → nem teljesül a halmaz tulajdonság
    - sorok sorrendje lényegtelen
  - **integritási kritériumok** = érvényességi szabályok (milyen hosszú lehet egy szöveg)
    - *tartományi integritás* → egy adott oszlopban milyen tartományból származó érték lehet (adattípus, hossz, értéktartomány megszabása)
    - *entitás integritás* → 1 rekord mikor érvényes (elsődleges kulcs sehol se lehet NULL)
    - *referenciális integritás* → táblák között, pl külső kulcsok (csak létezőre hivatkozhat, hivatkozott rekord nem törölhető)
    - *működési korlátozás* → üzletvitelből származó, üzleti logika feladata mert relációs modellen túlmutat (pl tárgy csak tárgyfelvételi időben vehető fel)
  - **adatmanipulációs nyelv**
    - relációs algebrából adódóan → SQL
- elnevezések
  - **attribútum** → nevezett oszlop
  - **rekord** → sor
  - **fokszám** → attribútumok száma
  - **kardinalitás** → sorok száma
- **felhasználói séma** = objektumok az adatbázisban
  - adatbázisban levő objektumok összessége
  - platform függő elemek

Microsoft SQL Server platform - MSSQL

- stabil adatbázisrendszer
- szerver komponensek → sok van belőlük, komplex rendszer

Felhasználói séma elemei

- **tábla**
  - oszlop
  - **computed column** = számított oszlopok
    - lehet virtuális (mindig újra számolódik), és tárolt is
- **nézetek** = lekérdezések eredményei
  - indexelhető → ilyenkor tárolódik (amúgy nem tárolódik)
- **indexek** = megmondja az adatbázisnak, hogy hogyan fogunk benne keresni
  - jobb index → gyorsabb keresés az adatbázison
- **szekvencia**
  - számláló
  - megadjuk, hogy honnan indul, hányasával lépked, stb
- **programmodul**
  - eljárás, függvény, trigger, assembly

Adattípusok

- szöveges adattípusok
  - char(n), varchar(n)
  - nchar(n), nvarchar(n) → ékezetes karaktereket is tud tárolni (unicode)
  - varchar(max), nvarchar(max) → ezt kerüljük
  - char és nchar → fix hosszú, többi helyre szóközt tesz
  - varchar, nvarchar → változó hosszú
- numerikus
  - int, float, numeric(p, s)
- dátumok
  - datetime → 1753. január 1-től tud számot reprezentálni
  - datetime2 → időzónát is tárol, minvalue-t is tud
- nagyméretű objektumok
  - Image, TEXT
- egyéb
  - Money, SQL_VARIANT, VARBINARY (ez jobb a nagyméretű objektumoknál), XML

Elsődleges kulcsok generálása

- Identity kulcsszó

```
create table Statusz(
    ID int identity(1, 1) primary key,
    Nev nvarchar(20))
insert into Statusz values ('Kész')
```

  - lekérdezése ennek
    - ident_current('Status'), TODO itt is hiányzik még valami
  - ha lehet mindig generált elsődleges kulcsot használjunk

Tranzakciós határ

- kapcsolat szintjén létezik a tranzakció
- tranzakció kezdés módjai (beállítás függő)
  - *auto commit* → minden utasítás önálló tranzakció (alapértelmezett)
  - *explicit tranzakciók* → explicit tranzakció `begin tran`, egymásba is ágyazhatók
  - *implicit tranzakciók* → tranzakció vége jel után jön új tranzakció
- DML és DDL utasítások is tranzakciók része

Izolációs szintek támogatottsága

- minden SQL szabvány szerinti szintet támogat
- a **Read committed** az alapértelmezett
  - olvasás: megosztott zárakat használ
  - írás: más nem olvashatja
- szabványtól eltérő → **snapshot**
  - tranzakció kezdetekor pillanatkép, az olvasható
  - adatbázis szinten engedélyezni kell
  - jó cucc, de NAGYON drága (külön engedélyezni kell adatbázis szinten)

Adatbázisok tárolása

- adatbázis
  - adatfájl (.mdf), filegroup is lehet
  - tranzakciós napló (.ldf)
  - több sémát tartalmazhat
    - alapértelmezett séma: dbo
    - ez jó logikai csoportosításra, hozzáférés szabályozására
  - rendszer adatbázisok: Master, Model, stb

Hozzáférés szabályozás

- rendszer szintű → adatbázis szerverhez ki férhet hozzá
- adatbázis szintű → konkrét adatbázishoz
- séma szintű → sémák csoportosítása
- objektum szintű
  - konkrét objektumhoz, esetleg objektum szinten is megadható
- tiltás is megadható
- sor szintű hozzáférés nem szabályozható

Tranzakciós naplózás

TODO → jegyzetből elolvasni

Adatbázis szerver oldali programozás

- Adatmodell: táblák
- Adatmanipuláció: SQL nyelv

- Vannak feladatok, amik kimutatnak a relációs adatmodellből
  - Üzleti logikai rétegben valósítjuk meg
  - Adatrétegben valósítjuk meg → **Adatbázis szerveroldali programozása**
- Szerveroldali programozás előnyei
  - Adatbázis felelős a konzisztenciáért (innentől már adatforrás és szolgáltatás is)
  - Adatbiztonság
  - Teljesítmény növelés (csökkenő hálózati forgalom, cache is jobb lesz)
  - Termelékenység (egyszerűbb karbantartás, több komponens által hívható modulok készítése)
- Szerveroldali programozás hátrányai
  - Nem szabványos dolgok (platformfüggő elemek)
  - Interpretált (azaz nem lefordított kód, teljesítménye rosszabb)
  - Növeli a szerver terhelését
  - Nem illetve nehezen skálázható

Transact-SQL nyelv (T-SQL)

- Csak az MSSQL Server nyelve
- amik lesznek: változók, utasítás blokkok, ciklusok, strukturált hibakezelés, új konstrukciók
- utasítás blokk

```
BEGIN
    /*TSQL utasítások*/
END
```

- változók
  - DECLARE-el deklaráljuk, @-al kezdődik a nevük
  - deklarálás után a változó értéke NULL

```
DECLARE @nev nvarchar(30), @szam int = 5
```

```
DECLARE @szam int
SET @SZAM = 3
```

```
DECLARE @szam int, @nev nvarchar(30)
SELECT @szam = id, @nev = nev FROM termek
WHERE ... /*ha a lekérdezés több sorral tér vissza, a változó értéke az
utolsó sor értékével fog megegyezni*/
```

- vezérlési szerkezetek

```
IF ... ELSE
WHILE
/*ciklusvezérlő utasítások*/
BREAK
CONTINUE
```

Tárolt eljárások

- tárolt, mert az adatbázis szerverben van
- eljárás, mert nincs visszatérési értéke
- Belső
  - T-SQL nyelv
  - interpretált
  - zárt környezetben van
- Külső → .NET assembly

Tárolt eljárás létrehozása

- `create or alter procedure`
  - létrehozza/módosítja a tárolt eljárást a sémában
  - értelmezi, ellenőrzi, eltárolja
- első futtatáskor → fordítás, optimalizálás
- stored prucedure cache → használaton kívüli tervek elöregednek

Tárolt eljárás újrafordítása

- ha releváns tartalom változik az adatbázisban → újra lesz fordítva
- újrafordítás kérése → sp_recomplie

Tárolt függvények

- van visszatérési értéke, nem változtathat az adatbázisban (csak olvashat, írni nem írhat)
- Típusai
  - Scalar-valued → értéket számol ki (min, max)
  - Table-valued → rekordhalmazzal tér vissza
  - Aggregate → saját oszlopfüggvény, .NET Assembly-ben írunk ilyet
- van jónéhány beépített (string hossza, string üsszefűzése, stbstb)

Tárolt eljárások kezelése

- módosítás → `ALTER PROCEDURE`
- törlés → `DROP PROCEDURE`

Hibakezelés

- @@Error függvény
  - minden utasítás után lekérdezhető
  - ha nincs hiba, akkor 0-t ad vissza
- struktúrált kivétel kezelés

```
BEGIN TRY
    /*utasítások*/
END TRY
BEGIN CATCH
    /*utasítások*/
END CATCH
```

- hiba generálás
  - Raiserror

    - Throw

Triggerek

- eseménykezelő tárolt eljárások
- mire használható:
    - származtatott értékek karbantartása (denormalizáció)
    - naplózás
    - statisztikák gyűjtése (hányszor történt meg valami)
    - máshogy nem kifejezhető referenciális integritás
- események
    - DML események → mikor módosul a tábla, táblához kötődik
    - DDL triggerek → create, alter, drop, sémákhoz kötődnek
    - rendszeresemény → logon, logoff, syserror, ...
    - instead of triggerek → adott utasítás helyett valami más hajtódik végre
- DML triggerek → utasítás szintű
    - ha 10 sort módosító utasítás van, akkor egybe kapja a trigger azt a 10-et, egyszer hívódik meg
    - adatmódosítás után hajtódik végre
- **Módosított rekord elérése**
    - Napló táblákon keresztül
        - napló tábla struktúrája:

	Insert	Delete	Update
Inserted	Új rekordok	Üres	Rekordok új értékei
Deleted	Üres	Törölt rekordok	Rekordok régi értékei

        - csak a trigger-ben értelmezett
- Triggerek egymásra hatása
    - Kaszkád triggerek → ami kivált egy következő triggert (32 mélységig)
    - Rekurzív triggerek → megengedett, de nagyon hagyjuk
    - Ugyan ahhoz az eseményhez több trigger kapcsolása
        - lehet, de sorrend nem befolyásolható, nem ismert
- Triggerek és tranzakciók
    - szülő tranzakció részét képzi egy trigger
- Instead of triggerek → náluk lehet nem okoz rekurziót, ha belőlük a saját táblájukra hivatkozunk

Kurzor

- foreach iterátor
- több rekordot visszaadó lekérdezések feldolgozására