Az SQL nyelv

SQL nyelv szerepe

Sequental Query Language, deklaratív nyelv
Halmaz orientált megközelítés, a relációs algebra műveleteinek megvalósítására

Előzménye a SEQUEL (IBM)

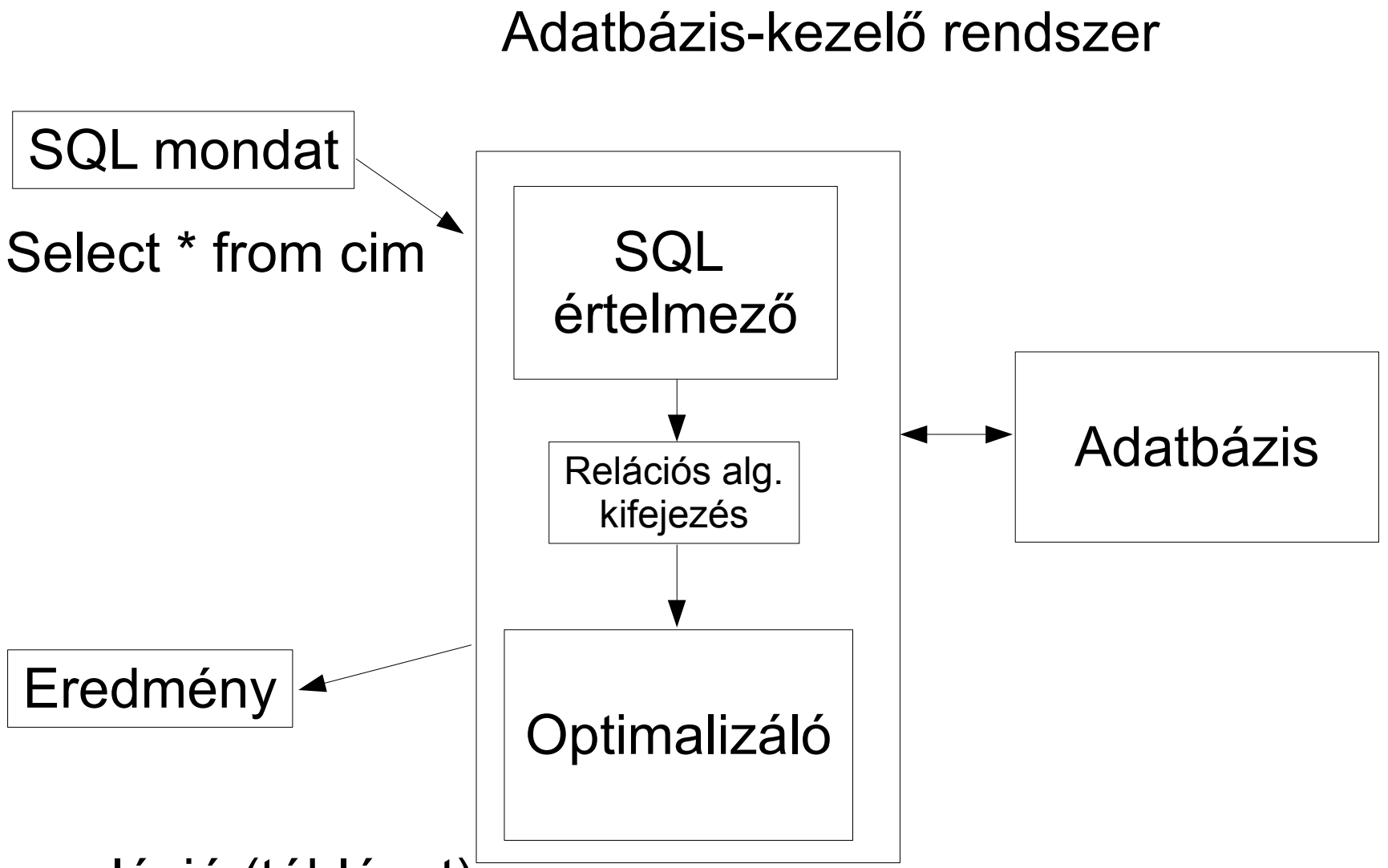
Algoritmus szerkezeteket (elágazás, ciklus) nem tartalmaz

A relációs kalkulus logikájára épül, nem azt kell megmondani, hogy hogyan kapjuk meg a kívánt adatokat, hanem azt, mely adatokat szeretnénk látni

A parancsok angol mondatok, egyszerűen értelmezhetők

Az SQL parancsok végrehajtása közben egy optimalizáló algoritmus határozza meg, hogy milyen elemi lépések során állítja elő az adatbázis-kezelő az eredményt, mely indexeket használ közben

SQL parancsok feldolgozása



Egy reláció (táblázat),
érintett sorok száma,
hibaüzenet

SQL szabványok

SQL-86
SQL-89
SQL-92
SQL:1999
SQL:2003
SQL:2006
SQL:2008

Követő szabvány
Számos tájszólás

Az SQL nyelv funkciói

Jóval több mint egy lekérdező nyelv

A lekérdezés mellett lehetővé teszi:

- Sémák definiálását (relációk, attribútumok)
- Tárolt adatok visszakeresését
- Tárolt adatok karbantartását (új adatok, módosítás, törlés)
- Hozzáférés korlátozását (felhasználók)
- Adatok megosztását (konkurens felhasználók)
- Adat integritás ellenőrzését (integritás kényszerek)

Az SQL az adatbázis-kezelő rendszer integrált része, nem egy önálló szoftver

Az SQL egy hatékony eszköz, mely a felhasználókat, a programokat és a számítógép rendszereket összeköti a relációs adatbázisban tárolt adatokkal

SQL felhasználási lehetőségei

Fejlesztőeszközökbe beépítve (űrlap, jelentés)
  CASE eszközök

Interaktív felhasználás (SQL konzol)
  Ad hoc lekérdezések
  adatbázis karbantartás

Más programnyelvekbe beépítve
  procedurális nyelvekben (pl. C, C++, Java, Php, stb.)
  adatbázis-kezelők saját procedurális kiegészítőibe (PL/SQL)

SQL szkriptek

Szintaktikai szabályok

Az SQL mondatokat pontosvesszővel zárjuk le

A kis és nagybetűket nem különbözteti meg

A parancsok szóközökkel, tabulátorokkal tagolhatók és több sorba írhatók

Szöveg konstansok szimpla aposztrófok között pl. 'alma'

Három értékű logika (3VL)

A	B	A OR B	A AND B	NOT A
True	True	True	True	False
True	Unknown	True	Unknown	False
True	False	True	False	False
Unknown	True	True	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
False	True	True	False	True
False	Unknown	Unknown	False	True
False	False	False	False	True

Használt jelölések

Az SQL utasítások leírásánál a következő jelöléseket alkalmazzuk

Nagybetűs szavak - az SQL nyelv fenntartott szavai
[szöveg] – opcionális része az utasításnak
elem1, *elem2*, … - egy vagy több elemből álló felsorolás
döltbetűs - behelyettesítendő érték
| - vagylagos elemek, a „|” jel baloldalán vagy jobboldalán szereplő érték

Az SQL nyelv elemei

Adat definíciós nyelv DDL (séma definíció és módosítás)
  CREATE, ALTER, DROP

Hozzáférés (Access Control)
  GRANT, REVOKE

Adat manipulációs nyelv DML
  INSERT, UPDATE, DELETE

Lekérdezés
  SELECT

Tranzakció kezelés
  COMMIT, ROLLBACK

Adat definíciós nyelv DDL

Adatbázis objektumok létrehozása, módosítása, megszüntetése

Adatbázis objektumok: tábla (reláció), index, nézet tábla, felhasználó, ...

CREATE TABLE *reláció_név*
(*attribútum_név adattípus* [(*szélesség*)] [CONSTRAINT *megszorítás_név*] [*oszlop_megszorítás*],
attribútum_név adattípus [(*szélesség*)] [CONSTRAINT *megszorítás_név*] [*oszlop_megszorítás*],
...) [CONSTRAINT *megszorítás_név*] [*reláció_megszorítás*];

Nevek – max. 128 karakter
Az angol ABC betűi, számjegyek és _ karakter
Betűvel vagy _ karakterrel kell kezdődnie

Adattípusok

CHAR [(*hossz*)]
megadott maximális hosszúságú karakterlánc, csak a karakterláncnak megfelelő hosszúságú területet foglalja el, szinonimája a VARCHAR. Maximum 255 karakterig.

NUMBER [(*szélesség*, *tizedes*)]
valós szám megadása, a szélesség mellett a tizedespont utáni jegyek száma is megadható, hasonlóan használható a FLOAT

INTEGER
egész típusú érték megadása, hozzá hasonló, de számábrázolási tartományában eltérő típus még a SMALLINT, szinonimája a DECIMAL

DATE
dátum megadása

TIME
Időpont megadás

DATETIME
Dátum és időpont megadás

RAW
a karakterhez hasonló típus, de az adatok értelmezésére nincs semmilyen feltételezés, így segítségükkel tetszőleges bináris adatokat tárolhatunk, például ábrákat is. Maximális hossza 255.

LONG
Maximum 64 KByte hosszú szöveg. Relációnként csak egy ilyen lehet, és csak korlátozott helyeken használható

LONGRAW
Mint a RAW, de 64 KByte hosszú adat.

Megszorítások

Egy oszlopra vonatkozó megszorítások

NULL az attribútum definíciójában arra utal, hogy az adat megadása nem kötelező, ez az alapértelmezés ezért a legritkább esetben írják ki.
NOT NULL az attribútum definíciójában arra utal, hogy az adat megadása kötelező, azaz nem vihető be olyan sor a relációban, ahol az így definiált adat nincs kitöltve.
PRIMARY KEY ez az oszlop a tábla elsődleges kulcsa.
UNIQUE ez az oszlop a tábla kulcsa.
CHECK(*feltétel*) csak feltételt kielégítő értékek kerülhetnek be az oszlopba.
[**FOREIGN KEY**] **REFERENCES** *reláció_név* [(*oszlop_név*)], ez az oszlop külső kulcs

Több oszlopra vonatkozó megszorítások

PRIMARY KEY(*oszlop1*[, *oszlop2*, ...]) ezek az oszlopok együtt alkotják az elsődleges kulcsot.
UNIQUE(*oszlop1*[, *oszlop2*, ...]) ezek az oszlopok együtt kulcsot alkotnak.
CHECK(feltétel) csak feltételt kielégítő sorok kerülhetnek be a táblába.
FOREIGN KEY (*oszlop1*[, *oszlop2*, ...]) **REFERENCES** *reláció*(*oszlop1*[, *oszlop2*, ...]), az oszlopok külső kulcsot alkotnak a megadott tábla oszlopaihoz.

Példák

```
CREATE TABLE osztaly (
   oszt_azon       INT2 PRIMARY KEY,
   nev             VARCHAR(14) NOT NULL,
   varos           VARCHAR(13) NOT NULL,
   CONSTRAINT  unev UNIQUE(nev));

CREATE TABLE alkalmazott (
   alk_azon      INT2 PRIMARY KEY,
   nev           VARCHAR(10) NOT NULL,
   beosztas      VARCHAR(9) NOT NULL,
   fonok         INT2 CONSTRAINT fonok_korl REFERENCES alkalmazott (alk_azon),
   belepes       DATE NOT NULL,
   fizetes       FLOAT4 NOT NULL,
   jutalom       FLOAT4,
   oszt_azon     INT2 NOT NULL,
   CONSTRAINT alk_kulso_kulcs FOREIGN KEY (oszt_azon) REFERENCES
                  osztaly (oszt_azon));

CREATE TABLE fiz_oszt (
   f_oszt          INT2 PRIMARY KEY,
   min             FLOAT4 NOT NULL,
   max             FLOAT4 NOT NULL,
   CONSTRAINT  min_max CHECK (min < max));
```

Reláció definíció módosítás

ALTER TABLE *reláció_név* ADD *attribútum_név adattípus* [(*szélesség*)];

ALTER TABLE *reláció_név*
MODIFY *attribútum_név adattípus* (*új_szélesség*) [NOT NULL];

ALTER TABLE *reláció_név* RENAME COLUMN *régi_név* TO *új_név*;

ALTER TABLE *reláció_név* DROP COLUMN *attribútum_név*;

DROP TABLE [IF EXISTS] *reláció_név*;

Példák

ALTER TABLE osztaly ADD oszt_vez int2 NOT NULL;
ALTER TABLE osztaly DROP COLUMN oszt_vez;
DROP TABLE osztaly; **kényszerek (alkalmazott külső kulcs hivatkozás) miatt nem hajtható végre!**

Indexek

CREATE [UNIQUE] INDEX *index_név* ON *reláció*(*oszlop1*, *oszlop2*, ...);

Egyedi index = kulcs

Példák

CREATE INDEX belepes_index ON alkalmazott(belepes);

DROP INDEX belepes_index;

Felhasználók

CREATE USER *név* IDENTIFIED BY *jelszó*;

ALTER USER *név* IDENTIFIED BY *jelszó*;

DROP USER *név*;

Adatmanipulációs nyelv

Táblák tartalmának módosítása

INSERT – új sorok hozzáadása a relációhoz
UPDATE – egy vagy több oszlop tartalmának módosítása
DELETE – sorok törlése a relációból

INSERT utasítás

INSERT INTO *reláció* [(*attribútum_név1*, *attribútum_név2*, ...)]
VALUES (*érték1*, *érték2*, ...);

INSERT INTO *reláció* VALUES (*érték1*, *érték2*, ...); rövid forma

INSERT INTO *reláció* SELECT … más táblák tartalmából
redundancia?

Példák

```
INSERT INTO osztaly VALUES (10,'SZAMLÁZÁS','BUDAPEST');
INSERT INTO osztaly oszt_azon, nev, varos VALUES (20,'KUTATÁS','ESZTERGOM');
INSERT INTO alkalmazott VALUES
    (7934,'MÜLLER','ELŐADÓ',7782,'23-JAN-82',1300,NULL,10);
```

UPDATE utasítás

UPDATE *reláció_név* SET *attribútum_név1* = *kifejezés1*, *attribútum_név2* = *kifejezés2*, ...
[WHERE *feltétel*];

Feltétel

Operátor	**Értelmezés**
=	egyenlő
!= <> ^=	nem egyenlő
>	nagyobb
>=	nagyobb egyenlő
<	kisebb
<=	kisebb egyenlő

Operátor	**Értelmezés (3VL)**
NOT	Logikai tagadás
AND	Logikai és
OR	Logikai vagy

Operátor	**Értelmezés**
BETWEEN x AND y	adott értékek közé esik
IN (a, b, c, ...)	az értékek között található
LIKE minta	hasonlít a mintára

Minta speciális elemei % és _

Precedencia

1. =, !=, <>, ^=, >, >=, <, <=
2. NOT
3. AND
4. OR

```
UPDATE alkalmazott SET fizetes = fizetes * 1.1;
UPDATE osztály SET varos = 'SZEGED' WHERE varos = 'ESZTERGOM';
```

DELETE utasítás

DELETE FROM *reláció_név* [WHERE *feltétel*];

Feltétel nélkül a tábla teljes tartalmát törli!
Feltétel elemei azonosak az UPDATE utasításnál használtakkal

Példák

DELETE FROM osztaly WHERE oszt_azon = 40;
DELETE FROM osztaly WHERE oszt_azon = 30; **Nem hajtható végre**

DELETE FROM alkalmazott; **Minden alkalmazott törlése**

Lekérdező nyelv

Egyetlen, de összetett utasítás, mellyel a relációs algebra műveletei megvalósíthatók

Legegyszerűbb alak, egy teljes tábla tartalma:
SELECT * FROM *reláció_név*;
Az oszlopok a táblázat sémájában szereplő sorrendben jelennek meg

Néhány SQL implementációban a FROM és a reláció név is Elmaradhat pl.
SELECT 2 + 4 * 6;

Példa
SELECT * FROM alkalmazott;
ELECT * FROM osztály;

Lekérdező nyelv folyt.

Vertikális megszorítás (projekció művelet)
SELECT *attribútum_név1*, *attribútum_név2*, … FROM *reláció_név*;

Az attribútumok a felsorolás sorrendjében jelennek meg, nem a sémában megadott sorrendben

Horizontális megszorítás (szelekció művelet)
WHERE klauzula
SELECT * FROM *reláció_név* WHERE *feltétel*;

Példák

SELECT nev, beosztas FROM alkalmazott;
SELECT * FROM alkalmazott WHERE fizetes > 1500;
SELECT nev, beosztas FROM alkalmazott WHERE jutalom IS NOT NULL;
SELECT nev, beosztas FROM alkalmazott WHERE jutalom IS NULL;

NULL értékkel összehasonlítás speciális művelettel!

Lekérdező nyelv folyt.

Összetett feltételek

Operátor	**Értelmezés**
BETWEEN x AND y	adott értékek közé esik
IN (a, b, c, ...)	az értékek között található
LIKE minta	hasonlít a mintára

Feltétel	**Értelmezés**
LIKE 'a%'	minden 'a' betűvel kezdődő
LIKE 'x_'	minden 'x'-el kezdődő kétbetűs
LIKE '%a%'	minden 'a' betűt tartalmazó
LIKE '_a%x'	második betű 'a' és 'x'-re végződő

A sorok egyediségének biztosítása

SELECT DISTINCT *attribútum1*, *attribútum2*, … FROM *reláció_név*;

Példák

```
SELECT * FROM alkalmazott WHERE belepes > '04-JUN-81';
SELECT * FROM alkalmazott WHERE jutalom >= 500;
SELECT * FROM alkalmazott WHERE fizetes BETWEEN 1000 and 2000;
SELECT * FROM alkalmazott WHERE fonok IN (7902, 7566, 7788);
SELECT * FROM alkalmazott WHERE nev LIKE 'S%';
SELECT * FROM alkalmazott WHERE nev LIKE '____';
SELECT * FROM alkalmazott WHERE oszt_azon in (10,20)
```

Lekérdező nyelv folyt.

Kifejezések használata

Az attribútumok helyén kifejezés is megjelenhet a SELECT után és a feltételben (AS - átnevezés)

SELECT *kifejezés1* [AS *alias_név1*], *kifejezés2* [AS *alias_név2*], … FROM *reláció_név*;

Kifejezésekben alapműveletek és függvények lehetnek

LOWER(), UPPER(), LEN()
ROUND(), EXP(), LOG()
NOW()

A függvények köre és nevei adatbázis-kezelőnként elég változatosak

Példák

```
SELECT nev, beosztas, fizetes * 12 AS evesfiz, oszt_azon FROM alkalmazott;

SELECT nev, beosztas, fizetes * 12 AS evesfiz, oszt_azon FROM alkalmazott
      WHERE fizetes * 12 > 20000;

SELECT nev, fizetes * 12 + jutalom  AS evesjov FROM alkalmazott;
```

NULL adat kifejezésben!

Lekérdező nyelv folyt.

Lekérdezés eredményének rendezése, ORDER BY klauzula

SELECT * | *kifejezés1*, *kifejezés2*, … FROM *reláció* [WHERE *feltétel*]
ORDER BY *attribútum1* [ASC | DESC] [, *attribútum2* [ASC | DESC], ...;]

A rendezés több attribútum figyelembevételével történhet,
az attribútum nevek helyett az oszlop sorszám is használható

Példák

```
SELECT nev, beosztas, belepes FROM alkalmazott ORDER BY belepes DESC;
SELECT oszt_azon, beosztas, nev FROM alkalmazott ORDER BY oszt_azon, fizetes DESC;
SELECT nev, fizetes * 12 FROM alkalmazott ORDER BY 2;
```

Lekérdező nyelv folyt.

Aggregátor függvények: min(), max(), sum(), avg(), count()
Több rekordból számított érték a teljes relációra vagy a rekordok csoportjaira, NULL értékeket figyelmen kívül hagyja

SELECT min(*attribútum*) FROM *reláció* WHERE *feltétel*;

Csoportképzés, GROUP BY klauzula

SELECT *attribútum1*, *attribútum2*, ..., min(*attribútum*) FROM *reláció* [WHERE *feltétel*]
GROUP BY *attribútum1*, *attribútum2*, … [HAVING *feltétel*]
[ORDER BY *attribútum1*, *attribútum2*, …];

Példák

```
SELECT max(fizetes), sum(fizetes), avg(fizetes), count(fizetes) FROM alkalmazott;
SELECT avg(fizetes) FROM alkalmazott WHERE oszt_azon=10;
SELECT nev, avg(fizetes) FROM alkalmazott;     HIBA!!!
SELECT oszt_azon, avg(fizetes) FROM alkalmazott GROUP BY oszt_azon ORDER BY 1;
SELECT oszt_azon, beosztas, avg(fizetes), count(*) FROM alkalmazott
      GROUP BY oszt_azon, beosztas ORDER BY 1, 2;
SELECT count(jutalom) FROM alkalmazott;
SELECT oszt_azon, avg(fizetes) from alkalmazott WHERE beosztas != 'ELOADO'
      GROUP BY oszt_azon HAVING COUNT(*) > 2;
```

Lekérdező nyelv folyt.

Több táblás lekérdezések

Descartes szorzat

SELECT * FROM *reláció1*, *reláció2*;

Egyen összekapcsolás

SELECT * FROM *reláció1*, *reláció2* WHERE *reláció1.attribútum1 = reláció2.attribútum2*;
Vagy
SELECT * FROM *reláció1* INNER JOIN *reláció2*
ON *reláció1.attribútum1 = reláció2.attribútum2;*

Példák

SELECT * FROM alkalmazott, osztaly;
SELECT * FROM alkalmazott, osztaly WHERE alkalmazott.oszt_azon = osztaly.oszt_azon;
SELECT * FROM alkalmazott INNER JOIN osztaly ON
alkalmazott.oszt_azon = osztaly.oszt_azon;
SELECT alkalmazott.nev, osztaly.nev FROM alkalmazott INNER JOIN osztaly ON
alkalmazott.oszt_azon = osztaly.oszt_azon WHERE osztaly.oszt_azon = 10;

Lekérdező nyelv folyt.

Nem egyen összekapcsolás

SELECT * FROM *reláció1*, *reláció2* WHERE *reláció1.attribútum1* >|< *reláció2.attribútum2*;

Reláció aliasok

SELECT *alias.attribútum* FROM *relació* [AS] *alias*;

Tábla összekapcsolása önmagával

SELECT *alias1.attribútum1*, *alias2.attribútum2*, …
FROM *reláció* [AS] *alias1*, *reláció* AS *alias2;*

Külső összekapcsolás

SELECT * FROM *reláció1* LEFT | RIGHT JOIN *reláció2*
ON *reláció1.attribútum1 = reláció2.attribútum2;*

Példák

```
SELECT a.nev, a.fizetes, f.f_oszt FROM alkalmazott a, fiz_oszt f
    WHERE a.fizetes BETWEEN f.min AND f.max;
SELECT a.nev AS főnök, b.nev AS beosztott FROM alkalmazott a, alkalmazott b
    WHERE b.fonok = a. alk_azon;  összekapcsolás önmagával
SELECT a.nev AS főnök, b.nev AS beosztott FROM alkalmazott a RIGHT JOIN  alkalmazott b
        ON b.fonok = a.alk_azon;
```

Lekérdező nyelv folyt.

Halmaz műveletek megvalósítása,
két SELECT összekapcsolásával, a két SELECT-nek azonos szerkezetű relációt kell visszaadnia

SELECT … UNION SELECT ...;
SELECT … INTERSECT SELECT ...;
SELECT … EXCEPT SELECT ...;

Példák

```
SELECT DISTINCT oszt_azon FROM alkalmazott WHERE beosztas = 'MANAGER'
UNION   osztályok, ahol manager vagy ügynök dolgozik, in-nel is megfogalmazható
SELECT DISTINCT oszt_azon FROM alkalmazott WHERE beosztas = 'ÜGYNÖK';

SELECT DISTINCT oszt_azon FROM alkalmazott WHERE beosztas= 'MANAGER'
INTERSECT  azok az osztályok ahol manager és ügynök is dolgozik
SELECT DISTINCT oszt_azon FROM alkalmazott WHERE beosztas = 'ÜGYNÖK';

SELECT oszt_azon FROM osztaly
EXCEPT   azok az osztályok, ahol nem dolgozik senki, not in-nel is lehet
SELECT DISTINCT oszt_azon FROM alkalmazott;
```

Lekérdező nyelv folyt.

Egymásba ágyazott lekérdezések, pl. Szabóval azonos osztályon dolgozók megválaszolása egy lekérdezéssel.
SELECT utasítás feltételébe (WHERE) egy másik SELECT utasítást helyezünk el
Két lehetőség
- a beágyazott SELECT egy rekordot ad vissza
- a beágyazott SELECT több rekordot add vissza

Speciális összehasonlítások: exists, in, not in, > all, < any

Példák

szabóval azonos munkakörben dolgozók, ha több Szabó nevű van akkor nem működik

```
SELECT nev, beosztas FROM alkalmazott WHERE beosztas =
      (SELECT beosztas FROM alkalmazott WHERE nev = 'SZABÓ');
```

valamelyik Szabó alkalmazottal azonos beosztásban dolgozók

```
SELECT nev, beosztas FROM alkalmazott WHERE beosztas IN
      (SELECT beosztas FROM alkalmazott WHERE nev = 'SZABÓ');
```

az átlagfizetésnél magasabb átlagú beosztások

```
SELECT beosztas, avg(fizetes) FROM alkalmazott GROUP BY beosztas
      HAVING avg(fizetes) > (SELECT avg(fizetes) FROM alkalmazott);
```

Lekérdező nyelv folyt.

Korrelált lekérdezés – a beágyazott lekérdezés visszautal a külső lekérdezésre, a belső lekérdezés a külső lekérdezés minden egyes rekordjára újra kiértékelésre kerül

Példák

akiknek nagyobb a fizetésük, mint az osztályuk átlaga

```
SELECT oszt_azon, nev, fizetes FROM alkalmazott a WHERE fizetes >
    (SELECT avg(fizetes) FROM alkalmazott WHERE oszt_azon = a.oszt_azon)
    ORDER BY oszt_azon;
```

akiknek legalább egy beosztottja van

```
SELECT nev, beosztas, oszt_azon FROM alkalmazott a WHERE EXISTS
    (SELECT * FROM alkalmazott WHERE fonok = a.alk_azon);
```

osztályonként az utolsónak belépett dolgozó

```
SELECT oszt_azon, nev, belepes FROM alkalmazott where (oszt_azon, belepes) in
    (SELECT oszt_azon, max(belepes) FROM alkalmazott GROUP BY oszt_azon)
    ORDER BY belepes desc;
```

osztályok, melyeknek nincs dolgozója

```
SELECT nev FROM osztaly o WHERE NOT EXISTS
    (SELECT * FROM alkalmazott WHERE oszt_azon = o.oszt_azon);
```

Gyakorlópéldák

Alkalmazott nevének, az osztály nevének és a fizetési osztály lekérdezése az alkalmazott nevek ABC sorrendjében.

Osztályonként a dolgozók száma, az osztálynév sorrendben.

A 4. fizetési osztályba tartozók közül a legmagasabb jövedelmű dolgozó(k) neve