Adatbázisrendszerek – gyakorlati segédlet

1. Alapok

1.1 SQL - Structured Query Language

Relációs adatbázisok kezelésére alkalmas, szabványosított lekérdező nyelv. Az 1970-es években kezdődött a fejlesztése (IBM). Iparági összefogással deklarálták az alapjait. 1986-től ANSI, 1987-től ISO szabvány. Szinte minden relációs DBMS alkalmazza (módosításokkal).

1.2 SQL operátorok

Aritmetikai operátorok: + (unáris), - (unáris), +, -, /, *
Karaktersorozatok összefűzése: || (bináris)
Logikai operátorok: AND, OR, NOT
Hasonlító operátorok: <, <=, >, >=, =, <>
null érték vizsgálata: kifejezés IS [NOT] NULL

1.3 Lekérdezések általános alakja

```
SELECT [{ALL|DISTINCT}] mezőkifejezés [álnév] [, mezőkifejezés [álnév]]...
FROM táblakifejezés [álnév]
[WHERE feltétel]
[GROUP BY csoportosítómező [, csoportosítómező]...]
[HAVING feltétel]
[ORDER BY mezőkifejzés [, mezőkifejezés]...]
```

A SELECT utasítás az adatok egy halmazát válogatja ki egy táblázatba az adatbázisból.
Opcionálisan megadható:
WHERE: Az utána álló feltételnek megfelelő sorok leválogatása.
GROUP BY: Az utána álló mezőkifejezések alapján csoportosítja az adatokat.
HAVING: A feltételnek megfelelő sorok leválogatása a csoportosítás után.
ORDER BY: Sorok rendezése a megadott mezők alapján.

1.4 A null érték

Az a mező, amelynek értéke nem került megadásra a null értéket veszi fel. A null nem nulla és nem üres sztring!
Ha két érték összehasonlításánál vagy egy műveletnél az egyik operandus null értékű, az eredmény is null lesz.
pl.: a > null, a <> null, null * 2, a - null stb.
Egy null értéket tartalmazó logikai kifejezés eredménye csak abban az esetben lesz null, ha a kifejezés értéke függ attól, hogy a null érték helyén igaz, vagy hamis érték szerepelne.
pl.: 1 = 0 AND null (hamis), 1 = 0 OR null (null)
A szelekciós feltétel null értéke esetén az érintett sorok nem kerülnek leválogatásra.

1.5 Mintaillesztés

```
kifejezés LIKE minta
```

Mintaillesztést a LIKE operátor segítségével végezhető. A mintában szerepelhet aláhúzás jel (_), mely egy darab tetszőleges karaktert helyettesít vagy százalék jel (%), amelyre tetszőleges karaktersorozat illeszkedik.

1.6 A DUAL tábla

A DUAL tábla egy speciális, egy attribútummal („DUMMY”) és egy sorral rendelkező tábla. Az egyetlen tárolt érték az „X” karakter. Ezt a táblát akkor használjuk, ha tábláktól független lekérdezéseket akarunk indítani.

2. Csoportosítás

```
SELECT ... FROM ...
GROUP BY csoportosítókif1 [, csoportosítókif2] ... ]
[HAVING feltétel]
```

Az SQL lehetővé teszi, hogy a sorokat csoportosítsuk, és az egyes csoportokon számításokat végezzünk. A GROUP BY kulcsszó után álló csoportosító kifejezés(ek) alapján történik a csoportképzés. Több csoportosító kifejezés esetén először az első alapján történik a csoportosíts, majd az egyes csoportokon belül történik az újabb csoportosítás a következő kifejezés alapján, stb.

HAVING feltétel megadása esetén a csoportosítás után előállt sorok közül csak azok kerülnek be az eredményhalmazba, amelyek megfelelnek a feltételnek. Az attribútum-listában a csoportosítás alapját képző kifejezések, konstans kifejezések, valamint aggregáló függvények vagy azokkal alkotott kifejezések szerepelhetnek. Olyan attribútum nem szerepelhet aggregáló függvényen kívül, amely a csoportosításban nem vesz részt.

Aggregáló függvények:
AVG(kifejezés): átlagolás
MIN(kifejezés): minimum kiválasztás
MAX(kifejezés): maximum kiválasztás
SUM(kifejezés): összegzés
COUNT({attribsorszám|attribnév}): megszámolja a nem null értékeket
- COUNT(DISTINCT attribútum): a különböző értékek száma
- COUNT(*): a sorok száma

Az AVG, MIN, MAX és a SUM a null értékeket figyelmen kívül hagyják.

3. Táblák összekapcsolása

3.1 Cross join

```
SELECT ... FROM táblakifejezés { , |CROSS JOIN} táblakifejezés
```

Kereszt összekapcsolás létrehozása a táblakifejezések vesszővel elválasztott felsorolásával, vagy a CROSS JOIN kulcsszóval lehetséges. Az eredményként kapott táblában a két tábla sorainak összes lehetséges kombinációja pontosan egyszer jelenik meg, azaz Descartes-szorzatot képez. Egy WHERE feltétel megadásával lehet elérni, hogy csak a szükséges sorok jelenjenek meg.

3.2 Inner join

```
SELECT ... FROM táblakifejezés [INNER] JOIN táblakifejezés ON feltétel
```

Belső összekapcsolás létrehozásához a táblakifejezések között az INNER JOIN vagy JOIN kulcsszavakat kell használni. Az összekapcsolás feltételét az ON kulcsszó után kell megadni. Az eredményként kapott táblában a két tábla azon sorainak kombinációi szerepelnek, amelyek esetében igaz az összekapcsolási feltétel. Ugyan azt az eredményt adja, mint egy megfelelő WHERE feltételt tartalmazó kereszt összekapcsolás. Itt is van lehetőség WHERE feltétel megadására.

3.2.1 Equi join

A belső összekapcsolás olyan speciális esetének tekinthető, amikor a táblák kapcsolódó attribútumainak páronkénti egyenlőségét írjuk elő.
Ha a kapcsolódó attribútumok nevei megegyeznek, akkor lehetőség van a következő szintaktika használatára:

```
SELECT ...
FROM táblakif1 JOIN táblakif2 USING (attrib)
```

Az Oracle DBMS ebben az esetben csak az egyik táblából hagyja meg a kapcsolódó attribútumot.

3.2.2 Natural join

A természetes összekapcsolás esetén a két tábla azonos nevű attribútumait tekintjük a kapcsolódó attribútumoknak. Az equi join-hoz hasonlóan itt is az attribútumok egyenlőségét írjuk elő. A kapcsolódó attribútum-párok közül csak az egyik tag jelenik meg az eredménytáblában.
Szintaxis: `SELECT ... FROM táblakif1 NATURAL JOIN táblakif2`

3.2.3 Több táblás belső összekapcsolás

```
SELECT ...
FROM táblakifejezés
[INNER] JOIN táblakifejezés
ON feltétel
[INNER] JOIN táblakifejezés
ON feltétel ...
```

3.3 Külső összekapcsolás

```
SELECT ... FROM táblakifA
{LEFT|RIGHT|FULL} OUTER JOIN táblakifB
ON feltétel
```

A külső összekapcsolásnak három típusa van:

- LEFT OUTER JOIN (baloldali külső összekapcsolás): A belső összekapcsolás eredményéhez hozzáveszi a baloldali tábla többi sorát is.
- RIGHT OUTER JOIN (jobboldali külső összekapcsolás): A belső összekapcsolás eredményéhez hozzáveszi a jobboldali tábla többi sorát is.
- FULL OUTER JOIN (teljes külső összekapcsolás): A belső összekapcsolás eredményéhez hozzáveszi a baloldali és jobboldali tábla többi sorát is.

4. Néhány adattípus Oracle SQL-ben

4.1 CHAR(n): rögzített hosszúságú (n karakteres) sztring tárolására használható. Az alapértelmezett hossz 1 karakter, a maximális hossz 2000 karakter. Ha a tárolandó sztring rövidebb, mint a megadott hossz, akkor az a végén szóközökkel kerül kiegészítésre. Jellemzően akkor használjuk, ha az attribútum minden értéke azonos hosszúságú lesz. Például: személyi igazolvány szám, termékkód, stb.

4.2 VARCHAR2(n): legfeljebb n karakter hosszúságú sztring tárolására használható. Jellemzően akkor használjuk, ha az attribútum értékei eltérő hosszúságú sztringek lesznek. Például: nevek, címek stb. Ezen esetekben a VARCHAR2 használatával tárhelyet lehet spórolni a CHAR-hoz képest.

4.3 NUMBER: legfeljebb 38 értékes számjeggyel rendelkező egész és lebegőpontos számok, valamint a mínusz és plusz végtelen értékek tárolására használható.
NUMBER(p): p a számjegyek száma (csak egész számok tárolása)
NUMBER(p, s): p a számjegyek száma, amiből s a tizedespont utáni számjegyek száma
NUMBER(*, s): ha csak a tizedes jegyek számát szeretnénk meghatározni, akkor a p paraméter helyett a '*' szimbólumot kell megadni

4.4 DATE: időpontok (dátum és idő) tárolására használható.
A DATE típusra értelmezettek az =, <>, >, <, >=, <= operátorok, azaz dátum és idő értékek természetes módon hasonlíthatóak össze.
<dátum> {+|-} <szám>: napokat ad hozzá/von ki a dátumból. Például: SYSDATE+0.5 hozzáad fél napot az aktuális időponthoz.
Két DATE típusú érték különbsége a dátumok közötti napok száma lesz.

Az INTERVAL kulcsszó segítségével növelhető vagy csökkenthető egy dátum típusú érték.

Szintaxis: <dátum> {+|-} INTERVAL '<egész érték>' <egység>, ahol az egység: YEAR, MONTH, DAY, HOUR, MINUTE, vagy SECOND.

Az EXTRACT függvénnyel egy dátum típusú érték egy része nyerhető ki numerikus értékként.
Szintaxis: EXTRACT(<egység> FROM <dátum>), ahol az egység ugyan az lehet, mint az INTERVAL esetében.

5. Függvények

5.1 Az NVL függvény

`NVL(attribnév,érték)` : a null érték kezelésére szolgál.
A második paraméter az az érték, amelyet vissza akarunk kapni, ha az első paraméterként kapott attribútum null értéket vesz fel. Az értéknek az attribútum típusára konvertálhatónak kell lennie.

5.2 A DECODE függvény

`DECODE(mezőkif, érték, eredmény, [érték, eredmény] ..., [alapért_eredmény])`
Oracle DBMS specifikus függvény. Egy mezőkifejezés értékét hasonlítja a felsorolt keresési értékekhez és ha egyezést talál, akkor visszaadja az ahhoz tartozó eredményt. Ha nem talál egyezést, de meg van adva egy alapértelmezett eredmény, akkor azt adja vissza. Ha nem talál egyezést, és nincs megadva alapértelmezett eredmény, akkor null-t ad vissza.

5.3 Néhány sztringkezelő függvény

`UPPER(sztring)`: nagybetűssé alakítja a sztringet
`LOWER(sztring)`: kisbetűssé alakítja a sztringet
`INITCAP(sztring)`: minden szót nagy kezdőbetűssé alakít
`LENGTH(sztring)`: a sztring hosszát adja meg
`CONCAT(sztring,sztring)`: két sztringet konkatenál (Eredménye ugyan az, mint a || operátornak.)
`SUBSTR(sztring,m[,n])`:az m-dik karaktertől kezdődően n karaktert ad vissza a sztringből. Ha az m negatív, akkor a sztring végétől vett m-dik karaktert jelenti. Ha az n hiányzik, akkor az összes karaktert visszaadja a sztring végéig.

5.4 Konverziós függvények

`TO_CHAR(szám)`**:** szám típusú értékek sztring típusra történő konvertálásra használható.
`TO_NUMBER(sztring)`**:** a sztring tartalmának szám típusra történő konvertálására használható.
`TO_NUMBER(sztring, formátumsztring)`**:** egy formátumsztring segítségével adhatjuk meg, hogy hogyan értelmezze a DBMS a sztringet. A formátumsztringben a '9'-es karakter jelöli a számjegyek helyét, a '.' karakter a tizedespontot, a ',' pedig az ezreseket elválasztó szimbólum.

5.5 Konverziós függvények – dátumok

`TO_DATE(sztring, formátum)`**:** sztringek konvertálását teszi lehetővé dátum típusra egy formátumsztring alapján. Első paramétere egy sztring, amely egy dátumot tartalmaz, a második paramétere pedig egy formátumsztring, amely alapján a DBMS értelmezi az első paraméterként kapott értéket.

`TO_CHAR(dátum, formátum)`**:** dátumok formázott megjelenítésére használható. Első paramétere egy dátum típusú attribútum, a második paramétere pedig egy formátumsztring, amely azt írja le, hogy hogyan szeretnénk látni a dátum és az idő egyes részeit.

A formátumsztring a következő elemekből állhat:
YEAR – az év kiírva
YYYY, YYY, YY, Y – az év négy, három, kettő vagy egy számjeggyel
Q – a negyedév sorszáma
MM – a hónap sorszáma két számjeggyel kiírva
MON – a hónap rövidítése

MONTH – a hónap kiírva
W – a hónap hetének sorszáma
WW – az év hetének sorszáma
D – a hét napjának sorszáma,
DD – a hónap napjának sorszáma
DDD – az év napjának sorszáma
DAY – a hét napjának neve
DY – a nap rövidítése
HH, HH24 – az óra 12 és 24 órás formátumban
MI – a perc
SS – a másodperc, SSSSSS – a másodpercek éjféltől

A formátumsztringben az egyes elemek kis- és nagybetűvel is szerepelhetnek. Ennek akkor van jelentősége, ha az eredmény szöveges részt is tartalmaz, mivel ezek a formátumsztringnek megfelelően kis- vagy nagybetűvel jelennek majd meg.

FM (fill mode): ha a formátumsztring az FM módosítóval kezdődik, akkor a kezdő nullák és a kitöltő szóközök nem jelennek meg az eredményben.

6. Beágyazott lekérdezések

Az SQL lehetővé teszi SELECT utasítások egymásba ágyazását, akár többszörösen is. Oracle SQL esetében ez maximum 255 szintig lehetséges.
A beágyazott lekérdezéseknek („alszelekteknek") két fő típusa különböztethető meg az alapján, hogy a „külső" és a „belső" lekérdezés milyen viszonyban áll egymással:
- Korrelálatlan (egyszerű) alszelekt
- Korrelált (kapcsolt) alszelekt

6.1 Korrelálatlan (egyszerű) alszelekt
Önmagában is kiértékelhető, azaz nem függ a külső SELECT utasítástól.
Kiértékelés: Először az alszelekt értékelődik ki, és az az által visszaadott értékek kerülnek át a külső SELECT utasításhoz. Csak egyszer fut le.

6.2 Korrelált (kapcsolt) alszelekt
Önmagában nem értékelhető ki, mert hivatkozik egy külső SELECT utasításban szereplő attribútumra/kifejezésre.
Kiértékelés: A kiértékelés a külső SELECT-nél kezdődik, majd az átadja a hivatkozott értéket a belső SELECT-nek, amely előállítja az értékekhez tartozó eredményt. Ezután a belső SELECT által előállított értékkel folytatódik a külső SELECT kiértékelése. Újra lefut minden átadott értéknél.
Fontos: Ha a külső és a belső SELECT ugyan azt a táblát használja, akkor vagy a külső vagy a belső lekérdezésben annak alias nevet kell adni, és annak attribútumait minősített nevekkel kell hivatkozni.

6.3 Alszelektek kezelése
Az alszelekteknek mindig zárójelek közt kell szerepelniük.
Minden alszelekt eredménye kezelhető táblaként, azaz bárhol szerepelhet, ahol táblakifejezés szerepelhet. Ha az alszelekt feltételben szerepel, akkor annak megfelelően kell megírni és kezelni, hogy hány sort és oszlopot adhat az adott helyen vissza.

6.4 ROWNUM
A ROWNUM egy pszeudo-oszlop, azaz fizikailag nincs tárolva, de ugyanúgy kezelhető, mint a többi attribútum.
Megadja, hogy hányadikként olvasta ki a DBMS a táblából vagy az eredményhalmazból az adott sort.
A számozás 1-től indul és egyesével növekszik.

Olyankor használható, amikor egy táblának vagy halmaznak a legelső n sorára van szükség. Használata rendezett adatok esetében gyakoribb. Például: 5 legdrágább könyv, 3 legfiatalabb tag stb.

6.5 Halmazt visszaadó alszelektek

Az alszelektek által visszaadott értékhalmazok a következő operátorokkal kezelhetőek feltételekben:
ALL, ANY, IN, EXISTS, NOT IN, NOT EXISTS
Minden operátor esetében szükséges, hogy az összehasonlítandó értékek típusa és száma egymásnak megfelelő legyen.

ALL: A feltétel akkor teljesül, ha az alszelekt által visszaadott összes sorra teljesül a kifejezés1 operátor kifejezés2 feltétel.

… kifejezés1 operátor ALL (SELECT kifejezés2 FROM …) …

ANY: A feltétel akkor teljesül, ha az alszelekt által visszaadott sorok közül legalább egyre teljesül a kifejezés1 operátor kifejezés2 feltétel.

… kifejezés1 operátor ANY (SELECT kifejezés2 FROM …) …

IN/NOT IN: A feltétel akkor teljesül, ha az IN (NOT IN) előtt álló kifejezés vagy zárójelbe tett kifejezések (nem) szerepelnek az alszelekt által visszaadott sorok közt.

… kifejezés1 IN (SELECT kifejezés2 … FROM …) …
… (kifejezés11, … , kifejezés1n) IN (SELECT kifejezés21 … kifejezés2n FROM …) …
… kifejezés1 NOT IN (SELECT kifejezés2 … FROM …) …
… (kifejezés11, … , kifejezés1n) NOT IN (SELECT kifejezés21 … kifejezés2n FROM …) …

EXISTS/NOT EXISTS: A feltétel akkor teljesül, ha az utána álló alszelekt legalább 1 sort visszaad (egy sort sem ad vissza).

… EXISTS (SELECT kifejezés2 FROM …) …
… NOT EXISTS (SELECT kifejezés2 FROM …) …

7. Data Definition Language (DDL)

Az SQL adatdefiníciós (DDL) utasításai segítségével alakítható ki az adatok tárolására szolgáló séma.
A DDL utasítások többek között táblák, táblák közötti kapcsolatok, indexek és megszorítások létrehozására, módosítására és törlésére használhatók.

7.1 Táblák létrehozása

Egy tábla létrehozásának általános alakja:

```
CREATE TABLE táblanév(
    oszlopdefiníció
    [, oszlopdefiníció]…
    [táblaszintű megszorítás]…
);
```

Oszlopdefiníció:

```
oszlopnév típus [oszlopszintű_megszorítások]
```

Oszlop szintű megszorítások: Egyetlen oszlopra vonatkoznak. Ezek az oszlopdefiníció végén adhatók meg.

```
[CONSTRAINT név] megszorítás [megszorítás]...
```

Táblaszintű megszorítások: Több oszlopra is vonatkozhatnak. A `CREATE TABLE` utasításban az oszlopdefiníciók után kell állniuk.

```
[CONSTRAINT név] megszorítás ( oszlopnév [, oszlopnév] … )
```

7.2 Megszorítások

7.2.1 Elsődleges kulcs megszorítás (PRIMARY KEY)
Minden táblának egy elsődleges kulcsa lehet, azonban ez lehet összetett (azaz több attribútumból álló) is. Az elsődleges kulcsot alkotó attribútumok között nem lehet olyan, amelynek az értéke NULL és nem szerepelhet két olyan sor a táblában, amelyek esetében az elsődleges kulcsot alkotó attribútumok értéke rendre megegyezik.
Összetett kulcs esetében egy-egy attribútumot tekintve lehet azonosság, de az összes attribútumot tekintve a kulcsnak egyedinek kell lennie.

7.2.2 Egyediség megszorítás (UNIQUE)
Az értékeknek egyedieknek kell lenniük, de NULL érték is szerepelhet. Több ilyen megszorítás is szerepelhet egy táblára vonatkozóan.

7.2.3 NOT NULL megszorítás
A NOT NULL megszorítás tiltja, hogy egy oszlopban NULL érték szerepeljen. A táblaszintű megszorítások között nem szerepelhet.

7.2.4 Külső kulcs megszorítás
Két tábla közti kapcsolat szabályozására szolgál. Biztosítja, hogy a külső kulcsot tartalmazó tábla külső kulcsot alkotó attribútumaiban csak NULL érték vagy olyan értékek állhatnak, mely a hivatkozott tábla (amelyhez ez a tábla kapcsolódik) hivatkozott oszlopaiban ténylegesen szerepel.

Állhat oszlop után írva oszlopmegszorításként:
```
[CONSTRAINT megszorításnév] REFERENCES hivatkozott_tábla (oszlopnév)
```

Vagy állhat az oszlopdefiníciók után táblaszintű megszorításként:
```
[CONSTRAINT megszorításnév] FOREIGN KEY (oszlopnév [, oszlopnév]...) REFERENCES
hivatkozott_tábla (oszlopnév [, oszlopnév]...)
```

7.2.5 Általános megszorítás
Ezzel biztosítható, hogy egy adott oszlopba csak egy feltételnek eleget tevő értékek kerülhessenek be. A feltétel több oszlopra is vonatkozhat, ekkor táblaszintű megszorításként kell megadni.
```
[CONSTRAINT megszorításnév] CHECK (feltétel)
```

7.3 Táblák módosítása

Oszlop hozzáadásának általános alakja:
```
ALTER TABLE táblanév
ADD oszlopdefiníció [, oszlopdefiníció] … ;
```

Oszlop módosításának általános alakja:
```
ALTER TABLE táblanév
MODIFY oszlopnév új_típus új_megszorítás [, oszlopnév új_típus új_megszorítás]…;
ALTER TABLE táblanév
RENAME COLUMN oszlopnév TO új_oszlopnév;
```

Oszlop törlésének általános alakja:
```
ALTER TABLE táblanév
DROP COLUMN oszlopnév;
```

7.4 Táblák törlése
Egy tábla törlésének általános alakja:
```
DROP TABLE táblanév;
```

Egy tábla teljes tartalmának végleges törlése, a tábla törlése nélkül:
```
TRUNCATE TABLE táblanév;
```

8. Data Manipulation Language (DML)

Az SQL adatmanipulációs (DML) utasításai segítségével tölthető fel és módosítható egy tábla tartalma.
`INSERT`: sorok beszúrása a táblába
`UPDATE`: sorok tartalmának frissítése
`DELETE`: sorok törlése

Az adatmanipulációs utasítások eredménye nem kerül automatikusan rögzítésre az adatbázisban.
`COMMIT`: tranzakciók véglegesítése
`ROLLBACK`: tranzakciók visszagörgetés

8.1 Az INSERT, UPDATE és DELETE utasítások

`INSERT`: sorok beszúrását teszi lehetővé egy táblába. Általános alak:
```
INSERT INTO táblanév[(mezőnév1, mezőnév2, … )] VALUES(érték1, érték2, … )
```

`UPDATE`: sorok frissítését teszi lehetővé egy táblában. Általános alak:
```
UPDATE táblanév SET mezőnév = érték WHERE feltétel
```

`DELETE`: sorok törlését teszi lehetővé egy táblából. Általános alak:
```
DELETE FROM táblanév [WHERE feltétel]
```
Ha nincs megadva feltétel, az összes sort törli a táblából!

9. SQL halmazoperátorok

Az SQL halmazoperátorai lehetővé teszik kettő vagy több lekérdezés által adott eredményhalmazok egy eredményhalmazba történő kombinálását.

Halmazoperátorok (Oracle SQL):
`UNION [ALL]`: unióképzés
`INTERSECT`: metszetképzés
`MINUS`: különbségképzés

A UNION, INTERSECT és MINUS operátorok rendre két vagy több lekérdezés eredményhalmazának unióját, metszetét és különbségét állítják elő. Az összekapcsolt lekérdezéseknek ugyanannyi és páronként kompatibilis típusú oszlopot kell visszaadnia. Az megfeleltetett oszlopoknak egyező sorrendben kell szerepelniük.
Fontos: Azon sorok közül, amelyek többször is szerepelnek csak egyet tartanak meg és nem garantálják az eredmény sorainak sorrendjét!
Unióképzés esetében az összes sor megtartása a UNION ALL kulcsszavak használatával van lehetőség.

Szintaxis:
```
SELECT oszlopnév1 [, oszlopnév2] … FROM táblanév1
[ UNION [ALL] | INTERSECT | MINUS ]
SELECT oszlopnév3 [, oszlopnév4] … FROM táblanév2
[ [ UNION [ALL] | INTERSECT | MINUS ]
SELECT oszlopnév5 [, oszlopnév6] … FROM táblanév3] …
```