SQL ALAPOK

Bevezetés

A MYSQL szintaxisa

Táblák, adatok kezelésének alapjai

BEVEZETÉS

- SQL: Structured Query Language – Strukturált Lekérdező Nyelv
- Szabvány határozza meg, azonban számos "nyelvjárása" létezik – egy ilyen a MySQL rendszerben használt változat **(ezt vesszük)**
- Nincs jelentős eltérés a különböző változatok közt
- **Nem algoritmikus nyelv,** jellemző használata:
  - Beágyazzuk egy algoritmikus nyelven írt programba az SQL-utasítást
  - Az SQL-utasítás hatására az adatbázis előáll a válasszal
  - Az algoritmikus nyelven íródó programban feldolgozzuk az adatokat

AZ SQL KÉT RÉSZE

- Az SQL-utasításokat a jellegük alapján két részre osztjuk

DDL	DML
Data Definition Language – adatdefiníciós nyelv	Data Manipulation Language – adatkezelő utasítások

AZ SQL SZINTAXISA

- "Beszédes" nyelv, az utasítások neve általában a megfelelő angol szó, ezekből alkotunk "mondatokat"
- a nyelv alapszavait általában nagybetűkkel írjuk, de a kisbetűs írásmód is egyenértékű

SELECT attrib FROM tábla;	**select attrib from tábla;**

- A parancsokat írhatjuk sorfolytonosan, vagy akár tagoltabban, a végüket ; jelzi
- Változó nincs, csak attribútumorkra lehet hivatkozni – ha egy utasításon belül nem egyértelmű, mire, elérírjuk a tábla nevét: tábla.attrib

AZ SQL SZINTAXISA, SQL LOGIKA

- Szövegkonstans: ‘idézőjelek közt’
- Relációjelek: =, <=, >=, !=, <>
- A logikában szokásos műveletek: AND, OR, NOT.
- Az SQL logika nem a klasszikus “igaz-hamis” logika, hanem úgymond **háromértékű:** TRUE, FALSE, UNKNOWN.
- Ha valahol NULL (“nem definiált, nem adott”) érték fordul elő, a rá vonatkozó logikai kifejezés kiértékelése ismeretlent ad válaszul
  - Ha azt akarjuk ellenőrizni, hogy egy adott atttribútum NULL-e, akkor:
  - attrib IS NULL
  - NEM fog helyes választ adni: attrib=NULL

ADATBÁZIS LÉTREHOZÁSA

- `CREATE DATABASE adatbázis_neve;`

- Létrehoz egy új, üres adatbázist a kiszolgálón (pl. kiadható a konzolon.)
- Konzolos elérésnél a `USE adatbázis_neve;` paranccsal lehet kiválasztani egy adatbázist, programozási környezetekből elérve általában valamilyen metódushívással.

SÉMÁK DEFINIÁLÁSA

- A relációs adatbázissémákat definiálnunk kell az adatbázis létrehozása után

```
CREATE  TABLE [IF NOT EXISTS] séma_neve (
      attr1 {ADATTÍPUS} [{megszorítások}],
      attr2 {ADATTÍPUS} [{megszorítások}],
      …
      attrn {ADATTÍPUS} [{megszorítások}]
      [,{táblaszintű_megszorítások}] )
[{táblára_vonatkozó_megszorítások}];
```

SQL ADATTÍPUSOK

- a korábbi parancsban {adattípus} helyére a következők valamelyike kerül:

INT(n)	n jegyű egész szám
CHAR(n)	pontosan n hosszú szöveg
VARCHAR(n)	legfeljebb n hosszú szöveg
DATE	dátum, pl. 2012-10-17
TIME	időpont, pl. 11:10:45
REAL	valós szám

- (létezik néhány egyéb adattípus is)

PÉLDA: TÁBLA DEFINIÁLÁSA

- Egy autók adatait tároló táblát így definiálnánk:

Elsődleges kulcs

```
CREATE TABLE IF NOT EXISTS auto (
        sorszam INT(10) PRIMARY KEY,
        rendszam CHAR(6),
        gyarto VARCHAR(128),
        evjarat INT(4),
        tipus VARCHAR(128)
);
```

MEGSZORÍTÁSOK

A MySQL nem támogatja! Használjunk táblaszintűt!

- Oszlopszinten:
  - PRIMARY KEY – elsődleges kulcs
  - UNIQUE – kulcs
  - REFERENCES más_tábla(attrib) [{ON-feltételek}] – külső kulcs
  - NOT NULL – nem lehet null
  - AUTO_INCREMENT – mindig növekszik az értéke
- Táblaszinten:
- PRIMARY KEY (oszlop_lista) – ha több attribútumból áll a kulcs
- FOREIGN KEY (oszlop_lista) REFERENCES másik_tábla(oszlop_lista) [{ON-feltételek}] – hasonlóan

MEGSZORÍTÁSOK

- ON feltételek: külső kulcsoknál megadható, hogy történjen-e valami az aktuális táblában, ha a táblában, ahol elsődleges kulcsként fordul elő az attribútum, valamilyen változás áll be – pl. ha egy vevőt törölnek egy adatbázisból, elérhető így, hogy automatikusan törlődjön az összes vásárlása is.

- ON UPDATE CASCADE – adatmódosításnál a friss adat bemásolódik
- ON DELETE CASCADE – ha törlik a hivatkozott sort, törlődnek a rá hivatkozó sorok is
- ON DELETE SET NULL – a hivatkozott sor törlésekor a hivatkozó sorokban NULL kerül a külső kulcsba

TÁBLÁK MÓDOSÍTÁSA

- Létrehozás után lehetőség van a tábák szerkezetén, kulcsain, kapcsolatain változtatni – valamint törölhetők

- ALTER TABLE tábla_neve ADD (oszlopnév {TÍPUS} [{feltételek}]);
- ALTER TABLE tábla_neve MODIFY (oszlopnév [{feltételek}]);
- ALTER TABLE tábla_neve DROP(oszlopnév1, …, oszlopnévk);

- DROP TABLE tábla_neve;

ADATBEVITEL

- INSERT INTO tábla_neve VALUES ({az oszlopok értékei az új rekordban});
- ha csak néhány oszlopot töltenénk fel, és a többi maradjon null: INSERT INTO tábla_neve (oszlop1,…,oszlopk) VALUES (érték1,…,értékk);

- Például:

INSERT INTO hallgato (eha, nev, szak, eletkor) VALUES ('mintaat.sze', 'Minta Áron', 'programtervező informatikus BSc', 22);

ADATMÓDOSÍTÁS

```
UPDATE tábla_neve
SET
        oszlop1 = érték1,
        …,
        oszlopk = értékk
[WHERE
        feltétel];
```

A WHERE feltételben valamilyen feltételt kell megadni arra, mely sorok módosuljanak.

Példa: UPDATE hallgato SET eletkor=23 WHERE eha='mintaat.sze';

ADATOK TÖRLÉSE

DELETE FROM tábla_neve
[WHERE feltéltel];

Törli a feltétel szerinti sort vagy sorokat az adott táblából

Példa: Töröljük a hallgatókat, akiknek 34 és 40 közti az életkoruk:

DELETE FROM hallgato WHERE eletkor BETWEEN 34 AND 40;

FELADATOK

Osztály (<u>osztálykód</u>, osztálynév, *vezAdószám*)

Dolgozó (<u>adószám</u>, név, lakcím, fizetés, *osztálykód*)

6.1 Hozzuk létre a sémák feletti táblákat! Hogyan adnánk meg, hogy vezAdószám külső kulcs?

6.2 Szúrjunk be a 'hulladékgatdálkodási' osztályt, és egy új dolgozót erre az osztályra. Nevezzük is ki osztályvezetőnek.

6.3 Növeljük az összes 10 és 20 közti osztálykódú dolgozó fizetését 20%-kal.

6.4 Kovács Márta nyugdíjba ment. Töröljük az adatait.

6.5 Töröljük ki a 'hulladékgazdálkodási' osztályt.