Ez a különnyomat szabadon és ingyenesen terjeszthető.

Jenei Imre

Triggerek, tárolt eljárások és függvények alkalmazása MySQL-ben

A könyv megvásárolható az
Ad Librum internetes könyvesboltjában

A bekeretezett szövegekre kattintva további információkhoz juthat.

A könyvvel kapcsolatban további információ érhető el a kötet weboldalán:
http://podpress.hu/Jenei-Imre/

A könyv adatai:

Szerző:	Jenei Imre
Cím:	Triggerek, tárolt eljárások és függvények alkalmazása MySQL-ben.
Kiadás:	Ad Librum Kiadó
Kiadási év:	2008
ISBN:	978-963-9888-33-3
Terjedelem:	206 oldal
Méret:	A4
Ár:	2990 Ft
Közvetlen vásárlás:	shop.adlibrum.hu

Tartalomjegyzék

Részletes tartalomjegyzék

Előszó

A számítógépes hálózatok terjedésével egyre több helyen alkalmaznak adatbázis-kezelő rendszereket, ahol általában egy vagy több központi számítógépen (szerveren) tárolnak minden adatot és ezekhez kapcsolódnak az ügyfélgépek (kliensek), hogy hozzáférjenek a számukra szükséges adatokhoz.

Az adatbázis-kezelés minél hatékonyabb megvalósítása mindig is kulcsfontosságú volt. Fontos, hogy a szerver a lehető leggyorsabban szolgálja ki az ügyfeleket, miközben arra is ügyelni kell, hogy ne legyen túlterhelve a hálózat. Hiszen ha egyszerre túl sok adat áramlik a hálózaton, akkor ez a rendszer lelassulását vagy akár összeomlását eredményezheti.

A hatékonyság növelésére több módszer is létezik. Ide tartozik a triggerek és a tárolt alprogramok (eljárások és függvények) alkalmazása. Ezek az adatbázisban (szerveren) vannak eltárolva és ott helyben hajtódnak végre amikor meghívjuk őket a kliens gépről. Így a triggerek és a tárolt alprogramok használatával jelentősen csökkenthető a hálózati adatforgalom és ezáltal nő a rendszer hatékonysága. Hasonlóan a többi magas szintű programozási nyelvhez, a triggerekben, valamint a tárolt eljárásokban és függvényekben is változókat deklarálhatunk, különböző vezérlési szerkezeteket, utasításokat alkalmazhatunk, meghívhatunk függvényeket stb.

Könyvünkben ezt a módszert fogjuk bemutatni. Először a fontosabb elméleti ismereteket tárgyaljuk, majd létrehozunk egy adatbázist a szükséges táblákkal, triggerekkel és tárolt alprogramokkal együtt, amely öttusázó sportolók adatait tárolja majd. Ezen példaadatbázison keresztül mutatjuk be a triggerek és tárolt alprogramok használatát. Adatbázisszerverként a MySQL 5.x (vagy magasabb) verzióját választottuk. (A MySQL csak az 5.x verziótól támogatja a triggereket és a tárolt alprogramokat.)

A tárolt alprogramokat Delphi, C# és webalkalmazásokból fogjuk meghívni. Vagyis három különböző programnyelven készítjük majd el kliens alkalmazásainkat. Így az Olvasó látni fogja a különböző megvalósítások közötti különbségeket és hasonlóságokat és eldöntheti, hogy számára melyik megoldás a szimpatikusabb.

3.
Sportolók adatait nyilvántartó adatbázis elkészítése

Ebben a fejezetben egy példán keresztül fogjuk bemutatni, hogy miként hozhatunk létre egy adatbázist, a hozzá tartozó táblákat, triggereket és tárolt alprogramokat a MySQL konzol alkalmazásában. A konzolalkalmazást root felhasználóként indítsuk el!

A következőkben hozzunk létre egy olyan adatbázist, amely öttusázó sportolók törzsadatait és a sportolók eredményeit tárolja két táblában (`racer, result`). A táblákhoz létre fogunk hozni triggereket és tárolt alprogramokat is.

A későbbi fejezetekben az itt létrehozott adatbázist használjuk majd fel Delphi, C# és webalkalmazásainkhoz.

3.1. Adatbázis létrehozása

Létrehozzuk a `racing` nevű adatbázisunkat, amelynek az alapértelmezett karakterkészletét magyar nyelvűre állítjuk:

```
mysql> create database racing character set=latin2 collate latin2_hungarian_ci;
Query OK, 1 row affected (0.02 sec)

mysql> use racing;
Database changed
```

Beléptünk a `racing` adatbázisba és a továbbiakban itt hozzuk létre a táblákat, triggereket és a tárolt alprogramokat.

3.2. A szükséges táblák létrehozása

Először hozzuk létre a versenyzők törzsadatait tartalmazó táblát `racer` néven, majd hozzunk létre indexeket is a tábla néhány oszlopához:

```
mysql> create table racer(
    -> id int not null auto_increment,
    -> name varchar(30) not null,
    -> age int not null,
    -> generic char(1) not null,
    -> primary key(id)) engine=MYISAM;
Query OK, 0 rows affected (0.03 sec)

mysql> create index name_ind on racer (name);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> create index gen_ind on racer (generic);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Listázzuk ki a tábla szerkezetét:

```
mysql> desc racer;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| id      | int(11)     | NO   | PRI | NULL    | auto_increment |
| name    | varchar(30) | NO   | MUL | NULL    |                |
| age     | int(11)     | NO   |     | NULL    |                |
| generic | char(1)     | NO   | MUL | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
4 rows in set (0.02 sec)
```

A `racer` tábla négy oszlopból áll:

- `id`: a versenyző egyedi azonosítója (elsődleges kulcs)
- `name`: a versenyző neve (indexelt)
- `age`: a versenyző kora
- `generic`: a versenyző neme (indexelt)

Most pedig hozzuk létre a `result` nevű táblát, amely az öttusázó versenyzők adott évben elért eredményeit tárolja. Minden versenyző egy adott évben csak egyszer versenyezhet. Vagyis nem szerepelhet két olyan rekord a táblában, ahol a versenyző és a verseny éve azonos. Ennek elkerülésére ellenörzéseket iktatunk majd be a táblát módosító tárolt eljárásokba (lásd. később).

```
mysql> create table result(
    -> id int not null auto_increment,
    -> rid int not null,
    -> run int not null,
    -> fence int not null,
    -> shoot int not null,
    -> riding int not null,
    -> swim int not null,
    -> position int not null,
    -> r_year year not null,
    -> primary key(id)) engine=MYISAM;
Query OK, 0 rows affected (0.00 sec)
```

Indexek létrehozása:

```
mysql> create index rid_ind on result (rid);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> create index year_ind on result (r_year);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Jelenítsük meg a tábla szerkezetét:

```
mysql> desc result;
+----------+---------+------+-----+---------+----------------+
| Field    | Type    | Null | Key | Default | Extra          |
+----------+---------+------+-----+---------+----------------+
| id       | int(11) | NO   | PRI | NULL    | auto_increment |
| rid      | int(11) | NO   | MUL | NULL    |                |
| run      | int(11) | NO   |     | NULL    |                |
| fence    | int(11) | NO   |     | NULL    |                |
| shoot    | int(11) | NO   |     | NULL    |                |
| riding   | int(11) | NO   |     | NULL    |                |
| swim     | int(11) | NO   |     | NULL    |                |
| position | int(11) | NO   |     | NULL    |                |
| r_year   | year(4) | NO   | MUL | NULL    |                |
+----------+---------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

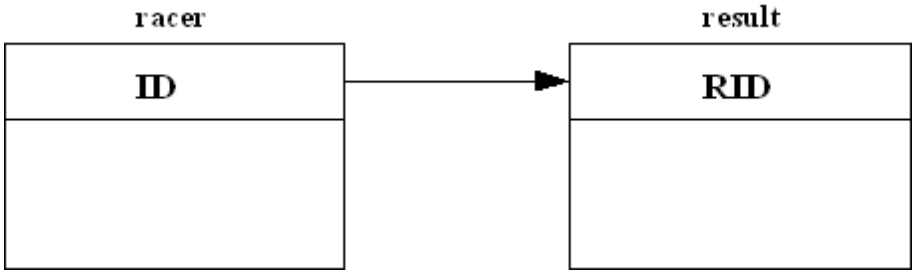
A `result` tábla 9 oszlopból áll:

- `id`: a tábla sorának egyedi azonosítója
- `rid`: a versenyző egyedi azonosítja (akinek az eredményét tároljuk)
- `run`: a versenyző futásban elért pontszáma (0-10-ig)
- `fence`: a versenyző vívásban elért pontszáma (0-10-ig)
- `shoot`: a versenyző lövésben elért pontszáma (0-10-ig)
- `riding`: a versenyző lovaglásban elért pontszáma (0-10-ig)

- `swim`: a versenyző úszásban elért pontszáma (0-10-ig)
- `position`: milyen helyezést ért el a versenyző az adott évben az összes pontszám alapján
- `r_year`: a verseny éve (amikor a versenyt megrendezték)

Majd szükség lesz a versenyzők összes elért pontjára (`run+fence+shoot+riding+swim`) és az átlagpontjukra (`run+fence+shoot+riding+swim`)/5. Ezek ún. számított mezők, ezért nem kell külön eltárolni őket a táblában.

A `racer` és `result` táblákat a versenyzők egyedi azonosítóját tartalmazó mezők segítségével fogjuk összekapcsolni, vagyis a `racer` (versenyző) tábla egy rekordjához több rekord is tartozhat a `result` (eredmény) táblában. A táblák egy-sok kapcsolatban lesznek egymással (22. ábra).



22. ábra: Táblák összekapcsolása (egy-sok kapcsolat)

3.3. Triggerek létrehozása

Először a `racer` táblához hozzunk létre három triggert a `before insert, before update` és az `after delete` eseményekhez.

```
mysql> delimiter //
mysql> create trigger new_racer before insert on racer
    -> for each row
    -> begin
    ->   set new.name=trim(new.name);
    ->   set new.generic=upper(new.generic);
    ->   if new.age<18 then set new.age=0; end if;
    ->   if new.age>50 then set new.age=0; end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.06 sec)

mysql> delimiter ;

mysql> delimiter //
mysql> create trigger upd_racer before update on racer
    -> for each row
    -> begin
    ->   set new.name=trim(new.name);
    ->   set new.generic=upper(new.generic);
    ->   if new.age<18 then set new.age=0; end if;
    ->   if new.age>50 then set new.age=0; end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

A fenti triggerek közül az egyik új rekord beszúrása előtt (`before insert`), a másik pedig rekord módosítása előtt (`before update`) hajtódik végre. Amikor módosítjuk a `racer` táblát az `insert` és az `update` SQL-utasításokkal, akkor a módosítás előtt a fenti triggerek lefutnak és adatkorrigálást, valamint adatellenörzést hajtanak végre:

- a név előtti és utáni szóközöket levágják (`trim`)

- a nem betűjelét nagybetűssé alakítják (`upper`)
- korlátozzák a felvihető versenyzőket életkor szerint (ha a versenyző életkora nem 18-50 év közötti, akkor lenullázzuk az age mező értékét figyelmeztetésként)

Hozzunk létre még egy triggert, amely rekord törlése után (`after delete`) aktivizálódik:

```
mysql> delimiter //
mysql> create trigger del_racer after delete on racer
    -> for each row
    -> begin
    ->   delete from result where rid=old.id;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Amikor törlünk egy rekordot a `racer` táblából a `delete` utasítással, akkor a törlést követően lefut a fenti triggerünk, mely a `racer` táblából törölt versenyző eredményeit törli a `result` táblából.

A `racer` és a `result` táblák egy-sok kapcsolatban állnak egymással, ezért ha törlünk egy rekordot a `racer` táblából, akkor törölni kell a hozzá tartozó tételeket (rekordokat) a `result` táblából is.

Most pedig írjuk meg a szükséges triggereket a `result` táblához a `before insert` és a `before update` eseményekhez.

```
mysql> delimiter //
mysql> create trigger new_result before insert on result
    -> for each row
    -> begin
    ->   if new.run<0 then set new.run=0; end if;
    ->   if new.fence<0 then set new.fence=0; end if;
    ->   if new.shoot<0 then set new.shoot=0; end if;
    ->   if new.riding<0 then set new.riding=0; end if;
    ->   if new.swim<0 then set new.swim=0; end if;
    ->   if new.run>10 then set new.run=10; end if;
    ->   if new.fence>10 then set new.fence=10; end if;
    ->   if new.shoot>10 then set new.shoot=10; end if;
    ->   if new.riding>10 then set new.riding=10; end if;
    ->   if new.swim>10 then set new.swim=10; end if;
    ->   if new.r_year<2000 then set new.r_year=2000; end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> delimiter //
mysql> create trigger upd_result before update on result
    -> for each row
    -> begin
    ->   if new.run<0 then set new.run=0; end if;
    ->   if new.fence<0 then set new.fence=0; end if;
    ->   if new.shoot<0 then set new.shoot=0; end if;
    ->   if new.riding<0 then set new.riding=0; end if;
    ->   if new.swim<0 then set new.swim=0; end if;
    ->   if new.run>10 then set new.run=10; end if;
    ->   if new.fence>10 then set new.fence=10; end if;
    ->   if new.shoot>10 then set new.shoot=10; end if;
    ->   if new.riding>10 then set new.riding=10; end if;
    ->   if new.swim>10 then set new.swim=10; end if;
    ->   if new.r_year<2000 then set new.r_year=2000; end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

- a triggerek korlátozzák az egyes sportágakhoz felvihető pontszámot (0 és 10 közé eshet a pontszám)
- továbbá nem engednek 2000-nél kisebb évszámot megadni

3.4. Tárolt eljárások és függvények létrehozása

Ebben a részben létrehozzuk a tárolt alprogramokat, melyek főleg lekérdezéseket, DML-utasításokat és ellenőrzéseket hajtanak végre az adattáblákon.

Ezeket a tárolt eljárásokat és függvényeket fogjuk majd a különböző programnyelven megírt kliens alkalmazásokból meghívni a 4., 5. és 6. fejezetekben.

3.4.1. Új sportolók felvitele tárolt eljárással

Az alábbi eljárással új rekordot szúrhatunk be a `racer` táblába:

```
mysql> delimiter //
mysql> create procedure new_racer(pname varchar(30),page int,pgen char(1))
    -> begin
    ->   insert into racer values(null,pname,page,pgen);
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Amint látható, az eljárás a paraméterekben megadott értékeket viszi fel a táblába. Próbaként vigyünk fel néhány új rekordot:

```
mysql> call new_racer('Kis Andi',30,'n');
Query OK, 1 row affected (0.03 sec)

mysql> call new_racer('Nagy Peti',25,'f');
Query OK, 1 row affected (0.00 sec)

mysql> call new_racer('Balogh Judit',20,'n');
Query OK, 1 row affected (0.00 sec)

mysql> call new_racer('Lakatos Pista',22,'f');
Query OK, 1 row affected (0.00 sec)
```

Ezután listázzuk ki a `racer` tábla tartalmát:

```
mysql> select * from racer;
+----+---------------+-----+---------+
| id | name          | age | generic |
+----+---------------+-----+---------+
|  1 | Kis Andi      |  30 | N       |
|  2 | Nagy Peti     |  25 | F       |
|  3 | Balogh Judit  |  20 | N       |
|  4 | Lakatos Pista |  22 | F       |
+----+---------------+-----+---------+
4 rows in set (0.00 sec)
```

Látható, hogy az `id` mező értéke automatikusan nőtt mindig 1-gyel és a `generic` mező értékét a trigger (a `before insert` esemény bekövetkezésekor) nagybetűssé alakította.

3.4.2. Sportolók adatainak módosítása tárolt eljárással

Az eljárást az alábbiak szerint hozzuk létre:

```
mysql> delimiter //
mysql> create procedure upd_racer(pid int,pname varchar(30),page int,pgen char(1))
    -> begin
    ->   update racer set name=pname,age=page,generic=pgen where id=pid;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Módosítsuk a fenti eljárás segítségével Balogh Judit életkorát 24-re:

```
mysql> call upd_racer(3,'Balogh Judit',24,'n');
Query OK, 1 row affected (0.00 sec)

mysql> select * from racer;
+----+---------------+-----+---------+
| id | name          | age | generic |
+----+---------------+-----+---------+
|  1 | Kis Andi      |  30 | N       |
|  2 | Nagy Peti     |  25 | F       |
|  3 | Balogh Judit  |  24 | N       |
|  4 | Lakatos Pista |  22 | F       |
+----+---------------+-----+---------+
4 rows in set (0.00 sec)
```

A listában látható, hogy a módosítás sikeres volt.

3.4.3. Sportoló törlése tárolt eljárással

Az eljárást a következőképpen hozzuk létre:

```
mysql> delimiter //
mysql> create procedure del_racer(pid int)
    -> begin
    ->   delete from racer where id=pid;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Próbáljuk ki, hogy helyesen működik-e a fenti eljárás:

```
mysql> call del_racer(4);
Query OK, 1 row affected (0.00 sec)

mysql> select * from racer;
+----+--------------+-----+---------+
| id | name         | age | generic |
+----+--------------+-----+---------+
|  1 | Kis Andi     |  30 | N       |
|  2 | Nagy Peti    |  25 | F       |
|  3 | Balogh Judit |  24 | N       |
+----+--------------+-----+---------+
3 rows in set (0.00 sec)
```

A 4-es azonosítóval rendelkező versenyzőt sikerült törölni a táblából.

3.4.4. Sportolók számának lekérdezése tárolt eljárással

Most hozzunk létre egy olyan eljárást, amely a bemeneti paramétertől függően kiírja a versenyzők számát egy kimenő paraméterbe. A bemeneti paraméter lehetséges értékei a következők:

- 0: a racer táblában szereplő összes versenyző számát megadja
- 1: csak a férfi sportolók számát adja meg
- 2: csak a női sportolók számát adja meg

Az eljárást így hozzuk létre:

```
mysql> delimiter //
mysql> create procedure num_racer_proc(x int, out n int)
    -> begin
    ->   case x
    ->     when 0 then select count(*) into n from racer;
    ->     when 1 then select count(*) into n from racer where generic='F';
    ->     when 2 then select count(*) into n from racer where generic='N';
    ->   else set n=0;
    ->   end case;
    -> end;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;
```

Próbáljuk ki az eljárás működését:

```
mysql> call num_racer_proc(0,@a);
Query OK, 0 rows affected (0.01 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    3 |
+------+
1 row in set (0.00 sec)

mysql> call num_racer_proc(1,@a);
Query OK, 0 rows affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql> call num_racer_proc(2,@a);
Query OK, 0 rows affected (0.00 sec)

mysql> select @a;
+------+
| @a   |
+------+
|    2 |
+------+
1 row in set (0.00 sec)
```

Az eljárás kimenő paraméterének értékét egy globális változónak (`@a`) adtuk át, amit külön lekérdeztünk, így kapva meg az eredményt (sportolók számát).

3.4.5. Sportolók számának lekérdezése tárolt függvénnyel

Ugyanazt a feladatot oldjuk meg, mint az előbb, csak most tárolt függvényt alkalmazunk. A függvény bemenő paramétere ugyanaz, mint az előbb ismertetett eljárásnak.

A függvény létrehozása így történik:

```
mysql> delimiter //
mysql> create function num_racer(x int) returns int
    -> begin
    ->   declare n int;
    ->   case x
    ->     when 0 then select count(*) into n from racer;
    ->     when 1 then select count(*) into n from racer where generic='F';
    ->     when 2 then select count(*) into n from racer where generic='N';
    ->   else set n=0;
    ->   end case;
    ->   return n;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

A függvényt a `select` utasítással futtatjuk:

```
mysql> select num_racer(0);
+--------------+
| num_racer(0) |
+--------------+
|            3 |
+--------------+
1 row in set (0.00 sec)

mysql> select num_racer(1);
+--------------+
| num_racer(1) |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)

mysql> select num_racer(2);
+--------------+
| num_racer(2) |
+--------------+
|            2 |
+--------------+
1 row in set (0.00 sec)
```

Ugyanazokat az eredményeket kaptuk, mint az eljárás esetében.

3.4.6. Sportolók kilistázása tárolt eljárással

Létrehozunk egy eljárást, amely a `racer` táblából kilistázza a sportolók adatait. Bemenő paraméterének lehetséges értékei:

- 0: minden sportolót kilistáz nem és név szerint rendezve
- 1: csak a férfi versenyzőket listázza ki név szerint rendezve
- 2: csak a női sportolókat listázza ki név szerint rendezve

Az ejárás így fog kinézni:

```
mysql> delimiter //
mysql> create procedure list_racer(p int)
    -> begin
    ->   case p
    ->     when 0 then select * from racer order by generic,name;
    ->     when 1 then select * from racer where generic='F' order by name;
    ->     when 2 then select * from racer where generic='N' order by name;
    ->   end case;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Hívjuk meg az eljárást az alábbiak szerint:

```
mysql> call list_racer(0);
+----+--------------+-----+---------+
| id | name         | age | generic |
+----+--------------+-----+---------+
|  2 | Nagy Peti    |  25 | F       |
|  3 | Balogh Judit |  24 | N       |
|  1 | Kis Andi     |  30 | N       |
+----+--------------+-----+---------+
3 rows in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> call list_racer(1);
+----+-----------+-----+---------+
| id | name      | age | generic |
+----+-----------+-----+---------+
|  2 | Nagy Peti |  25 | F       |
+----+-----------+-----+---------+
1 row in set (0.02 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> call list_racer(2);
+----+--------------+-----+---------+
| id | name         | age | generic |
+----+--------------+-----+---------+
|  3 | Balogh Judit |  24 | N       |
|  1 | Kis Andi     |  30 | N       |
+----+--------------+-----+---------+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Jegyezzük meg, hogy a fenti feladatot függvénnyel nem lehetne megoldani, mert a függvény nem tud eredményhalmazt értékként visszaadni. Csak az eljárások tudnak visszaadni eredményhalmazt (`select`).

3.4.7. Sportolók keresése tárolt eljárással

Írjunk két eljárást, ahol az egyik azonosítószám szerint, a másik pedig név szerint keresi meg a versenyzőt a `racer` táblában, majd kilistázza a talált rekordot.

Keresés név szerint:

```
mysql> delimiter //
mysql> create procedure search_racer(pname varchar(30))
    -> begin
    ->   select * from racer where name like concat('%',pname,'%');
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Nézzük egy példát a fenti eljárásra:

```
mysql> call search_racer('jud');
+----+--------------+-----+---------+
| id | name         | age | generic |
+----+--------------+-----+---------+
|  3 | Balogh Judit |  24 | N       |
+----+--------------+-----+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Keresés azonosítószám (`id` mező) szerint:

```
mysql> delimiter //
mysql> create procedure search_racer2(pid int)
    -> begin
    ->   select * from racer where id=pid;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Hívjuk meg az eljárást:

```
mysql> call search_racer2(2);
+----+-----------+-----+---------+
| id | name      | age | generic |
+----+-----------+-----+---------+
|  2 | Nagy Peti |  25 | F       |
+----+-----------+-----+---------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

3.4.8. Új eredmények felvitele a result táblába tárolt eljárással

A következő eljárással új rekordot tudunk beszúrni a `result` táblába:

```
mysql> delimiter //
mysql> create procedure new_result(prid int,prun int,pfence int,pshoot int,
                                   priding int,pswim int, pyear year)
    -> begin
    ->   declare x int default 0;
    ->   select count(*) into x from result where rid=prid and r_year=pyear;
    ->   if x=0 then insert into result values(null,prid,prun,pfence,pshoot,
                                               priding,pswim,0,pyear); end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Az eljárásban először egy ellenörzést végzünk. Megnézzük, hogy szerepel-e már az adott sportoló az adott évben. Ha nem, akkor engedélyezzük az új rekord felvitelét a `result` táblába.

Vigyünk fel néhány új tételt az eljárás segítségével:

```
mysql> call new_result(1,6,6,7,7,8,'2006');
Query OK, 1 row affected (0.02 sec)

mysql> call new_result(2,5,8,8,6,9,'2006');
Query OK, 1 row affected (0.00 sec)

mysql> call new_result(3,6,6,7,8,10,'2006');
Query OK, 1 row affected (0.02 sec)
```

A `racer` táblában szereplő versenyzőkhöz felvittünk egy-egy tételt a `result` táblába. Listázzuk ki a `result` táblát:

```
mysql> select * from result;
+----+-----+-----+-------+-------+--------+------+----------+--------+
| id | rid | run | fence | shoot | riding | swim | position | r_year |
+----+-----+-----+-------+-------+--------+------+----------+--------+
|  1 |   1 |   6 |     6 |     7 |      7 |    8 |        0 |   2006 |
|  2 |   2 |   5 |     8 |     8 |      6 |    9 |        0 |   2006 |
|  3 |   3 |   6 |     6 |     7 |      8 |   10 |        0 |   2006 |
+----+-----+-----+-------+-------+--------+------+----------+--------+
3 rows in set (0.00 sec)
```

A `position` mezők értékét egyelőre lenullázzuk. A későbbiekben írunk majd egy eljárást, amely kiszámolja a versenyzők helyezését (pozícióját) az elért összpontszám szerint és azt beírja majd a `position` mezőbe (a `result` táblában).

3.4.9. Eredmények módosítása tárolt eljárással

Csak az egyes sportágakban elért pontszámokat lehet módosítani az eljárásban, a többi mező értékét nem engedjük megváltoztatni:

```
mysql> delimiter //
mysql> create procedure upd_result(pid int,prun int,pfence int,pshoot int,
                                   priding int,pswim int)
    -> begin
    ->   update result set run=prun,fence=pfence,shoot=pshoot,
                           riding=priding,swim=pswim where id=pid;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Próbaként módosítsuk az elsőként felvitt rekord pontszámait:

```
mysql> call upd_result(1,5,5,5,6,6);
Query OK, 1 row affected (0.00 sec)
```

Ha kilistázzuk a táblát, akkor láthatjuk, hogy valóban megtörtént a módosítás:

```
mysql> select * from result;
+----+-----+-----+-------+-------+--------+------+----------+--------+
| id | rid | run | fence | shoot | riding | swim | position | r_year |
+----+-----+-----+-------+-------+--------+------+----------+--------+
|  1 |   1 |   5 |     5 |     5 |      6 |    6 |        0 |   2006 |
|  2 |   2 |   5 |     8 |     8 |      6 |    9 |        0 |   2006 |
|  3 |   3 |   6 |     6 |     7 |      8 |   10 |        0 |   2006 |
+----+-----+-----+-------+-------+--------+------+----------+--------+
3 rows in set (0.00 sec)
```

3.4.10. Rekord törlése a result táblából tárolt eljárással

Ha törölni kívánunk egy tételt, akkor azt az alábbi eljárással tehetjük meg:

```
mysql> delimiter //
mysql> create procedure del_result(pid int)
    -> begin
    ->   delete from result where id=pid;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

3.4.11. Versenyzők eredményeinek kilistázása

Két eljárást fogunk készíteni. Az egyik a `racer` és a `result` táblák összekapcsolásával listázza ki a versenyzők eredményeit, a másik csak egy adott sportoló eredményeit írja ki.

Az első eljárás az alábbiak szerint néz ki:

```
mysql> delimiter //
mysql> create procedure list_result(x int)
    -> begin
    ->   case x
    ->     when 0 then select result.id as reid,rid,name,age,generic,run,fence,s
hoot,riding,swim,(run+fence+shoot+riding+swim)/5 as aver,(run+fence+shoot+riding
+swim) as total,position,r_year from racer,result where racer.id=rid order by ge
neric,r_year desc,position asc;
    ->     when 1 then select result.id as reid,rid,name,age,generic,run,fence,s
hoot,riding,swim,(run+fence+shoot+riding+swim)/5 as aver,(run+fence+shoot+riding
+swim) as total,position,r_year from racer,result where racer.id=rid and generic
='F' order by generic,r_year desc,position asc;
    ->     when 2 then select result.id as reid,rid,name,age,generic,run,fence,s
hoot,riding,swim,(run+fence+shoot+riding+swim)/5 as aver,(run+fence+shoot+riding
+swim) as total,position,r_year from racer,result where racer.id=rid and generic
='N' order by generic,r_year desc,position asc;
    ->   end case;
    -> end;
    -> //
Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;
```

A bemenő paramétertől függően vagy mindenkit kilistáz, vagy csak a férfiakat ill. csak a nőket listázza ki az eljárás.

Az alábbi eljárás pedig egy adott sportoló eredményeit jeleníti meg a `result` táblából:

```
mysql> delimiter //
mysql> create procedure list_result2(pid int)
    -> begin
    ->   select run,fence,shoot,riding,swim,(run+fence+shoot+riding+swim)/5
         as aver,(run+fence+shoot+riding+swim) as total,position,r_year
         from result where rid=pid order by r_year desc,position asc;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

A fenti eljárás bemeneti paramétere a versenyző egyedi azonosítója (`id`) lesz. Ha találunk a paraméternek megfelelő értéket a `result` tábla `rid` mezőjében, akkor azokat a rekordokat kilistázzuk.

A következő eljárás ugyanazt csinálja, mint az előző csak itt az adott eredményrekord (sor) azonosítóját (`id`) is kiírjuk:

```
mysql> delimiter //
mysql> create procedure list_result3(pid int)
    -> begin
    ->   select id,run,fence,shoot,riding,swim,(run+fence+shoot+riding+swim)/5
        as aver,(run+fence+shoot+riding+swim) as total,position,r_year
        from result where rid=pid order by r_year desc,position asc;
    -> end;
    -> //
Query OK, 0 rows affected (0.03 sec)

mysql> delimiter ;
```

3.4.12. Versenyzők helyezésének meghatározása tárolt eljárással

Most pedig írunk egy eljárást, melynek segítségével kiszámoljuk a versenyzők helyezését az elért összpontszámuk szerint, majd a `result` tábla `position` mezőjébe beírjuk ezt a kiszámolt értéket.

Íme az eljárás:

```
mysql> delimiter //
mysql> create procedure rposition()
    -> begin
    ->   declare done int default 0;
    ->   declare pid,ppos,ptotal,ototal int;
    ->   declare pyear,oyear year;
    ->   declare pgen,ogen char(1);
    ->   declare cur cursor for select result.id,generic,r_year,(run+fence
         +shoot+riding+swim) as total from racer,result where racer.id=rid
         order by generic,r_year desc,total desc;
    ->   declare continue handler for not found set done=1;
    ->   set ppos=0;
    ->   set pgen='F';
    ->   set ogen='F';
    ->   set pyear='1980';
    ->   set oyear='1980';
    ->   set ptotal=0;
    ->   set ototal=0;
    ->   open cur;
    ->   repeat
    ->     fetch cur into pid,pgen,pyear,ptotal;
    ->     if pgen<>ogen then set ogen=pgen; set ppos=0; end if;
    ->     if pyear<>oyear then set oyear=pyear; set ppos=0; end if;
    ->     set ppos=ppos+1;
    ->     if (ptotal=ototal) and (ppos>1) then set ppos=ppos-1; end if;
    ->     set ototal=ptotal;
    ->     if not done then update result set position=ppos where id=pid;
    ->     end if;
    ->   until done end repeat;
    ->   close cur;
    -> end;
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Elemezzük az `rposition` nevű eljárást egy kicsit részletesebben. A deklarációs részben két új dologgal is találkoztunk:

- A `declare cur cursor for select...` sorral egy kurzor típusú változót adtunk meg, amelyben a lekérdezés (`select`) eredményhalmazát tároljuk el.
- A `declare continue handler...` sorral pedig egy kivételkezelőt deklaráltunk, amely a későbbi `repeat` ciklusból való kilépéshez kell. Vagyis amikor feldolgoztuk a kurzor minden sorát, akkor egy `not found` kivétel keletkezik, ami azt jelenti, hogy nincs több feldolgozandó sor, mert a kurzorban eltárolt eredményhalmaz végére értünk. Ekkor a `done` változó értékét 1-re állítjuk és így lépünk ki a `repeat` ciklusból.

A `cur` nevű kurzor feldolgozásának lépései a következők:

- `open cur;` először megnyitjuk a kurzort
- `fetch cur into...;` a kurzorban eltárolt eredményhalmaz (rekordhalmaz) következő sorát (rekordját) adja meg, melynek mezőértékei bekerülnek a megadott változókba (`pid, pgen, pyear, ptotal`)
- `close cur;` a kurzort feldolgozás után lezárjuk

A versenyzők adott évben elért helyezésének kiszámítása az eljárás `repeat` ciklusában történik meg. A kurzorban eltárolt rekordokat a versenyzők neme szerint, ezen belül a verseny éve szerint csökkenő sorrendbe, ezen belül pedig az összpontszám szerint csökkenő sorrendbe rendeztük. Nem véletlenül rendeztük így a rekordokat, hiszen külön kell számolni a férfiak és külön a nők elért helyezését. Arra az esetre is figyelni kell, hogy azonos pontszámmal rendelkező versenyzőknek azonos helyezést adjunk. Végül a kiszámolt helyezést (`ppos`) beírjuk a versenyző `position` mezőjébe (a `result` táblában).

Amikor minden versenyzőnek kiszámoltuk és beírtuk a helyezését, akkor a `done` változó értékét 1-re állítjuk, így kilépünk a ciklusból.

3.4.13. Séma adatok lekérdezése

Most pedig listázzuk ki az eddig létrehozott eljárásokat és függvényeket a következőképpen:

```
mysql> select name,type from mysql.proc where db='racing';
+---------------+-----------+
| name          | type      |
+---------------+-----------+
| del_racer     | PROCEDURE |
| del_result    | PROCEDURE |
| list_racer    | PROCEDURE |
| list_result   | PROCEDURE |
| list_result2  | PROCEDURE |
| new_racer     | PROCEDURE |
| new_result    | PROCEDURE |
| num_racer     | FUNCTION  |
| num_racer     | PROCEDURE |
| rposition     | PROCEDURE |
| search_racer  | PROCEDURE |
| search_racer2 | PROCEDURE |
| upd_racer     | PROCEDURE |
| upd_result    | PROCEDURE |
+---------------+-----------+
14 rows in set (0.00 sec)
```

Kilistáztuk a `racing` adatbázisban létrehozott összes eljárást és függvényt a `mysql` rendszeradatbázis `proc` táblájából. A tárolt alprogramjaink mindig a `proc` táblában kerülnek eltárolásra.

3.5. Jogosultságok beállítása

A tárolt eljárások és függvények nagy előnye a hatékonyságon kívül az, hogy könnyen rendelhetünk hozzájuk jogosultságokat. A jogosultságokkal meghatározhatjuk, hogy adott felhasználó melyik eljárás és függvény futtatására jogosult. Ezzel ki tudjuk zárni az illetéktelen felhasználókat. Jogosultságokat a `grant` utasítással adhatunk és a `revoke` utasítással vonhatunk vissza. (Később a kliens alkalmazásainkban azokat a felhasználókat kell megadni az adatbázis-kapcsolat beállításánál, akik jogosultságot kaptak a tárolt alprogramok futtatására.)

Hozzunk létre egy felhasználót, aki jogosult minden eljárás és függvény futtatására:

```
mysql> grant execute on racing.* to 'jimi'@'localhost' identified by 'pwd';
Query OK, 0 rows affected (0.02 sec)

mysql> flush privileges;
```

Létrehozhatunk olyan felhasználókat is, akik csak bizonyos eljárások vagy függvények meghívására jogosultak:

```
mysql> grant execute on procedure list_racer to 'john'@'localhost' identified by
  'pwd';
Query OK, 0 rows affected (0.05 sec)
```

A john nevű felhasználó csak a `list_racer` eljárást futtathatja.

```
mysql> grant execute on function num_racer to 'steve'@'localhost' identified by
'pwd';
Query OK, 0 rows affected (0.00 sec)
```

A steve nevű felhasználó pedig csak a `num_racer` függvényt hívhatja meg.

A `grant` utasításról részletes leírás található a MySQL dokumentumában.