Lecture SQL08
Qt and SQL – Part III

continued

Outline

- Last Time
  - **QSqlDatabase** Class
    - Default connection
    - Named connections
  - QSqlTableModel/QTableView
- This Time
  - **DROP TABLE**
  - **ALTER TABLE**

DROP TABLE Statement

- Deletes table and data within table
- Some DBMS provide a rollback mechanism

DROP TABLE - 1

```
#include <QtCore/QCoreApplication>
#include <QtSql>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(":memory:");

    if (!db.open())
        qDebug() << db.lastError();

    QSqlQuery q;

    // Create and populate table
    q.exec("CREATE TABLE customers (uid INTEGER, lastname TEXT, firstname TEXT);");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (128, 'Smith', 'John');");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (324, 'Doe', 'John');");
    q.exec("INSERT INTO customers VALUES (245, 'Jones', 'Mark');");
    q.exec("INSERT INTO customers VALUES (756, 'Smith', 'Jane');");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Moore', 'Sara', 459);");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Parks', 'Ralph', 721);");
```

← Create database in memory

DROP TABLE - 2

```
    // Print all rows from table
    qDebug() << "************** Start **************";
    q.exec("SELECT * FROM customers;");
    while (q.next())
        qDebug() << "(" << q.value(0).toInt()
                 << ", " << q.value(1).toString()
                 << ", " << q.value(2).toString() << ")";
    qDebug() << "**************  End  ***************";

    // Attempt to delete table
    q.exec("DROP TABLE customers;");

    // Print all rows from table
    qDebug() << "************** Start **************";
    q.exec("SELECT * FROM customers;");
    while (q.next())
        qDebug() << "(" << q.value(0).toInt()
                 << ", " << q.value(1).toString()
                 << ", " << q.value(2).toString() << ")";
    qDebug() << "**************  End  ***************";

    return a.exec();
}
```

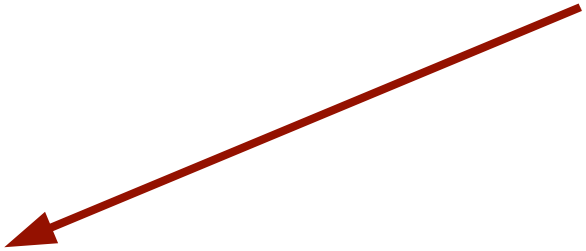
```
Sample Output
************** Start **************
( 128 , "Smith" , "John" )
( 324 , "Doe" , "John" )
( 245 , "Jones" , "Mark" )
( 756 , "Smith" , "Jane" )
( 459 , "Moore" , "Sara" )
( 721 , "Parks" , "Ralph" )
************** End **************
************** Start **************
QSqlError(1, "Unable to execute
statement", "no such table:
customers")
************** End **************
```

ALTER TABLE Statement

- Used to add or remove columns from table

ALTER TABLE - 1

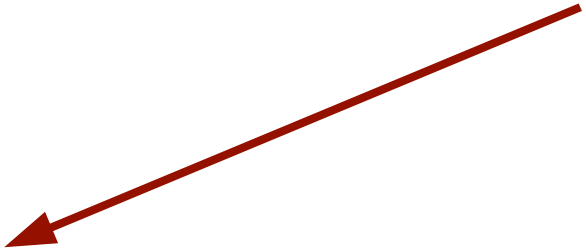
```
#include <QtCore/QCoreApplication>
#include <QtSql>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(":memory:");

    if (!db.open())
        qDebug() << db.lastError();

    QSqlQuery q;

    // Create and populate table
    q.exec("DROP TABLE customers;");
    q.exec("CREATE TABLE customers (uid INTEGER, lastname TEXT, firstname TEXT);");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (128, 'Smith', 'John');");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (324, 'Doe', 'John');");
    q.exec("INSERT INTO customers VALUES (245, 'Jones', 'Mark');");
    q.exec("INSERT INTO customers VALUES (756, 'Smith', 'Jane');");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Moore', 'Sara', 459);");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Parks', 'Ralph', 721);");
```



Remove any previous table named customers

ALTER TABLE - 1

```
#include <QtCore/QCoreApplication>
#include <QtSql>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName(":memory:");

    if (!db.open())
        qDebug() << db.lastError();

    QSqlQuery q;

    // Create and populate table
    q.exec("DROP TABLE customers;");
    q.exec("CREATE TABLE customers (uid INTEGER, lastname TEXT, firstname TEXT);");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (128, 'Smith', 'John');");
    q.exec("INSERT INTO customers (uid, lastname, firstname) VALUES (324, 'Doe', 'John');");
    q.exec("INSERT INTO customers VALUES (245, 'Jones', 'Mark');");
    q.exec("INSERT INTO customers VALUES (756, 'Smith', 'Jane');");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Moore', 'Sara', 459);");
    q.exec("INSERT INTO customers (lastname, firstname, uid) VALUES ('Parks', 'Ralph', 721);");
```



Remove any previous table named customers

ALTER TABLE - 2

```
// Print all rows from table
qDebug() << "************** Start **************";
q.exec("SELECT * FROM customers;");
while (q.next())
    qDebug() << "(" << q.value(0).toInt()
             << ", " << q.value(1).toString()
             << ", " << q.value(2).toString() << ")";

q.exec("SELECT balance FROM customers;");
qDebug() << q.lastError();
qDebug() << "**************  End  ***************";
```

```
Sample Output
************** Start **************
( 128 , "Smith" , "John" )
( 324 , "Doe" , "John" )
( 245 , "Jones" , "Mark" )
( 756 , "Smith" , "Jane" )
( 459 , "Moore" , "Sara" )
( 721 , "Parks" , "Ralph" )
QSqlError(1, "Unable to execute statement", "no such column: balance")
************** End **************
```

ALTER TABLE - 3

```
// Attempt to add column
q.exec("ALTER TABLE customers ADD balance INTEGER;");

// Print all rows from table
qDebug() << "************** Start **************";
q.exec("SELECT * FROM customers;");
while (q.next())
    qDebug() << "(" << q.value(0).toInt()
             << ", " << q.value(1).toString()
             << ", " << q.value(2).toString()
             << ", " << q.value(3).toInt() << ")";
qDebug() << q.lastError();
qDebug() << "**************  End  ***************";
```

```
Sample Output
************** Start **************
( 128 ,  "Smith" ,  "John" ,  0 )
( 324 ,  "Doe" ,  "John" ,  0 )
( 245 ,  "Jones" ,  "Mark" ,  0 )
( 756 ,  "Smith" ,  "Jane" ,  0 )
( 459 ,  "Moore" ,  "Sara" ,  0 )
( 721 ,  "Parks" ,  "Ralph" ,  0 )
QSqlError(-1, "", "")

**************  End  ***************
```