Szabóné Nacsa Rozália: nacsa@inf.elte.hu

SQL – Struktúrált (adatbázis) lekérdező nyelv

Az SQL – (*Structured Query Language - Struktúrált lekérdező nyelv*) relációs adatbázisok kezelésére kifejlesztett szabvány, amely alkalmazható relációs adatbázisok létrehozására, módosítására, valamint arra, hogy az így létrehozott adatbázisból különféle adatokat nyerjünk ki és jelenítsünk meg.

Néhány relációs adatbáziskezelő rendszer: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, MySQL.

Valamennyi relációs adatbáziskezelő rendszer értelmezi a legalapvetőbb SQL parancsokat, az összetettebb parancsok azonban az egyes rendszereknél eltérőek lehetnek.

Adatbázis táblázatok

Egy *relációs adatbázis* egy vagy több táblázatból áll. Az adatokat és az adatbázisra vonatkozó információkat ezen táblázatok (relációk) tartalmazzák. A táblázatoknak egyedi nevük van. Az adatbázis tárolja az oszlop egyedi nevét, adata típusát és egyéb tulajdonságait.

Az „Iskola" adatbázis felépítése

DIÁKOK

dkód	vnév	knév
1001	Kiss	János
1002	Nagy	János
1003	Nagy	Katalin
1004	Szabó	Kinga
1005	Kiss	Hajnalka
1006	Nagy	Benedek
2001	Szabó	Péter
2002	Kiss	Katalin
2003	Nagy	János
2004	Szabó	Benedek

REGISZTRÁCIÓ

dkód	okód	dátum
1001	nem01	2002-05-07
1002	ang03	2002-05-14
1003	ang01	2002-05-14
1004	nem01	2002-05-07
1005	ang01	2002-05-09
1006	ang03	2002-05-14
2001	ang03	2002-05-09
2002	ang01	2002-05-07
2003	nem01	2002-05-14
2004	ang03	2002-05-09
2004	nem01	2002-05-09

ÓRÁK

okód	tárgy	szint	nap	kezdete	vége
ang01	angol	1	Hétfö	08:00:00	10:00:00
ang02	angol	1	Hétfö	10:00:00	12:00:00
ang03	angol	2	Kedd	08:00:00	10:00:00
nem01	német	2	Kedd	10:00:00	12:00:00
nem02	német	1	Hétfö	10:00:00	12:00:00

Megjegyzés: *A táblázatokban szürkével jelölt oszlopok a táblázat kulcsmezői. Ezeknek az adatoknak egyedieknek kell lennie.* (Primary key)

A MySQL adatbáziskezelő

Ebben a fejezetben az SQL – (Structured Query Language - Struktúrált lekérdező nyelv) relációs adatbázisok kezelő nyelv használatával szeretnénk megismerkedni. Ahhoz, hogy a nyelv parancsait kipróbálhassuk, szükségünk van egy „élő” adatbáziskezelőre. Mi erre a célra a MySQLAB által kifejlesztett, szabadon forgalmazott MySQL adatbáziskezelő rendszert választottuk. (www.mysql.com), amely több platformra is létezik.

A MySQL installálása

Ha Ön olyan gépnél dolgozik, amelyre még nem installálták fel a mysql-t, akkor hajtsa végre az 1-4 lépéseket:

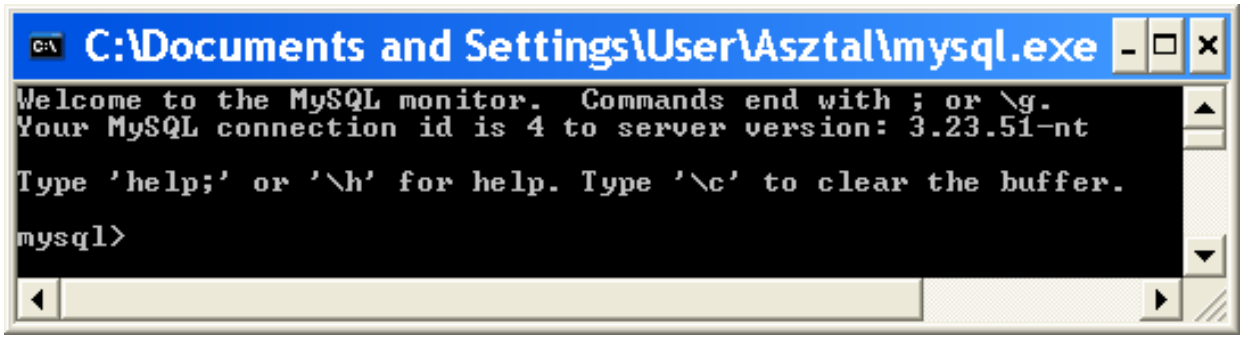
1. A **mysql.com** weboldalról töltse le a **mysql-3.23.52-win.zip,** és a **mysql++-1.7.1-1-win32-vc++.zip** fájlokat.
2. A **mysql-3.23.51-win.zip** kicsomagolása után indítsa el a **setup.exe** programot. Legyen a célkönyvtár neve **mysql**.
3. Csomagolja ki a **mysql++-1.7.1-1-win32-vc++.zip** fájlt. Legyen a célkönyvtár neve szintén **mysql**.
4. **A mysql\bin\winmysqladmin.exe** program segítségével elindíthatja, bezárhatja, módosíthatja a MySql szervert.

Megjegyzés:

- Ha parancsablakból elindítja a **mysql\bin\mysql.exe** programot, akkor itt közvetlenül adhat ki parancsokat a MySQL adatbáziskezelőnek.
- A MySQL futtatásához számítógépünknek rendelkeznie kell TCP/IP kapcsolattal.
- Hasznos információkat találhat a **mysql/doc** alkönyvtárban.

A MySQL adatbáziskezelő használata

Ha parancsablakból elindítjuk a **mysql\bin\mysql.exe** programot, akkor itt közvetlenül adhatunk ki parancsokat a MySql databáziskezelőnek. (*előtte a winmysqladmin.exe programmal el kell indítani egy adatbáziskezelő szervert*)



MySQL parancsok

Help	(\h)	Display this help.
?	(\?)	Synonym for `help'.
Clear	(\c)	Clear command.
connect	(\r)	Reconnect to the server. Optional arguments are db and host.
Ego	(\G)	Send command to mysql server, display result vertically.
Exit	(\q)	Exit mysql. Same as quit.
Go	(\g)	Send command to mysql server.
Note	(\t)	Don't write into outfile.
Print	(\p)	Print current command.
Quit	(\q)	Quit mysql.
Rehash	(\#)	Rebuild completion hash.
Source	(\.)	Execute a SQL script file. Takes a file name as an argument.
Status	(\s)	Get status information from the server.
Tee	(\T)	Set outfile [to_outfile]. Append everything into given outfile.
Use	(\u)	Use another database. Takes database name as argument.

Saját jegyzeteim

Adatbázis létrehozása: CREATE DATABASE

CREATE DATABASE adatbázis neve;

A MySQL a c:\mysql\data alkönyvtárban létrehoz egy „adatbázis neve” nevű alkönyvtárat.

☹ *Az SQL parancsokat* **MINDIG** *le kell zárni egy* **PONTOSVESSZŐVEL**.

Adatbázis használatbavétele: USE

USE adatbázis neve;

Az adatbázist minden felhasználásnál „használatba” kell venni.

Táblázat struktúra létrehozása: CREATE TABLE

Az SQL-ben a **CREATE TABLE** utasítás segítségével hozunk létre táblázatot (alaprelációt).

Egyszerű create parancs:

CREATE TABLE táblanév
(oszlop1 típus, oszlop2 típus, oszlop3 típus, . . .);

Create parancs megszorítással:

CREATE TABLE táblanév
(oszlop1 típus [megszorítás], oszlop2 típus [megszorítás], . . .)

Példák

CREATE TABLE diakok (dkod **INT**(4) **PRIMARY KEY NOT NULL**, vnev **VARCHA**r(10), knev **VARCHAR**(10));

CREATE TABLE REGISZTRACIO (dkod **INT**(4) **NOT NULL**, okod **CHAR**(4) **NOT NULL**, datum **DATE, PRIMARY KEY(**dkod**,**okod**)**);

CREATE TABLE orak (okod **CHAR**(5) **PRIMARY KEY NOT NULL**, targy **CHAR**(10) **NOT NULL**, szint **INT**(1), nap **VARCHAR**(9), kezdete **TIME**, vege **TIME**);

A tábla neve betűvel kezdődhet, belsejében betűket, aláhúzás jeleket és számokat lehet használni. A név ne legyen 30 jelnél hosszabb, és ne használjuk az SQL foglalt szavait (min pl. select, create, describe, where, stb.)

A **típusban** definiálhatjuk, milyen adatot tárolunk az adott oszlopban.

A **PRIMARY KEY** jelzi, hogy az oszlopban minden értéknek egyedinek kell lennie. Figyeljük meg, hogy a regisztracio táblában két oszlop közösen alkotja az elsődleges kulcsot.

Leggyakoribb típusok:

CHAR(méret):	Fix, méret hosszúságú karakterlánc. max: 255 hosszú.
VARCHAR(méret):	változó (maximum méret) hosszúságú karakterlánc.
INT(méret):	egész szám
DATE:	dátum (’éé-hh-nn’ alakban);
TIME:	idő (’oo:pp:mm’ alakban)

Táblázat struktúra megtekintése: DESCRIBE

DESCRIBE táblanév;

Példák

DESCRIBE DIAKOK;

Field	Type	Null	Key	Default	Extra
dkod	int(4)		PRI	0	
vnev	varchar(10)	YES		NULL	
knev	varchar(10)	YES		NULL	

DESCRIBE REGISZTRACIO;

Field	Type	Null	Key	Default	Extra
dkod	int(4)		PRI	0	
okod	char(4)		PRI		
datum	date	YES		NULL	

DESCRIBE ORAK;

Field	Type	Null	Key	Default	Extra
okod	varchar(5)		PRI		
targy	varchar(10)	YES		NULL	
szint	int(1)			0	
nap	varchar(9)	YES		NULL	
kezdete	time	YES		NULL	
vege	time	YES		NULL	

Táblázat struktúra módosítása: ALTER TABLE

Létező tábla struktúrájának módosítása: új oszlop beszúrása, oszlopnév/tulajdonságok módosítása.

ALTER [**IGNORE**] **TABLE** táblanév alter_spec

néhány alter_specifikáció:

ADD [**COLUMN**] oszlop_def [**FIRST** | **AFTER** oszlopnév]
or **ADD** [**COLUMN**] (oszlop_def, oszlop_def,...)
or **CHANGE** [**COLUMN**] régi_oszlopnév oszlop_def
or **MODIFY** [**COLUMN**] oszlop_def
or **DROP** [**COLUMN**] oszlopnév
or **DROP PRIMARY KEY**
or **RENAME** [**TO**] új_tablanév
or **ORDER BY** oszlop

oszlop_def: oszlopnév típus [megszorítás]

Példa: **ALTER TABLE** diakok **CHANGE** vnev vnev **VARCHAR**(15) not null;

Új adatsor beszúrása: INSERT

Egy új sort az **insert** paranccsal szúrhatunk be a táblázatunkba.

INSERT into táblanév (első_oszlop_neve,...,utolsó_oszlop_neve)
VALUES (első_érték,...utolsó_érték);

Ha ki szeretné listázni a táblázatot, akkor adja ki a **select * from táblanév;** *parancsot!*

Példák

INSERT INTO diakok **VALUES** ((1001,'Kiss','János');
INSERT INTO orak **VALUES** ('ang01','angol',1,'Hétfő','08:00:00','10:00:00);
INSERT INTO regisztracio **VALUES** (1001,'nem01','2002-05-07');

Gyakorlás: Tandíjak tábla létrehozása

Hozzunk létre egy új táblázatot az alábbiak szerint!

TANDIJAK

okod	tandij
ang01	5000
ang02	10000
ang03	4000
nem01	6000
nem02	7000

CREATE TABLE tandijak **(**okod **CHAR**(5) **PRIMARY KEY NOT NULL,** tandij **INT**(5)**);**

INSERt INTO tandijak **VALUES**("ang01",5000);
INSERT INTO tandijak **VALUES**("ang02",10000);
INSERT INTO tandijak **VALUES**("nem01",6000);
INSERT INTO tandijak **VALUES**("nem02",7000);
INSERT INTO tandijak **VALUES**("ang03",4000);

Adatok módosítása: UPDATE

Az Update parancsot használjuk egy adott feltételnek eleget tévő rekordkészlet módosítására.

UPDATE táblanév"
SET oszlopnév új_érték1[,köv_oszlopnév = új_érték2...]
WHERE oszlop1 **OPERATOR** érték [**AND** | **OR** oszlop2 **OPERATOR** érték];

Példák

Az alábbi paranccsal áttehetjük az ORAK táblázat valamennyi hétfői 10-12-ig tartott óráját szerda: 8-10-ig-re.

UPDATE orak **SET** kezdete='08:00:00', vege='10:00:00', nap='Szerda"
WHERE nap="Hétfö" **AND** kezdete='10:00:00';

Adatsorok törlése: DELETE

DELETE FROM táblanév
WHERE oszlop1 **OPERATOR** érték [**AND** | **OR** oszlop2 **OPERATOR** érték];

Figyelem: Ha nem adunk **WHERE** kiegészítést, akkor a táblázat összes sorát törli. – **NEM KÉRDEZ!!!**

Példák

Az alábbi parancs törli az ORAK táblázatunkból a 1 szintű (alapfokú) német órákat.

DELETE FROM orak **WHERE** targy='német' **AND** szint=1;

Táblázat struktúra törlése: DROP TABLE

DROP TABLE táblanév;
A fenti parancs végrehajtása után nem lesz többé az adatbázisunkban táblanév nevű táblázat.

NEM KÉRDEZ!!!

Saját jegyzeteim

Egyszerű lekérdezések – Select

A **SELECT** paranccsal gyűjthetjük ki az adatbázisunkból az általunk megfogalmazott feltételeknek eleget tévő adatokat.

A **SELECT** parancs egyszerűsített formája:

SELECT oszlop1 [oszlop2, oszlop3, stb.]
FROM táblanév
[**WHERE** feltétel]

A **SELECT** kulcsszót követő oszlopnevekkel adjuk meg, mely oszlopokat szeretnénk az eredménylistában megjeleníteni.

A **FROM** szócskát követő táblanév jelöli ki azt a táblázatot, amelyből az eredménylistát fel szeretnénk építeni.

A **WHERE** kiegészítésben megfogalmazhatjuk, hogy milyen feltételeknek eleget tévő adatok/rekordok kerüljenek be az eredménybe.

Feltételes operátorok a **WHERE** kiegészítésben: =, `>, <, >=, <=, <> or !=, LIKE,`

ahol a **LIKE** mintaillesztő operátorral a % jelet helyettesítőként használva adhatunk meg feltételt.

Példák

SELECT * from diakok;

dkod	vnev	knev
1001	Kiss	János
1002	Nagy	János
1003	Nagy	Katalin
2001	Szabó	Péter
2002	Kiss	Katalin
2003	Nagy	János
1004	Szabó	Kinga
1005	Kiss	Hajnalka
2004	Szabó	Benedek
1006	Nagy	Benedek

SELECT dkod,knev
FROM diakok
WHERE vnev="Nagy";

dkod	knev
1002	János
1003	Katalin
2003	János
1006	Benedek

"Nagyék" listája

SELECT dkod,datum
FROM regisztracio
WHERE okod LIKE '%01';

dkod	dátum
1001	2002-05-07
1003	2002-05-14
1004	2002-05-07
1005	2002-05-09
2002	2002-05-07
2003	2002-05-14
2004	2002-05-09

Az ang01és nemö1 csoport listája

Statisztikai függvények: min, max, sum, avg, count

MIN	Az adott oszlop legkisebb értéke
MAX	Az adott oszlop legnagyobb értéke
SUM	Az adott oszlop adatainak összege
AVG	Az adott oszlop adatainak átlaga
COUNT	Az adott oszlopban szereplő sorok száma
COUNT(*)	A táblázatban szereplő sorok száma

Példák

SELECT COUNT(*) FROM diakok;

Count(*)
10

SELECT AVG(tandij) **FROM** tandijak;

Avg(tandij)
6400

Matematikai műveletek és matematikai függvények

Matematikai műveletek: + , -, *, /, % (moduló)
Matematikai függvények:

ABS(x)	x abszolút értéke
SIGN(X)	x előjelétől függően értéke -1, 0, vagy 1 (negatív, zéró, vagy pozitív)
MOD(X,Y)	Ugyanaz, mint az x%y
POWER(X,Y)	x^y
ROUND(X)	Egészre kerekített érték
ROUND(X,D)	D db tizedesre kerekített érték
SQRT(X)	x négyzetgyöke

Példák

SELECT okod,**ROUND**(tandij/15)
FROM tandijak;

okod	**round(tandij/15)**
ang01	333
ang02	667
nem01	400
nem02	467
ang03	267

Saját jegyzeteim

Összetett lekérdezések – SELECT

SELECT [**ALL** | **DISTINCT**] *oszlop1*[,*oszlop2*]
FROM *table1*[,*table2*]
[**WHERE** feltétel]
[**GROUP BY** oszloplista]
[**HAVING** feltételek]
[**ORDER BY** oszloplista [**ASC** | **DESC**]]

A **SELECT** parancs öt bővítményt tartalmaz, bár csak a **FROM** bővítmény használata kötelező.

Példa

SELECT neve, kora, fizetése
FROM alkalmazottak
WHERE kora > 40;

A **DISTINCT** kulcsszót akkor használjuk, ha az eredményben az ismétlődő sorokat csak egyszer szeretnénk megjeleníteni.

Példa:

SELECT okod **FROM** regisztracio;

okod
nem01
ang03
ang01
nem01
ang01
ang03
ang03
ang01
nem01
ang03
nem01

SELECT DISTINCT okod **FROM** regisztracio;

okod
nem01
ang03
ang01

Azon órák listája, amelyekre már jelentkezett valaki.

Saját jegyzeteim

Összesítés készítése: GROUP BY

SELECT oszlop1, [**SUM**(oszlop2)]
FROM táblák listája
[GROUP BY oszlopok listája;]

A **GROUP BY** kikötéssel összegyűjtjük azokat a sorokat, amelyek értéke a jelölt oszlopban megegyezik, és ezekre a sorokra együttesen végzi el a megadott műveletet.

Példa

Szeretném megtudni, hogy melyik órára hányan diák regisztrált. **SELECT** okod, **COUN**t(dkod) **FROM** regisztracio **GROUP BY** okod; **okod** | **count(dkod)** ang01 | 3 ang03 | 4 nem01 | 4	✍ *Saját jegyzeteim*

Feltételes összesítés készítése: HAVING

SELECT oszlop1, [**SUM**(oszlop2)]
FROM táblák listája
[**GROUP BY** táblák listája]
[**HAVING** feltételek;]

A **GROUP BY** bővítménnyel kialakított csoportokból csak azok kerülnek be a listába, amelyek a **HAVING** feltételnek is eleget tesznek.

Példa

Keresem azokat az órákat, ahol 3-nál többen regisztráltak. **SELECT** okod, **COUNT**(dkod) **FROM** regisztracio **GROUP BY** okod **HAVING COUNT**(dkod) > 3; **okod** | **count(dkod)** ang03 | 4 nem01 | 4	✍ *Saját jegyzeteim*

Rendezett lista készítése: ORDER BY

SELECT oszlop1, [**SUM**(oszlop2)]
FROM táblák listája
[**ORDER BY** oszlopok listája **[ASC** | **DESC**]];

Az **ORDER BY** egy olyan tetszőlegesen választható kiegészítés, amellyel az **ORDER BY** után megadott oszlopok szerint rendezett listát készíthetünk.

Kászítsünk névsorszerint rendezett listát a „nem János" diákokról!

SELECT vnev, knev, dkod **FROM** diakok **WHERE** knev<>"János" **ORDER BY** vnev, knev;

vnev	knev	dkod
Kiss	Hajnalka	1005
Kiss	Katalin	2002
Nagy	Benedek	1006
Nagy	Katalin	1003
Szabó	Benedek	2004
Szabó	Kinga	1004
Szabó	Péter	2001

Saját jegyzeteim

Tartalmazás feltétel: IN és BETWEEN

SELECT oszlop1, **SUM**(oszlop2)
FROM táblázatok listája
[**WHERE** oszlop3 **IN** (értékek listája)];

SELECT oszlop1, **SUM**(oszlop2)
FROM táblázatok listája
[**WHERE** oszlop3 **BETWEEN** érték1 **AND** érték2] ;

Az **IN** feltételes operátor egy igazi halmazelméleti "eleme" operator. Segítségével eldönthetjük, hogy az eredményadatok adott oszlopának értéke benne van-e az **IN** után megadott listában. Az eredmény listába csak azok a tételek kerülnek be, amelyek eleget tesznek az "eleme" feltételnek.

A **BETWEEN** feltételes operátorral eldönthetjük, hogy a megjelölt oszlop adatai benne vannak-e a megadott intervallumban.

Példa: *Az alábbi paranccsal **Katalin** és **Kinga** keresztnevű diákjainkat listázhatjuk ki.*

SELECT vnev, knev
FROM diakok
WHERE knev **IN** ("Katalin","Kinga");

Több táblázat összekapcsolása: JOIN

Lehetőség van olyan lekérdezések készítésére, ahol egyszerre több táblázatot is összekapcsolunk. Az "összekapcsolás" a táblázatok vesszővel elválasztott felsorolásával történik.
Ha kiadjuk a

```
SELECT regisztracio.okod,tandij
    FROM regisztracio, tandijak;
```

parancsot, akkor kapunk egy nagyon hosszú listát és abban a meglepetésben lesz részünk, hogy mindenki mindenkivel össze van kapcsolva ("ez a két halmaz direkt szorzata").

Értelmesebb végeredményt kapunk, ha a fenti, nagyon hosszú listából kiszűrjük azokat a sorokat, amelyekben az okod megegyezik (a két táblázatot az okod oszlop mentén kötjük össze). Ehhez az alábbi parancsot kell kiadni:

```
SELECT regisztracio.okod,tandij
    FROM regisztracio, tandijak
    WHERE regisztracio.okod=tandijak.okod;
```

okod	tandíj
ang01	5000
ang01	5000
ang01	5000
nem01	6000
nem01	6000
nem01	6000
nem01	6000
ang03	4000
ang03	4000
ang03	4000
ang03	4000

Ha csoportonkénti bontásban szeretnénk megkapni a tandíjbevételeket, akkor a fenti parancsot tovább kell finomítani:

```
SELECT regisztracio.okod,SUM(tandij)
    FROM regisztracio, tandijak
    WHERE regisztracio.okod=tandijak.okod
    GROUP BY okod;
```

okod	sum(tandij)
ang01	15000
ang03	16000
nem01	24000

Ha csak azokat a csoportokat szeretnénk kilistázni, amelyeknél a tandíjbevétel 15000 Ft felett van, akkor az alábbiak szerint kell módosítani a parancsot:

```
SELECT regisztracio.okod, SUM(tandij)
    FROM regisztracio, tandijak
    WHERE regisztracio.okod = tandijak.okod
    GROUP BY okod
    HAVING SUM(TANDIJ) > 15000;
```

okod	sum(tandij)
ang03	16000
nem01	24000

Módosítsuk a fenti listát oly módon, hogy a tantárgy nevét is megjelentetjük az eredmény listában.

```
SELECT regisztracio.okod, orak.targy, SUM(tandij)
    FROM regisztracio, tandijak,orak
    WHERE regisztracio.okod=tandijak.okod AND regisztracio.okod=orak.okod
    GROUP BY okod;
```

okod	targy	sum(tandij)
ang01	angol	15000
ang03	angol	16000
nem01	német	24000