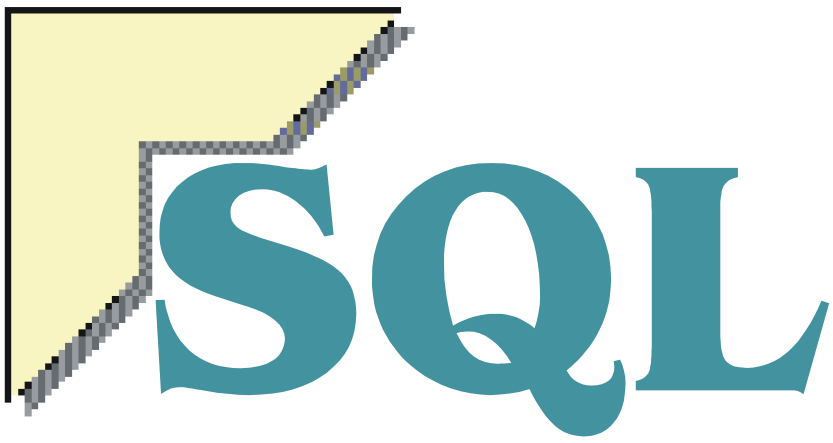

BUJDOSÓ GYÖNGYI

Bevezetés az SQL-be

OKTATÁSI SEGÉDANYAG AZ
ADATBÁZISKEZELÉS CÍMŰ GYAKORLATHOZ

DEBRECENI EGYETEM ◇ INFORMATIKAI KAR

Tartalomjegyzék

I. Adatbázisok

Repülőtér adatbázisa

A jaratok tábla szerkezete

Név	Null?	Típus
SZAM	NOT NULL	NUMBER(3)
INDULAS	NOT NULL	NUMBER(4)
ERKEZES	NOT NULL	NUMBER(4)
HONKOD	NOT NULL	CHAR(3)
HOVKOD	NOT NULL	CHAR(3)
NAPOK		CHAR(7)

A jaratok tábla tartalma

SZAM	INDULAS	ERKEZES	HON	HOV	NAPOK
123	1152	1436	001	002	hksc-sv
111	1023	1145	002	001	hkscp-v
112	1535	1643	001	003	hkscpsv
113	2032	2115	003	001	hks-psv
114	1322	1423	003	004	-ks-p—
115	1415	1510	001	003	hkscpsv

A repter tábla szerkezete

Név	Null?	Típus
KOD		CHAR(3)
VAROS		CHAR(25)
CHECKIN		NUMBER(1)
ILLETEK		NUMBER(5)

A repter tábla tartalma

KOD	VAROS	CHECKIN	ILLETEK
001	Budapest, Ferihegy 1	1	5000
002	Praha	2	5000
003	Warszawa Internacionalna	1	8000
004	Paris Orly	2	10000
005	London Heathrow		12000

A szamlak tábla szerkezete

Név	Null?	Típus
RKOD	NOT NULL	CHAR(3)
OSSZEG	NOT NULL	NUMBER(5)
KELT		DATE
SZLA		CHAR(10)
ROGZ	NOT NULL	CHAR(30)

A szamlak tábla tartalma

RKO	OSSZEG	KELT	ROGZ	SZLA
001	10000	99-OKT-08	kovacs	99/M182
003	30000	99-OKT-08	NAGY	99/R436
001	11328	99-DEC-28	Virag	99/M183
002	23560	00-JAN-13	Lakatos A.	00/M182
002	2356	00-FEB-09	Virag	00/M011
002	48000	00-SZE-11	Hegyi	00/M032
003	42000	01-JÚN-12	Magyar Bela	01/R136
004	9364	01-JÚL-30	Kiss Edit	01/R658
004	12000	01-AUG-28	Tóth Ádám	01/R200

Ezen kézikönyv mintái a fenti adatbázisnak a felhasználásával mutatják be a parancsok, kulcsszavak, kulcskifejezések működését.

Például:

---

1) Tábla létrehozása – CREATE TABLE ← A parancs leírása

```
SQL> CREATE TABLE szamlak
  2  (rkod CHAR(3) PRIMARY KEY,
  3  osszeg NUMBER(5) NOT NULL,
  4  kelt DATE NOT NULL
  5  szla CHAR(20));
```

← Minta a parancs kiadására, működésére. (Az „SQL>” prompt és a sorok sorszáma nem gépelendő be!)

A tábla létrejött. ← A parancs lefuttatása utáni üzenet.

A következő oldalon lévő, a mesehősök kis boltjának adatait felhasználó tábla használható gyakorlásra, és elérhető a http://oracle.inf.unideb.hu/isqlplus/ felületről.

Mesehős adatbázis

Reláció	*Attribútumok*	*Megjegyzés*
MESEHOS	AZON, NEV, CIM, EGYENLEG	megadja a mesehősök azonosítóját, nevét, címét és egyenlegét
RENDELES	RENDSZAM, MAZON, DATUM	megadja, hogy az adott rendelési számú rendelést mikor adta fel az adott azonosítójú mesehős
TARTALMAZ	RENDSZAM, CIKKNEV, MENNYISEG	megadja, hogy az adott rendelési számon az adott cikkből mekkora mennyiségben rendeltek
CIKK	CIKKNEV, EGYSEGAR	megadja, hogy az adott cikknek mekkora az egységára

SQL> desc **mesehos**;

```
 Név                            Null?    Típus
 ------------------------------ -------- --------
 AZON                           NOT NULL CHAR(3)
 NEV                            NOT NULL CHAR(15)
 CIM                                     CHAR(20)
 EGYENLEG                                NUMBER
```

SQL> desc **cikk**

```
 Név                  Null?    Típus
 -------------------- -------- --------
 CIKKNEV              NOT NULL CHAR(12)
 EGYSEGAR                      NUMBER
```

SQL> select * from **mesehos**;

```
AZO NEV             CIM                    EGYENLEG
--- --------------- -------------------- ---------
M01 Vasorru Baba    Mezeskalacs Haz          2500
M02 Vadasz          Vadaszles                   0
M03 Micimacko       Szazholdas Pagony       -1000
M04 Maci Laci       Hollywood                5030
M05 Torpilla        Gombahaz
M06 Hetfeju sarkany Sarkanybarlang            700
M07 A harom nyul    Nyuszihaz               -3400
M08 Okoska          Gombahaz
M09 Piroska         Erdoszeli haziko         6700
M10 Tapsi Hapsi     Hollywood                9920
M11 Susu            Sarkanybarlang           1130
M12 Vuk             Rokalyuk                 -250
M13 Torpapa         Gombahaz                  400
```

SQL> select * from **cikk**;

```
CIKKNEV       EGYSEGAR
------------ ---------
repa                20
magnespor           20
cipo               300
puskagolyo          50
mezesbodon          60
kalacs              90
sutolapat          150
kosar              200
szemfestek         150
puska              300
smirgli             20
salata             100
rokacsapda         300
pipihus            200
eros paprika        20
eleszto             15
```

SQL> desc **tartalmaz**

```
 Név                            Null?    Típus
 ------------------------------ -------- --------
 RENDSZAM                       NOT NULL NUMBER
 CIKKNEV                        NOT NULL CHAR(12)
 MENNYISEG                               NUMBER
```

SQL> desc **rendeles**

```
 Név                  Null?    Típus
 -------------------- -------- -------
 RENDSZAM             NOT NULL NUMBER
 MAZON                         CHAR(3)
 DATUM                         DATE
```

SQL> select * from **tartalmaz**;

```
 RENDSZAM CIKKNEV      MENNYISEG
--------- ------------ ---------
        6 cipo                 6
       13 szemfestek          12
       16 rokacsapda           3
       17 repa               200
       22 magnespor          319
       23 szemfestek           4
       23 salata               6
       25 puska                2
       34 cipo                 2
       37 smirgli             17
       37 mezesbodon           3
       37 eleszto             20
       46 szemfestek           8
       49 puskagolyo          56
       53 mezesbodon           5
       61 kalacs               1
       66 salata              12
       78 sutolapat            1
       80 cipo                 2
       97 kosar                1
```

SQL> select * from **rendeles**;

```
 RENDSZAM MAZ DATUM
--------- --- ---------
        6 M07 94-OKT-07
       13 M05 93-JAN-01
       16 M02 94-MÁR-15
       17 M07 92-JAN-21
       22 M01 94-AUG-25
       23 M02 96-OKT-12
       25 M02 97-FEB-28
       34 M05 95-DEC-03
       37 M01 00-JAN-12
       46 M01 96-JÚL-21
       49 M02 98-MÁR-05
       53 M03 01-JÚN-02
       61 M09 03-ÁPR-05
       66 M07 93-OKT-30
       78 M01 95-JÚL-02
       80 M02 94-NOV-18
       97 M09 96-OKT-20
```

II. Adatdefiníciós nyelv (DDL)

A lekérdező nyelv adatdefiníciós nyelvének (Data Definition Language, DDL) szokták nevezni azon parancsok összességét, amelyek táblák és egyéb objektumok létrehozását, módosítását és törlését teszik lehetővé.

Ezen kézikönyvben az alábbiakat tárgyaljuk:

II.1. Adattípusok

II.1.1.1.1. Táblanév, oszlopnév

Minden Oracle adatbázisban alkalmazhatók azok a tábla- és oszlopnevek, amelyeket az alábbi szabályok figyelembevételével adunk meg:

- max. 30 karakter
- használható karakterek:
  - → Ha aposztrófok közé zárt, bármilyen ASCII karakter lehet
  - → Ha ettől eltér, az alábbiak használhatók:
    - angol acb kis- és nagybetűi,
    - számjegyek,
    - $ _ #
- betűvel kezdődik,
- kis- és nagybetűk nem megkülönböztetettek, kivéve, ha aposztrófok között szerepelnek,
- a táblanevek az adatbázisban, az oszlopnevek a táblában egyediek.

II.1.1.1.2. Néhány adattípus

CHAR ASCII/EBCDIC, max 240 karakter változó hosszal tárolódnak [pl. CHAR (6)]	VARCHAR, LONG VARCHAR
NUMBER 0–9, előjel, tizedes pont, max. 42 karakterből állhat [pl. NUMBER (3) vagy NUMBER (5,2)]	DECIMAL, INTEGER, SMALLINT, FLOAT
DATE Beviteli és kiírási formátuma beállításfüggő. Beviteli forma itt: 'ÉÉ-HHH-NN' [pl. '99-Okt-2' vagy '2001-Jún-18']	
LONG ASCII/EBCDIC, max 65 536 1 táblában max. 1 LONG lehet. Nem használható: – függvények argumentumában, – kifejezésekben, – a SELECT – WHERE, GROUP BY, CONNECT BY, ORDER BY, DISTINCT opcióival,	
RAW ASCII/EBCDIC + grafikák, bitképek, stb.	
NULL Ha egy adott sor adott oszlopába még nem történt adatbevitel, vagy töröltek adatott, akkor ott a NULL adat szerepel.	LONG RAW

II.1.1.1.3. NULL vagy NOT NULL

NULL: a mező egy adott sor felvételénél maradhat üres (alapértelmezés)
NOT NULL: a mezőnek az adott sor felvételénél kötelező értéket adni.

II.2. Táblák létrehozása, módosítása, törlése

CREATE TABLE, ALTER TABLE, DROP TABLE

1) Tábla létrehozása – CREATE TABLE

```
SQL> CREATE TABLE szamlak
  2  (rkod CHAR(3) PRIMARY KEY,
  3  osszeg NUMBER(5) NOT NULL,
  4  kelt DATE NOT NULL
  5  szla CHAR(20));
```

A tábla létrejött.

2) Módosítás – új oszlop hozzáadása: ALTER TABLE …ADD…

```
SQL> ALTER TABLE szamlak
  2  ADD rogz CHAR(30);
```

A tábla módosítva.

Az új oszlopban a már létező sorok NULL értéket kapnak, tehát így nem adható meg a NOT NULL, kötelező kitöltést beállító opció.

3) Módostás – létező oszlop egyes adatainak módosítása:
ALTER TABLE …MODIFY…

```
SQL> ALTER TABLE szamlak
  2  MODIFY rogz CHAR(30) NOT NULL;
```

A tábla módosítva.

```
SQL> desc szamlak
```

Név	Null?	Típus
RKOD	NOT NULL	CHAR(3)
OSSZEG	NOT NULL	NUMBER(5)
KELT		DATE
SZLA		CHAR(10)
ROGZ	NOT NULL	CHAR(30)

4) Tábla törlése – DROP TABLE

```
SQL> CREATE TABLE eldobando
  2  (egyetlen date);
```

A tábla létrejött.

```
SQL> DROP TABLE eldobando;
```

A tábla eldobva.

II.3. Nézettáblák létrehozása és törlése

```
CREATE VIEW nézettábla_név [alias_név1 alias_név2…]
        AS szelekciós_utasítás;
DROP VIEW nézettábla_név;
```

Ha valamilyen táblán/táblákon nagyon gyakran akarunk valamilyen bonyolult, speciális lekérdezést alkalmazni, akkor érdemes egy olyan nézettáblát létrehozni, amely az illető tábla/táblák szükséges feltételeknek eleget tevő részét mutatja.

Sok szempontból ugyanúgy viselkednek, mint a táblák.

A nézettáblák definíciója mindig valamilyen másik nézettábla és/vagy valódi tábla adatainak szelektív megjelenítésére szolgál.

Az AS utáni SELECT-re egyetlen kikötés: nem tartalmazhat rendezést (ORDER BY).

A nézettábla fizikailag nem tárolódik. Ha valamilyen műveletet akarunk rajta végrehajtani, akkor előbb kiértékeli az őt létrehozó feltételeket, majd az így kapott adatokon végrehajtódik a művelet.

A nézettábla folyamatosan frissítődik.

Ha a nézettábla pontosan egy valódi táblán van meghatározva és a szelekciós része nem tartalmaz sem GROUP BY sem DISTINCT kulcsszót, bizonyos feltételek mellett az adatokon végrehajtható módosítás, illetve törlés (lásd ORACLE kézikönyv).

a) Készítsünk most egy olyan nézettáblát, amely a Budapestről induló járatok számát, indulási idejét, és cél városának a nevét jeleníti meg!

```
SQL> CREATE VIEW Budapest AS
  2  SELECT szam jarat, indulas, varos hova
  3  FROM jaratok, repter
  4  WHERE honkod='001' AND hovkod=kod;
```

A nézet létrejött.

b) Töröljük a nézettáblát!

```
SQL> DROP VIEW Budapest
```

III. Adatmanipulációs nyelv (DML)

Az adatmanipulációs nyelvbe (Data Manipulation Language, DML) szokták egyesek sorolni azon parancsokat, amelyek az adatok bevitelét, módosítását és törlését teszik lehetővé.

Tárgyalásra kerülnek itt az

INSERT INTO
UPDATE
DELETE

parancsok.

Először a parancsok legegyszerűbb használati módjait tekintjük át:

1) Sor létrehozása és teljes feltöltése

```
SQL> INSERT INTO szamlak
  2  VALUES  ( '001',  10000,     '99-okt-08',     'kovacs',     '99/M182' );
1 sor létrejött.
```

2) Sor létrehozása és részleges feltöltése

```
SQL> INSERT INTO szamlak (rkod, osszeg, rogz, szla)
  2  VALUES  ( '002',  30000,                    'NAGY',     '99/R436' );
1 sor létrejött.
```

3) Már létrehozott sor egy vagy több adatának módosítása

```
SQL> UPDATE szamlak
  2  SET kelt='99-Okt-08', rkod = '003'
  3  WHERE  szla = '99/R436';
1 sor módosítva.
```

4) Létező adat törlése, azaz helyettesítése a NULL értékkel

```
SQL> UPDATE szamlak
  2  SET kelt=NULL
  3  WHERE  szla = '99/R436';
```

5) Teljes sor törlése

```
SQL> DELETE FROM szamlak  WHERE rkod='003';
```

Bonyolultabb formák:

Az INSERT INTO utasítás után a VALUES… helyett SELECT is állhat, amellyel valamely másik táblából másolhatunk át adatokat.
Például a ha létrehozunk a egy masrepter nevű táblát:

```
SQL> CREATE TABLE masrepter (rep CHAR(3));
```

oszloppal, amelybe át szeretnénk másolni az összes reptér kódját, azt megtehetjük az

```
SQL> INSERT INTO masrepter SELECT kod FROM repter;
```

az eredmény 5 új rekord lesz az új táblában:

```
SQL> SELECT * FROM masrepter;

REP
---
001
002
003
004
005
```

Ezzel a módszerrel teljes táblatartalmak is létrehozhatók.

IV. QUERY

A lekérdezésnek, vagyis a QUERY-nek gyakorlatilag egyetlen parancsa van, a SELECT*, amelyhez több kulcsszót és kulcskifejezést alkalmazhatunk a keresés szűkítéséhez.

A témakör itt tárgyalt részei:

* Más nézőpontot tekintve azonban a SELECT a DML (Data Manipulation Language) része.

IV.1. Projekció

SELECT [DISTINCT] oszlopnév [alias_név], oszlopnév [alias_név],…
FROM táblanév;

```
SQL> SELECT * FROM jaratok;

SZAM INDULAS   ERKEZES   HON    HOV   NAPOK
--------- --------- --------- --- --- -------
123          1152   1436          001      002     hksc-sv
111          1023   1145          002      001     hkscp-v
112          1535   1643          001      003     hkscpsv
113          2032   2115          003      001     hks-psv
114          1322   1423          003      004     -ks-p---
```

```
SQL> SELECT szam, honkod Bol, hovkod Ba
  2  FROM jaratok;

    SZAM       BOL   BA
---------      -----   -----
       123      001   002
       111      002   001
       112      001   003
       113      003   001
       114      003   004
```

```
SQL> SELECT
  2  DISTINCT honkod bol
  3  FROM jaratok;

      BOL
      -----
      001
      002
      003
```

IV.2. Szelekció:

```
SELECT [DISTINCT] oszlopnév [alias_név], oszlopnév [alias_név],…
    FROM táblanév
        WHERE keresési feltétel;
```

Keresési feltételek:

1) Egyszerű összehasonlítás

- oszlopnév összehasonlító operátor konstans
- oszlopnév összehasonlító operátor oszlopkifejezés

Összehasonlító operátorok:
= | < | <= | > | >= |
!= | <> | ^=

Oszlopkifejezés:
- oszlopnév
- oszlopokra alkalmazható beépített függvények, aritmetikai vagy karakterműveletek

Konstans:
- CHAR, NUMBER vagy DATE típus
- CHAR, DATE: aposztrófok között
- CHAR esetén Case sensitive

2) Összehasonlítás egy halmaz elemeivel

oszlopnév halmazösszehasonlító_operátor halmazdefiníció

Beépített halmazokat összehasonlító operátorok (logiakai típusú vagy predikátum függvények):
BETWEEN érték AND érték (lehet numerikus, date, karakter)
IN (lista)
LIKE karakterminta (benne _ %)

3) Összehasonlítás a NULL értékkel

Az IS NULL operátor azt a feltételt adja meg, hogy az az oszlop, amelynek neve az operátor előtt áll, a NULL értéket tartalmazza. Használata:

oszlopnév IS NULL

(lásd pl. a *d)* feladatot).

4) Összetett keresési feltételek

Logikai operátorok: NOT, AND, OR, IN, NOT IN
Precedenciájuk csökkenő sorrendben:

1. relációjelek: = , != stb.
2. NOT
3. AND
4. OR

A végrehajtási sorrend megváltoztatható a kerek zárójelek használatával.

a) Keressük azon reptereket, ahol az illeték 8000 Ft-tól különbözik:

```
SQL> SELECT varos, illetek
  2  FROM repter
  3  WHERE illetek != 8000;

VAROS                     ILLETEK
------------------------  ----------
Budapest, Ferihegy 1          5000
Praha                         5000
Paris Orly                   10000
London Heathrow              12000
```

b) Szeretnénk tudni, hányas járatok hová indulnak Budapestről.

```
SQL>  SELECT szam jarat, honkod bol
  2  FROM jaratok
  3  WHERE hovkod='001';

   JARAT   BOL
  ---------  -----
     111      002
     113      003
```

c) Kilistázzuk a hétfői járatokat.

```
SQL> SELECT szam, honkod bol, hovkod ba
  2  FROM jaratok
  3  WHERE napok LIKE 'h%';

  SZAM BOL BA
--------- ----- -----
     123 001 002
     111 002 001
     112 001 003
     113 003 001
     115 001 003
```

d) Mely repülőterekre vonatkozóan nem tudjuk a bejelentkezési időt?

```
SQL> SELECT varos
  2  FROM repter
  3  WHERE checkin IS NULL;

VAROS
------------------------
London Heathrow
```

e) Listázzuk ki, hogy a 112, 113, 114, 115-ös járatok honnan hová mennek!

```
SQL> SELECT szam, honkod bol, hovkod ba
  2   FROM jaratok
  3   WHERE szam BETWEEN '112' AND '115';
```

```
    SZAM BOL BA
---------- ----- ----
       112   001   003
       113   003   001
       114   003   004
       115   001   003
```

f) Budapestről Varsóba menő járatot keresünk keddi 11:00 és 16:00 óra közötti érkezéssel.

```
SQL> SELECT szam jarat, indulas
  2  FROM jaratok
  3  WHERE honkod='001'
  4    AND hovkod='003'
  5    AND napok LIKE '__s%'
  6    AND erkezes BETWEEN 1100 AND 1600;

   JARAT   INDULAS
  --------- -------------
      115         1415
```

g) Kilistázzuk a Budapestre Varsóból és Paris Orly-ról érkező járatokat.

```
SQL> SELECT szam jarat, erkezes
  2  FROM jaratok
  3  WHERE hovkod='001'
  4     AND honkod IN ('003','004');

   JARAT   ERKEZES
   --------- -------------
      113         2115
```

h) Kilistázzuk a Budapestre a nem Varsóból és nem Paris Orly-ról érkező járatokat.

```
SQL> SELECT szam jarat, erkezes
  2  FROM jaratok
  3  WHERE hovkod='001'
  4      AND honkod  NOT IN ('003','004');

   JARAT   ERKEZES
   --------- -------------
      111         1145
```

IV.3. Kiválasztott sorok rendezése

```
SELECT oszlopnév,…
        FROM táblanév
                [WHERE feltétel]
                        ORDER BY oszlopnév [DESC], oszlopnév [DESC],…
```

A feltételnek megfelelő sorok rendezése 1 vagy több oszlop szerint alapértelmezésben növekvő, a DESC opcióval kiegészítve csökkenő sorrendbe.

A rendezés a rendezéshez felsorolt oszlopok sorrendje szerint történik.

Ha a rendezendő oszlopokban NULL értékek is szerepelnek, a NULL értéket tartalmazó sorok nem vesznek részt a rendezésben, hanem a rendezett lista előtt jelennek meg.

a) Rendezzük a pénteki járatokat elsődlegesen a HOVKOD szerint növekvő, másodlagosan a HONKOD szerint csökkenő, s még ezen belül a járat száma (SZAM) szerint növekvő sorrendbe!

```
SQL> SELECT szam, hovkod, honkod, indulas
  2  FROM jaratok
  3    WHERE napok LIKE '%p%'
  4    ORDER BY hovkod, honkod DESC, szam;

    SZAM HOV HON   INDULAS
---------- ------ ------   ----------
     113   001   003        2032
     111   001   002        1023
     112   003   001        1535
     115   003   001        1415
     114   004   003        1322
```

b) Rendezzük a repülőtereket a bejelentkezési idő szerint csökkenő, azon belül a repülőtér kódja szerint növekvő sorrendbe!

```
SQL> SELECT * FROM repter
  2  ORDER BY checkin DESC, kod;

KOD VAROS                      CHECKIN  ILLETEK
------ ------------------------ ---------  ---------
005 London Heathrow                        12000
002 Praha                              2     5000
004 Paris Orly                         2    10000
001 Budapest, Ferihegy 1               1     5000
003 Warszawa Internacionalna           1     8000
```

IV.4. Csoportok képzése

```
SELECT oszlopkifejezések
    FROM táblanév
        [WHERE keresési feltétel]
            GROUP BY oszlopnév_gb
                [HAVING csoportkiválasztási feltétel]
```

1) Csoportokra alkalmazható (aggregáló/aggregátor) függvények

Fajtái: AVG, SUM, MIN, MAX, COUNT

Szintakszis: függvénynév ([DISTINCT|ALL] kifejezés)
(Az alapértelmezett az ALL)

SUM, AVG – numerikus adatokra.
MIN, MAX, COUNT – numerikus, karakter és dátum típusú adatokra.

A *kifejezés* által meghatározott oszlopba eső csoportok értékeire vonatkoznak.

Jelentésük:
AVG([DISTINCT|ALL] *kifejezés*) – Átlagot ad. A NULL értékeket figyelmen kívül hagyja.
SUM([DISTINCT|ALL] *kifejezés*) – Összeget ad. A NULL értékeket figyelmen kívül hagyja.
MIN([DISTINCT|ALL] *kifejezés*) – A legkisebb értéket adja vissza.
MAX([DISTINCT|ALL] *kifejezés*) – A legnagyobb értéket adja vissza.
COUNT([DISTINCT|ALL] *|kifejezés*) – Adatok számát adja vissza (* esetén az összesét), a NULL értékeket vigyelmen kívül hagyva.

Általában nem alkalmazhatók együtt az egyszerű lekérdezésekkel.

2) Csoportok képzése

Az eredményül kapott sorok csoportosítása.

GROUP BY *oszlopnév_gb*:

A csoportosítás ezen oszlop/oszlopok azonos értékei alapján történik, majd ezután valamely másik oszlop csoportosított soraiba tartozó értékeit összegezhetjük, rendezhetjük, átlagolhatjuk stb.

HAVING *csoportkiválasztási feltétel*:

A HAVING-et akkor használjuk, ha nem az összes, hanem csak bizonyos GROUP BY által kiválasztott csoportokra van szükségünk.

A GROUP BY által kialakított csoportokból kiválasztja azon sorokat, amelyek eleget tesznek az őt követő csoportkiválasztási feltételnek. Ha GROUP BY nincsen megadva, akkor a tábla egésze egy csoportnak tekinthető.

Megjegyzések:

1. Ha a feltétel összetett, akkor minden elemi feltételnek aggregáló függvényeket kell tartalmaznia.

2. A feltételben szereplő aggregáló függvényeknek a SELECT oszlopnév-listájában is szerepelnie kell.

a) Nézzük meg, hogy az azonos bejelentkezési idővel rendelkező repülőtereknél mennyi az illetékek összege és átlaga, illetve hogy mennyi a minimális és a maximális illeték!

```
SQL> SELECT checkin,  SUM (illetek),  AVG ( illetek ) , MIN(illetek), MAX(illetek)
  2   FROM repter
  3   GROUP BY checkin;
```

CHECKIN	SUM(ILLETEK)	AVG(ILLETEK)	MIN(ILLETEK)	MAX(ILLETEK)
1	13000	6500	5000	8000
2	15000	7500	5000	10000
	12000	12000	12000	12000

b) Hány járat indul az egyes repülőterekről? És mikor indul a legkorábbi?

```
SQL> SELECT honkod, COUNT(honkod), min(indulas)
  2  FROM jaratok
  3  GROUP BY honkod;
```

HON	COUNT(HONKOD)	MIN(INDULAS)
001	3	1152
002	1	1023
003	2	1322

c) Nézzük meg, hogy az azonos bejelentkezési idővel rendelkező repülőterek illetékének mennyi az átlaga, de csak azokban az esetekben, ahol az nem haladja meg a 7000 Ft-ot!

```
SQL> SELECT checkin,  AVG ( illetek )
  2  FROM repter
  3   GROUP BY checkin
  4      HAVING AVG ( illetek ) <= 7000;
```

CHECKIN	AVG(ILLETEK)
1	6500

IV.5. Származtatott adatok – függvények és műveletek

A származtatott adatok az adatbázisban lévő valódi adatokból számíthatók ki különböző beépített függvények és műveletek felhasználásával. Az eredmények nem tárolódnak, így a kifejezések, függvények minden egyes használatkor újra kiértékelődnek.

A függvény egy előre meghatározott művelet, amely egy SQL utasításban nevének és aktuális paramétereinek megadásával hívható meg. Minden meghívásakor kiértékelődik, és mindig értéket ad vissza.

Felhasználható

- adatokkal végzett számításokra
- egyedi adatelemek módosítására
- sorok csoportján végzett műveletekre
- megjelenítendő számok, dátumok, sztringek formázására
- oszlopok adattípusának konvertálására.

1) Aritmetikai függvények és műveletek

Megjeleníthetők a SELECT-ben, felhasználhatók annak WHERE és ORDER BY részében.

Néhány függvény:

Paramétere: *szám*
Eredménye: *szám*

ABS({oszlop | kifejezés}) – Az oszlop, a kifejezés vagy a kifejezés által meghatározott oszlop soraiban lévő értékek abszolút értékét adja Vissza.

GREATEST(kifejezés, kifejezés,…) – A legnagyobb értéket adja vissza.

LEAST(kifejezés, kifejezés,…) – A legkisebbet adja vissza.

POWER(kifejezés, kitevő) – Hatványozás.

ROUND({oszlop|kifejezés}, tizedes jegyek száma) – Kerekítés a megadott számú tizedesre.

TRUNC({oszlop|kifejezés}, tizedes jegyek száma) – A megadott számú tizedes meghagyása.

Műveletek: + – * / Precedenciájuk a szokásos.

Megjegyzés:

A NULL ≠ 0, így a legtöbb függvény nem tudja értelmezni. Megoldás: ld. az NVL függvényt.

2) Karakterkezelő függvények

Megjeleníthetők a SELECT-ben, felhasználhatók annak WHERE és ORDER BY részében.

Néhány függvény:

Paramétere: *karakter vagy karaktersorozat*
Eredménye: *karakter vagy szám*

INITCAP({oszlop | kifejezés}) – KOVACS vagy kOVAcs → Kovacs

LOWER({oszlop | kifejezés}) – KOVACS vagy kOVAcs → kovacs

UPPER({oszlop | kifejezés}) – kovacs vagy kOVAcs → KOVACS

LENGTH({oszlop | kifejezés}) – értékek hossza

CONCAT({oszlop1 | kifejezés1}, {oszlop2 | kifejezés2}) – a két megadott karakterláncot összefűzi. Azonos a (||) operátorral.

INSTR(kifejezés, 'keresendő sztring') – az első előfordulás pozíciószámát adja vissza

SUBSTR(kifejezés, kezdő_pozíció, hossz) – A *kezdő_pozíció*-tól kezdődően a *hossz* hosszúságú sztringet adja vissza

RPAD({oszlop | kifejezés}, *n*, *'kitöltő'*) – az oszlop értékeit *n* karakteren balra igazítva jeleníti meg úgy, hogy a jobb oldalon fennmaradó „üres" helyet a *kitöltő* karakterrel tölti fel.

LPAD({oszlop | kifejezés}, *n*, 'kitöltő') – az oszlop értékeit *n* karakteren jobbra igazítva jeleníti meg úgy, hogy a bal oldalon fennmaradó „üres" helyet a *kitöltő* karakterrel tölti fel.

3) Dátum típusú adatok

A teljes dátum az évszázad, év, hónap, nap, óra, perc, másodperc részekből áll.

Megjelenítése (így begépelése is) változatos, a rendszer beállításaitól függ. Az Oracle magyar nyelvű verziójában az alapértelmezett megjelenítési és beviteli forma:

04-MÁR-18

ami megfelel az alapértelmezett YY-MON-DD Magyar nyelvű formátumnak

A SYSDATE a rendszerdátumot eredményezi.

Értékadásnál pl.:

```
SQL>UPDATE szamlak
  2  SET kelt=sysdate
  3  WHERE szla='01/R200';
```

Lekérdezése:

```
SQL> SELECT  sysdate FROM  dual;
```

Műveletek dátum típusú adatokkal

Dátum + szám	→ dátum	a szám a hozzáadandó napok száma
Dátum – szám	→ dátum	a szám a kivonandó napok száma
Dátum – dátum	→ napok száma	két dátum különbsége napokban

Néhány függvény:

ADD_MONTHS(dátum, hozzáadandó_hónapok_száma) – hozzáadás.

GREATEST(dátum1, dátum2) – a nagyobbikat adja vissza.

LEAST(dátum1, dátum2) – a kisebbiket adja vissza.

MONTHS_BETWEEN(dátum1, dátum2) – a két dátum különbsége hónapokban. Eredménye egy szám.

NEX_DAY(dátum, 'nap') – meghatározza a megadott dátum utáni 'nap' nevű nap dátumát.

LAST_DAY(dátum) – meghatározza a megadott dátum által meghatározott hónap utolsó napjának dátumát.

ROUND(dátum, 'formátum') – a legközelebbi egész napra kerekíti.

TRUNC(dátum, 'formátum') – a befejezett egész napot adja vissza.

4) Konverziós függvények

A teljes dátum az évszázad, év, hónap, nap, óra, perc, másodperc részekből áll.

TO_CHAR({szám | 'dátum'} ['formátummaszk']) – konverziós függvény:
a szám és a dátum típusú adatot → karakteres típusú adattá konvertálja

Értékadásnál pl.:

```
SQL>SELECT TO_CHAR(datum,'YYYY. month fmDD.') datumok
  2  FROM rendeles;
```

TO_NUMBER('karakterlánc' ['formátummaszk']) – konverziós függvény:
a számot tartalmazó karakteres típusú adatot → számmá konvertálja

TO_DATE('karakter' [,'formátummaszk']) – konverziós függvény:

a karakteres típusú adatot → dátum típusú adattá konvertálja

Értékadásnál pl.:

```
SQL>UPDATE szamlak
  2  SET kelt= TO_DATE('00/09/11','YY/MM/DD')
  3  WHERE szla = '00/M032';
```

Formátummaszk elemei:

YY	évszám két karakteren (alapért.)
YYYY	teljes évszám
YEAR	teljes évszám betűkkel
MM	a hónap neve két számjeggyel
MON	a hónap nevének első három karaktere nagybetűkkel (alapért.)
mon	a hónap nevének első három karaktere kisbetűkkel
MONTH	a hónap teljes nev
W	a hét sorszáma a hónapban
WW	a hét sorszáma az évben
D	a nap sorszáma a héten
DD	a nap sorszáma a hónapban
DDD	a nap sorszáma az évben
DY	a hét napjának hárombetűs rövidítése
DAY	a hét napjának teljes neve
fm	leszedi az egyébként megjelenítendő nullákat és kitöltő szóközöket a számok elől

Műveletek dátumokkal:

Összeadhatók, kivonhatók egymásból, hozzáadhatók, kivonhatók napok.

Értékadásnál, például mai nap + 7 nap:

```
SQL> UPDATE szamlak
  2  SET kelt=SYSDATE+7, rogz='Tóth Ádám'
  3 WHERE osszeg=12000;
```

5) Az NVL függvény

NLV(oszlopkifejezés, helyettesítő_érték) – Ha egy adott helyen NULL szerepel, a függvény a *helyettesítő_érték*-et adja vissza, ellenkező esetben a tárolt adatot.

Lényeges pl. az AVG függvénynél, ahol a NULL eltorzíthatja az eredményt.

a) Listázzuk ki, hogy mely járatok menetideje kisebb, mint 2 óra (azaz 120 perc), és írassuk ki a menetidőt is!

```
SQL>  SELECT szam jarat, erkezes-indulas
  2   FROM jaratok
  3   WHERE erkezes–indulas<120;

 JARAT ERKEZES–INDULAS
--------- ---------------
      112             108
      113              83
      114             101
      115              95
```

b) Mennyi a bejelentkezési idők valós átlaga?

```
SQL> SELECT  AVG(checkin)
  2  FROM repter;

AVG(CHECKIN)
------------
         1.5
```

```
SQL> SELECT  AVG(NVL(checkin,0))
  2  FROM repter;

AVG(NVL(CHECKIN,0))
-------------------
                1.2
```

c) Írassuk ki a 10 000-nél kisebb számlaösszegeket, azok harmadát, majd azok harmadát 2 tizedes jeggyel megjelenítve!

```
SQL> SELECT osszeg, NVL(osszeg,0)/3, TRUNC(NVL(osszeg,0)/3,2)
  2  FROM szamlak
  3   WHERE osszeg<10000;

   OSSZEG   NVL(OSSZEG,0)/3   TRUNC(NVL(OSSZEG,0)/3,2)
----------   ---------------   ---------------------------------
     2356          785.33333                          785.33
     9364          3121.3333                         3121.33
```

d) Írassuk ki a számlák rögzítőinek nevét (mindenkiét csak egyszer)!

```
SQL> SELECT  DISTINCT  INITCAP(rogz)
  2  FROM szamlak;

INITCAP(ROGZ)
------------------------------
Hegyi
Kiss Edit
Kovacs
Lakatos A.
Magyar Bela
Nagy
Toth Adam
Virag
```

e) Listázzuk ki a számlák keltét, valamint az esedékességi dátumukat, amely a keltezéstől számított 30. nap!

```
SQL>  SELECT szla szamlaszam, kelt, kelt+30 esedekes
  2 FROM szamlak
```

IV.6. Join

```
SELECT oszlopnév, oszlopnév,…
    FROM táblanév [alias név], táblanév [alias név]
        WHERE kapcsoló_oszlop1 öszehasonlító_operátor kapcsoló_oszlop2
            AND…
```

A JOIN lehetővé teszi a különböző táblákban lévő adatok kapcsoló feltételek alapján való összekapcsolását.

a) Listázzuk ki, hogy melyik járat melyik repülőtérről indul!

```
SQL> SELECT szam jarat, varos honnan
  2  FROM jaratok, repter
  3  WHERE honkod = kod;
```

JARAT	HONNAN
123	Budapest, Ferihegy 1
112	Budapest, Ferihegy 1
115	Budapest, Ferihegy 1
111	Praha
113	Warszawa Internacionalna
114	Warszawa Internacionalna

b) Egészítsük ki a fenti listát azzal, hogy kiíratjuk, hová megy a járat! A megjelenített lista legyen a járatok száma szerinti növekvő sorrendben!

```
SQL> SELECT szam jarat, r1.varos honnan, r2.varos hova
  2   FROM jaratok, repter r1, repter r2
  3   WHERE honkod = r1.kod AND hovkod = r2.kod
  4   ORDER BY szam;
```

JARAT	HONNAN	HOVA
111	Praha	Budapest, Ferihegy 1
112	Budapest, Ferihegy 1	Warszawa Internacionalna
113	Warszawa Internacionalna	Budapest, Ferihegy 1
114	Warszawa Internacionalna	Paris Orly
115	Budapest, Ferihegy 1	Warszawa Internacionalna
123	Budapest, Ferihegy 1	Praha

c) Mely repülőterek küldtek 10 000 Ft-nál nagyobb számlákat?

```
SQL> SELECT varos, osszeg
  2  FROM repter, szamlak
  3  WHERE kod = rkod
  4      AND osszeg>10000;
```

VAROS	OSSZEG
Budapest, Ferihegy 1	11328
Praha	23560
Praha	48000
Warszawa Internacionalna	30000
Warszawa Internacionalna	42000
Paris Orly	12000

d) A fenti példa összegzéssel együtt:

```
SQL>  SELECT varos repuloter, SUM(osszeg) osszesen
  2  FROM repter, szamlak
  3  WHERE kod = rkod AND osszeg > 10000
  4  GROUP BY varos;

REPULOTER                 OSSZESEN
------------------------  ----------
Budapest, Ferihegy 1           11328
Paris Orly                     12000
Praha                          71560
Warszawa Internacionalna       72000
```

e) Melyik reptérről nem érkezett még számla?

```
SQL> SELECT UNIQUE varos
  2  FROM repter, szamlak
  3  WHERE kod NOT IN (SELECT rkod FROM szamlak);
```

IV.7. Egymásba ágyazott lekérdezések

A SELECT utasítás WHERE részébe újabb SELECT utasítások ágyazhatók.

Az egymásba ágyazott lekérdezéseket az alaprendszer belülről kifelé haladva dolgozza fel.

Ha egynél több oszlopot kívánunk átvenni a belső SELECT-ből, akkor a külső SELECT WHERE részében az átvenni kívánt oszlopokat ( ) zárójelek közé kell zárni.

A belső SELECT-ből is hivatkozhatunk a külső SELECT-re, ezeket az egymásba skatulyázott SELECT-eknek nevezzük.

A belső SELECT által visszaadott listát kezelhetjük az ANY, ALL, illetve EXISTS eszközökkel.

a) Listázzuk ki a 12:00 után induló járatok számát és indulási idejét!

```
SQL>  SELECT szam, indulas
  2  FROM jaratok
  3  WHERE szam IN
  4     (SELECT szam
  5      FROM jaratok
  6     WHERE indulas>1200);

   SZAM  INDULAS
------------  ------------
     112     1535
     113     2032
     114     1322
     115     1415
```

b) Milyen járatszámmal és mikor indítanak járatot azok a repülőterek, amelyek 20 000 Ft-nál nagyobb számlát küldtek?

```
SQL> SELECT szam jarat,  indulas indul, r1.varos repterrol, r2.varos repterre
  2     FROM jaratok, repter r1, repter r2
  3     WHERE honkod=r1.kod AND hovkod=r2.kod AND r1.kod IN
  4        (SELECT DISTINCT rkod
  5         FROM szamlak
  6        WHERE osszeg > 20000);

JARAT INDUL REPTERROL                      REPTERRE
--------- ----------- -------------------------------- ------------------------
      111      1023 Praha                       Budapest, Ferihegy 1
      113      2032 Warszawa Internacionalna    Budapest, Ferihegy 1
      114      1322 Warszawa Internacionalna    Paris Orly
```

c) Mely repülőterekről érkezett M kódú számla?

```
SQL> SELECT DISTINCT varos                    VAROS
  2  FROM repter, szamlak sz                  ------------------------
  3  WHERE rkod = kod AND                     Budapest, Ferihegy 1
  4  EXISTS ( SELECT * FROM szamlak           Praha
  5  WHERE sz.szla LIKE '__/M%');
```