Adatbáziskezelés (Oracle SQL)

----------Adatbázishoz kapcsolódó parancsok, jogok kiosztása, tranzakciók----------

1. ADATBÁZIS LÉTREHOZÁSA:

CREATE DATABASE adatbázisnév;

adatbázisnév a létrehozandó adatbázis neve

2. INFORMÁCIÓ ADATBÁZISRÓL:

SHOW DATABASE;

3. ADATBÁZIS MEGNYITÁSA:

START DATABASE adatbázisnév;

adatbázisnév a megnyitandó adatbázis neve (**csak létező lehet!!!**)

4. ADATBÁZIS BEZÁRÁSA:

STOP DATABASE;

5. ADATBÁZIS TÖRLÉSE:

DROP DATABASE adatbázisnév;

adatbázisnév a törlendő adatbázis neve

6. HOZZÁFÉRÉSI JOGOK ADOMÁNYOZÁSA/VISSZAVONÁSA:

A) A táblákra vonatkozó jogosultságok adományozása

GRANT ALL[PRIVILEGES] /jogosultséglista **ON [TABLE]** táblalista **TO PUBLIC** /felhasználólista **[WITH GRANT OPTION];**

A parancs minden jogot (*ALL PRIVILEGES*) vagy a jogosultséglistában szereplő műveletekre való jogot adja át a táblalistában szereplő táblákra mindenkinek (*PUBLIC*), vagy a felhasználólistában szereplő személyeknek. Amennyiben a WITH GRANT OPTION szerepel, az e jogokat kapók át is adathatják ezeket a jogokat másoknak. A ***kioszthatók jogok*** listája:

ALTER	Jogosultság a tábla módosítására
DELETE	Jogosultság a tábla törlésére
INDEX	Jogosultság indextábla létrehozására
INSERT	Jogosultság új sor felvételére a táblázatba
SELECT	Jogosultság lekérdezésre
UPDATE	Jogosultság a tábla módosítására

Példa:

GRANT INSERT,SELECT **ON** *table1,table2* **TO** *user1,user2* **WITH GRANT OPTION**;

B) Jogosultság adományozása az adatbázison végzett műveletekre

GRANT adatbázisjog **TO PUBLIC** /felhasználónév;

A parancs jogosultságot ad az adatbázisra vonatkozóan mindenkinek (PUBLIC), vagy adott felhasználóknak a felhasználólista szerint. A **kioszthető adatbázisjogok**:

CONNECT – Hozzáférés a teljes adatbázishoz
– Jog arra, hogy SELECT,INSERT, DELETE, UPDATE műveleteket végezzen más felhasználók tábláin, ha ilyen jogosultságot kapott a táblára vonatkozó GRANT-tal
– Jog nézettáblák és szinonim táblák létrehozására

RESOURCE – Minden CONNECT jogosultság
– Jogosultság táblák és indextáblák létrehozására, jogosultságok adományozása ezekre a táblákra

DBA – Teljes adatbázis-adminisztrátori jog

Példa:

GRANT DBA **TO** user1;

C) Táblára vonatkozó jogok visszavonása

REVOKE ALL [PRIVILEGES] /jogosultséglista **ON [TABLE]** táblalista **TO PUBLIC** /felhasználólista;

Az összes jogosultságot (ALL PRIVILEGES), vagy csak a jogosultséglistában felsoroltakat a megadott táblákra vonatkozóan mindenkitől (PUBLIC), vagy csak a listában szereplő felhasználóktól visszavonja.

Példa:

REVOKE SELECT **ON** *table1* **TO** *user1*;

D) Adatbázis-jogosultságok visszavonása

REVOKE adatbázisjog **FROM PUBLIC** /felhasználólista;

Az adatbázisjogokat mindenkitől (PUBLIC), vagy a listában szereplőktől visszavonja.

Példa:

REVOKE DBA **FROM** *user1*;

----------Táblák létrehozása, feltöltése, karbantartása, törlése----------

1. TÁBLÁK LÉTREHOZÁSA

CREATE TABLE táblanév
(oszlopnév adatleírás,
oszlopnév adatleírás);

táblanév a létrehozandó tábla neve
oszlopnév egy oszlop neve
adatleírás az oszlopban tárolandó adatok típusának* és méretének megadása

- *Típusok:
Szám: NUMBER(számjegyek, tizedesek)

Karakter: CHAR(hossz) - fix hossz, max 2000 bájt
VARCHAR2(max hossz) - változó hossz, max 4000 bájt
LONG - változó hossz, max 2GB
CLOB - karaktersorozat tárolása, max 4GB

Bináris adat: RAW(méret) - max 2000 bájt
LONG RAW - max 2GB
BLOB - max 4GB

Dátum: DATE
SYSDATE - a rendszerdátum

Az oszlopnéven és adatleíráson túl megadhatunk oszlopelemekre vonatkozó ***alapértelmezett*** értéket is.
Például: ...oszlopnév adatleírás ***DEFAULT*** kifejezés,
konkrétan,
...evfolyam number(1) DEFAULT 1,

Továbbá vannak különféle **oszlopmegszorítások**:

NULL Nem azonos a nullával!!! Annyit jelent hogy az adott oszlophoz tartozó mező lehet üres is. (Ez az alapértelmezett)
NOT NULL Ezzel azt adjuk meg hogy az adott oszlophoz tartozó mező nem lehet üres.
UNIQUE A mező értéke minden rekordnál(sornál) különböző kell hogy legyen.
CHECK A mező értékének és a CHECHK utáni feltételnek meg kell egyeznie.
PRIMARY KEY Elsődleges kulcs
FOREIGN KEY Idegen kulcs

Az elsődleges és idegen kulcsot is többféleképpen meg lehet adni.

Például: CREATE TABLE hallgatok
(neptunkod char(6) PRIMARY KEY,

```
 nev varchar2(30),
 szak_azon char(1) references szak(szakkod));

CREATE TABLE hallgatok
(neptunkod char(6) PRIMARY KEY,
 nev varchar2(30),
 szak_azon char(1),
 PRIMARY KEY(neptunkod)
 FOREIGN KEY(szak_azon) references szak(szakkod));
```

Miután átvettük azokat a lehetőségeket, amelyek a tábla létrehozásakor adódhatnak, nézzünk egy összefoglaló példát.

```
CREATE TABLE summery
        (id number(6) PRIMARY KEY,
         nev varchar2(30) NOT NULL,
         evfolyam number(1) DEFAULT 1,
         tan_kod char(5) UNIQUE,
         letszam number(3) CHECK (letszam>0),
         szak_azon char(1) REFERENCES szak(szakkod));
```

Miután már létrehoztuk a táblá(ka)t ami(ke)t szerettünk volna, fel kell töltenünk adatokkal. Erre az INSERT INTO parancs szolgál.

2. TÁBLÁK FELTÖLTÉSE

INSERT INTO táblanév (oszlopnév,oszlopnév) ***VALUES*** (érték,érték);

táblanév	Egy létező táblának a neve.
oszlopnév	A tábla oszlopának egy neve. Megadása opcionális (csak ha nem minden oszlophoz rendelünk értéket).
érték	Az adott oszlopnak megfelelő típusú érték.

Ha nem adunk meg oszlopneveket, akkor az új sorban a tábla minden oszlopának adunk értéket. Ha megadjuk az oszlopneveket, akkor csak azok az oszlopok kapnak értéket, amelyek a felsorolásban szerepelnek. A többi oszlop értéke - ha nem adtunk meg DEFAULT értéket - NULL lesz. Ezek az oszlopok nem lehetnek NOT NULL opcióval definiálva.

Példák:

```
INSERT INTO hallgato (hkod,nev,szak,evf)
        VALUES (10001,'KEREK ERVIN','KONYVTAR',2)

INSERT INTO anyag VALUES(2233,'AGYAG','T')
```

Ha ***automatikus sorszámozás***t szeretnénk megvalósítani, arra is van lehetőségünk a ***CREATE SEQUENCE*** utasítással.

```
CREATE SEQUENCE szekvencianév
```

INCREMENT BY növelés_értéke
START WITH kezdőérték
MAXVALUE maximumérték/***NOMAXVALUE***
MINVALUE minimumérték/***NOMINVALUE***
CYCLE/***NOCYCLE***
CACHE darabszám/***NOCACHE***
ORDER/***NOORDER***;

A paraméterek bármelyike elhagyható, ekkor az alapértelmezett értékek kerülnek beállításra.

- MAXVALUE/MINVALUE: Az elérhető maximális/minimális érték. Ha eddig eljut a generált érték, ennél nagyobb/kisebb érték nem kerül generálásra.
- CYCLE/NOCYCLE: Ha eléri a szélsőértékeket (MAXVALUE/MINVALUE), újrakezdődik a generálás (CYCLE), vagy nem generálódik több új érték (NOCYCLE). Az alapértelmezés a NOCYCLE.
- CACHE: Megadható, hogy hány értéket generáljon le a gép előre és tartson a memóriában a gyorsabb elérés végett. Alapértelmezés a NOCACHE.
- ORDER/NOORDER: Garantáltan sorrendben generál (ORDER), vagy erre nincs garancia (NOORDER). Alapértelmezés a NOORDER.

Példa:

CREATE SEQUENCE sorszam START WITH 1 INCREMENT BY 2 MAXVALUE 4
CYCLE CACHE 2 ORDER;

A rekordok felvitele egy táblába amiben sorszámozást szeretnénk használni a következőképpen néz ki:

INSERT INTO tabla VALUES (sorszam.nextval,'valami');
INSERT INTO tabla VALUES (sorszam.currval,'valami');

- ***NEXTVAL***: Első híváskor a kezdőértékkel tér vissza, a többi híváskor növeli a szekvencia értékét és azzal tér vissza. Ha egy SQL parancsban többször is meghíjuk, akkor csak egyszer növeli az értékét.
- ***CURRVAL***: A szekvencia aktuális értékével tér vissza, nem növeli azt.

3. TÁBLÁK KARBANTARTÁSA:

A) Most már, hogy vannak **adat**ok a táblázatunkban. Előfordulhat, hogy azokat **módosít**ani szeretnénk. Erre az UPDATE… SET… parancs használható.

UPDATE táblanév ***SET*** oszlopnév=kifejezés, oszlopnév=kifejezés ***WHERE*** feltétel;

táblanév	Egy létező tábla neve.
oszlopnév	A tábla egy oszlopának neve.
kifejezés	Az adott oszlopnak megfelelő típusú értéket eredményező kifejezés
feltétel	Logikai kifejezés, értéke igaz vagy hamis.

Az adatok módosítása abban a sorban, vagy sorokban történik meg, melyekre a feltétel igaz. A megadott oszlopokba az adott értékek kerülnek beírásra. **VIGYÁZAT** a WHERE záradék nélkül az összes sorra érvényes!!!

Példák:

```
UPDATE hallgato
        SET eredm=3.85 WHERE nev='KIS KATALIN';
```

Értelmezése: A hallgato táblában az eredmény oszlop mezőjét 3,85-re változtatja ahol a nev egyenlő KIS KATALIN-nal.

```
UPDATE hallgato
        SET eredm=0 WHERE evf=2;
```

Értelmezése: A hallgato táblában az eredmény oszlop mezőjét 0-ra változtatja ahol az evf egyenlő 2-vel.

B) Természetesen nemcsak módosítani lehet, hanem ***töröl***ni is. A törlés az nem mezőre hanem ***sor***ra vonatozik!!!

DELETE FROM táblanév ***WHERE*** feltétel;

táblanév	Egy létező tábla neve.
feltétel	Logikai kifejezés, értéke igaz vagy hamis.

A táblából azon sor, vagy sorok törlődnek, melyekre a megadott feltétel igaz. Ha nem adunk meg feltételt, akkor minden sor törlődik (üres táblát eredményez!!!).

Példák:

```
DELETE FROM hallgato WHERE hkod=11234;
```

```
DELETE FROM anyag;
```

C) Egy táblához ***hozzáad***hatunk új oszlopokat, vagy ***módosít***hatjuk a meglevő oszlopok definícióját az **ALTER TABLE… ADD/MODIFY** utasítással.

```
ALTER TABLE táblanév
        ADD (oszlopnév adatleírás, oszlopnév adatleírás)
        MODIFY (oszlopnév adatleírás, oszlopnév adatleírás);
```

táblanév	Egy létező tábla neve
oszlopnév	A hozzáadni vagy módosítani kívánt oszlop neve
adatleírás	Az oszlopban tárolandó adatok típusának és méretének megadása

Az ADD opcióval új oszlopot tudunk hozzávenni a táblázathoz.
A MODIFY opcióval egy meglevő oszlop típusát, méretét és a NOT NULL opciót tudjuk megváltoztatni.

Példa:

```
ALTER TABLE hallgato
        ADD (szhely CHAR(20))
        MODIFY (nev char(30));
```

D) Arra is van mód, hogy a felesleges ***oszlopokat töröl***jük. Erre az ***ALTER TABLE …DROP COLUMN*** utasítás szolgál.

ALTER TABLE táblanév DROP COLUMN oszlopnév;

táblanév	Egy létező tábla neve.
oszlopnév	A törölni kívánt oszlop neve.

Példa:

ALTER TABLE elso DROP COLUMN osztondij;

E) Az oszlopokat ***át*** is lehet ***nevez***ni az ***ALTER TABLE… RENAME COLUMN… TO*** paranccsal.

ALTER TABLE táblanév RENAME COLUMN oszlopnév1 TO oszlopnév2;

táblanév	Egy létező tábla neve.
oszlopnév1	Az átnevezni kívánt oszlop neve.
oszlopnév2	Az oszlop új neve.

Példa:

ALTER TABLE elso RENAME COLUMN osztondij TO zsebpenz;

F) Nemcsak az oszlopokat, hanem a ***táblázat***ot is ***átnevez***hetjük. Erre az ***ALTER TABLE… RENAME TO…*** utasítással van lehetőségünk.

ALTER TABLE táblanév RENAME TO új_táblanév;

táblanév	Egy létező tábla neve.
új_táblanév	A tábla új neve.

Példa:

ALTER TABLE elso RENAME TO first;

4. TÁBLÁK TÖRLÉSE

DROP táblanév;

Példa:

DROP elso;

----------Lekérdezés az adatbázisból----------

SELECT * ***FROM*** táblanév;

A fent látható példa a legegyszerűbb SELECT utasítást prezentálja. Hatására az adott tábla teljes tartalma kerül kilistázásra a képernyőn. Természetesen van rá mód, hogy különféle opciókkal bővítsük a lehetőségeinket.

- Oszlopszűrés a SELECT segítségével:

SELECT oszlop1,oszlop2 ***FROM*** táblanév;

Az így megadott utasítással a táblának csak a felsorolt oszlopai kerülnek megjelenítésre. Ha egy adott oszlophoz tartozó éték többször is előfordul, akkor ***többször lesz kilistázva. Ha ezt nem szeretnénk*** (tehát minden érték csak egyszer forduljon elő) akkor a ***DISTINCT*** opciót kell használnunk.

Példa:

SELECT DISTINCT oszlop1,oszlop2 FROM táblanév;

A későbbiek folyamán találkozni fogunk olyan helyzettel amikor több táblával dolgozunk egy lekérdezésen belül. Ekkor ugyan így megadhatjuk, hogy mely oszlopokat akarjuk lekérni a táblából. DE!!! mód van arra is, hogy az egyik tábla mindegyik oszlopát lekérjük míg a másikból csak néhányat.

Példa:

SELECT oszlop1,oszlop2,táblanév2.* FROM táblanév1,táblanév2;

A fenti utasítást kiadva az első táblának a megadott 2 oszlopa illetve a második táblának minden oszlopa belekerül a lekérdezésbe és így ezek lesznek kilistázva is.

A lekérdezés során különböző ***oszlopokat össze is fűzhetünk*** és ezeket már összefűzve egy új név alatt ki tudjuk listázni. Az összefűzésre a || karakter (2x AltGr+W) használandó.

Példa:

SELECT oszlop1||' '||oszlop2 AS uj_oszlopnev from táblanév;

Arra is van lehetőségünk, hogy egy oszlopot összefűzés nélkül új névvel jelenítsünk meg.

Példa:

SELECT oszlop1 AS uj_oszlopnev from táblanév;

A lekérdezések során használhatók matematikai operátorok is.

+ - -> Előjel 1 operandus esetén.
+ - -> Összeadás, kivonás 2 operandus esetén.
** /* -> Szorzás, osztás

(konkrét)Példák:

1.SELECT -fizetés FROM táblanév;
- Jelentése: A fizetés oszlop értékeit listázza negatív előjellel.
2.SELECT fizetés+1 FROM táblanév;
- Jelentése: A fizetés oszlop értékei 1-el megnövelve kerülnek kilistázásra.
3.SELECT (oszlopnev+100)/2 FROM táblanév;
- Jelentése: Az oszlop értékei 100-al megnövelve és 2-vel osztva listázódnak.
4.SELECT oszlopnev*2 FROM táblanév;
- Jelentése: Az oszlop értékei megduplázza jelennek meg.

- Sorszűrés a SELECT segítségével:

SELECT oszlop1,oszlop2 ***FROM*** táblanév ***WHERE*** feltétel;

Csak azok a sorok kerülnek kilistázásra, amelyek a WHERE után megadott feltételnek eleget tesznek. A feltétel megadásakor használhatunk: ***ÖSSZEHASONLÍTÓ OPERÁTOROK***AT,

=	-> Egyenlő.
<>	-> Nem egyenlő.
<,>,<=,>=	-> Kisebb, nagyobb, kisebb egyenlő, nagyobb egyenlő (Szövegre is működik az ABC sorrend alapján).
IS NULL	-> NULL érték esetén teljesül.
IS NOT NULL	-> Kitöltött érték esetén teljesül.
LIKE	-> Egy adott maszkhoz való illeszkedést vizsgál.
NOT LIKE	-> Maszktól való eltérést vizsgál.

A ***LIKE*** maszkban használható helyettesítő karakterek:

_	1 db tetszőleges karaktert helyettesít.
%	Tetszőleges számú tetszőleges karaktert helyettesít.

Példák:

```
SELECT * FROM dolgozo WHERE nev='Bármi Áron';
SELECT * FROM dolgozo WHERE dolgozik IS NOT NULL;
SELECT * FROM dolgozo WHERE nev LIKE 'J%o_';
```

LOGIKAI OPERÁTOROKAT (A precedencia sorrend is ez, a NOT a legerősebb),

NOT - Tagadás.
AND - Logikai ÉS.
OR - Logikai VAGY.

Igazságtábla

A	B	NOT A	A AND B	A OR B
I	I	H	I	I
I	H	H	H	I
H	I	I	H	I
H	H	I	H	H

Példák:

SELECT * FROM dolgozo WHERE NOT(nev='Bármi Áron');
SELECT * FROM dolgozo WHERE nev='Bármi Áron' AND kor<25;
SELECT * FROM dolgozo WHERE hely='Debrecen' OR hely='Berettyóújfalu';

HALMAZJELLEGŰ ÖSSZEHASONLÍTÓ OPERÁTOROK,

IN	-> Teljesül, ha a vizsgált elem a halmazban van.
NOT IN	-> Teljesül, ha nincs a vizsgált elem a halmazban.
ANY	-> Teljesül, ha a halmaz legalább egy elemére teljesül az ANY előtti feltétel.
ALL	-> Teljesül, ha a halmaz minden elemére teljesül az ALL előtti feltétel.
BETWEEN X AND Y	-> Teljesül, ha a vizsgált elem X és Y közé esik.
NOT BETWEEN X AND Y	-> Teljesül, ha a vizsgált elem nem esik X és Y közé.

Példák:

SELECT * FROM dolgozo WHERE varos IN('Berettyóújfalu','Debrecen','Budapest');

SELECT * FROM dolgozo WHERE
varos=ANY('Berettyóújfalu','Debrecen','Budapest');

SELECT * FROM dolgozo WHERE
varos>ANY('Berettyóújfalu','Debrecen','Budapest');

SELECT * FROM dolgozo WHERE
varos<ALL('Berettyóújfalu','Debrecen','Budapest');

SELECT * FROM dolgozo WHERE fizetes BETWEEN 60000 AND 150000;

SELECT * FROM dolgozo WHERE nev BETWEEN 'K' AND 'M';

- Kiválasztott sorok ***csoportosítása***, ***rendezése***

A SELECT utasításban rendelkezhetünk a kiválasztott sorok csoportosításáról, rendezéséről is, erre szolgál a ***GROUP BY*** és az ***ORDER BY*** opció.

Mielőtt a fenti két utasítással foglalkoznánk, előbb az ***oszlopfüggvények***et mutatom be.

COUNT - Darabszám meghatározása.

Példa:

SELECT COUNT(szigszam) FROM dolgozo;

Magyarázat: Mesgzámolja, hogy a dolgozo táblában hány darab szigszam oszlophoz tartozó érték található. Ami többször fordul elő, azt többször is számolja.

SELECT COUNT(distinct varos) FROM dolgozo;

Magyarázat: Az előzőhöz képest annyi az eltérés, hogy mindegy hányszor fordul elő egy adott érték, az egynek fog számítani.

MIN - Minimum kiválasztása.

Példa:

SELECT MIN(fizetes) FROM dolgozo;

MAX - Maximum kiválasztása.

Példa:

SELECT MAX(fizetes) from dolgozo;

AVG - Átlagszámítás.

Példa:

SELECT AVG(fizetes) from dolgozo;
SELECT AVG(fizetes) from dolgozo WHERE varos<>'Dunaújváros';

SUM - A megadott mezőben szereplő értéek összege.

Példa:

SELECT SUM(fizetes) from dolgozo;

A GROUP BY opcióban megadhatunk egy, vagy több oszlopot, amelyben azonos értéket tartalmazó sorokat csoportosítani szeretnénk, általában valamilyen összesítés céljára.

Csoportosítás esetén a lekérdezés csak két fajta mezőre vonatkozhat:
1. Olyanra amely szerepel a GROUP BY záradékban is
2. Olyanra amelyre valamelyik oszlopfüggvény van alkalmazva

Példák:

SELECT varos, AVG(fizetes) from dolgozo group by varos;
SELECT varos, AVG(fizetes) from dolgozo where dolgozik in('info','karb') group by varos;

A csoportképzés kiegészíthető a *HAVING* záradékkal. Ez a GROUP BY által létrehozott csoportok további szűrését valósítja meg.

Példa:

SELECT varos, AVG(fizetes) from dolgozo group by varos HAVING AVG(fizetes)>70000;

Az ORDER BY opcióban azokat az oszlopokat definiálhatjuk, melyek szerint az eredménytábla rendezett lesz. Az ASC opció növekvő sorrendet jelent - ez az alapértelmezés - a DESC pedig csökkenőt. Ha több oszlopot definiálunk az ORDER BY parancs után akkor mindegyik szerint rendezve lesz az eredménytábal és a kiértékelés a megadás sorrendjében történik.

Példák:

SELECT * FROM dolgozo ORDER BY nev;
SELECT nev,fizetes FROM dolgozo ORDER BY nev DESC;
SELECT nev,fizetes FROM dolgozo ORDER BY fizetes,nev;

- BELSŐ SELECT (alkérdés):

Egy SELECT utasítás WHERE feltételében állhat egy újabb SELECT utasítás (allekérdezés). Erre általában akkor van szükség, ha a sorok kiválasztása valamely más táblázatban található információn alapul. A belső SELECT hivatkozhat ugyanarra a táblázatra is, mint a külső. Ez akkor fordul elő, ha nem tudjuk egyetlen kifejezésben megfogalmazni a leválogatási feltételt. A belső SELECT-ben szerepelhet oszlopfüggvény is.
A belső SELECT-ek használata több szintű is lehet, vagyis egy belső SELECT-nek is lehet belső SELECT-je. A lekérdezések végrehajtása, belűről kezdődik, és kifelé halad. A legbelső SELECT hajtódik végre legelőször, majd sorban a fölötte levő szintek.

Példa:

SELECT nev ***FROM*** hallgato

WHERE evf=(***SELECT*** evf ***FROM*** hallgato ***WHERE*** nev='KOVÁCS JÓZSEF')

Jelentése: Ezzel az utasítással azoknak a hallgatóknak a nevét kapjuk meg, akik Kovács Józseffel egy évfolyamra járnak. A belső SELECT-ben meghatározzuk , hogy melyik évfolyamra jár Kovács József, a külső SELECT segítségével pedig azoknak a hallgatóknak a nevét választjuk ki, akiknél az évfolyam értéke megegyezik a belső SELECT eredményével.

```
SELECT nev,eredm FROM hallgato

    WHERE eredm=(SELECT MAX(eredm) FROM hallgato)
```

Jelentése: Ezzel az utasítással megkapjuk a hallgato táblázatból a legjobb eredményt elért hallgató nevét és eredményét. A belső SELECT kiválasztja az eredményoszlopból a legnagyobb értéket, a külső SELECT segítségével meghatározzuk azokat a hallgatókat, akiknek az eredménye megegyezik ezzel az értékkel.

Az egyenlőségként megfogalmazott belső SELECT-nek mindig egyetlen értéket kell eredményeznie. Az előző belső SELECT csak egyetlen értéket ad eredményül, hiszen a legjobb eredmény értéke egyetlen érték akkor is, ha több hallgató is elérte azt. **A külső SELECT eredményezhet több sort is**:

```
SELECT tkod FROM gyakjegy

    WHERE jegy=(SELECT max(jegy) FROM gyakjegy)
```

Magyarázat: A belső SELECT eredménye egyetlen érték (5), a külső SELECT azonban azoknak a tantárgyaknak a kódját fogja kiválasztani a gyakjegy táblából, amelyekből volt 5-ös gyakorlati jegy.

Ha a belső SELECT több értéket eredményez, akkor a külső SELECT-ben az IN operátort alkalmazhatjuk:

```
SELECT nev FROM hallgato
  WHERE hkod IN (SELECT hkod FROM vizsga WHERE jegy=1)
```

Jelentése: Ezzel az utasítással azoknak a hallgatóknak a nevét kapjuk eredményül, akiknek elégtelen vizsgajegye van. A belső select a vizsga táblából leválogatja azokat a hallgatókódokat, amelyekhez elégtelen vizsgajegy tartozik. A külső SELECT segítségével a hallgató táblából leválogatjuk a hallgatókódokhoz tartozó neveket.

Láthattuk az előzőekben, hogy ezt az információt a hallgato és a vizsga táblák összekapcsolásával is megkaphatjuk.

- A SELECT utasításban alkalmazható további operátorok

ALL (lista) - relációs operátorral együtt alkalmazható.
Akkor ad igaz értéket eredményül, ha a reláció a lista minden elemére teljesül. A lista helyett belső SELECT is állhat.

ANY (lista) - relációs operátorral együtt alkalmazható.
Akkor ad igaz értéket eredményül, ha a reláció a lista valamelyik elemére teljesül. A lista helyett belső SELECT is állhat.

EXISTS - belső SELECT-tel együtt alkalmazható.
Akkor ad igaz eredményt, ha a belső SELECT legalább egy sort eredményezett.

Az *ALL*, *ANY* és *EXISTS* operátorok mindegyike előtt használható a *NOT* kulcsszó. Ekkor a művelet eredményének negált értékét kapjuk.

- KORRELÁLT ALKÉRDÉSEK

Az alkérdés többször kerül kiértékelésre.

Példa:

Írassuk ki az azonos fizetésű dolgozók igazolványszámát, nevét és fizetését. A példában szereplő BELSŐ SELECT többször lefut, annyiszor ahány sort vizsgál a KÜLSŐ SELECT.

SELECT szigszam,nev,fizetes FROM dolgozo, kulso WHERE
fizetes IN(SELECT fizetes FROM dolgozo WHERE szigszam<>kulso.szigszam);

----------Táblák összekapcsolása----------

- Keresztszorzat

Az egyik tábla minden sorához a másik tábla minden sora hozzákapcsolódik.

SELECT * FROM dolgozo,osztaly;

Az összekapcsolás történhet idegenkulcs alapján is.

SELECT * FROM dolgozo,osztaly WHERE dolgozo.dolgozik=osztaly.oid;

Természetesen a WHERE záradékban további szűrés is lehetséges.

SELECT * FROM dolgozo,osztaly WHERE dolgozo.dolgozik=osztaly.oid AND
fizetes>70000;

- INNER JOIN (Belső összekapcsolás)

Eredménye azonos az előző összekapcsolási módszerrel.

- Keresztszorzat (1-1 összekapcsolás)

SELECT * FROM dolgozo JOIN osztaly ON 1=1;

- Kapcsolómező (idegenkulcs alapján)

SELECT * FROM dolgozo JOIN osztaly ON dolgozo.dolgozik=osztaly.oid;

- OUTER JOIN (Külső összekapcsolás)

Az összekapcsolás eredményében azok a rekordok is részt vesznek, amelyeknek nincs párjuk a másik táblában.

- A baloldali tábla minden oszlopa megjelenik az eredményben.

SELECT * FROM dolgozo LEFT OUTER JOIN osztaly ON dolgozo.dolgozik= osztaly.oid;

- A jobboldali tábla minden oszlopa megjelenik az eredményben.

SELECT * FROM dolgozo RIGHT OUTER JOIN osztaly ON dolgozo.dolgozik= osztaly.oid;

Mindkét tábla minden rekordja megjelenik az eredményben.

SELECT * FROM dolgozo FULL OUTER JOIN osztaly ON dolgozo.dolgozik= osztaly.oid;

- ***Halmazműveletek***:

UNION Két lekérdezés eredményét összegzi, az eredményben a mindkét lekérdezésben szereplő elemek egyszerre szerepelnek.

UNION ALL A közös elemek az eredményben többször szerepelnek.

INTERSECT Két lekérdezés eredményének metszetét képezi.

MINUS Egyik lekérdezés eredményéből kivonja egy másik eredményét.

Példák:

SELECT * FROM dolgozo WHERE nev LIKE 'K%' UNION
SELECT * FROM dolgozo WHERE fizetes>70000;

Eredménye: A dolgozo táblából azok a sorok listázódnak ki amelyekre teljesül a fenti két feltétel valamelyike. Azok a sorok amelyek mind a két lekérdezés eredményhalmazában benne vannak csak egyszer kerülnek kilistázásra.

SELECT * FROM dolgozo WHERE nev LIKE 'K%' UNION ALL
SELECT * FROM dolgozo WHERE fizetes>70000;

Eredménye: Ugyan az mint az előzőé azzal a különbséggel, hogy itt azok az elemek amelyek mindkét lekérdezés eredményhalmazában szerepelnek többször fognak szerepelni a listában.

SELECT * FROM dolgozo WHERE nev LIKE 'K%' INTERSECT
SELECT * FROM dolgozo WHERE fizetes>70000;

Eredménye: A listázás során csak azok a sorok fognak szerepelni amelyekre mind a két feltétel egyszerre teljesül.

SELECT * FROM dolgozo WHERE nev LIKE 'K%' MINUS
SELECT * FROM dolgozo WHERE fizetes>70000;

Eredménye: Az eredménylistában azok a sorok fognak szerepelni amelyekre teljesült az első feltétel, ám ezekből kiesnek azok amelyekre a második feltétel is teljesült.

----------Nézettáblák----------

Az SQL nyelvben lehetőségünk van arra, hogy egy vagy több táblából nézettáblát hozzunk létre, melynek segítségével az adatokat a számunkra könnyebben kezelhető formában láthatjuk. A nézettábla csak logikai tábla, fizikailag nem létezik. Az adatbázisban a nézettábla kialakítására szolgáló lekérdezés kerül tárolásra.

- Nézettábla létrehozása:

CREATE VIEW <név> ***AS*** <lekérdezés>

név	A nézettábla neve.
lekérdezés	azon táblák oszlopainak és sorainak megadása, amelyeken a nézettábla alapul. Általában ez egy SELECT utasítás, amely nem tartalmazhat ORDER BY opciót.

Nézettáblákat bármelyik SQL utasításban használhatunk, ahol táblák használata megengedett, azzal a megszorítással, hogy amennyiben a nézettábla nem egyetlen tábla adatain alapul, vagy a nézettáblát definiáló lekérdezésben a GROUP BY opció szerepel, akkor csak lekérdezhetünk adatokat a nézettáblából. Egyébként módosításokat is végezhetünk a nézettáblában szereplő adatokon.

Példák:

```
CREATE VIEW atlag AS SELECT hkod,nev,eredm FROM hallgato
```

Ezen utasítás hatására létrejön az *atlag* nézettábla, amely a *hallgato* táblából a *hkod*, *nev*, *eredm* oszlopokat tartalmazza.

```
CREATE VIEW statjegy
      AS SELECT nev,evf,csop,jegy FROM hallgato,vizsga
            WHERE hallgato.hkod=vizsga.hkod AND
                tkod=(SELECT tkod FROM tantargy
                      WHERE tnev='STATISZTIKA')
```

A *statjegy* nézettábla a *hallgato* és a *vizsga* táblákon alapul. Az belső SELECT a *vizsga* táblából csak azokat a sorokat választja ki, ahol a tantárgy a statisztika.

- Csak olvasható nézet létrehozása:

CREATE VIEW olvashato ***AS SELECT*** * ***FROM*** dolgozo ***WITH READ ONLY***;

- A nézettábla megszüntetése:

DROP VIEW <név>

név	A törölni kívánt nézettábla neve.

----------Indextáblák----------

Indexállomány: Egy adott táblából kiemelt néhány [rendezett] oszlopból áll.

CREATE [***UNIQUE***] **INDEX** indextábla-név **ON** táblanév (oszlopnév [[***ASC/DESC***][,oszlopnév[***ASC/DESC***]]…);

Az ON után adott tábla felsorolt oszlopait rendezi (növekedően ASC esetén és csökkenően DESC esetén) és belőlük egy INDEX szó után megadott nevű táblát készít. A UNIQUE azt jelenti, hogy az oszlop értékei egyediek, s ha ez esetben ismétlődő értékek is vannak az oszlopban, a rendszer hibát jelez.

----------Tranzakciós eljárások----------

Az SQL tranzakciók logikailag egybetartozó SQL utasítások sorozata. Abban az esetben, ha egy tranzakció lefut, akkor ennek eredményeként ellentmondásmentes adatbázishoz jutunk. Amennyiben egy tranzakció nem fut le, akkor az adatbázis ellentmondásos.
A tranzakció érvényesítése az SQL-ben a **COMMIT** paranccsal történik, ezzel jelezzük, hogy ha sikeres volt, azaz a tranzakció előtti állapot visszaállítása a **ROLLBACK** paranccsal történik.