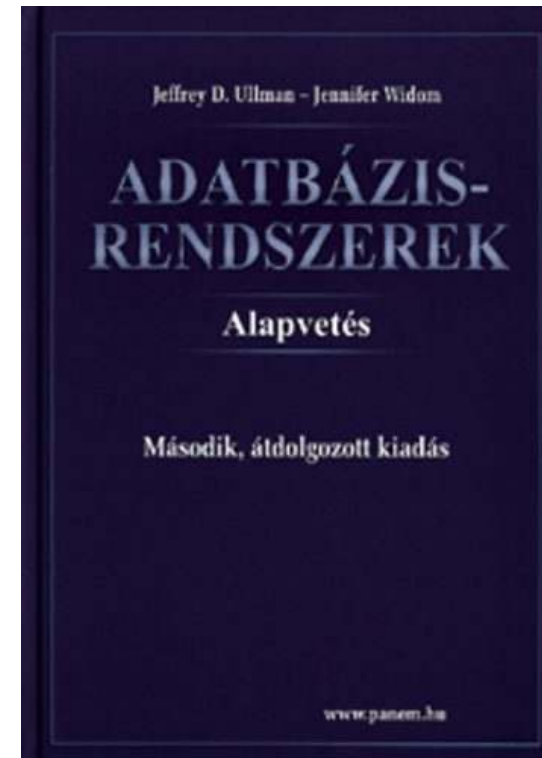


Bevezetés az SQL-be

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009

2.3. Relációsémák definiálása SQL-ben Kulcsok megadása (folyt.köv.7.fej.)

-- még: Relációs algebrai kifejezések
átírása az SQL SELECT utasításra:
Tk. 2.4.1.Termék-feladatai SQL-ben



SQL története, szabványok

- Szabvány adatbázis-kezelő nyelv: SQL
- SQL (angol kiejtésben SEQUEL) uis az SQL előfutára
IBM fejlesztette ki a 70-es években: SEQUEL → SQL
más is volt pl. Ingres : QUEL (ez kalkulus alapú lekérdezés)
- Szabványok (ANSI, ISO)
SQL86, SQL89, SQL92 (SQL2), SQL:1999 (SQL3),
SQL: 2003, SQL:2006, SQL:2008
- Nyelvjárások (Oracle, Sybase, DB2, Progress, MSSQL,
mySQL, SQL Server, PostgreSQL, Access,...)
- Az SQL megvalósítások között vannak különbségek,
gyakorlatokon az Oracle SQL-t nézzük meg részletesen.

SQL fő komponensei

- **Adatleíró nyelv, DDL** (Data Definition Language)
CREATE, ALTER, DROP
- **Adatkezelő nyelv, DML** (Data Manipulation Language)
INSERT, UPDATE, DELETE, SELECT
-- **Az SQL elsődlegesen lekérdező nyelv** (Query Language)
SELECT utasítás (az adatbázisból információhoz jussunk)
- **Adatvezérlő nyelv, DCL** (Data Control Language)
GRANT, REVOKE
- **Tranzakció-kezelés**
COMMIT, ROLLBACK, SAVEPOINT
- **Procedurális kiterjesztések**
Oracle PL/SQL (Ada alapján), SQL/PSM (PL/SQL alapján)

Adatbázis relációsémák definiálása

- Az SQL használata szerint építjük fel az előadást, vagyis a Tk.2.3. fejezete után a Tk.6.SQL fejezetével folytatjuk, majd visszatérünk újra a megszorításokra (Tk.7.fejezet)
- Az SQL tartalmaz **adateleíró részt (DDL)**, az adatbázis **objektumainak** a leírására és megváltoztatására. **Objektumok** leíró parancsa a **CREATE** utasítás.
- A relációt az SQL-ben táblának (TABLE) nevezik, az SQL alapvetően háromféle táblát kezel:
 - ❑ Alaptáblák (permanens) CREATE TABLE
 - ❑ Nézettáblák CREATE VIEW (ezt később nézzük)
 - ❑ Átmeneti munkatáblák (WITH utasítás is később)
- **Alaptáblák** megadása: **CREATE TABLE**

Tábla/reláció sémák SQL-ben

- A legegyszerűbb formája:

```
CREATE TABLE relációnév (  
    Attribútum deklarációk listája,  
    További kiegészítések  
);
```

- Az attribútum deklaráció legalapvetőbb elemei:

Attribútumnév típus [kiegészítő lehetőségek]

- itt: a **típus** olyan, amit az SQL konkrét megvalósítása támogat (gyakorlaton Oracle környezetben nézzük meg),
Típusok, pl: INTEGER, REAL, CHAR, VARCHAR, DATE
- A **kiegészítő lehetőségek** például [PRIMARY KEY] vagy [DEFAULT érték] (köv.lapon példa, ill. később lesz: 7.fej)

Példa: sörivők adatbázis

Sörök(sör, gyártó)

Bárok(bár, város, tulaj, engedély)

Ivók(név, város, tel)

Kedvel(név, sör)

Felszolgál(bár, sör, ár)

Látogat(név, bár)

- Az aláhúzás jelöli a **kulcsot** (a sorok a kulcs összes attribútumán nem vehetik fel ugyanazt az értékeket).
 - Ez a kulcs, külső kulcs és hivatkozási épség megszorításoknak lesz később kiváló példája.

Egyszerű példák táblák létrehozására

```
CREATE TABLE Bárok (  
    bár CHAR(20) ,  
    város VARCHAR2(40) ,  
    tulaj CHAR(30) ,  
    engedély DATE DEFAULT SYSDATE  
);
```

```
CREATE TABLE Felszolgál (  
    bár CHAR(20) ,  
    sör VARCHAR2(20) ,  
    ár NUMBER(10,2) DEFAULT 100  
);
```

Az SQL értékekről (bővebben gyakorlaton)

- INTEGER, REAL, stb, a szokásos értékek, számok.
- STRING szintén, de itt egyes-aposztróf közé kell tenni a 'szöveget' (vagyis nem „macskaköröm” közé).
 - Két egyes-aposztróf = egynek felel meg, például ' Joe' ' s Bar' megfelel a Joe's Bar szövegnek.
- Bármely érték lehet **NULL** (erről később a 6.1.fejezetben)
- DATE és TIME típusok is vannak az SQL-ben.
- A dátum formátumát meg kell adni DATE 'yyyy-mm-dd'
 - **Például:** DATE '2007-09-30' (2007. szept. 30)
- Az idő formátumát is meg kell adni TIME 'hh:mm:ss'
 - **Például:** TIME '15:30:02.5' (délután fél 4 múlt két és fél másodperccel)

Kulcs megadása

- **PRIMARY KEY** vagy **UNIQUE**
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEAFULT értéknek.
- Kulcsok megadásának két változata van:
 - Egyszerű kulcs (egy attribútum) vagy
 - Összetett kulcs (attribútumok listája)(példákat lásd a következő oldalon)

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

<attribútumnév> <típus> **PRIMARY KEY**

vagy <attribútumnév> <típus> **UNIQUE**

- Példa:

```
CREATE TABLE Sörök (  
    név          CHAR(20) UNIQUE,  
    gyártó       CHAR(20)  
);
```

Összetett kulcs megadása

- Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: **PRIMARY KEY (attrnév₁, ... attrnév_k)**
- Példa:

```
CREATE TABLE Felszolgal (
    bár          CHAR(20) ,
    sör          VARCHAR2(20) ,
    ár           NUMBER(10,2) ,
    PRIMARY KEY (bár, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen **PRIMARY KEY** lehet a relációban, viszont **UNIQUE** több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet **NULL érték** egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL értékű, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke **NULL** vagyis **hiányzó érték**. Köv.ea: 6.1.fejezetben: NULL az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni.
- Itt csak: kulcsok, majd idegen kulcsok megadása lesz. Később, a 7.fejezetben visszatérünk a megszorításokra, arra is, hogy milyen lehetőségeink vannak a hivatkozási épség fenntartására, valamint további megszorításokra.

Idegen kulcsok megadása

- Még egy kiegészítő lehetőség Mi köthet össze két táblát? **Idegen kulcs (foreign key) megadása**
- Az egyik tábla egyik oszlopában szereplő értékeknek szerepelnie kell egy másik tábla bizonyos attribútumának az értékei között.
- **A hivatkozott attribútumoknak** a másik táblában kulcsnak kell lennie! (PRIMARY KEY vagy UNIQUE)
- **Példa: Felszolgál(bár, sör, ár)** táblára megszorítás, hogy a sör oszlopában szereplő értékek szerepeljenek a **Sörök(sör, gyártó)** táblában a sör oszlop értékei között.

Idegen kulcsok megadása: attribútumként

REFERENCES kulcsszó használatának két lehetősége:
attribútumként vagy sémaelemként lehet megadni.

1.) Attribútumként (egy attribútumból álló kulcsra)

PÉLDA:

```
CREATE TABLE Sörök (  
    sör CHAR(20) PRIMARY KEY,  
    gyártó CHAR(20) );  
  
CREATE TABLE Felszolgál (  
    bár CHAR(20) ,  
    sör CHAR(20) REFERENCES Sörök,  
    ár REAL );
```

Idegen kulcsok megadása: sémaelemként

2. Sémaelemként (egy vagy több attr.-ból álló kulcsra)

FOREIGN KEY (attribútum lista)

REFERENCES relációnév (attribútum lista)

```
PÉLDA: CREATE TABLE Sörök (  
    név          CHAR(20) PRIMARY KEY,  
    gyártó       CHAR(20) );
```

```
CREATE TABLE Felszolgál (  
    bár          CHAR(20) ,  
    sör          CHAR(20) ,  
    ár           REAL ,  
    FOREIGN KEY (sör) REFERENCES Sörök(név) );  
-- itt most másképp neveztük a hivatkozott attribútumot
```

Rel.algebrai kifejezések felírása SELECT-tel

- A lekérdezések megadására formálisan relációs algebrai kifejezéseket adtunk meg (lásd előző 02A_RelAlg2kif-ben). Ezeket hogyan tudjuk átírni az SQL SELECT utasítására?
- **Példa:** Termék (gyártó, modell, típus)
PC (modell, sebesség, memória, merevlemez, cd, ár)
Laptop (modell, sebesség, memória, merevlemez, képernyő, ár)
Nyomtató (modell, színes, típus, ár) --- típus itt mást jelent!
- Mi volt a leggyakrabban előforduló típus, amiből építkezek?

$\Pi_{\text{lista}}(\sigma_{\text{feltétel}}(\text{táblák szorzata}))$

Ezt a komponenst támogatja legerősebben majd az SQL:

SELECT s-lista FROM f-lista WHERE feltétel

Halmazműveletek SQL-ben SELECT... lekérdezések között

UNION | EXCEPT/MINUS | INTERSECT