SQL

Gugolya László

Az SQL nyelv

SQL = Structured Query Language.
Nem algoritmikus nyelv.

1976: **SEQUEL** (IBM)

1981: ORACLE 2 (SQL alapú DBMS)

1983: IBM: DB2 (SQL alapú DBMS)

1986: **szabvány SQL** (ANSI)

1992: **SQL-92** szabvány (SQL2)

1999: **SQL:1999** szabvány (SQL3) (objektum-relációs)

2003: **SQL:2003** szabvány (XML)

2006, 2008: további bővítések

Az SQL szabvánnyal kapcsolatos tévhitek

- **„Minden rendszer betartja a szabványt.”**
  Valójában szinte egyetlen rendszer sem tartja be pontosan.
- **„A nagy fejlesztők inkább betartják a szabványt.”**
  Valójában ők tartják be kevésbé, mert fejlesztéseik megelőzik a szabvány kidolgozását.
- **„A régebbi szabványokban leírtak mind működnek, az újabbakban leírtak fokozatosan megvalósulnak.”**
  Valójában a legújabb rendszerek sem valósítanak meg mindent például az SQL’92-ből.

Az SQL részei

- **DDL** (= Data Definition Language): adatstruktúrát definiáló utasítások.

- **DML** (= Data Manipulation Language): adatokon műveletet végző utasítások:
    1. adatok aktualizálása
    2. lekérdezések

Az SQL részei más felbontás szerint

- **DDL** (= Data Definition Language):
  adatstruktúrát definiáló utasítások.
- **DML** (= Data Manipulation Language):
  adatok aktualizálása
- **DQL** (=Data Query Language)
  lekérdezések
- **DCL** (=Data Control Language)
  vezérlés (jogosultságkezelés, tranzakciókezelés)

Szintaxis

- Kisbetű és nagybetű egyenértékű
- Utasítások sorfolytonosan, lezárás: pontosvessző
- ***Változó nincs***, csak tábla- és oszlopnevek
  Jelölés: [tábla **.**] oszlop
- ***Alias név:*** név AS másodnév
- ***Szövegkonstans:*** 'szöveg'
- ***Stringek konkatenációja:*** + vagy ||
- ***Relációjelek:*** =, <=, >=, !=, <>

Logikai kifejezések - 1

- ***Logikai műveletek:*** AND, OR, NOT.
- ***Háromértékű logika:*** TRUE, FALSE, NULL. (SQL-szabvány szerint UNKNOWN)
- **x IS NULL** (x=NULL értéke UNKNOWN)
- **x BETWEEN a AND b**
  a ≤ x ≤ b
  Példa: `évszám BETWEEN 1950 AND 2000`
- **x LIKE minta**
  Példa: `lakcím LIKE '%Vár u.%'`

Logikai kifejezések - 2

- **x IN halmaz**
  Példa: `város IN ('Szeged','Szolnok','Pécs')`
  Példa: `város IN (lekérdezés)`
- **x relációjel ALL halmaz**
  Példa: `fizetés != ALL (81000, 136000, 118000)`
- **x relációjel ANY halmaz**
  Példa: `fizetés < ANY (81000, 136000, 118000)`
- **EXISTS halmaz**
  Példa: `EXISTS (lekérdezés)`

NOT használata: például NOT IN

Relációsémák definiálása (DDL)

```
CREATE TABLE táblanév
  ( oszlopnév  adattípus  [feltétel],
       ...       ...,
    oszlopnév  adattípus  [feltétel]
    [, táblaFeltételek]
  );
```

Adattípusok

- **CHAR(n)** n hosszúságú karaktersorozat
- **VARCHAR(n)** max. n hosszú karaktersorozat
- **INTEGER** egész szám (röviden INT)
- **REAL, FLOAT** valós (lebegőpontos) szám
- **DECIMAL(n[,d])** *n* jegyű decimális szám, ebből *d* tizedesjegy
- **DATE** dátum (év, hó, nap)
- **TIME** idő (óra, perc, másodperc)
- **TIMESTAMP** dátum + idő

Speciális adattípusok

- **SERIAL** automatikusan növelt szám (kulcs)
- **BLOB** Binary Large Object
- **CLOB** Character Large Object
- **VARRAY** változó hosszúságú tömb
- **POINT** (x, y) koordinátákkal adott

...stb.

Feltételek (adott oszlopra)

- **PRIMARY KEY** *elsődleges kulcs*
- **UNIQUE** *kulcs*
- **REFERENCES tábla(oszlop) [ON-feltételek]**
  külső kulcs
- **DEFAULT** *alapértelmezett érték*

Megjegyzések:

- Csak egy PRIMARY KEY, de több UNIQUE lehet.
- Külső kulcs általában PRIMARY KEY-re hivatkozik.
- PRIMARY KEY-hez automatikus index generálás lehet.

Táblafeltételek (egész táblára)

- **PRIMARY KEY (oszloplista)** *elsődleges kulcs*
- **UNIQUE (oszloplista)** *kulcs*
- **FOREIGN KEY (oszloplista) REFERENCES tábla(oszloplista) [ON-feltételek]** *külső kulcs*

Megjegyzések:

A feltételek és táblafeltételek valójában funkcionális függéshalmaz leírását valósítják meg.

Összetett kulcs és összetett külső kulcs csak táblafeltételként adható meg.

Példa

Osztály (osztálykód, osztálynév, vezAdószám)

Dolgozó (adószám, név, lakcím, *osztálykód*)

```
CREATE TABLE Osztály
 ( osztálykód  CHAR(3) PRIMARY KEY,
   osztálynév  VARCHAR(20),
   vezAdószám  DECIMAL(10)
 );
CREATE TABLE Dolgozó
 ( adószám DECIMAL(10) PRIMARY KEY,
   név     VARCHAR(30),
   lakcím  VARCHAR(40) DEFAULT 'ismeretlen',
   osztálykód CHAR(3)
         REFERENCES Osztály(osztálykód)
 );
```

A *Dolgozó* sémát így is lehetne definiálni:

```
CREATE TABLE Dolgozó
    ( adószám     DECIMAL(10),
      név         VARCHAR(30),
      lakcím      VARCHAR(40),
      osztálykód CHAR(3),
      PRIMARY KEY (adószám),
      FOREIGN KEY (osztálykód)
        REFERENCES Osztály(osztálykód)
    );
```

Példa összetett kulcsra

Fuvar (gkvez, rendszám, indul, érkezik)

```
CREATE TABLE Fuvar
   ( gkvez       CHAR(20),
     rendszám    CHAR(10),
     indul       TIMESTAMP,
     érkezik     TIMESTAMP,
     PRIMARY KEY (rendszám, indul)
   );
```

REFERENCES tábla(oszlop) [ON-feltételek]

- ON UPDATE CASCADE
- ON DELETE CASCADE
- ON UPDATE SET NULL
- ON DELETE SET NULL

Példa ON-feltételre:

```
CREATE TABLE Dolgozó
 ( adószám DECIMAL(10) PRIMARY KEY,
   név     VARCHAR(30),
   lakcím  VARCHAR(40) DEFAULT 'ismeretlen',
   osztálykód CHAR(3)
     REFERENCES Osztály(osztálykód)
       ON UPDATE CASCADE
       ON DELETE SET NULL
   );
```

Relációséma törlése

DROP TABLE táblanév;

Relációséma módosítása

ALTER TABLE táblanév

[ADD (újelem, ..., újelem)]

[MODIFY (módosítás, ..., módosítás)]

[DROP (oszlop, ..., oszlop)];

újelem: oszlopnév adattípus [feltétel]

Példák:

```
ALTER TABLE Dolgozó
  ADD (szüldátum DATE);

ALTER TABLE Dolgozó
  MODIFY (lakcím VARCHAR(60));

ALTER TABLE Osztály
   MODIFY (vezAdószám
     REFERENCES Dolgozó(adószám));
```

Indexek létrehozása

Nem része a szabványnak:

```
CREATE [UNIQUE] INDEX indexnév
  ON tábla(oszloplista);
```

Példák:

```
CREATE INDEX SzerInd ON Könyv(szerző);
CREATE INDEX DolgInd ON
  Dolgozó(osztálykód,név);
```

```
DROP INDEX indexnév;
```

Adattábla aktualizálása (DML)

Új sor felvétele:

INSERT INTO táblanév [(oszloplista)]
VALUES (értéklista);

Példák:

Dolgozó (adószám, név, lakcím, *osztálykód*)

```
INSERT INTO Dolgozó (név, adószám)
  VALUES ('Tóth Aladár', 1111);

INSERT INTO Dolgozó
  VALUES (1111, 'Tóth Aladár',,'12');
```

Sor(ok) módosítása:

UPDATE táblanév

SET oszlop = kifejezés, ..., oszlop = kifejezés

[WHERE feltétel];

Példák:

```
UPDATE Dolgozó
  SET lakcím = 'Szeged, Rózsa u. 5.'
    WHERE adószám = 1234;

UPDATE Dolgozó
    SET osztálykód = '013'
    WHERE osztálykód = '003';

UPDATE Dolgozó SET osztálykód = NULL;
```

Sor(ok) törlése:

DELETE FROM táblanév [WHERE feltétel];

Példák:

```
DELETE FROM Dolgozó
    WHERE név = 'Kovács József';
DELETE FROM Osztály;
```

Példa: kulcs nélküli tábla:

Dolgozó (név, lakcím, fizetés)

Ha két azonos sor van, nem lehet csak az egyiket törölni vagy módosítani.

Lekérdezések (DML)

Lekérdezés: Adattáblák → Eredménytábla

A SELECT utasítás alapváltozata:

```
SELECT [DISTINCT] oszloplista
   FROM táblanévlista
  [WHERE feltétel];
```

`SELECT DISTINCT` $A_1,\ldots,A_n$ `FROM` $T_1,\ldots,T_m$
`WHERE feltétel;`

egyenértékű az alábbival:

$E = \pi_{A1,\ldots,An}(\sigma_{\text{feltétel}}(T_1 \times \ldots \times T_m))$

Tábla listázása: `SELECT * FROM T;`

A relációs algebra műveletei

Projekció:

SELECT [DISTINCT] $A_1,\ldots,A_n$ FROM T;

Példák:

```
SELECT DISTINCT szerző,cím FROM Könyv;
SELECT könyvszám,cím FROM Könyv;
```

Szelekció:

SELECT * FROM *T* WHERE feltétel;

Példa:

```
SELECT * FROM Könyv
  WHERE kivétel<2009.01.01;
```

Descartes-szorzat:

SELECT * FROM T_1, T_2;

Természetes összekapcsolás:

$T_1(A)$ és $T_2(B)$, $X = A \cap B$.

$E = \pi_{A \cup B}(\sigma_{T1.X=T2.X}(T_1 \times T_2))$

Ezzel egyenértékű:

SELECT $A \cup B$ FROM T_1, T_2 WHERE $T_1.X = T_2.X$;

Példa természetes összekapcsolásra - 1

Áru (cikkszám, megnevezés)

Vásárlás (*cikkszám*, mennyiség)

E (cikkszám, megnevezés, mennyiség)

```
SELECT Áru.cikkszám,megnevezés,mennyiség
  FROM Áru,Vásárlás
  WHERE Áru.cikkszám=Vásárlás.cikkszám;
```

Más szintaxis:

```
SELECT Áru.cikkszám,megnevezés,mennyiség
  FROM Áru INNER JOIN Vásárlás
  ON Áru.cikkszám=Vásárlás.cikkszám;
```

Példa természetes összekapcsolásra - 2

Könyv (könyvszám, szerző, cím, *olvszám*, kivétel)
Olvasó (olvszám, név, lakcím)
E (könyvszám, szerző, cím, név, kivétel)

```
SELECT könyvszám,szerző,cím,név,kivétel
  FROM Könyv,Olvasó
   WHERE Könyv.olvszám=Olvasó.olvszám;
```

Megjegyzés: nem minden attribútumot tartunk meg, például az összekapcsoló olvszám-ot sem.

Külső összekapcsolás:

Korábbi Oracle szintaxis: (+)=, =(+), (+)=(+)

```
SELECT könyvszám,szerző,cím,név,kivétel
  FROM Könyv,Olvasó
   WHERE Könyv.olvszám (+)= Olvasó.olvszám;
```

Szabványos szintaxis: LEFT, RIGHT vagy FULL OUTER JOIN:

```
SELECT könyvszám,szerző,cím,név,kivétel
  FROM Könyv LEFT OUTER JOIN Olvasó
   ON Könyv.olvszám = Olvasó.olvszám;
```

Théta összekapcsolás:

$T = \sigma_{\text{feltétel}}(T_1 \times T_2)$

SELECT * FROM T_1, T_2 WHERE feltétel;

Példa:

Raktár (raktárkód, mennyiség)

Vevő (vevőkód, igény)

E (raktárkód, mennyiség, vevőkód, igény)

```
SELECT * FROM Raktár,Vevő
   WHERE mennyiség >= igény;
```

Halmazműveletek (kompatibilis táblák között):

Unió:

```
(SELECT * FROM T1)
UNION
(SELECT * FROM T2);
```

Metszet:

```
(SELECT * FROM T1)
INTERSECT
(SELECT * FROM T2);
```

Különbség:

```
(SELECT * FROM T1)
EXCEPT
(SELECT * FROM T2);
```

Példa:

Helyiség (épület, ajtószám, név, alapterület)

Tanterem (*épület*, *ajtószám*, férőhely, tábla, vetítő)

Gépterem (*épület*, *ajtószám*, gépszám)

Kérjük le az oktatási célú géptermek listáját:

```
(SELECT épület,ajtószám FROM Tanterem)
INTERSECT
(SELECT épület,ajtószám FROM Gépterem);
```

Más megoldás, természetes összekapcsolással:

```
SELECT Tanterem.épület,Tanterem.ajtószám
  FROM Tanterem,Gépterem
  WHERE Tanterem.épület=Gépterem.épület
    AND Tanterem.ajtószám=Gépterem.ajtószám;
```

Alias nevek - 1

SELECT kif$_1$ AS másodnév$_1$, kif$_2$ AS másodnév$_2$ FROM …;

Raktár (cikkszám, név, egységár, mennyiség)

táblából E (áru, érték) *tábla létrehozása:*

```
SELECT név AS áru,
  egységár*mennyiség AS érték
   FROM Raktár;
```

Személy (adószám, név, születésiév)

táblából E (név, életkor) *tábla létrehozása:*

```
SELECT név, 2013-születésiév AS életkor
  FROM Személy;
```

Alias nevek - 2

SELECT oszloplista
FROM tábla$_1$ AS másodnév$_1$, tábla$_2$ AS másodnév$_2$
WHERE …;

Dolgozó (adószám, név, lakcím, osztkód, fizetés)

Azonos nevű dolgozók lekérése: E (név, adószám1, adószám2)

```
SELECT d1.név AS név,
       d1.adószám AS adószám1,
       d2.adószám AS adószám2
  FROM Dolgozó AS d1, Dolgozó AS d2
  WHERE d1.név=d2.név AND
        d1.adószám < d2.adószám;
```

Függvények - 1

ABS(n): abszolút érték

Példa: **`ABS(-15) = 15`**

LOWER(char): konverzió kisbetűsre

Példa: **`LOWER('Kovács') = 'kovács'`**

UPPER(char): konverzió nagybetűsre

Példa: **`UPPER('Kovács') = 'KOVÁCS'`**

LTRIM(char): balról szóközök eltávolítása

Példa: **`LTRIM('  alma  ') = 'alma  '`**

RTRIM(char): jobbról szóközök eltávolítása

Példa: **`RTRIM('  alma  ') = '  alma'`**

Függvények - 2

SUBSTR(char, m[, n]): részstring az *m*-edik karaktertől *n* hosszan. (Ha *n* nem szerepel, akkor a végéig.)

Példa: **`SUBSTR('ABCDEFG',2,3) = 'BCD'`**

TO_CHAR(n): konverzió numerikusról vagy dátumról karakteresre

Példa: **`TO_CHAR(123) = '123'`**

TO_DATE(char): konverzió karakteresről dátumra

Példa: **`TO_DATE('15-JAN-06')`**

TO_NUMBER(char): konverzió karakteresről numerikusra

Példa: **`TO_NUMBER('123') = 123`**

Összesítő függvények

függvénynév ([DISTINCT] oszlopnév)

- AVG( ): átlagérték

Példa: Dolgozó (adószám, név, lakcím, osztkód, fizetés)

```
SELECT AVG(fizetés) FROM Dolgozó;
```

- SUM( ): összeg

```
SELECT SUM(fizetés) FROM Dolgozó;
```

- MAX( ): maximális érték
- MIN( ): minimális érték
- COUNT( ): elemek száma

```
SELECT COUNT(*) FROM Dolgozó;
SELECT COUNT(DISTINCT osztkód) FROM Dolgozó;
```

Csoportosítás

GROUP BY oszloplista

Valójában csoportonkénti összevonás történik!

Dolgozó (adószám, név, lakcím, osztkód, fizetés)

```
SELECT osztkód, AVG(fizetés) AS átlagfizetés
   FROM Dolgozó GROUP BY osztkód;
```

E (osztkód, átlagfizetés)

Osztkód	*Átlagfizetés*
005	130434
007	147108
012	122700

Csoportosítási szabály

SELECT után csak összesítő függvény vagy összesített oszlop szerepelhet!

Példa: Dolgozó (adószám, név, lakcím, osztkód, fizetés)

Helyes:

```
SELECT osztkód, MAX(fizetés) AS maxfiz
  FROM Dolgozó GROUP BY osztkód;
```

Helytelen:

```
SELECT osztkód, név, MAX(fizetés) AS maxfiz
   FROM Dolgozó GROUP BY osztkód;
```

További példák

Projóra (dolgozó, projekt, óra)

```
SELECT dolgozó, SUM(óra) AS összóraszám
  FROM Projóra
  GROUP BY dolgozó;
```

E (dolgozó, összóraszám)

```
SELECT projekt, SUM(óra) AS összóraszám
  FROM Projóra
  GROUP BY projekt;
```

E (projekt, összóraszám)

Csoport-szelekció

HAVING feltétel

```
SELECT osztkód, AVG(fizetés) AS átlagfizetés
  FROM Dolgozó GROUP BY osztkód
  HAVING AVG(fizetés) > 130000;
```

Osztkód	*Átlagfizetés*
005	130434
007	147108

WHERE feltétel: csoportosítás előtti szelekció

HAVING feltétel: csoportosítás utáni

Az eredménytábla rendezése

ORDER BY oszlopnév [DESC], ..., oszlopnév [DESC]

- ASC (ascending): növekvő (alapértelmezés)
- DESC (descending): csökkenő

```
SELECT * FROM Dolgozó ORDER BY név;
```

```
SELECT osztkód, név, fizetés
   FROM Dolgozó
   ORDER BY osztkód, fizetés DESC;
```

A SELECT utasítás általános alakja

SELECT [DISTINCT] oszloplista	*projekció*	*5.*
FROM táblanévlista	*Descartes-szorzat*	*1.*
[WHERE feltétel]	*szelekció*	*2.*
[GROUP BY oszloplista]	*csoport-összevonás*	*3.*
[HAVING feltétel]	*csoport-szelekció*	*4.*
[ORDER BY oszloplista];	*rendezés*	*6.*

Megjegyzések:

- "feltétel" helyére tetszőleges logikai kifejezés írható.
- "oszloplista" általában oszlopkifejezéseket is tartalmazhat.
- A végrehajtási sorrendből következik, hogy hol milyen paramétert lehet használni.

Példa

Dolgozó (adószám, név, lakcím, *osztkód*, fizetés)
Osztály (osztkód, osztálynév, *vezAdószám*)

Feladat: ábécé sorrendben azon osztályok névlistája, ahol
a legkisebb fizetés is nagyobb, mint 200 000:

```
SELECT osztálynév
FROM Dolgozó, Osztály
WHERE Dolgozó.osztkód = Osztály.osztkód
GROUP BY Dolgozó.osztkód, osztálynév
HAVING MIN(fizetés)>200000
ORDER BY osztálynév;
```

Kérdés: mi történik, ha két azonos nevű osztály van?

Alkérdések

Dolgozó (adószám, név, lakcím, osztkód, fizetés)

Speciális logikai kifejezések:

```
'Tóth Pál' IN (SELECT név FROM Dolgozó
                WHERE osztkód='015')

EXISTS (SELECT * FROM Dolgozó
         WHERE fizetés < 80000)
```

Ilyen kifejezések WHERE és HAVING feltételben használhatók.

Alkérdés: *SQL utasítás belsejében lekérdezés*

Dolgozó (adószám, név, lakcím, osztkód, fizetés)

Példa: *az átlagfizetésnél kisebb fizetésűek listája*

```
SELECT név, fizetés
  FROM Dolgozó
   WHERE fizetés <
   ( SELECT AVG(fizetés) FROM dolgozó );
```

Az alkérdés csak egyszer értékelődik ki.

Példa: *az osztályukon belül legnagyobb fizetésűek névlistája*

Hibás megoldás:

```
SELECT osztkód, név, MAX(fizetés)
  FROM Dolgozó GROUP BY osztkód;
```

Jó megoldás:

```
SELECT osztkód, név, fizetés
  FROM Dolgozó AS D1
   WHERE fizetés =
     ( SELECT MAX(fizetés)
       FROM Dolgozó AS D2
       WHERE D1.osztkód=D2.osztkód );
```

Az alkérdés többször értékelődik ki.

Ügyeljünk a típuskompatibilitásra!

Hibás WHERE feltétel, mert az alkérdés rekordhalmazt ad vissza, amely nem hasonlítható össze a fizetés értékkel:

```
SELECT adószám, név FROM Dolgozó
  WHERE fizetés = (SELECT *
  FROM Dolgozó WHERE név='Kovács');
```

Helyesen:

```
SELECT adószám, név FROM Dolgozó
  WHERE fizetés = (SELECT fizetés
  FROM Dolgozó WHERE adószám=1234);
```

Összekapcsolás helyett alkérdés

Példa: *A pécsi olvasók által kikölcsönzött könyvek szerzője és címe:*

Könyv (könyvszám, szerző, cím, *olvasószám*, kivétel)
Olvasó (olvasószám, név, lakcím)

```
SELECT szerző, cím FROM Könyv, Olvasó
  WHERE Könyv.olvasószám = Olvasó.olvasószám
        AND lakcím LIKE 'Pécs,%';

SELECT szerző, cím FROM Könyv
  WHERE olvasószám IN
  (SELECT olvasószám FROM Olvasó
     WHERE lakcím LIKE 'Pécs,%');
```

Alkérdés UPDATE utasításban

UPDATE táblanév
SET oszlop = kifejezés, ..., oszlop = kifejezés
[WHERE feltétel];

Példa. *Fizetésemelés A12 projekt dolgozóinál:*

Dolgozó (adószám, név, fizetés)
Projekt (*adószám*, pkód, óraszám)

```
UPDATE Dolgozó
  SET fizetés=fizetés+10000
  WHERE adószám IN
    ( SELECT adószám FROM Projekt
      WHERE pkód='A12' );
```

Alkérdés INSERT utasításban

INSERT INTO táblanév [(oszloplista)] VALUES (értéklista);

INSERT INTO táblanév [(oszloplista)] alkérdés;

Példa: *származtatott tábla előállítása:*

Raktár (cikkszám, név, egységár, mennyiség)

Készlet (áru, érték)

```
CREATE TABLE Készlet
   ( áru   CHAR(20),
     érték INTEGER
   );
INSERT INTO Készlet
   SELECT név, egységár*mennyiség
     FROM Raktár;
```

További alkérdés-változatok

SQL szabványban szereplő, de nem minden DBMS által támogatott lehetőségek:

```
SELECT...FROM Tábla1
  WHERE (oszlop1,oszlop2) =
  (SELECT oszl1,oszl2 FROM Tábla2...);

SELECT ...
  FROM Tábla1, (SELECT...FROM Tábla2) AS t2

  WHERE ...;
```

Virtuális táblák (nézettáblák)

Adatbázis tartalma:

- alapadatok (törzsadatok),
- származtatott adatok.

Virtuális tábla = nézettábla = view:

nem tárol adatokat, tk. transzformációs formula.

Alkalmazása:

- származtatott adatok kezelése,
- adatok elrejtése.

CREATE VIEW táblanév [(oszloplista)] AS alkérdés;

- Csak a definíciót tárolja a rendszer.
- A nézettábla csak akkor generálódik, ha hivatkozunk rá.
- A nézettábla tartalma mindig aktuális.
- Lekérdezésben az alaptáblákkal azonos módon kezelhető.

1. példa: származtatott adatok kezelése

Raktár (cikkszám, név, egységár, mennyiség)

```
CREATE VIEW Készlet(áru, érték) AS
  SELECT név, egységár*mennyiség
    FROM Raktár;
```

Lekérdezés:

```
SELECT * FROM Készlet WHERE érték > 1000000;
```

2. példa: adatok elrejtése

Dolgozó (adószám, név, lakcím, osztálykód, fizetés)

```
CREATE VIEW Dolg2 AS
  SELECT adószám, név, lakcím
    FROM Dolgozó
    WHERE osztálykód='A02';
```

Lekérdezés:

```
SELECT név,lakcím FROM Dolg2
  ORDER BY név;
```

Nézettábla módosítása

A módosítás az alaptáblákon hajtódik végre.

Csak akkor lehet módosítani, ha a változás egyértelműen átvezethető az alaptáblákra.

Például nem lehet módosítani:

- kifejezéssel megadott oszlopot,
- DISTINCT opció esetén,
- FROM után több tábla esetén (join),
- GROUP BY alparancsnál.

Amikor nem lehet módosítani - 1

Raktár (cikkszám, név, egységár, mennyiség)

Kifejezéssel megadott oszlop:

```
CREATE VIEW Készlet(áru, érték) AS
  SELECT név, egységár*mennyiség
    FROM Raktár;
```

Hibás:

```
UPDATE Készlet SET érték=200000
 WHERE áru='alma';
```

Amikor nem lehet módosítani - 2

ProjÓra (adószám, projektkód, óra)

DISTINCT művelet:

```
CREATE VIEW HardProj(projkód) AS
  SELECT DISTINCT projektkód
    FROM ProjÓra
    WHERE óra>20;
```

Hibás:

```
DELETE FROM HardProj WHERE projkód='A12';
```

Amikor nem lehet módosítani - 3

Dolgozó (adószám, név, lakcím)
ProjÓra (*adószám*, projektkód, óra)

Join művelet:

```
CREATE VIEW DolgProj(név,pkód,óra) AS
  SELECT név, projektkód, óra
    FROM Dolgozó, ProjÓra
    WHERE Dolgozó.adószám = ProjÓra.adószám;
```

Hibás:

```
UPDATE DolgProj SET név='Kovácsné'
  WHERE projkód='A12';
```

Amikor nem lehet módosítani - 4

ProjÓra (adószám, projektkód, óra)

Csoportosítás:

```
CREATE VIEW SumProj(pkód,összóra) AS
  SELECT projektkód, SUM(óra)
    FROM Projóra
    WHERE óra<10
    GROUP BY projektkód;
```

Hibás:

```
UPDATE SumProj SET pkód='B12'
  WHERE pkód='A12';
```

Amikor lehet módosítani

Ha a nézettábla tartalmazza az alaptábla kulcsát, akkor általában lehet módosítani.

Példa:

Dolgozó (adószám, név, lakcím, osztálykód, fizetés)

Dolg2 (adószám, név, lakcím) az A02-es osztályra.

Nézettábla módosítása - 1

Dolgozó (adószám, név, lakcím, osztálykód, fizetés)

```
CREATE VIEW Dolg2 AS
  SELECT adószám,név,lakcím
    FROM Dolgozó
    WHERE osztálykód='A02';

UPDATE Dolg2
  SET lakcím=”Pécs, Kő u. 10.”
  WHERE adószám=1234;

INSERT INTO Dolg2
  VALUES (3333, 'Tóth Pál', 'Győr');
```

Probléma: osztálykód NULL lesz!

Nézettábla módosítása - 2

```
CREATE VIEW Dolg2 AS
   SELECT adószám,név,lakcím,osztálykód
   FROM Dolgozó
   WHERE osztálykód='A02';

INSERT INTO Dolg2
   VALUES (3333,'Tóth Pál','Győr','A02');
```

Probléma: Dolg2-n keresztül más osztályba is lehet új dolgozót felvenni!

Nézettábla módosítása - 3

```
CREATE VIEW Dolg2 AS
   SELECT adószám,név,lakcím,osztálykód
   FROM Dolgozó
   WHERE osztálykód='A02'
   WITH CHECK OPTION;
```

Lekérdezések kiértékelése

```
CREATE VIEW Dolg2 AS
  SELECT adószám,név,lakcím
    FROM Dolgozó
    WHERE osztálykód='A02';

SELECT lakcím FROM Dolg2
  WHERE név='Tóth Pál';
```

$E = \pi_{lakcim}(\sigma_{név='Tóth Pál'}(Dolg2))$

$Dolg2 = \pi_{adószám,név,lakcím}(\sigma_{osztkód='A02'}(Dolgozó))$

$\mathbf{E = \pi_{lakcim}(\sigma_{név='Tóth Pál'}(\pi_{adószám,név,lakcím} (\sigma_{osztkód='A02'}(Dolgozó))))}$