Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások fejlesztése I

9. előadás

Adatbázis-kezelés elemi eszközökkel

Giachetta Roberto

http://people.inf.elte.hu/groberto

A MySQL adatbázis-kezelő
Elérhetősége

- A *MySQL* az egyik legnépszerűbb SQL alapú relációs adatbázis-kezelő motor, jelenleg az Oracle tulajdonában van
  - letölthető a `www.mysql.com` honlapról
  - elérhető ingyenes és kereskedelmi változata is
  - több platformon elérhető, legfrissebb változata az 5.5
  - grafikus kezelőfelülete a *MySQL Workbench*
  - több programozási nyelvhez is ad elérési felületet a *MySql Connector* segítségével (pl. C++, Java, .NET)

A MySQL adatbázis-kezelő
Használata

- A MySQL beépített felhasználó-kezeléssel rendelkezik
  - a rendszergazda felhasználó a `root`, a többi felhasználónak tetszőlegesen szabályozható a jogosultsága
  - felhasználót létrehozni a `create user` utasítással, jogosultságot felvenni a `grant privileges` utasítással lehet, pl.:

```
CREATE USER 'root'@'localhost' IDENTIFIED BY
   'asd123';
GRANT ALL PRIVILEGES ON *.* TO
   'tanulo'@'localhost' WITH GRANT OPTION;
```

  - a felhasználó megadásával léphetünk be a konzol felületre:

```
mysql –u root -p
```

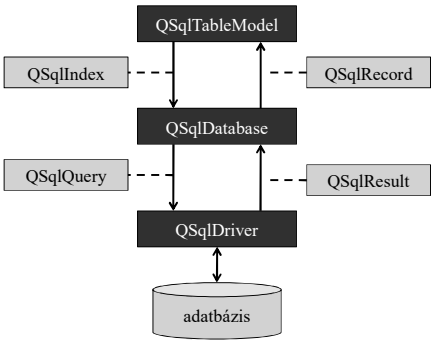

A MySQL adatbázis-kezelő
Használata

- Általános MySQL utasítások:
  - `status`: adatbázisszerver állapotának lekérdezése
  - `show databases`: adatbázisok listázása
  - `use <adatbázis>`: adatbázis használatba vétele
  - `show tables`: aktuális adatbázis tábláinak listázása
  - `desc <táblanév>`: tábla oszlopainak listázása
  - `show table status like '<táblanév>'`: tábla aktuális státuszának lekérdezése
  - `set password = '<jelszó>'`: jelszóváltás

Adatbázis-kezelés elemi eszközökkel
A Qt adatbázis-kezelő modul

- A *QtSQL* modul biztosítja az SQL alapú adatbázisok kezelésének osztályait, amelyeket három csoportba sorolunk
  - a modell osztályok biztosítják a logikai adatbázist, és az interfészt a felületi réteg számára: `QSqlQueryModel`, `QSqlTableModel`, `QSqlRelationalModel`
  - az alkalmazásprogramozói (API) osztályok biztosítják az SQL elemek kezelését: `QSqlDataBase`, `QSqlQuery`, `QSqlError`, `QSqlField`, `QSqlIndex`, `QSqlRecord`
  - a meghajtó osztályok biztosítják az adatbázis elérését és a kommunikációt: `QSqlDriver`, `QSqlDriverCreator<T>`, `QSqlDriverCreatorBase`, `QSqlDriverPlugin`, `QSqlResult`

Adatbázis-kezelés elemi eszközökkel
A Qt adatbázis-kezelő modul



Adatbázis-kezelés elemi eszközökkel
A Qt adatbázis-kezelő modul

- A modul osztályainak használatához a megfelelő osztályt kell meghivatkozni az aktuális fájlban, illetve lehetőség van a teljes modul betöltésére is:
  `#include <QtSql>`
- A modul alapból nem érhető el egy alap Qt alkalmazásban, használatát a projektfájlban jeleznünk kell a `QT += sql` utasítással
  - a betöltendő modulok egy sorban is lehetnek, pl.
    `QT += core gui widgets sql`
  - amennyiben konzol felületen szerkesztünk, a projektfájl létrehozásakor is hozzáadhatjuk a modult:
    `qmake -project "QT += sql"`

Adatbázis-kezelés elemi eszközökkel
Adatbázis meghajtók

- A QtSQL modul több beépített meghajtót is tartalmaz a különböző adatbázis-motorok kezelésére, valamint felületet biztosít további meghajtók készítésére, pl.:
  - `QMYSQL`: MySQL
  - `QSQLITE2`, `QSQLITE`: SQLite
  - `QOCI`: Oracle Call Interface Driver
  - `QODBC`: Open Database Connectivity (ODBC), Microsoft SQL Serverhez, valamint más ODBC adatforrásokhoz
- Nincs minden meghajtó előre telepítve, bizonyos esetekben további csomagként (pl. `libqt5sql5-mysql`) kell hozzáadnunk őket

Adatbázis-kezelés elemi eszközökkel
Adatbázis kapcsolat

- Adatbázismotort betölteni, és kapcsolatot létesíteni a `QSqlDatabase` osztály segítségével tudunk, pl.:

```
QSqlDatabase db =
    QSqlDatabase::addDatabase("QMYSQL");
db.setHostName("localhost"); // szerver
db.setDatabaseName("myDatabase"); // adatbázis
db.setUserName("root"); // felhasználónév
db.setPassword("root"); // jelszó
```

  - a betöltést az `addDatabase()` statikus metódussal végezzük a meghajtó megadásával, beállíthatjuk a szervert, az adatbázist, valamint a felhasználó adatait
  - a rendelkezésre álló meghajtókat a `drivers()` metódussal kérhetjük le

Adatbázis-kezelés elemi eszközökkel
Adatbázis kapcsolat

- Lehetőségünk van több kapcsolatot is kezelni a programban (statikus metódusokkal):
  - a kapcsolatok elnevezéssel rendelkeznek, ezen keresztül érhetjük el őket: `addDatabase(<meghajtó>, <név>)`
  - kapcsolatok listázhatóak (`connectionNames()`), lekérhetőek (`database(<név>)`), törölhetőek (`removeDatabase(<név>)`)
- Kapcsolatot megnyitni az `open()`, bezárni a `close()` művelettel tudunk
  - ha nem sikerül a megnyitás, hamissal tér vissza
  - a program bezárása nem zárja be a nyitott kapcsolatokat

Adatbázis-kezelés elemi eszközökkel
Parancskiadó objektumok

- SQL utasításokat `QSqlQuery` segítségével futtathatunk, pl.:

```
QSqlQuery query;
if (query.exec("select id, data from myTable"))
   // lekérdezés futtatása
   // … eredmények kiírása
else // sikertelen futtatás
   cout << query.lastError().text();
```

  - a konstruktorban megadható paraméterként az adatbázis kapcsolat, ha nem adjuk meg, az alapértelmezettet használja
  - az `exec()` művelettel tetszőleges utasítást végrehajthatunk, és igazzal tér vissza, amennyiben sikerült végrehajtania
  - a `lastError()` segítségével lekérdezhetjük a hiba okát

Adatbázis-kezelés elemi eszközökkel
Lekérdezések olvasása

- Amennyiben lekérdezést hajtottunk végre, az eredményt soronként kezeljük, azaz egyszerre nem látjuk a teljes eredményt csak egy sorát
  - lépegetni a `first()`, `next()`, `previous()`, `last()` utasításokkal
  - adott sorra ugrani a `seek(<sorszám>)` metódussal, mindegyik igazat ad, ha tudott lépni
  - alapból az eredmény első sora előtt állunk (tehát rögtön léptetni kell)
  - amennyiben csak előre akarunk lépkedni, a `setForwardOnly()` metódus optimalizálja a lekérdezést

Adatbázis-kezelés elemi eszközökkel

Lekérdezések olvasása

- értéket lekérdezni a kijelölt sorból a `value(<oszlopszám>)` metódussal tudunk
  - az érték `QVariant` típusú, így az tovább konvertálható alkalmas formára
- a `size()` megadja a lekérdezett sorok számát

- Pl.:

```
while (query.next())
{
   cout << query.value(0).toString();
      // az első oszlop egész számot tartalmaz
   cout << query.value(1).toInt();
      // a második oszlopot szövegesen kérjük le
}
```


Adatbázis-kezelés elemi eszközökkel

Példa

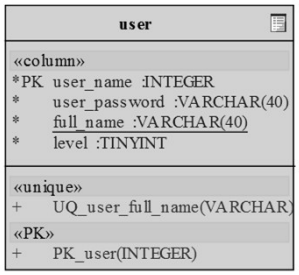
Feladat: Készítsünk egyszerű konzol alkalmazást, amely alkalmas az apartman adatbázis (`apartments`) felhasználók (`user`) táblájának beolvasására, sorok törlésére, valamint új értékek bevitelére.

- a program csak megfelelő felhasználónév/jelszó megadásával engedi elvégezni a tevékenységeket
- a program lekérdezés segítségével azonosít, beolvassa a táblatartalmat (azonosító, név, jelszó, szint), kilistázza, lehetőséget ad azonosító alapján törlésre, és beszúrásra
- a programot vezéreljük menün keresztül, ehhez hozzunk létre egy menü osztályt

Adatbázis-kezelés elemi eszközökkel

Példa

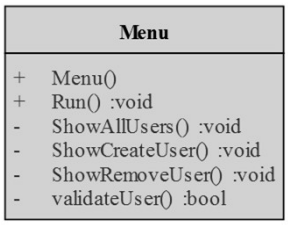
Tervezés (adatbázis):



Adatbázis-kezelés elemi eszközökkel

Példa

Tervezés (alkalmazás):



Adatbázis-kezelés elemi eszközökkel

Példa

Megvalósítás (`main.cpp`):

```
int main(int argc, char *argv[]){
   …
   QSqlDatabase db =
      QSqlDatabase::addDatabase("QMYSQL");
   … // kapcsolat létrehozása
   if (db.open()) { // kapcsolat megnyitása
      … // sikeres kapcsolódás
      Menu m; m.Run(); // menü futtatása
      db.close(); // kapcsolat bezárása
   }
   else { … } // sikertelen kapcsolódás
   return 0;
}
```


Adatbázis-kezelés elemi eszközökkel

Példa

Megvalósítás (`menu.cpp`):

```
void Menu::ShowRemoveUser() {
   …
   getline(cin, userNameString);
   QSqlQuery removeQuery; // törlő utasítás
   removeQuery.exec("delete from user where
                     user_name = '" +
         QString::fromStdString(userNameString) +
                     "'"); // törlés végrehajtása
}
```

Adatbázis-kezelés elemi eszközökkel
SQL injekció

- Az adatbázisban tárolt adatokat meg kell óvni az idegenek elől, biztonságossá kell tenni az adatokhoz való hozzáférést
  - a programok az adatbázis-szerveren SQL lekérdezéseket futtatnak, amelyeket a programkódban összeállítanak szövegként
  - az összeállítás során törekedni kell arra, hogy ne lehessen manipulálni az utasítást úgy, hogy az illetéktelen felhasználók hozzáférhessenek a tárolt adatokhoz
- Az SQL parancsok manipulációját nevezzük *SQL injekció*nak (*SQL injection*)
  - leggyakoribb webes környezetben

Adatbázis-kezelés elemi eszközökkel
SQL injekció

- Pl.:

```
// felhasználónév/jelszó bekérése
QSqlQuery query;
query.exec("select user_name from users where
            user_name = '" + userName + "' and
            password = '" + password + "'");

// userName = "", password = "' or '1' = '1"
// esetén:
query.exec("select user_name from users where
            user_name = '' and password = ''
            or '1' = '1'");
   // a feltétel minden sorra igazat ad, az egész
   // táblát lekéri
```


Adatbázis-kezelés elemi eszközökkel
SQL injekció

- Az SQL injekciók kivédésének több módja van:
  - a felhasználótól kapott értékek ellenőrzése, módosítása, pl.:
    `userName = userName.remove("'");`
  - felhasználó által megadható adatok korlátozása, pl.:

```
lineEdit.setInputMask("aaaaaaaaaaaaaaa");
   // csak alfabetikus karaktereket fogad el
lineEdit.setValidator(
   QRegExpValidator("[A-Za-z0-9]{1,8}"));
   // 1-8 db alfanumerikus karaktert fogad el
```

  - paraméteres utasítások használata
  - az injekciót eleve kizáró megoldások használata (pl. modell-nézet architektúra)

Adatbázis-kezelés elemi eszközökkel
Utasítások paraméterezése

- Az utasítás paraméterezhető, azaz előre legyárthatjuk az utasítást a `prepare(<utasítás>)` utasítással
  - a paramétereket a `bindValue(<paraméter>, <érték>)` metódussal helyettesíthetjük be tényleges értékekre
  - pl.:

```
QSqlQuery query;
query.prepare("INSERT INTO myTable(id, data)" +
              "VALUES (:id, :data)");
query.bindValue(":id", 1); // paraméter beírása
query.bindValue(":data", "something");
query.exec(); // végrehajtás
```

  - a behelyettesítéskor átkódolja a kapott szöveget

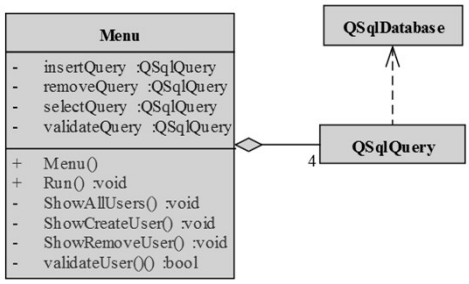
Adatbázis-kezelés elemi eszközökkel
Példa

Feladat: Készítsünk egyszerű konzol alkalmazást, amely alkalmas az apartman adatbázis (`apartments`) felhasználók (`user`) táblájának beolvasására, sorok törlésére, valamint új értékek bevitelére.

- a program csak megfelelő felhasználónév/jelszó megadásával engedi elvégezni a tevékenységeket
- a program paraméteres utasításokat fog használni, kikerülve az SQL injekció lehetőségét
- azonosít, beolvassa a táblatartalmat (azonosító, név, jelszó, szint), kilistázza, lehetőséget ad azonosító alapján törlésre, és beszúrásra

Adatbázis-kezelés elemi eszközökkel
Példa

Tervezés:



Adatbázis-kezelés elemi eszközökkel

Példa

Megvalósítás (`menu.cpp`):

```
bool Menu::validateUser(){
    …
    validateQuery.bindValue(":name",
       QString::fromStdString(userName));
    validateQuery.bindValue(":password",
       QString::fromStdString(userPassword));
       // paraméterek behelyettesítése
    validateQuery.exec(); // lekérdezés futtatása

    return validateQuery.next();
    // ha van eredmény, akkor sikeres a lekérdezés
}
```