Ajánlott irodalom:

- **Kende Mária-Kotsis Domokos-Nagy István: Adatbázis kezelés az ORACLE rendszerben**
- **Balogh Judit - Rutkovszky Edéné: SQL példatár http://lazarus.elte.hu/~hzsolt/oracle/ora.html**
- *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: szoftver, amely az adatbázis kezelését végzi
Típusai:

- Saját nyelvű rendszer
- Befogadó nyelvű rendszer (SQL)

Adatbáziskezelők legfontosabb feladatai:

- adatbázis szerkezetének kialakítása, karbantartása
- adatok bevitele, karbantartása
- adatok visszakeresése - lekérdezések
- adatvédelem, adatbiztonság megoldása
- konzisztencia megőrzése
- konkurens hozzáférés 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
  - a rendszer működésének figyelése
  - rendszerhibák kezelése
  - adatbázis tervezés figyelemmel kísérése
- felhasználó
  "érte van az egész"

Relációs adatmodell - alapfogalmak

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 értékkel nem hasonlítható össze (összeadás!)
Primary key (elsődleges kulcs): olyan oszlop(ok), amely egyedileg azonosítja a tábla minden sorát
Foreign key (külső kulcs): olyan oszlop(ok), amely egy másik tábla elsődleges kulcsára hivatkozik. Több táblát logikailag összekapcsolhatunk vele
Nézettábla: lehetővé teszi, hogy egy vagy több tábla adatait az igényeknek megfelelő formában tegyük láthatóvá. Adatokat nem tartalmaz
Index: Adatelérés gyorsítására, bizonyos adatok egyediségének biztosítására használják
Szekvencia: Egyedi egészek generálására használható, ezeket elsődleges kulcsként használjuk

Műveletek táblákkal

NEVEK

KOD	*NEV*
01	ALMA
02	KORTE
03	SZILVA

ARAK

SORSZ	*KOD*	*AR*
1	02	100
2	03	150
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:** két táblából indul ki; az eredmény tábla sorai úgy keletkeznek, hogy az első tábla sorait minden lehetséges módon folytatjuk a második tábla minden sorával

KOD	*NEV*	*SORSZ*	*KOD*	*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

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

KOD	*NEV*	*SORSZ*	*KOD*	*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

VII. **Táblák összekapcsolása - KÜLSŐ JOIN:** 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
Kapcsoló oszlop: KOD
Kapcsoló feltétel: NEVEK.KOD = ARAK.KOD

KOD	*NEV*	*SORSZ*	*KOD*	*AR*
01	ALMA			
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. Nincsenek benne ciklusok, feltételes elágazások, 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, Oracle Form, Oracle Report stb.
- Beágyazva procedurális programozási nyelvekbe. pl.: C/C++, ADA, COBOL, stb. befogadó nyelvekbe
- PL/SQL (az ORACLE saját nyelve, az SQL procedurális kiterjesztése) és JAVA nyelvekben

Az SQL nyelv utasításainak főbb csoportjai:

DDL	adatdefiníciós nyelv (**D**ata **D**efinition **L**anguage) adatbázis és adatbázis objektumok létrehozása: 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 információt keres vissza az adatbázisból, megvalósíthatók vele a korábban felsorolt táblaműveletek

SELECT ...	oszlopok kiválasztása (projekció)	**(6)**
FROM ...	táblanév(-ek) (Descartes szorzat)	**(1)**
[WHERE ...]	sorok kiválasztása (szelekció)	**(2)**
[CONNECT BY ... [START WITH ...]]	hierarchia kezelés	**(3)**
[GROUP BY ...]	csoportosítás	**(4)**
[HAVING ...]	csoportok közötti válogatás	**(5)**
[{UNION [ALL] | INTERSECT | MINUS} alselect]	halmazműveletek	**(7)**
[ORDER BY ...]	eredménysorok rendezése	**(8)**
[FOR UPDATE OF …]	kiválasztott sorok zárolása update idejére	**(9)**

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
Magyar: 'YY-MON-DD'

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

a) Egyszerű feltételek:

o_kifejezés relációs_operátor o_kifejezés
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 mindig NULL)

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?

b) Összetett keresési feltételek:

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

Igazságtáblázatok:

AND	True	False	Null
True	True	False	Null
False	False	False	False
Null	Null	False	Null

OR	True	False	Null
True	True	True	True
False	True	False	Null
Null	True	Null	Null

NOT	
True	False
False	True
Null	Null

c) 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

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 csoportján végeznek művelet 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.

Típusok: Karakteres, numerikus, dátum, konverziós, egyéb

a) **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:
  évszázad, év, nap, óra, perc, másodperc
- Az érvényes dátumok i. e. 4712. január 1-e és i. sz. 9999. december 31-e közöttiek
- Az alapértelmezett megjelenítési és beviteli dátumformátum és a nyelv beállítható: (SYS felhasználó)
  YY-MON-DD
- 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, MM, MONTH, MON, mon, DDD, DD, D, Dy, HH24, SS, WW, W, Q
fm -- vezető nullák letiltása

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) 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

Külső kulcs (Foreign Key):
- az egyik táblában lévő oszlop, amely hivatkozik egy másik tábla elsődleges (Primary Key) kulcsára (vagy egy egyedi Unique kulcsára)
- Több oszlop is alkothatja
- A hivatkozott tábla ugyanaz a tábla is lehet
- A külső kulcs értéke csak létező kulcs értékre hivatkozhat, vagy NULL értékű 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** előforduljon

mélység maximálva van

mindig zárójelbe kell tenni
hasonlító operátor jobb oldalán legyen

Allekérdezés szerepelhet:

- WHERE utasításrészben
- HAVING utasításrészben
- FROM utasításrészben -- INLINE nézet

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:

- az egysoros allekérdezés több sort ad vissza eredményként (IN operátor kell)
- belső lekérdezés egyetlen sort sem ad vissza

a) Egy értéket (egy sor, egy oszlop) ad át a külsőnek

b) Több sort ad át a külsőnek

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

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.

VII. Halmazműveletek selectek között:

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

UNION, UNION ALL, INTERSECT, MINUS

Kiértékelés: balról jobbra, felülről lefelé, ha az INTERSECT is szerepel használjunk zárójelet

UNION: 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)

VIII. ADATDEFINÍCIÓS NYELV (DDL)

Objektum elnevezési szabályok:

- Betűvel kezdődik, legfeljebb 30 karakterből áll
  - Kivétel: adatbázisnév 8 karakter
    adatbázis kapcsolat 128, tartalmazhat @ és . karaktereket
- Csak az A–Z, a–z, 0–9, _ (aláhúzás), $ és # karaktereket tartalmazhatják (az utóbbiak használata nem javasolt)
- Nem különböztetjük meg a kis- és nagybetűket
- Nem egyezhetnek meg az Oracle Szerver fenntartott szavaival
- Egy felhasználónak nem lehet két azonos nevű objektuma, ha azok azonos névterületen vannak

1. Táblák az Oracle adatbázisban

- **Adatszótár:**
  - Táblák és nézetek gyűjteménye, az adatbázis létrehozásakor jön létre, az Oracle rendszer kezeli, az adatbázisról tartalmaznak információt
  - A táblák tulajdonosa a SYS felhasználó, a felhasználók az adatszótár nézeteit használják
  - Tartalma: felhasználók nevei és jogosultságai, az adatbázis-objektumok nevei, a táblákra vonatkozó megszorítások, stb.

USER_ azon objektumok adatai, amelyeknek a felhasználó a tulajdonosa
USER_TABLES
ALL_ azon objektumok adatai, amelyekhez a felhasználónak hozzáférési jogosultsága van, de azok más tulajdonában is lehetnek
ALL_TABLES
DBA_ az adatbázisban lévő összes objektum adata
DBA_TABLES

- **Felhasználói táblák**: felhasználók hozzák létre

a) Tábla létrehozása:

CREATE TABLE tábla
(oszlop adattípus [DEFAULT kif][oszlopmegszorítás]], ...
[táblamegszorítás],…)
[fizikai és tábla tulajdonságok];

Adattípusok:

CHAR[(m)]	Rögzített hosszúságú karakteres adat (1<=**m**<=2000)
VARCHAR2(m)	Változó hosszúságú karakteres adat (1<=**m**<=4000) A maximális *méret* megadása kötelező CHAR: 'A '='A' igaz VARCHAR2: 'A '='A' hamis 'A '>'A' igaz
NUMBER[(p [,s])]	**p** pontosságú, **s** tizedes jegyet tartalmazó szám p: összes számjegy 1<=p<=38 s: a tizedesjeltől jobbra levő számjegyek száma -84<=s<=127
DATE	I. e. 4712.01.01 és i. sz. 9999. 12.31. közé eső dátum- és időérték

DEFAULT: alapértelmezett érték az oszlop számára, amelyet akkor kap meg, ha új sor beszúrásakor nem adunk értéket neki. Lehet literál, kifejezés vagy SQL függvény (SYSDATE, USER).

Integritási megszorítások, kényszerek (CONSTRAINT):

Konzisztencia: Annak megkövetelése, hogy

- Összetartozó adatok módosítása együtt, megfelelő sorrendben történjen (pl. ha a telephely megszűnik, előbb az alkalmazottak törlődjenek, aztán a telephely)
- Ha vannak redundáns adatok, azoknak nem szabad egymással ellentmondásban lenni

Megszorítások:

- Szabályok, melyek biztosítják az adatbázis konzisztenciáját (ne kerüljenek hibás értékek a táblákba, használható legyen az adatbázis, logikai ellentmondásokat ne tartalmazzon)
- Betartásukat az RDBMS automatikusan biztosítja. Nem enged meg olyan adatbázis műveletek, amelyek révén az adatok a szabályokat megszegnék

Megjegyzések:

- Adjunk nevet a megszorításoknak, különben a rendszer generál egyet számukra SYS_C*n* formában Ez a név hibaüzenetekben jelenik meg, vagy az ALTER TABLE utasításban használhatjuk
- A rendszer az adatszótárban tárolja a megszorításokat:
  USER_CONSTRAINTS
  USER_CONS_COLUMNS
- Megszorításokat definiálhatunk a CREATE TABLE vagy ALTER TABLE parancsokban

Megszorítások általános alakja:

[**CONSTRAINT** megszorításnév] megszorítás [megszorítás_állapot]

Megszorítás:

1) **NOT NULL**: biztosítja, hogy az adott oszlopba ne kerüljön null érték

2) **UNIQUE(oszlop[,oszlop]…)**: megköveteli, hogy az oszlop(ok)ban szereplő értékek különbözőek legyenek a tábla különböző soraiban. A megadott oszlopot (vagy oszlopok halmazát) egyedi kulcsnak hívjuk, több oszlop megadása esetén összetett egyedi kulcsról beszélünk.

Egyedi kulcs: tartalmazhat NULL értéket is, ha nem adtunk NOT NULL megszorítást ugyanazon oszlopokra. Ha nem adunk meg NOT NULL megszorítást, akkor tetszőleges számú sorban szerepelhet null érték, mert a rendszer a null értéket nem tekinti azonosnak semmi mással, tehát egy oszlopban (vagy akár az összes oszlopban) szereplő null érték mindig kielégíti a UNIQUE feltételt

3) **PRIMARY KEY(oszlop[,oszlop]…)**: elsődleges kulcsot hoz létre a táblához. Olyan oszlop(ok)at definiál, amelyben szereplő érték(ek) egyértelműen azonosítják a tábla minden sorát. A megszorítás egyrészt biztosítja a megadott oszlop(ok)ban szereplő érték(ek) egyediségét, másrészt azt, hogy az elsődleges kulcsot alkotó oszlopok egyike se tartalmazzon null értéket (UNIQUE és NOT NULL)

Az Oracle Szerver a UNIQUE és PRIMARY KEY kulcsmegszorítás betartatásához is létrehoz egy egyedi indexet a megfelelő oszlophoz.

4) [FOREIGN KEY (oszlop[,oszlop]…)]
REFERENCES tábla[(oszlop[,oszlop]…)]
[ON DELETE {CASCADE | SET NULL}

oszlop(ok)at jelöl ki **külső kulcsk**ént (hivatkozási integritási megszorításként) és kapcsolatot hoz létre a külső kulcs és ugyanazon vagy egy másik tábla elsődleges vagy egyedi kulcsa között.
A külső kulcs értékének vagy meg kell egyeznie a szülőtáblában szereplő értékek egyikével, vagy null értéknek kell lennie.
A külső kulcsot a gyermektáblában kell megadnunk, a hivatkozott oszlopot tartalmazó tábla lesz a szülőtábla. A külső kulcsot a következő kulcsszavak használatával definiáljuk:

- FOREIGN KEY: táblaszintű megszorításnál megadja a gyermektábla oszlopát, oszlopszintű megszorításnál elhagyható a kulcsszó
- REFERENCES: azonosítja a szülőtáblát és annak oszlopát/oszlopait
- ON DELETE CASCADE: a szülőtábla egy sorának törlése esetén a rendszer a gyermektábla függő sorait is törölje
- ON DELETE SET NULL: a szülőtábla egy sorának törlése esetén a rendszer a gyerektábla megfelelő oszlopaiban a külső kulcs értékét nullára konvertálja
- Ha az utóbbi két opció egyikét sem adjuk meg, nem törölhetjük a szülőtábla azon sorait, amelyekre a gyermektábla hivatkozik

Ha az ALKALMAZOTT táblában a TKOD oszlopot az alábbi megszorítással hoztuk létre:

```
tkod VARCHAR2(2) CONSTRAINT alk_fk
                 REFERENCES telephely(tkod) ON DELETE CASCADE,
```

akkor a

```
DELETE FROM telephely WHERE tkod=30;
```

parancs kiadása esetén a

```
DELETE FROM alkalmazott WHERE tkod=30;
```

is végrehajtódik.

5) **CHECK(feltétel)**: a feltételt minden sornak ki kell elégítenie. A feltételben a lekérdezések összeállításakor alkalmazott feltételeket és szerkezeteket használhatjuk, a következő kivételekkel:

- Nem hivatkozhatunk pszeudooszlopra, mint pl. CURRVAL, NEXTVAL, LEVEL vagy ROWNUM
- Nem használhatjuk a SYSDATE, UID, USER, USERENV függvényeket
- Nem alkalmazhatunk más sorok értékeire hivatkozó lekérdezéseket

Az egy oszlopra megadható CHECK megszorítások száma nincs korlátozva.

Megszorításokat definiálásának szintjei:

- **Oszlop**: Egy oszlopra vonatkozik, az adott oszlop definíciójának részeként adjuk meg. Bármilyen típusú megszorítást megadhatunk, közvetlenül az oszlop definíciója után írjuk, vesszőt nem kell tenni közéjük
- **Tábla:** Egy vagy több oszlopra vonatkozik, a NOT NULL kivételével bármilyen típusú lehet. A tábla oszlopainak definíciójától függetlenül adjuk meg, a több oszlopot érintő megszorításokat (pl. összetett kulcsok) csak tábla szintű megszorításként írhatjuk elő, vesszőt kell tenni közéjük.

Megszorítás_állapot:

- ENABLE VALIDATE: biztosítja, hogy minden új, a megszorítás alatt álló adatra vonatkozó DML művelet kielégítse a megszorítást. A régi adatokat is ellenőrzi. Alapértelmezés
- ENABLE NOVALIDATE: biztosítja, hogy minden új, a megszorítás alatt álló adatra vonatkozó DML művelet kielégítse a megszorítást. A régi adatokat nem ellenőrzi
- DISABLE: kikapcsolja/letiltja a megszorítást, az Oracle nem ellenőrzi azt

AS alselect:

- A rendszer létrehozza a táblát, és beszúrja a táblába a SELECT utasítás által visszaadott sorokat
- Ha nem adunk meg oszlopokat, a létrejövő tábla oszlopainak neve megegyezik az alselect oszlopainak nevével
- Ha megadunk oszlopokat, azok számának meg kell egyeznie az alselect SELECT utasítása által visszaadott oszlopok számával
- Ha az alselectben kifejezés szerepel, adjunk másodlagos nevet az oszlophoz
- *Az oszlopok definíciójában csak az oszlop neve, alapértelmezett értéke és integritási megszorítás szerepelhet, adattípus nem*
- *Hivatkozási integritási megszorítás nem adható meg, csak az ALTER TABLE-ben*
- *Az új táblára az integritási szabályok nem vonatkoznak, csak az oszlop adattípusú definíciók: név, típus, méret, NOT NULL*

b) Táblaszerkezet módosítása

ALTER TABLE [séma.]táblanév

ADD {oszlop | megszorítás} …	--új oszlop a tábla végére új megszorítás megadása
MODIFY …	--meglévő oszlop megváltoztatása, NOT NULL megszorítás megadása, megszorítások engedélyezése, letiltása
DISABLE CONSTRAINT	--megszorítás kikapcsolása
DROP CONSTRAINT…	--megszorítás eltávolítása
DROP [COLUMN]…	--oszlop eltávolítása
RENAME TO…	--tábla átnevezése

Általános szabály: az utasítás végrehajtása akkor lesz sikeres, ha a tábla módosított definíciójának eleget tesznek a táblában már meglévő adatok és a hivatkozási integritási megszorítások sem sérülnek.

I. Tábla bővítése új oszloppal, DEFAULT megadása:

- A meglevő sorok az új oszlopban NULL értéket kapnak
- Ha van DEFAULT, minden eddigi sor megkapja az adott DEFAULT értéket
- NOT NULL csak akkor adható meg új oszlophoz, ha még nem léteznek sorok a táblában vagy DEFAULT-al együtt

II. Megszorítás hozzáadása táblához:(tábla v. oszlop)

III. Oszlop módosítása, DEFAULT megadása:

- Növelhetjük a számértéket vagy karaktereket tartalmazó oszlopok szélességét
- Csökkenthetjük az oszlopok szélességét, ha csak null értéket tartalmaznak vagy ha a táblának nincsenek sorai
- Módosíthatunk CHAR típusú oszlopot VARCHAR2-re vagy fordítva, ha az oszlop csak null értékeket tartalmaz, vagy ha a méretet nem változtatjuk meg
- Módosíthatjuk az oszlopok adattípusát, ha csak null értékeket tartalmaznak
- Az alapértelmezett érték módosítása csak a táblába a módosítás után beszúrt sorokra van hatással

IV. NOT NULL megszorítás megadása:

Használható egy oszlophoz, ha minden létező sorban szerepel érték

V. Megszorítás letiltása/engedélyezése:

- ENABLE/DISABLE megadásával
- CASCADE a függő megszorításokat is kikapcsolja

VI. Oszlop törlése: (8i-től)

- Az oszlopok tartalmazhatnak adatot
- Legalább egy oszlopnak maradnia kell a táblában

- Törölt oszlopot nem tudunk helyreállítani
- Nem törölhetünk olyan oszlopot, amelyre más megszorítások hivatkoznak, de a CASCADE CONSTRAINTS opció megadásával az oszlopra hivatkozó megszorítások is törlődnek

VII. Megszorítás törlése:

- DROP CONSTRAINT utasításrésszel
- CASCADE a függő megszorításokat is törli

VIII. Tábla átnevezése:

c) Táblaszerkezet törlése

DROP TABLE táblanév [CASCADE CONSTRAINTS];

Adatok, indexek, jogosultságok törlődnek, szinonimák, nézetek maradnak, de érvénytelenek lesznek. A parancsot a tábla tulajdonosa, vagy DROP ANY TABLE jogosultsággal bíró felhasználó adhatja ki.
CASCADE CONSTRAINTS: a táblával együtt a táblára hivatkozó megszorítások is automatikusan törlődnek

d) Tábla csonkolása

TRUNCATE TABLE táblanév;

- Törli a tábla összes sorát, felszabadíthatja az elfoglalt tárterületet
- A sorok törlése nem visszagörgethető, a parancs nem generál visszagörgetési információt, így gyorsabb, mint a DELETE
- A parancsot a tábla tulajdonosa, vagy DELETE TABLE objektumjogosultsággal rendelkező felhasználó adhatja ki
- Ha a tábla egy hivatkozási integritási megszorítás szülője, akkor a táblát nem lehet csonkolni, tiltsuk le a megszorítást, mielőtt kiadnánk az utasítást

DELETE (DML): törli a tábla sorait, de nem szabadítja fel az elfoglalt tárterületet, a sorok törlése visszagörgethető.

e) Megjegyzés fűzése táblához

COMMENT {ON TABLE tábla | COLUMN oszlop} IS 'szöveg';

f) Objektum átnevezése

Az objektum tulajdonosa átnevezheti a táblát, nézetet, szekvenciát vagy szinonimát.

RENAME réginév TO újnév;

2. Nézettábla

Más táblá(k)ból és/vagy nézet(ek)ből SELECT utasítással származtatott adatbázis objektum. Ablakhoz hasonlít, amelyen keresztül megtekinthetők és módosíthatók az alapjául szolgáló táblák adatai. Adatokat nem tartalmaz, a rendszer a nézeteket SELECT utasítás formájában tárolja az adatszótárban.

USER_VIEWS

Használatának előnyei:

- Adathozzáférés szabályozása: bizonyos felhasználók az alaptáblához nem kapnak hozzáférési jogot, hanem nézeten keresztül az adatok egy részéhez férhetnek hozzá
- Adatbázis komplexitásának elrejtése: egy vagy több táblára, nézettáblára támaszkodó bonyolult lekérdezés eredménye könnyebben kezelhető, ha nézetet alkalmazunk
- Elérhetjük, hogy a felhasználók különböző csoportjai saját igényeiknek megfelelően különféle adatokhoz férhessenek hozzá

a) Nézettábla létrehozása, létező nézet felülírása

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW név [(oszlopnév,...)]
AS szelekciós_utasítás
[WITH CHECK OPTION] [CONSTRAINT megszorítás_név]
[WITH READ ONLY] [CONSTRAINT megszorítás_név];

FORCE: akkor is létrehozza a nézetet, ha az alaptábla nem létezik
NOFORCE: csak akkor hozza létre a nézetet, ha az alaptábla létezik
Szelekciós utasítás: tetszőleges, ORDER BY nélküli lekérdezés
WITH READ ONLY: DML műveletek letiltása
WITH CHECK OPTION: biztosítja, hogy csak a nézet számára hozzáférhető sorok szúrhatók be vagy módosíthatók (amilyen adatokat lekérdezhetünk a nézettáblán keresztül, csak azokat tarthassuk karban)

Amikor nézettáblán hajtunk végre lekérdezést, a rendszer az alaptáblákból veszi az adatokat, ha az alaptáblák tartalma módosul, akkor módosul a nézettábla is.

- **Egyszerű nézet**:
  - Adatai egy táblából származnak
  - Nem tartalmaz oszlopkifejezést, függvényt vagy adatcsoportokat
  - A nézeten keresztül végrehajthatók DML utasítások, ha a nézet tartalmazza a tábla minden NOT NULL oszlopát (kivéve, ha van DEFAULT érték)
- **Az összetett nézet:**
  - Adatai több táblából származnak
  - Tartalmazhat függvényt vagy adatcsoportokat
  - Nem mindig hajthatók végre DML utasítások a nézeten keresztül

Egyszerű nézettáblán keresztül az alaptábla karbantartható az INSERT, UPDATE, DELETE műveletekkel

b) Nézettábla törlése

DROP VIEW nézettábla_név;

A parancsot a nézet létrehozója, vagy DROP ANY VIEW jogosultsággal rendelkező felhasználó adhatja ki

I. Karbantartás nézeten keresztül:

Törölhetünk sorokat a nézetből, ha nem tartalmaz csoportfüggvényeket, GROUP BY utasításrészt, DISTINCT kulcsszót, a ROWNUM pszeudo-oszlop kulcsszavát
Módosíthatunk adatokat a nézetben, ha nem tartalmaz csoportfüggvényeket, GROUP BY utasításrészt, DISTINCT kulcsszót, a ROWNUM pszeudo-oszlop kulcsszavát vagy kifejezéssel definiált oszlopot
Felvehetünk adatokat a táblába, kivéve, ha a nézet nem tesz eleget az előző feltételeknek, illetve ha az alaptábla tartalmaz olyan alapértelmezett érték nélküli NOT NULL oszlopokat, amelyek nem szerepelnek a nézetben. Minden szükséges értéknek szerepelnie kell a nézetben. Ne felejtsük el, hogy a nézeten keresztül közvetlenül a nézet alaptáblájába írunk be új adatokat

3. Index

Sémaobjektum, amely

- mutató használatával felgyorsíthatja a sorok visszakeresését
- biztosíthatja egy vagy több oszlopon belül az értékek egyediségét
- ugyanakkor lassítja a táblával kapcsolatos DML műveleteket, mivel ezek végrehajtása után az összes indexet frissíteni kell

Egyedi index: értékei különbözők, az Oracle automatikusan létrehozza, ha egy tábla oszlopára PRIMARY KEY vagy UNIQUE megszorítást állítunk be, neve azonos lesz a megszorítás nevével
Felhasználói index: a felhasználó hozza létre, megfontolandó!!
Például egy FOREIGN KEY oszlopindex egy lekérdezésben megadott összekapcsoláshoz gyorsítja az adatok visszakeresését. Index egy vagy több oszlopra is létrehozható

Mikor hozzunk létre indexet?
Ha pl. van egy több ezer soros táblánk dátum típusú oszloppal és gyakran van szükségünk dátum vagy időszak szerinti lekérdezésre. Ha az oszlophoz nem tartozik index, akkor a rendszernek minden kereséskor a teljes táblát végig kell néznie.

Az index közvetlen és gyors hozzáférést biztosít a tábla soraihoz. Használatával csökkenthetjük a szükséges lemezműveletek számát, mert közvetlenül kijelöli a keresett adat helyét.

A rendszer automatikusan használja és karbantartja az indexeket, létrehozás után a felhasználónak nem kell foglalkoznia vele.
Az indexek logikailag és fizikailag is függetlenek a táblától, amelyet indexelnek. Ez azt jelenti, hogy bármikor létrehozhatók és törölhetők, ez nincs hatással az alaptáblára vagy más indexekre. Ha eldobunk egy táblát, a hozzá tartozó indexek is törlődnek.

a) Index létrehozása

```
CREATE [{UNIQUE | BITMAP}] INDEX indexnév
    ON táblanév (oszlopkif [ASC | DESC],...)    -- indexkulcsot alkotó oszlopok
    [tárolási és egyéb előírások];
```

UNIQUE: az oszlop(ok)nak, mely alapján indexelünk, egyedinek kell lenni
BITMAP: B-fa szerkezetű index helyett bitmap indexet hoz létre
ASC, DESC: kompatibilitás miatt van, az index létrehozása mindig az oszlopbeli értékek növekvő sorrendjében történik

b) Index módosítása

ALTER INDEX … utasítással csak a tárolási jellemzők változtathatók.

c) Index törlése

DROP INDEX indexnév;

Mikor célszerű indexet használni?

- Ha az oszlop értékei széles tartományba esnek
- Ha az oszlopban sok null érték szerepel
- Ha két vagy több oszlopot gyakran használunk együtt WHERE feltételben vagy összekapcsolásban
- Ha a tábla nagy, és a legtöbb lekérdezés csak a sorok legfeljebb 2–4 %-át adja vissza

Mikor ne használjunk indexet?

- Ha a tábla túl kicsi
- Ha az oszlopokat nem használjuk gyakran lekérdezések feltételeiben
- Ha a legtöbb lekérdezés a sorok több mint 2-4%-át adja vissza eredménytelenül
- Ha a tábla gyakran módosul
- Ha az indexszel ellátott oszlopokra kifejezés részeként hivatkozunk

IX. ADATMANIPULÁCIÓS NYELV (DML)

INSERT	Új sor(ok) felvitele táblába, nézetbe
UPDATE	Táblában meglévő sor(ok) módosítása
DELETE	Táblában meglévő sor(ok) törlése

Megjegyzések:

- A karbantartó műveletek által kezelt adatoknak ki kell elégíteni az integritási megszorítási szabályokat
- Egy-egy karbantartó utasítással csak egy táblát kezelhetünk ORACLE 8i-ig. Az utasításokban szereplő alselectek, más táblá(k)ra is hivatkozhatnak és korrelációban lehetnek a karbantartó utasítással kezelt táblával
- A rendszer a táblákhoz kapcsolódó index-objektumokat automatikusan karbantartja

I. Adatok felvitele

INSERT INTO {táblanév | nézet} [aliasnév] [(oszlopnév, ...)]
{**VALUES** (érték, ...) | alselect};

- Oszlopnév - érték pároknak típusban és értéktartományban illeszkedni kell
- Ha az oszlopnév felsorolás elmarad, akkor az összes oszlopra vonatkozik az értékadás a CREATE TABLE utasításban megadott sorrendben
- NULL értékek bevitele ott lehetséges, ahol ez megengedett, nem tiltja megszorítás
- Speciális értékek megadhatók kifejezéssel: SYSDATE+7, USER, DEFAULT stb.
- Dátumkonstans megadása az alapértelmezett dátumformátum szerint vagy TO_DATE konverziót alkalmazva lehetséges. Ha időpontot nem adunk meg, az alapértelmezett idő éjfél
- Az alselect más táblából nyert adatok felvitelét teszi lehetővé

a) Adatfelvitel a tábla minden oszlopába:

b) Adatfelvitel hiányos listával, NULL kulcsszóval

c) Adatfelvitel másik táblából

- Az alselect select listájának ugyanannyi oszlopa kell hogy legyen, mint a VALUES utasításrész oszlop listájának
- Az alaptábla oszlopaira vonatkozó összes szabályt követnünk kell, hogy az INSERT utasítás sikeresen működjön. Például, nem használhatunk többszörös alkalmazotti azonosítót, és nem hagyhatunk ki egyetlen értéket sem a kötelezően nem nulla oszlopra vonatkozóan

II. Adatok módosítása

UPDATE [séma.]{táblanév | nézet} [aliasnév]
SET {oszlop=kifejezés | oszlop=(alselect) | (oszlopnév,…)=(alselect)}, ...
[**WHERE** feltétel];

SET: megadja, hogy mely oszlopok régi értékét mely új értékre kell módosítani
Érték: NULL, DEFAULT, USER, skalár értéket adó alselect is lehet
WHERE feltétel: a módosításban résztvevő sorokat válogatja ki

III. Adatok törlése

DELETE [FROM] [séma.]{táblanév | nézet} [aliasnév]
[**WHERE** feltétel];

WHERE feltétel: a törlésben résztvevő sorokat válogatja ki

a) Egy sor törlése:

b) Tábla teljes tartalmának törlése:

c) Táblarészlet törlése:

Hivatkozási megszorítási hiba: ha olyan sort törlünk egy táblában, amelynek elsődleges kulcsára hivatkozás van egy másik vagy ugyanazon táblában. Szülő táblában olyan sor nem törölhető, melyhez vannak sorok a gyerek táblában!

X. ADATVEZÉRLŐ NYELV (DCL)

I. Tranzakciókezelés

Konzisztencia: Adatbázis fogalom, annak megkövetelése, hogy
- Összetartozó adatok módosítása együtt, megfelelő sorrendben történjen (pl. ha a telephely megszűnik, előbb az alkalmazottak törlődjenek, aztán a telephely
- Ha vannak redundáns adatok, azoknak nem szabad egymással ellentmondásban lenni

Tranzakció: az adatfeldolgozás logikai egysége. Egy feladat szempontjából **összetartozó DML utasítások** sorozata, amelyek mindegyikének sikeresen kell végrehajtódni ahhoz, hogy ellentmondásmentes (konzisztens) legyen az adatbázis.

Tranzakció kezdete:
Az első végrehajtható DML utasítás végrehajtásával kezdődik
Tranzakció vége:
- COMMIT utasítás (véglegesítés): rögzíti a tranzakció folyamán végrehajtott adatmódosításokat az adatbázisban

- ROLLBACK utasítás (visszagörgetés): visszaállítja a tranzakció megkezdésekor (a legutolsó COMMIT-nál) rögzített állapotot az adatbázisban, így a tranzakcióhoz tartozó utasítások egyikének sem érvényesül a hatása
- egy DDL utasítás, például a CREATE kiadása
- egy DCL utasítás kiadása
- a felhasználó kilépése az SQL*Plus környezetből
- a számítógép meghibásodása vagy a rendszer összeomlása

Explicit tranzakciókezelés:
- COMMIT
- ROLLBACK
- **SAVEPOINT** mentési_pont;
  lehetővé teszi, hogy a tranzakció közben olyan pontot képezzünk, amely pontig részlegesen visszagörgethetjük a tranzakciót
- **ROLLBACK TO** mentési_pont;
  A megadott pontig lehet a visszagörgetést elvégezni. Mindenféle COMMIT törli a mentési pontokat.

Implicit tranzakciókezelés:
- DDL (pl. CREATE), DCL (pl. GRANT) utasítások után automatikusan végrehajtódik a COMMIT
- Az eszközökből való normális kilépés után automatikusan, beállítástól függően, COMMIT vagy ROLLBACK történik
- Abnormális programbefejezés esetén automatikus ROLLBACK lesz kiadva a legközelebbi indításkor

Automatikus jóváhagyás (COMMIT):
- DDL utasítás kiadásakor
- DCL utasítás kiadásakor
- az SQL*Plus környezetből való normális kilépéskor (EXIT), a COMMIT vagy a ROLLBACK kiadása nélkül

Automatikus visszagörgetés (ROLLBACK):
- az SQL*Plus rendszer hibás befejeződése (pl. ablak bezárása)
- rendszer összeomlása

I. Felhasználók kezelése, jogosultságok

Többfelhasználós környezetben az adatbiztonság fenntartásához szabályoznunk kell az adatbázishoz való hozzáférést. Az Oracle a következő lehetőségeket nyújtja:

- adatokhoz való hozzáférés szabályozása
- hozzáférés biztosítása az adatbázis egyes objektumaihoz
- az adott és kapott jogosultságok ellenőrzése az adatszótárban

Adatbázis adminisztrátor (**DBA**: Database Administrator-SYS, SYSTEM): kitüntetett felhasználó, felügyeli az egész rendszer működését, a rendszer installálása során az alábbi teendők elvégzéséhez szükséges jogosultságokat megkapja

Jogosultság: adott SQL utasítás végrehajtási jogát jelenti, van rendszerjogosultság és objektumkezelési jogosultság

1. Felhasználó:

Azonosító, melynek segítségével kapcsolódni tudunk az adatbázishoz. DBA hozza létre

2. Séma

Objektumok gyűjteménye. A séma tulajdonosa egy adatbázis felhasználó, a neve megegyezik az adott felhasználó nevével. A sémát ténylegesen nem a CREATE SCHEMA parancs hozza létre, a CREATE USER parancs hatására létrejön, a CREATE SCHEMA csak benépesíti táblákkal, nézetekkel.
A CREATE SCHEMA parancs lehetővé teszi táblák és nézetek létrehozását, valamint jogosultságok adományozását **egy tranzakció alatt**. Ha minden parancs sikeres véglegesítés történik, ha valamelyik hibás, minden parancs visszagörgetése lesz végrehajtva

3. Szerepkör (ROLE):

- Jogosultságok névvel ellátott gyűjteménye, amely lehetővé teszi, hogy egyetlen hozzárendeléssel a benne szereplő összes jogot hozzárendeljük a felhasználóhoz vagy másik szerepkörhöz
- Egy felhasználónak több szerepköre is lehet, egy szerepkör több felhasználóhoz is tartozhat
- GRANT parancs jogosultságokat rendel a szerepkörhöz, majd a szerepkör felhasználóhoz vagy szerepkörhöz rendelhető
- Előredefiniált szerepkörök: CONNECT, RESOURCE, DBA, stb. (Elavult!)

4. Jogosultságok

a) Rendszer szintű jogok

DBA adja, lehetővé teszik az adatbázisba való belépést, valamint DDL és DCL (pl. GRANT ANY PRIVILEGE) parancsok végrehajtását

DBA alapvető jogosultságai:

Rendszerjogosultság	**Engedélyezett műveletek**
CREATE USER	Oracle felhasználók felvétele (a DBA szerepkörhöz szükséges jogosultság)
DROP USER	Felhasználó törlése
DROP ANY TABLE	Tábla törlése bármelyik sémában
BACKUP ANY TABLE	Tábla mentése bármelyik sémából az exportálási segédeszközzel
SELECT ANY TABLE	Tábla, nézet vagy frissített másolat lekérdezése bármelyik sémában
CREATE ANY TABLE	Tábla létrehozása bármelyik sémában

Alkalmazásfejlesztő alapvető jogosultságai:

Rendszerjogosultság	**Engedélyezett műveletek**
CREATE SESSION	Kapcsolódás az adatbázishoz
CREATE TABLE	Tábla létrehozása a felhasználó sémájában
CREATE SEQUENCE	Szekvencia létrehozása a felhasználó sémájában
CREATE VIEW	Nézet létrehozása a felhasználó sémájában
CREATE PROCEDURE	Tárolt eljárás, függvény vagy csomag létrehozása a felhasználó sémájában

Rendszerjogok adása:

GRANT {rendszer_jog | szerepkör | **ALL [PRIVILEGES]**} , ...
TO {felhasználó_név | szerepkör | **PUBLIC**}, …
[IDENTIFIED BY jelszó]
[WITH ADMIN OPTION];

PUBLIC: az összes adatbázis felhasználó
ALL PRIVILEGES: az összes rendszerjog

WITH ADMIN OPTION: a felhasználó a kapott jogokat továbbadhatja más felhasználóknak vagy szerepköröknek

Rendszerjogok visszavonása:

REVOKE {rendszer_jog | szerepkör | **ALL [PRIVILEGES]**},…
FROM {felhasználó_név | szerepkör | **PUBLIC**}, …;

Ha egy felhasználótól minden jogosultságot visszavonunk, az objektumai még megmaradnak az adatbázisban, előtte célszerű ezeket DROP-al törölni

b) Objektumjogok

Konkrét, névvel ellátott adatbázis objektumokra vonatkozó művelet végrehajtási jogosultságok. A felhasználó saját objektumaihoz minden jogosultságot megkap továbbadható módon, más felhasználók objektumaihoz csak konkrét jogok birtokában férhet hozzá. Hozzáférési jogokat az objektum létrehozója (tulajdonosa) vagy a DBA adhat másoknak. A konkrétan megadható objektum jogok az objektum típusától függenek.

Objektumkezelési jogosultságok	**Tábla**	**Nézet**	**Szekvencia**	**Eljárás, Függvény, Csomag**
ALTER	X		X	
DELETE	X	X		
EXECUTE				X
INDEX	X			
INSERT	X	X		
REFERENCES	X			
SELECT	X	X	X	
UPDATE	X	X		

UPDATE, REFERENCES, INSERT utasítások végrehajtására vonatkozó jogosultságot szűkíthetjük a módosítható oszlopok körének megadásával.
SELECT utasítás használata úgy korlátozható, hogy létrehozunk egy nézetet, amely csak az oszlopok egy részhalmazát tartalmazza, majd erre a nézetre adunk SELECT jogosultságot.
A szinonimákra adott jogosultságot a rendszer úgy értelmezi, mintha közvetlenül az adott szinonima alaptáblájára vonatkozó jogot adnánk.

Objektumjogok adása:

GRANT {objektum_jog [(oszlop, …)] | **ALL [PRIVILEGES]**},…
ON objektum
TO {felhasználó_név | szerepkör | **PUBLIC**}
[**WITH GRANT OPTION**];

PUBLIC: az összes adatbázis felhasználó
ALL PRIVILEGES: az összes objektumjog
WITH ADMIN OPTION: a felhasználó a kapott jogokat visszavonhatja más felhasználóktól vagy szerepköröktől (körkörösen nem lehet)

Objektumjogok visszavonása:

REVOKE {objektum_jog | **ALL [PRIVILEGES]**},…
ON objektum
FROM {felhasználó_név | szerepkör | **PUBLIC**},…
[CASCADE CONSTRAINTS];

A WITH GRANT OPTION lehetőséggel továbbadott jogosultságok is visszavonódnak.
CASCADE CONSTRAINTS hatására az objektumra a REFERENCES jogosultság segítségével megadott hivatkozási integritási kényszerek is törlődnek.
Oszlopszintű jogosultságok egyenként nem vonhatók vissza.