Practice 7.

1. Filling of Database.
1.1. You need to enter information into the database about the students who do this work, about you.

1.2. Change the displayed names for the table columns to the correct ones, for example: Name, Surname, etc.

1.3. Make the primary key column invisible.
To do this, print ***ui,*** click on the "***dot***" - and list of available elements will appear.
In it, select the ***tableView***, click "***Hide Column***" again.

1.4. Add the "***Average mark***" column to the table.
Sort the table by this item.

1.5. Resize the window, add the "***Exit***" button, implement it.
1.6. Add a column "***Date of birth***".

1.7. Create a shell for working with the database (like on fig.1):
- it should be possible to go to the next (Next), previous (Previous), first (First) and last (Last) records of the table (see item 2. in the Practice 7);
–the name of the table should be displayed in the title of the window;
–all fields must be displayed on the form;
- it should be possible to add a new student directly from the form.

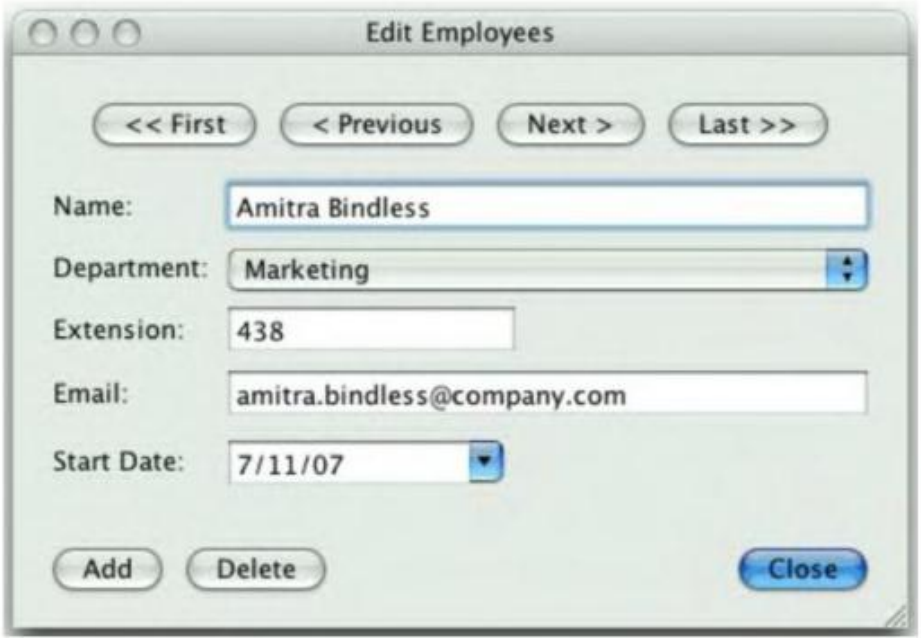

Figure 1 - Example for working with another table

2. Navigation through the records in the table.

2.1. Mapper.

In order to navigate through the records in the table, you need to use the ***QDataWidgetMapper*** class.

Create its description in the class (**mainwindow.h file, private section**):

```
QDataWidgetMapper * mapper;
```

//it must be connected in the same place:

```
#include <QDataWidgetMapper>
```

//and mention it in the constructor (**mainwindow.cpp, createConnection () function**) by //linking to the model:

```
mapper = newQDataWidgetMapper (this);
mapper-> setModel (model);
```

Switch to visual editing mode (by double clicking on ***mainwindow.ui***).

Break the layout  and narrow (reduce) the table.

Drag *LineEdit* and ***Label elements*** onto the form.

Use the *Ctrl* key to select both widgets and arrange them horizontally  .

Using the properties window (Fig.2), change the name ***LineEdit*** from *lineEdit* to ***nameEdit***. Change the text property of the **Label** to "***Name"***.

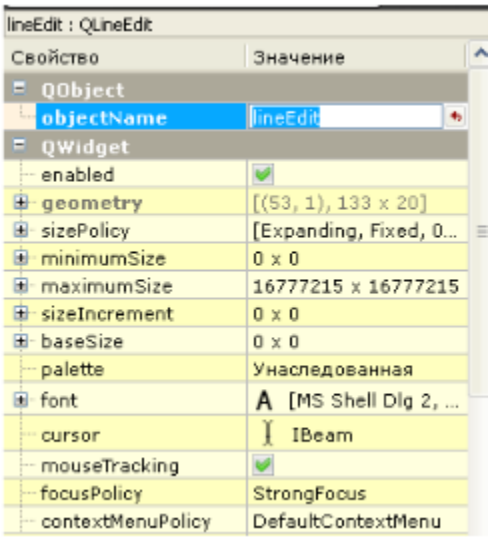

Figure 2.

Return to ***mainwindow.cpp***, and add:

```
mapper->addMapping(ui->nameEdit,model->fieldIndex("firstname"));
mapper->toFirst();
```

Please note - as an example, a connection is established with only one field "Name". You need to make cjnnections with other fields by yourself.
Run and test the application.

2.2. Navigation buttons.
Drag a ***PushButton*** element onto the form, name it ***next_Button.***
Change the displayed text on the "***Forward"***, "***Next"***, or "***>>"*** key.
Click on the button with the right mouse button, select "***Go to slot*** ...".
In the dialog box that opens, select the signal ***clicked ()***and click ***OK***.
In the function that appears, write:

```
mapper->toNext();
```

Do the same for the rest of the buttons: ***toPrevious(), toFirst(), toLast().***

! The buttons should be named correctly, i.e. **NOT** pushButton1, pushButton2...a previous_Button, first_Button...

For the delete button:

```
model->removeRow(mapper->currentIndex());
model->select();
```

3.Creating other SQL queries.
You can also use the following code to fill it in:

```
QSqlQueryquery;
query.prepare("INSERTINTOperson(id,firstname,lastname)"
"VALUES(:id,:firstname,:lastname);");
//query.bindValue(":id","7");
 // id will be updated by itself
query.bindValue(":firstname","Garry");
query.bindValue(":lastname","Potter");
query.exec();model->select();
mapper->toLast();
```

This SQL query can be replaced with operations on the model:

```
model->insertRows(0,1);
model->setData(model->index(0,1),"Tyrion");
model->setData(model->index(0,2),"Lannister");
model->setData(model->index(0,3),5);
```

```
model->submitAll();
model->select();//if you comment it, the up of the table will be added
mapper->toLast();
```

Let's say you want to make a query like this:
SELECT * FROM person WHERE ball>= 1 ORDER BY id DESC

You can do it in this way:

```
query.exec("SELECT*FROMpersonWHEREball>=1ORDERBYballDESC");
```

Or, it is enough to set a filter and a sorting condition:

```
model.setTable("person");// Database table name
model.setFilter("ball>=1");//condition WHERE.
model.setSort(0,Qt::DescendingOrder);// Sort by descending id.
model.select();//to get data
```

4. Save the table on compute, not in the operating memory.