



**数据库课程设计**

课程名称\_ 数据库系统概论 \_

题目名称 作业调度管理系统

专业班级 14级软件工程卓越工程师班

学 号 3114006513

学生姓名 李培锋

指导教师 周俭铁

2016 年 12 月 30 日

目录

[1 介绍 1](#_Toc470786195)

[1.1 选题简介 1](#_Toc470786196)

[1.2 使用的RDBMS 1](#_Toc470786197)

[1.3 应用程序开发环境 1](#_Toc470786198)

[1.4 系统运行环境 1](#_Toc470786199)

[1.5 开发工具 1](#_Toc470786200)

[2 需求分析 2](#_Toc470786201)

[2.1 业务流程分析 2](#_Toc470786202)

[2.2 功能需求分析 2](#_Toc470786203)

[2.3 信息需求分析 2](#_Toc470786204)

[2.4 安全性与完整性需求 2](#_Toc470786205)

[2.5 数据字典 3](#_Toc470786206)

[3 概念结构设计 4](#_Toc470786207)

[3.1 概念模型 4](#_Toc470786208)

[3.2 功能模块概要设计 4](#_Toc470786209)

[4 逻辑结构设计 5](#_Toc470786210)

[4.1 关系模型 5](#_Toc470786211)

[4.1.1 表结构 5](#_Toc470786212)

[4.1.2 范式分析 6](#_Toc470786213)

[4.2 用户子模式 7](#_Toc470786214)

[4.2.1 视图定义 7](#_Toc470786215)

[4.3 完整性定义 8](#_Toc470786216)

[4.3.1 主码 8](#_Toc470786217)

[4.3.2 外码 8](#_Toc470786218)

[4.3.3 用户定义完整性 8](#_Toc470786219)

[5 物理结构设计 10](#_Toc470786220)

[5.1 索引定义 10](#_Toc470786221)

[5.2 数据存放说明 10](#_Toc470786222)

[5.3 系统配置说明 10](#_Toc470786223)

[6 数据库连接与应用 11](#_Toc470786224)

[6.1 数据库连接 11](#_Toc470786225)

[6.2 数据库应用 11](#_Toc470786226)

[6.2.1 登录界面 11](#_Toc470786227)

[6.2.2 注册界面 12](#_Toc470786228)

[6.2.3 登录主界面 13](#_Toc470786229)

[6.2.4 用户个人信息界面 13](#_Toc470786230)

[7 收获与体会 18](#_Toc470786231)

[8 附录 19](#_Toc470786232)

[8.1 数据库初始化 19](#_Toc470786233)

[8.2 基本表创建 19](#_Toc470786234)

[8.3 视图创建 20](#_Toc470786235)

[8.4 用户操作 21](#_Toc470786236)

[8.5 图片操作 22](#_Toc470786237)

[8.6 Gif操作 23](#_Toc470786238)

[8.7 作业操作 23](#_Toc470786239)

# 

# 介绍

## 选题简介

作业调度管理系统是一个用户作业管理软件，类似操作系统调度，用户可以提交自己的作业，在运行完毕后，可以将作业保存至数据库中，以便后期观看。并且，失败的任务也可以被取出，留待重审。用户还可进行图片及gif的设置，图片可作为用户头像，而gif则是在作业开始调度时进行播放。

## 使用的RDBMS

MySQL 5.7.12

## 应用程序开发环境

编程语言：C++ 11

GUI：Qt 5.7

## 系统运行环境

Windows 10 Education x64

## 开发工具

Qt Creator 4.0.3

# 需求分析

## 业务流程分析

“作业调度管理系统”是一个客户端应用，用户安装后使用它。用户需要进行注册登录后才可使用。在进行注册时，需要验证用户名是否已存在，且该用户名是否在黑名单中，如果是则拒绝；同时验证两次输入的密码是否一致。

用户在登录之后，可以进行图片和gif的添加。若用户对其不满意，也可以在后期进行修改。而在作业进行调度时，gif会自动进行播放，在暂停作业时则暂停播放gif。

在作业调度完毕后，会询问用户是否保存作业至数据库。如果是，则将作业保存至数据库。另外，不管是调度成功完成或失败的任务，都会被保存至数据库。

用户可以在个人信息界面进行个人信息的修改，且可以观看以往的作业情况，并进行删除操作等。

## 功能需求分析

根据业务流程，系统需要提供以下功能：

1. 用户注册、登录
2. 修改用户信息
3. 提交、运行作业
4. 添加、修改图片
5. 添加、修改gif
6. 查看、删除作业

## 信息需求分析

系统中存在着几类实体：用户、作业、图片、gif。根据业务流程、系统需求以及用户需求，可以将系统的信息需求总结为如下表：

表格 2.3‑1 信息需求分析表

|  |  |
| --- | --- |
| **实体名** | **属性** |
| 用户 | 用户编号(Unique)，用户名(Unique，not null)，用户密码 |
| 作业 | 作业编号，用户编号(Unique)，作业名(not null)，提交时间(not null)，持续时间(not null)，最后期限(not null)，开始执行时间，运行时间，需要时间，优先级，轮转时间，带权轮转时间，结束时间 |
| 图片 | 图片编号，图片类型，图片数据 |
| Gif | Gif编号，gif路径 |

## 安全性与完整性需求

由于系统涉及账户登录功能，需要保证账户密码保存的安全性。为此，对于实体型“用户”，其密码字段应该使用散列算法进行运算后，再保存到数据库中。

为了能够有效管理各种实体，每种实体都具有一个主码，各实体型的主码已在上表中以下划线给出。完整性包括实体完整性、参照完整性和用户定义的完整性。表中各个主码有一些是参考的主码，即外码，已经以底纹标出。另外，对于一些字段，如用户名、用户密码等，要求非空，而这与现实中的业务是对应的。

## 数据字典

根据信息需求分析，可以得知系统的数据字典有以下：

表格 2.5‑1 用户数据字典

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 自动递增 | 备注 |
| userID | INT(10) |  | Yes | Yes | 用户ID |
| username | VARCHAR(20) |  | Yes |  | 用户名 |
| passwd | VARCHAR(20) | Null | No |  | 用户密码 |
| signupdate | DATE |  | No |  | 注册日期 |

表格 2.5‑2 作业数据字典

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 自动递增 | 备注 |
| jobID | INT(10) |  | Yes | Yes | 作业ID |
| userID | INT(10) |  | Yes | Yes | 用户ID |
| jobName | VARCHAR(20) |  | Yes | No | 用户明码 |
| joinTime | SMALLINT(5) |  | Yes | No | 提交时间 |
| lastTime | SMALLINT(5) |  | Yes | No | 持续时间 |
| startTime | SMALLINT(5) | 0 | No | No | 开始时间 |
| runTime | SMALLINT(5) | 0 | No | No | 运行时间 |
| needTime | SMALLINT(5) | 0 | No | No | 需要时间 |
| deadline | SMALLINT(5) | 0 | No | No | 最后期限 |
| PrioOrSlice | SMALLINT(5) |  | No | No | 优先级 |
| turnoverTime | FLOAT | 0 | No | No | 轮转时间 |
| weightTurnoverTime | FLOAT | 0 | No | No | 带权轮转时间 |
| finishedTime | SMALLINT(5) | 0 | No | No | 结束时间 |

表格 2.5‑3 图片数据字典

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 自动递增 | 备注 |
| graphID | CHAR(32) |  | Yes | No | 图片ID |
| graphType | VARCHAR(20) |  | No | No | 图片类型 |
| graph | LONGBLOB | Null | No | No | 图片 |

表格 2.5‑4 gif数据字典

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 自动递增 | 备注 |
| gifID | CHAR(32) |  | Yes | Yes | Gif ID |
| gifName | VARCHAR(256) |  | Yes | No | Gif路径 |

# 概念结构设计

## 概念模型

从信息需求中，得知系统有一下实体型：用户、作业、图片、gif。他们的属性在信息需求分析表中给出，在此不重复。根据上表，可以画出系统的概念结构E-R图，如下所示：



图 3.1‑1 系统概念E-R图

## 功能模块概要设计



图 3.2‑1 系统功能模块概略图

# 逻辑结构设计

## 关系模型

### 表结构

从系统的需求分析和概念结构图中，可以建立如下关系表：

1. 用户表

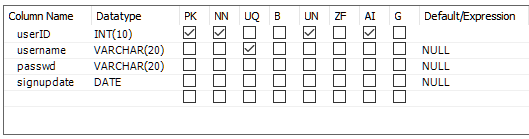


图 4.1‑1 用户表

1. 黑名单表

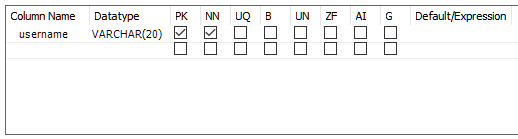


图 4.1‑2 黑名单表

1. gif表

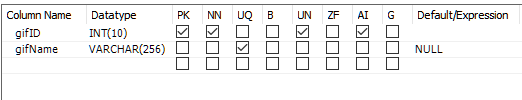


图 4.1‑3 gif表

1. 图片表

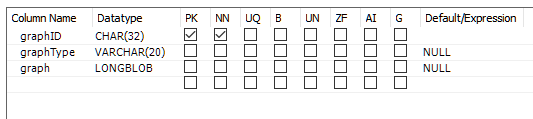


图 4.1‑4 图片表

1. 失败作业表

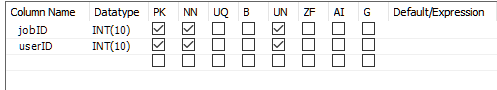


图 4.1‑5 失败作业表

1. 成功作业表

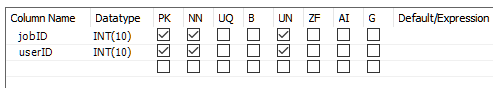


图 4.1‑6 成功作业表

1. 图片归属用户表

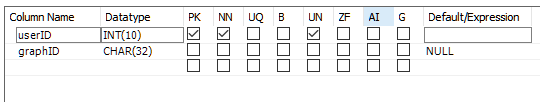


图 4.1‑7 图片归属用户表

1. gif归属用户表

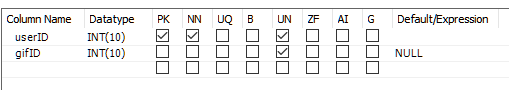


图 4.1‑8 gif归属用户表

### 范式分析

系统中的关系模型有：

1. User(用户编号，用户名，用户密码)
2. Job(作业编号，用户编号，作业名，提交时间，持续时间，最后期限，开始执行时间，运行时间，需要时间，优先级，轮转时间，带权轮转时间，结束时间)
3. graph(图片编号，图片类型，图片数据)
4. gif(Gif编号，gif路径)
5. blacklist(username)
6. jobfailed(作业编号，用户编号)
7. jobfinished(作业编号，用户编号)
8. user\_graph(用户编号，图片编号)
9. user\_gif(用户编号，Gif编号)

分析如下：

1. 关系模式User是4NF
2. 关系模式Job是4NF
3. 关系模式graph是4NF
4. 关系模式gif是4NF
5. 关系模式blacklist是4NF
6. 关系模式jobfailed是4NF
7. 关系模式jobfinished是4NF
8. 关系模式user\_graph是4NF
9. 关系模式user\_gif(是4NF

对系统的关系模型进行范式分析后，可以根据分析结果进行相应地调整，达到优化系统的目的，使其降低冗余度，同时避免插入异常、更新异常和删除异常。

## 用户子模式

### 视图定义

为简化系统的查询使用，将某些常用的较复杂的查询定义为视图，方便用户操作，使其可以在定义好的视图进行查询。

1. 作业记录

作业记录存储的是用户的ID及其所拥有的作业数目：



图 4.2‑1 作业记录视图

1. 失败作业记录

失败作业记录的是各用户的失败作业：



图 4.2‑2 失败作业记录视图

1. 成功作业记录

成功作业记录的是各用户的成功作业：



图 4.2‑3 成功作业记录视图

1. 所有作业总记录

所有作业总记录记录了各作业的数目情况：

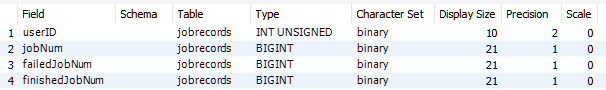


图 4.2‑4 所有作业总记录视图

1. 轮转时间记录

轮转时间记录的各用户作业的平均周转时间和平均带权周转时间：

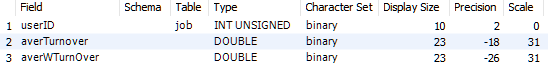


图 4.2‑5 轮转时间记录视图

## 完整性定义

### 主码

一般而言，为了避免表中有冗余的数据，每个表都需要有一个主码，用于使其元组唯一。因此，本系统的表结构设计也遵循这一原则。每个表格的主码已经在“逻辑结构设计”一节中通过下划线标出。

### 外码

本系统的表结构的外码已经在“逻辑结构设计”一节中以“字符底纹”效果标出。

### 用户定义完整性

为了能够管理用户，本系统对用户名等属性进行了非空限定，而对于图片ID，采用MD5hash计算获得唯一标识码。同时还将其中一些字段设置为自动增加，以方便操作。

# 物理结构设计

## 索引定义

索引的定义与各个基本表的操作方式密切相关。如果一个属性经常在查询条件中出现，则应考虑在其上建立索引。还有其他的一些情况会成为是否需要定义索引的考虑因素。

本设计由于基本表数据较少，但鉴于关系模式清晰，可以对主键建立索引，以加快查询速度。而且，由于主键是采用增量计数器来分配的，因此其逻辑与物理上的顺序是一致的。

## 数据存放说明

由于系统的开发环境是PC机，暂不涉及磁盘阵列或分布式存储等存储方法，因此，数据仅仅存放在本机上的磁盘上，而没有做异地、异机存放等。

## 系统配置说明

本系统是基于默认的MySQL配置进行的，没有使用不可移植特性，因此使用本系统，只需使用默认的MySQL配置即可。

# 数据库连接与应用

## 数据库连接

在本系统中，使用QT的数据库连接方法进行数据库的连接，其需要将MySql的驱动文件动态库和静态库放置于QT的安装目录bin文件夹下，该文件分别为libmysql.dll以及libmysql.lib。

## 数据库应用

### 登录界面

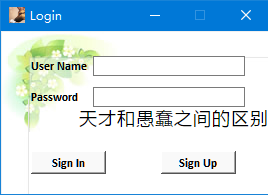


图 6.2‑1 登录界面

登录时的检查代码为：

|  |
| --- |
| /\* check if username valid or if user exists \*/  query.prepare("Select username, passwd From user Where username = (:username) Limit 1");  query.bindValue(":username", username.c\_str());    if(query.exec()){  if(query.next()){  std::string \_passwd = std::string((const char\*)query.value("passwd").toString().toLocal8Bit());  if(passwd.compare(\_passwd) == 0){  info.setStatus(true);  info.setInfo("Sign in success!");  }else{  info.setInfo("Sign in failed!\n"  "Password is not correct!");  }  }else{  qDebug() << query.lastError();  info.setInfo("The user does not exist!");  }  } |

### 注册界面

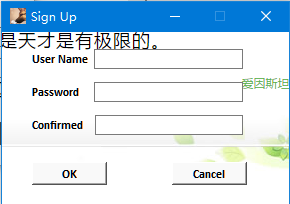


图 6.2‑2 注册界面

注册时检测代码为：

|  |
| --- |
| /\* check if it is on black list \*/  query.prepare("Select \* From blacklist Where username = (:username) Limit 1");  query.bindValue(":username", username.c\_str());  if(query.exec()){  if(query.next()){  info.setInfo("The user is on blacklist");  }else{  /\* check if it it exists yet \*/  query.prepare("Select \* From user Where username = (:username) Limit 1");  query.bindValue(":username", username.c\_str());    if(query.exec()){  if(query.next()){  info.setInfo("The user does exist!");  }  else{  /\* insert new user \*/  query.prepare("Insert Into user (username, passwd, signupdate) "  "Values(:username, :passwd, :signupdate)");  QDate date = QDate::currentDate();  query.bindValue(":username", username.c\_str());  query.bindValue(":passwd", passwd.c\_str());  query.bindValue(":signupdate", date.toString("yyyy-MM-dd"));  //only work when it is "yyyy-MM-dd"    if(query.exec()){  info.setStatus(true);  info.setInfo("Sign up success!");  }else{  qDebug() << query.lastError();  info.setInfo("Sign up failed!");  }  }  }  }  } |

### 登录主界面

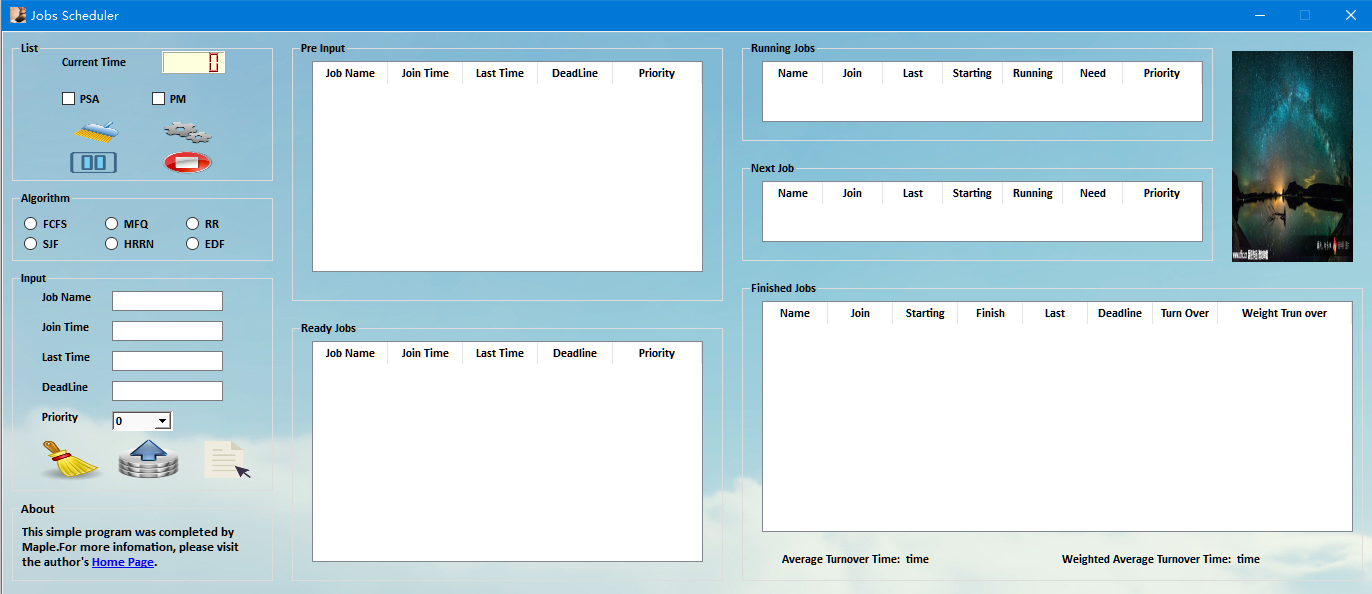


图 6.2‑3 用户操作主界面

### 用户个人信息界面

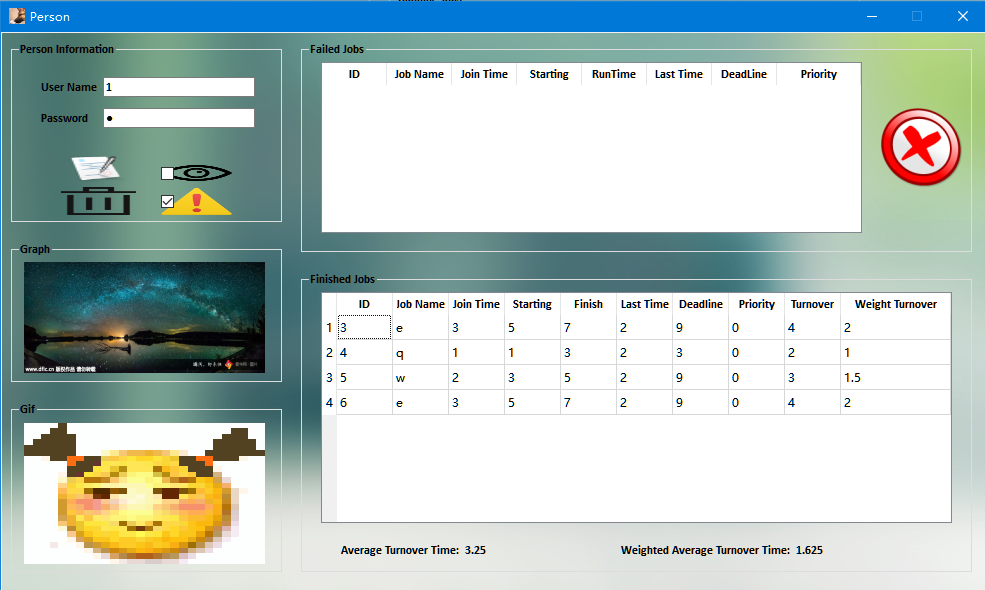


图 6.2‑4 用户个人信息界面

1. 添加图片代码

|  |
| --- |
| QSqlQuery query(con);  QVariant var(graph.getGraph());  /\* Insert \*/  query.prepare("Insert Into graph (graphID, graph, graphType) Values(:graphID, :graph, :graphType)");  query.bindValue(":graphID", graph.getGraphID().c\_str());  query.bindValue(":graph", var);  query.bindValue(":graphType", graph.getGraphType().c\_str());    if(query.exec()){  info.setStatus(true);  info.setInfo("Insert graph success!");  }else{  qDebug() << query.lastError();  info.setInfo("Insert graph failed!");  } |

1. 添加gif代码

|  |
| --- |
| QSqlQuery query(con);  /\* Insert \*/  query.prepare("Insert Into gif (gifName) Values(:gifName)");  query.bindValue(":gifName", gif.getGifName().c\_str());    if(query.exec()){  info.setStatus(true);  info.setInfo("Insert gif success!");  }else{  qDebug() << query.lastError();  info.setInfo("Insert gif failed!");  } |

1. 修改图片代码

|  |
| --- |
| QSqlQuery query(con);  /\* Insert \*/  query.prepare("Insert Into user\_graph (userID, graphID) Values(:userID, :graphID) "  "On Duplicate Key Update graphID = (:graphID)");  query.bindValue(":userID", uid);  query.bindValue(":graphID", graphID.c\_str());    if(query.exec()){  info.setStatus(true);  setHasGraph();  info.setInfo("Insert user\_graph success!");  }else{  qDebug() << query.lastError();  info.setInfo("Insert user\_graph failed!");  } |

1. 修改gif代码

|  |
| --- |
| QSqlQuery query(con);  /\* Insert \*/  query.prepare("Insert Into user\_gif (userID, gifID) Values(:userID, :gifID) "  "On Duplicate Key Update gifID = (:gifID)");  query.bindValue(":userID", uid);  query.bindValue(":gifID", gifID);    if(query.exec()){  info.setStatus(true);  setHasGif();  info.setInfo("Insert user\_gif success!");  }else{  qDebug() << query.lastError();  info.setInfo("Insert user\_gif failed!");  } |

1. 修改用户信息代码

|  |
| --- |
| QSqlQuery query(con);  /\* update user info \*/  query.prepare("Update user Set username = ?, passwd = ? Where userID = ?");  query.bindValue(0, username.c\_str());  query.bindValue(1, passwd.c\_str());  query.bindValue(2, uid);    if(query.exec()){  info.setStatus(true);  info.setInfo("Modify success!");  }else{  qDebug() << query.lastError();  info.setInfo("Modify failed!\n"  "Duplicate username!");  } |

1. 获取作业代码

|  |
| --- |
| QSqlQuery query(con);  query.prepare("select \* "  "From job Left Outer Join job" + QString(type) + " "  "Using (userID, jobID) "  "where userID = (:userID) and job.jobID = job" + QString(type) + ".jobID");  query.bindValue(":userID", uid);    if(query.exec()){  while(query.next()){  ptr job = std::make\_shared<Job>();  job->setDeadLine(query.value("deadline").toUInt());  job->setFinishedTime(query.value("finishedTime").toUInt());  job->setJobID(query.value("jobID").toUInt());  job->setJobName(std::string((const char\*)(query.value("jobName").toString().toLocal8Bit())));  job->setJoinTime(query.value("joinTime").toUInt());  job->setLastTime(query.value("lastTime").toUInt());  job->setNeedTime(query.value("needTime").toUInt());  job->setPrioOrSlice(query.value("PrioOrSlice").toUInt());  job->setRunTime(query.value("runTime").toUInt());  job->setStartTime(query.value("startTime").toUInt());  job->setTurnoverTime(query.value("turnoverTime").toUInt());  job->setWTurnoverTime(query.value("weightTurnoverTime").toFloat());    jobs.push\_back(job);  }  info.setStatus(true);  info.setInfo("Get job success!");  }else{  qDebug() << query.lastError();  info.setInfo("Query error!");  } |

1. 删除作业代码

|  |
| --- |
| QSqlQuery query(con);  /\* Delete \*/  query.prepare("Delete From job Where jobID = (:jobID) and userID = (:userID)");  QVariantList jobIDs, userIDs;  /\* bind value \*/  for(auto it = jobIDVec.begin(); it != jobIDVec.end(); ++it){  jobIDs << \*it;  userIDs << uid;  }  query.bindValue(":jobID", jobIDs);  query.bindValue(":userID", userIDs);  if(query.execBatch()){  info.setStatus(true);  info.setInfo("Insert jobs success!");  }else{  qDebug() << query.lastError();  info.setInfo("Insert jobs failed!");  } |

# 收获与体会

不得不说，开发整套系统还是花费颇多时间的。

由于我是把操作系统的课程设计与数据库概论的课程设计整合到一起做，所以在打通两者的关联时，颇费了一些功夫。在本系统实现早起，先是查找设计模式的书籍，看哪种好的设计模式可以适用于本系统。由于之前是做嵌入式系统开发，所以一直没有接触到设计模式的思想。所以在探索的前期，还是遇到了较多的问题。

在确定好设计模式之后，便开始着手操作系统方面的处理。前期断断续续，才完成了些许的工作。而真正开始开发数据库模块，还是要到操作系统考完试后。当天晚上，便开始思考系统有什么样的实体，应该为这些实体赋予什么样的属性。思虑良久，本打算用Visio画UML，但感觉太死板了，且发现网上有说VS可以自动搭建类图，所以便想在VS上搭建类图，先创建基本类，再看看各个类之间的联系，再进行补充。可天不遂人愿，在我想要打开VS的时候，它竟然罢工了。进行了很多修补方式都无效，浪费时间不说，还影响心情。最后不得不进行重装。唉！

但后来又磨磨蹭蹭，逐渐地把整个业务逻辑整理清楚。可是在真正开发的时候，还是会发现当时是对问题考虑不足的。有时是缺少某个属性，或者是属性类型设置不对（如早期对graphID设置为Int，后来使用MD5hash值后又转为varchar(32)），或者是主键的设计问题，等等。各种问题在实际开发中慢慢浮现。而且，由于对QT的很多组件控件不熟悉，在查询用法时也是话费了较多的时间。这不得不让我感叹“熟能生巧”的道理。

之前学习的时候，觉得sql语句还是蛮简单的，毕竟只是普通的增删查改操作。但在实际开发过程中，要考虑的问题实在很多。特别是随着操作趋向复杂时，那更是如此了。这也验证了那个道理，“看起来容易做起来难”。另外，由于是嵌入式sql语句，很多问题都不能直接看出来。由于没有语法错误的提示，所以在调试时花了很多的时间，尽管一条语句可能很简单，但由于一时的疏忽，如少打了个空格，则只能望洋兴叹，慢慢地重新审视sql语句才能把问题找出来。不过，这也让我更加懂得规范的重要性。如果每写一句语句都严格地按照规范来做的话，那么就可以很清晰的看到语法错误的地方，否则就如死沉大海般难找了。

最后，不得不说一句实践的重要性。平时缺少操作，到实际应用中确实会手生。但事实上，要学习的东西是在太多，一直专研于某种方面的学习还是不足的。尽管要求学习要有深度，但在这个技术更新换代猝不及防的年代，确实让人等待不起。但也没办法，该做的还是要做的。因此，还是要继续加油做下去。

不忘初心，方得始终。

# 附录

## 数据库初始化

|  |
| --- |
| /\* Initial work \*/  create database Jobs;  use Jobs; |

## 基本表创建

|  |
| --- |
| /\* create base table \*/  create table user (  userID INT unsigned AUTO\_INCREMENT,  username VARCHAR(20) UNIQUE,  passwd VARCHAR(20),  signupdate DATE,  PRIMARY KEY (userID)  )Engine = InnoDB;  create table graph(  graphID char(32), #MD5  graphType varchar(20),  graph longblob,  PRIMARY KEY (graphID)  )Engine = InnoDB;  create table user\_graph(  userID INT unsigned,  graphID char(32),  primary key (userID),  FOREIGN KEY (graphID) REFERENCES graph (graphID) ON DELETE CASCADE,  FOREIGN KEY (userID) REFERENCES user (userID) ON DELETE CASCADE  )Engine = InnoDB;  create table user\_gif(  userID INT unsigned,  gifID INT unsigned Unique,  primary key (userID),  FOREIGN KEY (gifID) REFERENCES gif (gifID) ON DELETE CASCADE,  FOREIGN KEY (userID) REFERENCES user (userID) ON DELETE CASCADE  )Engine = InnoDB;  create table gif(  gifID INT unsigned AUTO\_INCREMENT,  gifName varchar(256) unique,  PRIMARY KEY (gifID)  )Engine = InnoDB;  create table blackList (  username VARCHAR(20),  PRIMARY KEY (username)  );  create table job (  jobID INT unsigned,  userID INT unsigned,  joinTime VARCHAR(20) not null,  joinTime smallint unsigned not null,  lastTime smallint unsigned not null,  startTime smallint unsigned,  finishedTime smallint unsigned,  runTime smallint unsigned,  needTime smallint unsigned,  deadline smallint unsigned,  PrioOrSlice smallint unsigned,  turnoverTime float,  weightTurnoverTime float,  PRIMARY KEY (jobID, userID),  FOREIGN KEY (userID) REFERENCES user (userID) ON DELETE CASCADE  )Engine = InnoDB;  create table jobFinished (  jobID INT unsigned,  userID INT unsigned,  PRIMARY KEY (jobID, userID),  FOREIGN KEY (userID) REFERENCES user (userID) ON DELETE CASCADE,  FOREIGN KEY (jobID) REFERENCES job (jobID) ON DELETE CASCADE  )Engine = InnoDB;  create table jobFailed (  jobID INT unsigned,  userID INT unsigned,  PRIMARY KEY (jobID, userID),  FOREIGN KEY (userID) REFERENCES user (userID) ON DELETE CASCADE,  FOREIGN KEY (jobID) REFERENCES job (jobID) ON DELETE CASCADE  )Engine = InnoDB; |

## 视图创建

|  |
| --- |
| /\* create view \*/  #all job info records view  create view jobRecords (userID, jobNum, failedJobNum, finishedJobNum)  As  select jobRecord.userID, jobrecord.jobNum, jobRecordFail.failedJobNum, jobRecordFinished.finishedJobNum  from jobRecord, jobRecordFail, jobRecordFinished;  #job record view  create view jobRecord(userID, jobNum)  As  select userID, count(\*)  from job;  #failed job record view  create view jobRecordFail(userID, failedJobNum)  As  select userID, count(\*)  from jobRecordFail;  #finished job record view  create view jobRecordFinished(userID, finishedJobNum)  As  select userID, count(\*)  from jobRecordFinished;  #finished job turnover time view  create view overTime  As  select userID, (sum(turnoverTime) / count(\*)) as averTurnover,  (sum(weightTurnoverTime) / count(\*)) as averWTurnOver  from job  where needTime = 0  group by userID; |

## 用户操作

|  |
| --- |
| /\* User Operation \*/  #insert job instance  Insert Into job  (jobID, userID, jobName, joinTime, lastTime, startTime, runTime,  needTime, deadline, PrioOrSlice, turnoverTime, weightTurnoverTime, finishedTime)  Values(  1, 1, 'jobName', 1, 2, 2, 2,  0, 9, 0, 2, 2, 4);    #delete job instance  Delete From job Where jobID = 1 and userID = 1;  #get fail job instance  Select \*  From job left outer join jobfailed  Using (userID, jobID)  Where userID = 1  and job.jobID = jobfailed.jobID;    #get finished job instance  Select \*  From job left outer join jobfinished  Using (userID, jobID)  Where userID = 1  and job.jobID = jobfinished.jobID;    #get max job ID  Select Max(jobID)  From job  where userID = 1;  #view operate, select averTurnover, averWTurnOver instance  Select averTurnover, averWTurnOver  From overTime  where userID = 1; |

## 图片操作

|  |
| --- |
| /\* Graph Operation \*/  #insert graph instance  Insert Into graph (graphID, graph, graphType)  Values('d41d8cd98f00b204e9800998ecf8427e', '', 'JPG');  #insert user\_graph instance  Insert Into user\_graph (userID, graphID)  Values(1, 'd41d8cd98f00b204e9800998ecf8427e')  On Duplicate Key  Update graphID = 'd41d8cd98f00b204e9800998ecf8427e';  #get graph instance  Select graph.graphID, graphType, graph  From graph, user\_graph  Where graph.graphID = user\_graph.graphID  and user\_graph.userID = 1  Limit 1; |

## Gif操作

|  |
| --- |
| /\* Gif Operation \*/  #insert gif instance  Insert Into gif (gifName)  Values('name');  #update gif instance  Insert Into user\_gif (userID, gifID)  Values(1, 2)  On Duplicate Key  Update gifID = 1;  #get gif instance  Select gif.gifID, gifName  From gif, user\_gif  Where gif.gifID = user\_gif.gifID and user\_gif.userID = 1  Limit 1;  #get gifID instance  Select gifID  From gif  where gifName = 'name'; |

## 作业操作

|  |
| --- |
| /\* Job Operation \*/  #insert gif instance  Insert Into gif (gifName)  Values('name');  #update gif instance  Insert Into user\_gif (userID, gifID)  Values(1, 2)  On Duplicate Key  Update gifID = 1;  #get gif instance  Select gif.gifID, gifName  From gif, user\_gif  Where gif.gifID = user\_gif.gifID and user\_gif.userID = 1  Limit 1;  #get gifID instance  Select gifID  From gif  where gifName = 'name'; |