Ajánlott irodalom:

- **Kende Mária-Kotsis Domokos-Nagy István: Adatbázis kezelés az ORACLE rendszerben**
- **Balogh Judit - Rutkovszky Edéné: SQL példatár**
- *Loney - Koch: Oracle 8i Teljes Referencia*
- *Ullman-Widom: Adatbázisrendszerek. Alapvetés*
- *Stolniczki Gyula: SQL kézikönyv*

Alapfogalmak

Adatbázis: olyan adathalmaz, amely az adatokon kívül a köztük levő kapcsolatokat és a rájuk vonatkozó információkat is tárolja - "többfelhasználós adathalmaz"

Adatbáziskezelő rendszer: szoftver, amely az adatbázis kezelését végzi

Adatbáziskezelő rendszerek legfontosabb feladatai:

- az adatbázis és szerkezetének kialakítása, karbantartása
- adatok karbantartása (beszúrás, módosítás, törlés)
- adatok visszakeresése (lekérdezése)
- adatvédelem, adatbiztonság megoldása
- konzisztencia biztosítása integritási megszorítások használatával
- konkurens hozzáférések kezelése

Adatbázis rendszerek részei:

DBS = DB + DBMS + DBA + felhasználó

- DB (adatbázis)
  felhasználó adatai - fizikai állományokban
  adatszótár (DD)
  - adatbázis objektumok (tábla, nézet, szekvencia, index stb.)
  - felhasználói adatok (felhasználói név, jogosultságok)
- **DBMS (adatbáziskezelő rendszer)**
  RDBMS (relációs adatbáziskezelő rendszer)
  mindennemű tárolás alapja a tábla
  relációs adatbáziskezelő nyelv: DDL, DML, DCL

- DBA (adatbázis adminisztrátor)
  - felhasználók létrehozása, kezelése
  - jogosultságok kiosztása
  - a rendszer működésének figyelése
  - rendszerhibák kezelése
  - adatbázis tervezés figyelemmel kísérése
- felhasználó
  "érte van az egész"

Relációs adatmodell – alapfogalmak

Először az adatbázis sémáját kell definiálni (táblák+szabályok), utána történhet az adatokkal való feltöltés.

tábla	reláció	(itt tárolódnak az adatok)
oszlop	attribútum	(azonos adattípusú adatokat tart.)
sor	rekord	(összetart. oszlopértékeket tart.)
mező	érték	

NULL érték: valamelyik sorban egy oszlopnak nincs értéke. A NULL érték semmilyen más értékkel nem hasonlítható össze

Integritási megszorítások: Szabályok, melyek biztosítják az adatbázis konzisztenciáját (ne tartalmazzon az adatbázis hibás értékeket, logikai ellentmondásokat). Az RDBMS automatikusan biztosítja a szabályok betartatását.

Primary key (elsődleges kulcs): olyan oszlop, amelynek értéke azonosítja a tábla minden sorát. Több oszlop is alkothatja. Nem lehet NULL.
Unique key (egyedi kulcs): olyan oszlop, amelynek értéke a tábla minden sorában egyedi. Több oszlop is alkothatja. Lehet NULL értékű.
Foreign key (külső kulcs): olyan oszlop, amely egy másik (vagy ugyanazon) tábla elsődleges kulcsára hivatkozik. Több táblát logikailag összekapcsolhatunk vele. Több oszlop is alkothatja.

Műveletek táblákkal

NEVEK	
KOD	*NEV*
01	ALMA
02	KORTE

ARAK		
SORSZ	*KOD*	*AR*
1	02	100
2	03	150

03	SZILVA	3	03	50

I. **Projekció - oszlopok kiválasztása**
II. **Szelekció - sorok kiválasztása**
III. **Halmazelméleti műveletek (Egyesítés, Metszet, Különbség):** csak azonos oszlopokat tartalmazó táblákon értelmezhető
IV. **Táblák szorzása (Descartes szorzat):** két táblából indul ki; az eredmény tábla sorai úgy keletkeznek, hogy az első tábla sorait minden lehetséges módon folytatjuk a második tábla minden sorával

NEVEK.KOD	*NEVEK.NEV*	*ARAK.SORSZ*	*ARAK.KOD*	*ARAK.AR*
01	ALMA	1	02	100
01	ALMA	2	03	150
01	ALMA	3	03	50
02	KORTE	1	02	100
02	KORTE	2	03	150
02	KORTE	3	03	50
03	SZILVA	1	02	100
03	SZILVA	2	03	150
03	SZILVA	3	03	50

BELSŐ ÖSSZEKAPCSOLÁS

V. **Táblák összekapcsolása - EQUI-JOIN:** speciális szorzás; két táblából indul ki; mindkét táblában ki kell jelölni egy kapcsoló oszlopot és meg kell adni a kapcsolási feltételt, ami egyenlőség
Kapcsoló oszlop: **KOD**
Kapcsoló feltétel: **NEVEK.KOD = ARAK.KOD**

NEVEK.KOD	*NEVEK.NEV*	*ARAK.SORSZ*	*ARAK.KOD*	*ARAK.AR*
02	KORTE	1	02	100
03	SZILVA	2	03	150
03	SZILVA	3	03	50

VI. **Táblák összekapcsolása - NEM EQUI-JOIN:** speciális szorzás; két táblából indul ki; mindkét táblában ki kell jelölni egy kapcsoló oszlopot és meg kell adni a kapcsolási feltételt, ami nem egyenlőség
Kapcsoló oszlop: **KOD**
Kapcsoló feltétel: **NEVEK.KOD > ARAK.KOD**

KÜLSŐ ÖSSZEKAPCSOLÁS

VII. speciális szorzás; két táblából indul ki; mindkét táblában ki kell jelölni egy kapcsoló oszlopot és meg kell adni a kapcsolási feltételt; **DE** olyan, az első táblához tartozó sorokat is az eredménytáblába tesz, amelyhez nem létezik a második táblában kapcsolódó sor
Típusok: baloldali, jobboldali, kétoldali
Lehet: EQUI-JOIN és NEM-EQUI-JOIN

Példa: baloldali, equi-join
Kapcsoló oszlop: **KOD**
Kapcsoló feltétel: **NEVEK.KOD = ARAK.KOD**

NEVEK.KOD	*NEVEK.NEV*	*ARAK.SORSZ*	*ARAK.KOD*	*ARAK.AR*
01	ALMA	NULL	NULL	NULL
02	KORTE	1	02	100
03	SZILVA	2	03	150
03	SZILVA	3	03	50

SQL - Structured Query Language

Lekérdező nyelv relációkban tárolt információk visszanyerésére, minimális adatbeviteli és módosítási lehetőségekkel

Jellemzői:

- Nem algoritmikus: Parancsnyelv jellegű, megfogalmazhatjuk, mit akarunk csinálni, de a megoldási algoritmust nem kell megadni a felhasználónak. Nincsenek benne ciklusok, feltételes elágazások, változó deklarálás stb.
- Mintaillesztéses, halmazorientált: A táblákat mint a sorok (rekordok) halmazát tekintjük. Az adott utasításban megfogalmazott feltételnek eleget tevő összes sor részt vesz a műveletben
- Szabványos: Illeszkedik az SQL szabványhoz. A szabványban van egy SQL utasításcsoport, amelyet minden SQL alapú szoftver implementációnak meg kell valósítani, de mindegyik implementáció plusz lehetőséget is nyújt a standard SQL-hez képest, felülről kompatibilis a szabvánnyal

SQL fontosabb használati módjai:

- Önállóan fejlesztő eszközökben: pl.: SQL*Plus, iSQL*Plus, Oracle Form, Oracle Report stb.
- Beágyazva procedurális programozási nyelvekbe. pl.: C/C++, ADA, COBOL, stb. befogadó nyelvekbe – először előfordítóval fordítani kell
- PL/SQL (az ORACLE saját nyelve, az SQL procedurális kiterjesztése) és JAVA nyelvekben – a programegységek tárolhatók, futtathatók az adatbázis szerveren

Az SQL nyelv utasításainak főbb csoportjai:

DDL	adatdefiníciós nyelv (**D**ata **D**efinition **L**anguage) adatbázis és adatbázis objektumok létrehozása, kezelése: CREATE, ALTER, DROP, RENAME
DML	adatmanipulációs nyelv (**D**ata **M**anipulation **L**anguage) adatok karbantartása (bevitele, módosítása, törlése), lekérdezése: INSERT, UPDATE, DELETE, SELECT
DCL	adatvezérlő nyelv (**D**ata **C**ontrol **L**anguage) tranzakció kezelése: COMMIT, ROLLBACK, SAVEPOINT adatvédelem, felhasználói hozzáférés szabályozása: GRANT, REVOKE

SELECT lekérdező utasítás

SELECT utasítással végrehajtható feladatok:
A SELECT paranccsal egy vagy több tábla vagy nézettábla tartalmát tudjuk lekérdezni, vagy használhatók beépítve más SQL utasításokban. Megvalósíthatók vele a korábban felsorolt táblaműveletek

SELECT ...	oszlopok kiválasztása (projekció)
FROM ...	táblanév(-ek) (összekapcsolás)
[WHERE ...]	sorok kiválasztása (szelekció)
[CONNECT BY ... [START WITH ...]]	hierarchia kezelés
[GROUP BY ...]	csoportosítás
[HAVING ...]	csoportok közötti válogatás
[{UNION [ALL] | INTERSECT | MINUS} alselect]	halmazműveletek
[ORDER BY ...]	eredménysorok rendezése

I. Egyszerű lekérdezések, rendezés

SELECT [**ALL** | **DISTINCT**] {[táblanév.]* | o_kifejezés [o_alias]},...
FROM táblanév [t_alias],...
{ORDER BY {o_kifejezés | o_alias} [**ASC** | **DESC**], ...};

ALL	alapértelmezés, az összes sort visszaadja
DISTINCT	az egymástól különböző sorokat adja vissza
o_alias	az eredményben oszlop neve helyett jelenik meg fejlécként
t_alias	a táblanév rövidítésére használható *(Az o_alias és t_alias csak az adott utasításban érvényes)*
*	a tábla összes oszlopát jelenti
o_kifejezés	oszlopnév(ek), konstansok, függvényhívások összekapcsolva aritmetikai (*, /, +, -) vagy konkatenáló operátorral (||), zárójelezés is megadható

konstans: adattípusa: karakteres, numerikus, dátum
Karakteres: megkülönbözteti a kis- és nagybetűt
Dátum: függ az alapértelmezett dátum formátumtól és nyelvtől
Pl. Oracle 9i: '88-FEB-21', '2006-ÁPR-01'
Pl. Oracle 10g: '88-FEBR.-21', '2006-ÁPR.-01'

```
SELECT * FROM nls_session_parameters;
ALTER SESSION
    SET NLS_DATE_LANGUAGE='HUNGARIAN'
        NLS_DATE_FORMAT='formátum'
```

függvények: később
oszlopnév: [táblanév.]oszlopnév
(minősíteni kell, ha egy oszlopnév több táblában is előfordul

NULL érték:

- Olyan érték, amely nem ismert, nincs megadva vagy nem értelmezhető. Nem azonos a nullával (szám), a szóközzel (karakter). Bármely típusú oszlop tartalmazhatja, ha az oszlopot nem NOT NULL vagy PRIMARY KEY megszorítással hoztuk létre
- Nem lehet vele számolni (definiálatlan, azaz NULL lesz a kifejezés eredménye vagy a függvény értéke) és a legtöbb csoportfüggvény figyelmen kívül hagyja

- Kezelésére az **NVL** függvényt használhatjuk:

  NVL(kifejezés, helyettesítő_érték)

  A függvény eredménye azonos az első argumentum által meghatározott értékkel, ha az nem NULL, különben a második argumentum értékét adja vissza

Rendezés: (ORDER BY)

- Alapértelmezésben növekvő, DESC hatására csökkenő
- Értelmezve van minden adattípusnál
- A NULL érték növekvő rendezésnél utolsóként, csökkenőnél elsőként jelenik meg

II. Sorok kiválasztása, lekérdezés keresési feltétellel

SELECT...
FROM...
WHERE feltétel
…;

Csak azok a sorok vesznek részt a további műveletekben, amelyekre a **feltétel** igaz, amelyekre hamis vagy ismertelen (Unknown), azok nem.

Feltétel: oszlopnevekből, kifejezésekből, konstansokból és összehasonlító operátorból áll.

a) Egyszerű feltételek:

o_kifejezés relációs_operátor o_kifejezés

kifejezések értékének összehasonlítása
relációs_operátor: =, !=, <>, <, >, <=, >=

o_kifejezés [NOT] BETWEEN kif1 AND kif2

kif1 és kif2 közé esés
zárt intervallum: kif1 <= o_kifejezés <= kif2

o_kifejezés [NOT] LIKE 'karakterminta'

illeszkedés a megadott karaktermintára. Helyettesítő karakterek:

% tetszőleges hosszú karaktersorra illeszkedés az adott pozíciótól
_ tetszőleges karakterre illeszkedés az adott pozícióban

o_kifejezés IS [NOT] NULL

NULL értékkel való egyezés
(o_kifejezés=NULL eredménye definiálatlan)

b) Halmazos feltételek:

Halmaz: (kifejezés lista) vagy (alselect)
(az alselect több oszlopot és több sort is visszaadhat)

o_kifejezés [NOT] IN (halmaz)

a megadott halmazban szerepel-e a kifejezés?

o_kifejezés relációs_operátor {ANY|SOME} (halmaz)

teljesül-e a reláció a halmaz valamely (legalább egy) elemére?
(=ANY azonos az IN relációval, ANY és SOME azonos)

o_kifejezés relációs_operátor ALL (halmaz)

teljesül-e a reláció a halmaz minden egyes (összes) elemére?
(<>ALL azonos a NOT IN relációval)

[NOT] EXISTS (alselect)

az alselect visszaad-e legalább egy sort?

c) Összetett keresési feltételek:

A feltételeket összekapcsolhatjuk logikai operátorokkal: **NOT, AND, OR**

Igazságtáblázatok:

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

NOT	
True	False
False	True
Unkno wn	Unkno wn

d) **Műveletek kiértékelési sorrendje:**

Azonos precedenciájú műveletek esetén a balról-jobbra szabály érvényes.

1. Aritmetikai operátorok (*, /, +, -)
2. Karakteres operátor (||)
3. Összehasonlító operátorok
   (=, !=, <>, <, >, <=, >=,
   [NOT] IN, ANY, ALL, [NOT] BETWEEN, [NOT] EXISTS
   [NOT] LIKE, IS [NOT] NULL
   (precedenciájuk azonos)
4. Logikai operátorok (NOT, AND, OR)

III. Származtatott adatok, SQL sor-függvények

A származtatott adatokat tábla adataiból származtathatjuk oszlopkifejezések segítségével.

Oszlopkifejezés: oszlopnevek, konstansok és függvényhívások összekapcsolva aritmetikai (*, /, +, -) vagy konkatenáló karakteres operátorral (||), zárójelezés is megengedett.

Függvények típusai:

- **Sor-függvények**: egyszerre egy soron végeznek műveletet, soronként egy eredményt adnak vissza
- **Csoport-függvények:** sorok egy-egy csoportján végeznek műveletet és minden csoportra egy értéket adnak vissza

Sor függvények jellemzői

- Egy vagy több argumentumot fogadnak, és a lekérdezés által visszaadott minden egyes sorhoz egy értéket adnak vissza.
- Argumentum lehet: felhasználó által megadott konstans, oszlopnév, kifejezés
- A hivatkozott adat típusától eltérő típusú adatértéket is visszaadhatnak
- A SELECT, a WHERE és az ORDER BY utasításrészben is használhatók, és egymásba ágyazhatók

függvény_név [(arg1, arg2,...)]

DUAL tábla: a SYS felhasználó tulajdona, az összes felhasználó hozzáférhet. Egy DUMMY nevű oszlopot és egy sort tartalmaz, amelyben az X szerepel, mint érték.

Függvény típusok: Karakteres, numerikus, dátum, konverziós, egyéb

a) **Karakteres függvények:** LOWER, UPPER, INITCAP, LENGTH, SUBSTR, LPAD, LTRIM stb.

b) **Numerikus függvények:** ABS, POWER, SQRT, ROUND, TRUNC stb.

c) Konverziós függvények: TO_CHAR, TO_NUMBER, TO_DATE stb.

d) **Dátum függvények:** dátumokon végeznek műveletet. A MONTHS_BETWEEN numerikus, a többi DATE adattípusú értéket ad vissza

- Az Oracle rendszer a dátumokat belső numerikus formátumban tárolja:
  év (4 számjegy), nap, óra, perc, másodperc
- Az alapértelmezett megjelenítési és beviteli dátumformátum és a nyelv lekérdezhető és beállítható:

```
SELECT * FROM nls_session_parameters;
ALTER SESSION
    SET NLS_DATE_LANGUAGE='HUNGARIAN'
        NLS_DATE_FORMAT='formátum'
```

- Adatfelvitelnél, keresésnél az alapértelmezett formában megadott dátumoknál az évszázadot az aktuális évszázadnak, az időpontot pedig éjfélnek tekinti

RR dátumformátum-elem: az YY elemhez hasonló, de lehetővé teszi eltérő évszázad megadását is. Az így visszaadott dátumban az évszázad a megadott két számjegyű évtől és az aktuális év utolsó két számjegyétől függően változik.

	A 2 számjeggyel megadott év:	
Az aktuális év utolsó 2 számjegye:	**00-49**	**50-99**
00-49	Aktuális évszázad	Előző évszázad
50-99	Következő évszázad	Aktuális évszázad

Használjuk az RR formátumot, ha literált dátummá alakítunk és az év két karakteres!

Aktuális év	**Megadott dátum**	**RR formátum**	**YY formátum**
2002	17-OKT-27	2017	2017
2002	95-OKT-27	1995	2095
1995	95-OKT-27	1995	1995
1995	17-OKT-27	2017	1917

Dátumokkal végezhető aritmetikai műveletek:

	Eredmény:	
Dátum+szám	dátum	szám nappal későbbi dátum
Dátum-szám	dátum	szám nappal korábbi dátum
Dátum-dátum	napok száma	
Dátum+szám/24	dátum	órával későbbi dátum

Dátum formátumban használható elemek:
Century, **Y**ear, **M**onth, **D**ay, **H**our, **MI**nute, **S**econd, **W**eek, **Q**uarter, stb.

CC, YYYY, RRRR, MM, MONTH, MON, mon, DDD, DD, D, Dy, HH24, SS, WW, W, Q
fm -- vezető nullák letiltása

e) **Egyéb függvények:** USER, NVL, GREATEST, LEAST, DECODE stb.

IV. Csoportok képzése, csoport-függvények

Csoportfüggvények:

- Sorok csoportjain végeznek műveletet és egy, a csoportra jellemző értéket állítanak elő
- Csoport: a teljes tábla, vagy a tábla sorainak egy részhalmaza
- Egyszeres mélységben ágyazhatók egymásba
- SELECT listában, ORDER BY és HAVING utasításrészben szerepelhetnek

FV_NÉV([DISTINCT|ALL] kif)
FV_NEV: **AVG, SUM, MIN, MAX, STDDEV, VARIANCE**
NULL értéket figyelmen kívül hagyják
AVG, SUM, STDDEV,VARIANCE: csak numerikus értékkel
MIN, MAX: numerikus, karakteres, dátum értékkel működik
DISTINCT: a függvény csak a különböző értékeket veszi figyelembe
COUNT({*|[DISTINCT|ALL] kif})

COUNT(*):	az összes kiválasztott sor száma, a többször szereplő és a NULL értéket tartalmazó sorokat is beleszámolja
COUNT(kif):	azon sorok száma, ahol kif értéke nem NULL
COUNT(DISTINCT kif):	a kif által meghatározott oszlopban a különböző, nem NULL értéket tartalmazó sorok számát adja vissza

SELECT ...
FROM ...
WHERE ...
GROUP BY o_kifejezés,...
[HAVING csoportkiválasztási_feltétel]
ORDER BY …;

GROUP BY: az oszlopkifejezés alapján csoportosítja a leválogatott sorokat és egyetlen, összesített információt tartalmazó sort állít elő minden csoporthoz. Az eredmény növekvő sorrendbe rendezve jelenik meg

HAVING: megadható a **csoportok** közötti válogatás feltétele

a) Nincs GROUP BY -- egyetlen csoport készül:

b) Van GROUP BY: egy csoportot képeznek azon sorok, amelyekben a GROUP BY után álló csoportképző oszlopok (oszlopkifejezések) azonos értékűek. Minden csoportból csak egy sor lesz visszaadva. Azon sorok is részt vesznek a csoportosításban, ahol a csoportképző oszlop értéke NULL

Ha a SELECT utasításban csoportfüggvényt használunk, akkor egyedi (egy sorra vonatkozó) eredményt csak akkor választhatunk ki a SELECT listában, ha a GROUP BY utasításrészben szerepel az adott oszlop, egyébként hibaüzenetet kapunk

GROUP BY után: táblabeli oszlopokból álló kifejezés (másodlagos név nem)
HAVING után: feltétel (csoportfüggvény) (másodlagos név nem)
SELECT után:
- konstans
- paraméter nélküli függvény (SYSDATE, USER)
- csoportfüggvény
- GROUP BY után adott kifejezéssel azonos kifejezés,

vagyis olyan, amire a csoportosítás vonatkozik

V. Táblák összekapcsolása

- A táblák közötti kapcsolat megvalósítása kapcsolóoszloppal történik
- A kapcsolóoszlop meghatározza, hogy az egyik tábla egy adott sorához a másik tábla mely sorai tartoznak
- Összekapcsolásra általában az **elsődleges kulcs - külső kulcs** oszlopokat használjuk

Egyedi kulcs (Unique key):
- Az (egy vagy több) oszlopban szereplő értékek különbözőek a tábla különböző soraiban (lehet NULL érték is)

Elsődleges kulcs (Primary Key):
- Olyan (egy vagy több) oszlop, amelyben szereplő érték egyértelműen azonosítja a tábla minden sorát

Külső kulcs (Foreign Key):
- Olyan (egy vagy több) oszlop, amely hivatkozik egy másik (vagy ugyanazon) tábla elsődleges kulcsára
- A hivatkozott tábla ugyanaz a tábla is lehet
- A külső kulcs értéke NULL érték is lehet

Az Oracle9*i* egy SQL 1999 szabványaival kompatibilis összekapcsolási szintaxist ajánl. A 9*i* verziót megelőző formák nem feleltek meg az ANSI szabványainak

```
SELECT tábla1.oszlop, tábla2.oszlop,…
FROM tábla1
      [CROSS JOIN tábla2] |
      [NATURAL JOIN tábla2] |
      [JOIN tábla2 USING (oszlop)] |
      [JOIN tábla2 ON (tábla1.oszlop= tábla2.oszlop)] |
      [{LEFT|RIGHT|FULL} OUTER JOIN tábla2
            ON (tábla1.oszlop=tábla2.oszlop)];
```

Az összekapcsolható táblák száma nincs korlátozva, valódi vagy nézettáblák is lehetnek.

BELSŐ ÖSSZEKAPCSOLÁS: két táblában található összetartozó sorok visszaadása

a) Kereszt (cross) összekapcsolás

Egyenértékű a Descartes szorzat létrehozásával, amelyben a sorok összes lehetséges kombinációja megjelenik. Az első tábla összes sora össze lesz kapcsolva a második tábla összes sorával

```
SELECT *
  FROM telephely
  CROSS JOIN alkalmazott;
```

b) Természetes összekapcsolás (Natural joins)

Automatikus összekapcsolás minden olyan, a két táblában szereplő oszlop alapján, melyek neve és adattípusa megegyezik. Egyenértékű az egyen-összekapcsolással. A megadott oszlopok nem rendelkezhetnek minősítővel (tábla név és másodlagos név) sehol az SQL utasításon belül

```
SELECT *
     FROM alkalmazott
     NATURAL JOIN telephely;
```

Azon alkalmazottak, akik telephelye DEBRECEN-ben van:

```
SELECT *
     FROM alkalmazott
     NATURAL JOIN telephely
     WHERE varos='DEBRECEN'
```

c) Összekapcsolás USING utasításrésszel

Egyen összekapcsolás a USING utasításrészben kiválasztott oszlop alapján. A megadott oszlopok nem rendelkezhetnek minősítővel (tábla név és másodlagos név) sehol az SQL utasításon belül. A NATURAL JOIN és USING utasításrészek kölcsönösen kizárják egymást.

```
SELECT akod, anev, tnev
   FROM alkalmazott JOIN telephely
   USING (tkod)
   WHERE tnev LIKE '%AUTO%';
```

d) Összekapcsolás ON utasításrésszel

Egyen összekapcsolás, melyben tetszőleges kapcsolási feltételt megadhatunk az ON utasításrészben és a kapcsolási feltételt elkülöníthetjük az egyéb WHERE feltételektől

Eladók neve és telephelyük adatai:

```
SELECT X.anev, X.beosztas, X.tkod, Y.tnev, Y.varos
        FROM alkalmazott X JOIN telephely Y
        ON (X.tkod=Y.tkod)
        WHERE X.beosztas='ELADO';
```

Háromirányú összekapcsolás:

Az alkalmazottak és főnökük, valamint a főnökük telephelyének adatai:

```
  SELECT X.tkod, X.anev, Y.anev fonoknev, Y.tkod, tnev, varos
        FROM alkalmazott X
        JOIN alkalmazott Y
        ON X.fonok=Y.akod
        JOIN telephely T
        ON Y.tkod=T.tkod;
SELECT X.tkod, X.anev , Y.anev fonoknev, Y.tkod, tnev, varos
        FROM alkalmazott X, alkalmazott Y, telephely T
        WHERE X.fonok=Y.akod AND Y.tkod=T.tkod;
```


Nem egyenlőségen alapuló összekapcsolás:
A kapcsoló feltételben nem = jel van

KÜLSŐ ÖSSZEKAPCSOLÁS: ha két tábla esetében megkapjuk a belső összekapcsolás eredményeit, és ezen felül még a bal vagy jobboldali táblákban találunk nem társított sorokat is

Baloldali külső összekapcsolás (LEFT OUTER JOIN)

Kérjük le az összes sort a TELEPHELY táblából - amely a baloldali tábla -, azok a sorok is jelenjenek meg, amelyeknek nincs párja az ALKALMAZOTT táblában:

```
SELECT *
     FROM telephely t
     LEFT OUTER JOIN alkalmazott a
     ON (t.tkod=a.tkod);
SELECT *
     FROM telephely t, alkalmazott a
     WHERE t.tkod=a.tkod(+);
```

Jobboldali külső összekapcsolás (RIGHT OUTER JOIN)

Kétoldali külső összekapcsolás (FULL OUTER JOIN)
Olyan, táblák közti összekapcsolás, amely a belső, baloldali külső és jobboldali külső összekapcsolások eredményeit is visszaadja

VI. Egymásba ágyazott SELECT

Az SQL nyelvben megengedett, hogy egy SELECT (vagy más SQL) utasításban **allekérdezés (alselect)** előforduljon
mélység maximálva van
mindig zárójelbe kell tenni

Allekérdezés szerepelhet:

- WHERE utasításrészben, HAVING utasításrészben
- FROM utasításrészben -- INLINE nézet
- Skalár értéket adó allekérdezés minden olyan helyen szerepelhet, ahol oszlopkifejezés szerepel a szintaxisban; művelet is végezhető vele
- SQL utasításokban: INSERT, DELETE, UPDATE, stb.

1. **Egyszerű típus:** a belső SELECT önmagában kiértékelhető, a lekérdezések belülről kifelé haladva lesznek feldolgozva. A kiértékelés menete:
   - a belső SELECT kiértékelődik és egy vagy több sort vagy oszlopértéket átad a külső SELECT-nek
   - a külső SELECT ezen értékek alapján összeállítja az eredményt

Allekérdezéssel kapcsolatos problémák:
- egyszerű feltételben csak skalár értéket adó allekérdezés lehet, amely pontosan egy sort és egy oszlopot ad vissza
- belső lekérdezés egyetlen sort sem ad vissza
- az egysoros allekérdezés több sort ad vissza eredményként (IN operátor kell)

a) Egy értéket (egy sor, egy oszlop) ad át a külsőnek
Pl. a legrégebben belépett alkalmazottak:
```
SELECT * FROM ALKALMAZOTT
WHERE BELEPES=
(SELECT MIN(BELEPES) FROM ALKALMAZOTT)
```

b) Több sort ad át a külsőnek
Pl. azon alkalmazottak, akiknek a fizetése éppen annyi, mint az egyes telephelyeken kifizetett legkisebb bér:
```
SELECT * FROM ALKALMAZOTT
WHERE FIZETES IN
    (SELECT MIN(FIZETES)
    FROM ALKALMAZOTT
    GROUP BY TKOD)
```

Halmazos (többsoros) operátorok:
Ha **több sorból** egy vagy több értéket adunk át a külső lekérdezésnek, akkor a **halmazos operátorokat** kell használni:

[NOT] IN	a lista bármely elemével egyenlő
rel_op ANY	a hasonlítási feltételnek teljesülnie kell az allekérdezés által visszaadott **legalább egy** értékre
<ANY	kisebb mint a maximum
>ANY	nagyobb mint a minimum
=ANY	megegyezik az IN operátorral

rel_op ALL	a hasonlítási feltételnek teljesülnie kell az allekérdezés által visszaadott **összes** értékre
>ALL	nagyobb mint a maximum
<ALL	kisebb mint a minimum
EXISTS	feltétel teljesül, ha az allekérdezés legalább egy sort visszaad

c) Több sorból több értéket ad át a külsőnek
Azon alkalmazottak adatai, akiknek fizetése és prémiuma megegyezik az ELADO beosztásúak fizetésével és prémiumával
```
SELECT anev, beosztas, fizetes, NVL(premium,0)
FROM alkalmazott
WHERE (fizetes, NVL(premium, 0)) IN
(SELECT DISTINCT fizetes, NVL(premium, 0)
            FROM alkalmazott
           WHERE beosztas='ELADO');
```

2. **Korrelált (kapcsolt) típus:** a belső SELECT olyan, a külső SELECT-re történő hivatkozást tartalmaz, amely miatt a belső önmagában nem kiértékelhető. A kiértékelés menete:
   - a külső SELECT átad egy sort a belsőnek
   - a belső SELECT így már kiértékelhető, visszaadja az eredménysort vagy sorokat
   - a külső SELECT elvégzi a további értékelést.

Ez ismétlődik a külső SELECT minden sorára, míg összeáll az eredmény.

Amennyiben a külső és a belső SELECT is ugyanazt a táblát használja, a belső SELECT csak aliasnévvel tud a külső SELECT táblájára hivatkozni.

pl. a legnagyobb fizetésű alkalmazottak beosztásonként
```
select *
from alkalmazott x
where fizetes=
    (select max(fizetes)
    from alkalmazott
where beosztas=x.beosztas)
```

VII.Halmazműveletek select-ek között:

Két kompatibilis lekérdezés (eredménytáblájuk oszlopainak száma egyezik és típus szerint is rendre kompatibilisek) halmazműveletekkel kapcsolható össze:

UNION, UNION ALL, INTERSECT, MINUS

UNION: az összes sort eredményül adja több tábla esetén, de a többször előforduló sorok csak egyszer szerepelnek az eredményhalmazban
UNION ALL: az összes sort eredményül adja több lekérdezés esetén. A UNION operátorral ellentétben az ismétlődő sorokat nem küszöböli ki, és az eredmény rendezése nem az alapértelmezett szerint történik
INTERSECT: a közös sorokat adja eredményül több tábla esetén
MINUS használatával az első lekérdezés azon sorait kapjuk eredményül, amelyek a második lekérdezésben nem szerepelnek. (Az első SELECT utasítás MINUS a második SELECT utasítás)