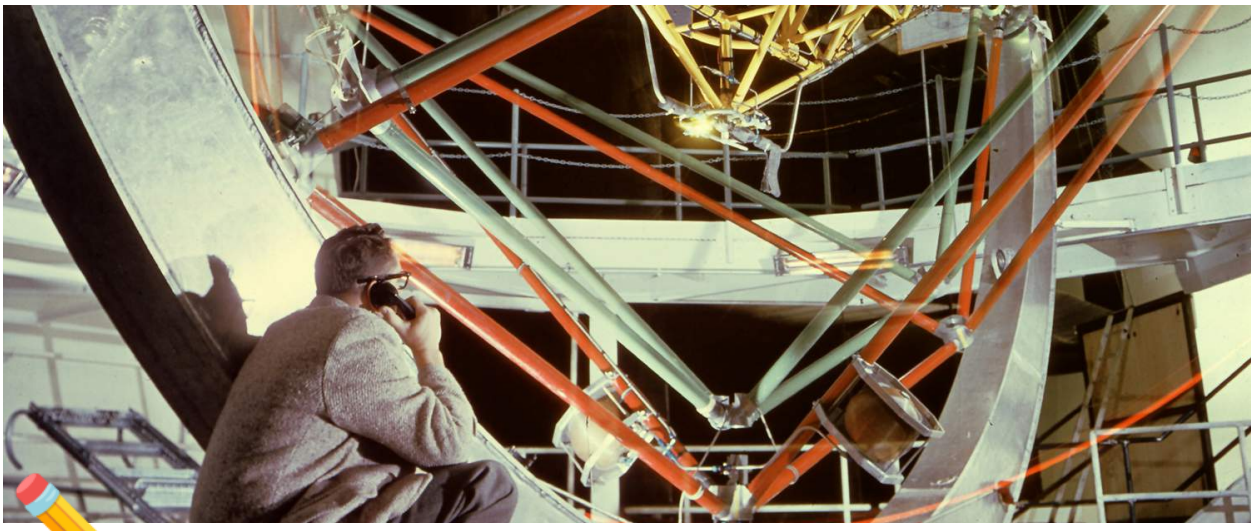


第21篇 (数据库) SQL关系表格模型 `QSqlRelationalTableModel`

代码地址：https://github.com/Lornatang/QtStartQuicklyTutorial/tree/main/SQL04

一、使用外键
二、使用委托

一、使用外键

1．新建Qt Gui应用，名称为`relationalTableModel`，基类为`QMainWindow`，类名为`MainWindow`。完成后打开`relationalTableModel.pro`项目文件，将第一行改为：

```
QT       += coregui sql
```

然后保存该文件。

2．下面向项目中添加新的C++头文件`connection.h`，并更改其内容如下：

```
#ifndef CONNECTION_H
#define CONNECTION_H
#include <QSqlDatabase>#include <QSqlQuery>static bool createConnection()
{
   QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
```

```
    db.setDatabaseName("database.db");
    if(!db.open()) return false;
    QSqlQuery query;
    query.exec("create table student (id int primary key, name vchar,course int)");
    query.exec("insert into student values(1,'yafei0',1)");
    query.exec("insert into student values(2,'yafei1',1)");
    query.exec("insert into student values(3,'yafei2',2)");

query.exec("create table course (id int primarykey, name vchar, teacher vchar)");
    query.exec("insert into course values(1,'Math','yafeilinux1')");
    query.exec("insert into course values(2,'English','yafeilinux2')");
    query.exec("insert into course values(3,'Computer','yafeilinux3')");
    return true;
}
#endif // CONNECTION_H
```

在这里建立了两个表，`student`表中有一项是`course`，它是`int`型的，而`course`表的主键也是int型的。如果要将`course`项和`course`表进行关联，它们的类型就必须相同，一定要注意这一点。

3．更改`main.cpp`文件内容如下：

```
#include "mainwindow.h"#include <QApplication>#include "connection.h"int main(int argc, ch
ar *argv[])
{
    QApplication a(argc, argv);
    if(!createConnection()) return 1;
    MainWindow w;
    w.show();

    return a.exec();
}
```

4．然后到`mainwindow.h`文件中，先包含头文件：

```
#include<QSqlRelationalTableModel>
```

然后添加`private`类型对象声明：

```
QSqlRelationalTableModel *model;
```

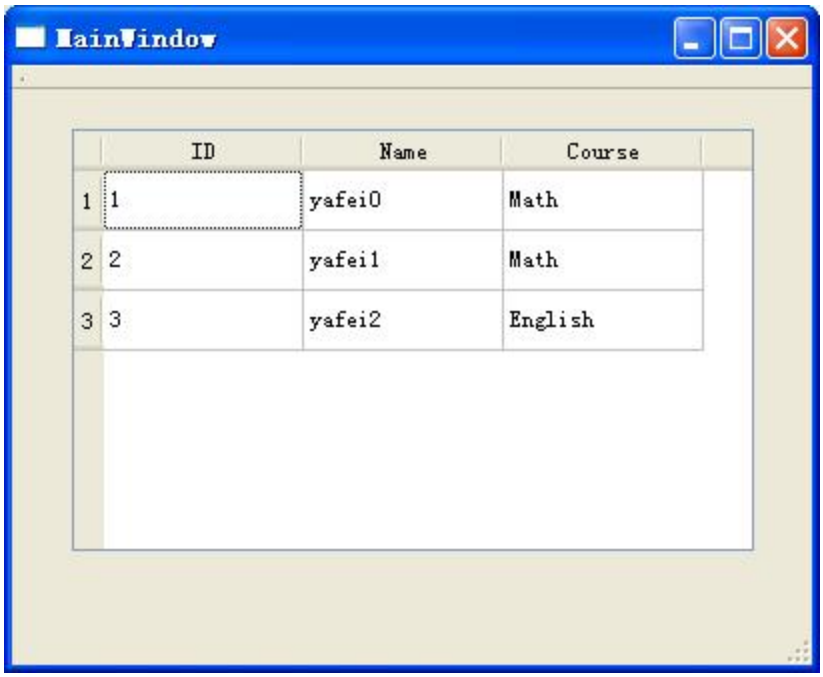
5．到设计模式，往界面上拖放一个`Table View`部件。

6．到`mainwindow.cpp`文件中，在构造函数里添加如下代码：

```
model = new QSqlRelationalTableModel(this);
//属性变化时写入数据库
model->setEditStrategy(QSqlTableModel::OnFieldChange);
model->setTable("student");
//将student表的第三个属性设为course表的id属性的外键，
//并将其显示为course表的name属性的值
model->setRelation(2,QSqlRelation("course","id","name"));
model->setHeaderData(0, Qt::Horizontal, QObject::tr("ID"));
model->setHeaderData(1, Qt::Horizontal, QObject::tr("Name"));
model->setHeaderData(2, Qt::Horizontal, QObject::tr("Course"));
model->select();
ui->tableView->setModel(model);
```

这里修改了`model`的提交策略，`OnFieldChange`表示只要属性被改动就马上写入数据库，这样就不需要我们再执行提交函数了。`setRelation()`函数实现了创建外键，注意它的格式就行了。

7．运行程序，效果如下图所示。



可以看到`Course`属性已经不再是编号，而是具体的课程了。关于外键，大家也应该有一定的认识了吧，说简单点就是将两个相关的表建立一个桥梁，让它们关联起来。

二、使用委托

有时我们也希望，如果用户更改课程属性，那么只能在课程表中有的课程中进行选择，而不能随意填写课程。Qt中还提供了一个`QSqlRelationalDelegate`委托类，它可以为`QSqlRelationalTableModel`显示和编辑数据。这个委托为一个外键提供了一个`QComboBox`部件来显示所有可选的数据，这样就显得更加人性化了。使用这个委托是很简单的，我们先在`mainwindow.cpp`文件中添加头文件`#include <QSqlRelationalDelegate>`，然后继续在构造函数中添加如下一行代码：

```
ui->tableView->setItemDelegate(
                new QSqlRelationalDelegate(ui->tableView));
```

运行程序，效果如下图所示。