同济大学

同济大学计算机系
数据库课程设计报告

高校信息管理系统

姓　　名　成威尔

学　　号　1853003

专　　业　计算机科学与技术

授课老师　李文根

同济大学

高校人事信息管理系统

一、需求及可行性分析

■ 背景

对机构以及人员的管理是每一个高校都必须具备的能力，随着信息化时代的到来，信息的形式越来越复杂，信息的交流越来越密切，传统的方法不仅安全性不高，而且信息更新复杂滞后，已经无法满足高校庞大的信息流动。以我们同济大学为例，学院、直属附属机构以及党政机构等等总计几十甚至上百个部门，每个部门的学生、教师、职工等人员上万个，为了避免信息的混乱和减少信息更新查找的代价，必须开发一个功能完善，操作简单，稳定性高，容量大的高校信息管理系统。

高校信息管理系统是典型的信息管理系统(MIS),其开发主要包括后台数据库的建立和维护以及前端应用程序的开发两个方面。对于前者要求建立起数据一致性和完整性强、安全性好、稳定性高的库。而对于后者则要求应用程序功能完备、界面友好、操作简单等特点。

■ 目的

针对高校信息管理系统的问题，设计一个高校人事信息管理系统。使高校人事信息管理更加便捷，信息更新更加快速，强化高校信息处

的管理能力，对各种工作进行分类管理，使高校信息管理专业化、规范化，使人事信息详实准确，以达到人事信息管理的稳定性、高效性。

本高校信息管理系统包括的内容及功能有：①人员各种信息：包括员工的基本信息，如编号、姓名、性别、学历、所属部门、毕业院校、健康情况、职称、职务、奖惩等。②人员各种信息的修改。③对转出、辞退、退休人员信息的删除。④按照一定条件，查询、统计符合条件的人员信息。⑤教师教学信息的录入：教师编号、姓名、课程编号、课程名称、课程时数、学分、课程性质等；学生选课信息的录入。⑥科研信息的录入：教师编号、研究方向、课题研究情况、专利、论文及著作发表情况等。

■ 核心流程

1.用户登录：不同用户登录查看以及修改权限不同，提高信息的安全性，方便管理。

2.部门基本信息：包括部门名称、部门编号、上级部门以及附属部门。

3.学生基本信息：包括姓名、学号、年龄、性别、入学时间、所属部门编号、奖惩信息等。

4.教职工基本信息：包括姓名、编号、年龄、性别、所属部门编号、奖惩信息、毕业院校等。

5.课程信息：课程号、课程名称、授课时间、学分、课程性质、所

属部门等。

6.科研信息：所属部门、项目名称、项目编号、项目状态、开始时间、结束时间等。

7.职位：职位编号、所属部门、职位名称、薪资等。

■ **用户需求**

1.登录：分为管理员登录，教职工登录以及学生登陆，登陆不同，操作权限也不同。

2.学生登录：可以修改登录学生的大部分信息，以及查看相关课程以及科研信息。

3.教职工登录：可以修改登录教职工的大部分信息，以及查看相关课程以及科研信息，也可以修改学生的部分信息。

4.管理员登录：最高权限，可以修改学生以及教职工的全部信息，还可以更新机构信息，包括机构的增加、删除、修改等；以及对学生和教职工的删除等。

5.注册：学生和教职工都可以输入基本信息完成注册。

6.教师开课：通过教师登录后，教师可以将自己的开课信息录入数据库。

7.开设科研项目：教师登陆后可以将自己的科研项目录入数据库。

8.职位信息：教职工和学生都可以选择自己的职位。

■ 数据字典

部门

属性名	字段	类型	长度	约束
编号	Dno	int		主键，非空
名称	Dname	char	20	非空

学生

属性名	字段	类型	长度	约束
姓名	Sname	char	20	非空
学号	Sno	int		非空，主键
年龄	Sage	int		非空
性别	Sgender	char	4	female 或 male，非空
入学时间	Scheckin	date	8	非空
所属部门	Sdept	int		非空，外键
奖惩信息	SRandP	char	50	

教职工

属性名	字段	类型	长度	约束
工号	Tno	int		主键，非空
姓名	Tname	char	20	非空

年龄	Tage	int		非空
性别	Tgender	char	4	female 或 male，非空
所属部门	Tdept	int		外键
奖惩信息	TRandP	char	50	
毕业院校	Tgra	char	50	

课程

属性名	字段	类型	长度	约束
课程号	Cno	int		主键，非空
课程名称	Cname	char	20	非空
授课时间	Ctime	char	20	非空
学分	Ccredit	int		非空
课程性质	Cchar	char	20	required 或 optional 非空
所属部门	Cdept	int		外键，非空

职位

属性名	字段	类型	长度	约束
职位编号	Jno	int		主键，非空
所属部门	Jdept	int		外键,非空
职位名称	Jname	char	20	非空
薪资	Jsalary	int		非空

科研

属性名	字段	类型	长度	约束
项目编号	Pno	int		主键，非空
项目名称	Pname	char	20	非空
所属部门	Pdept	int		外键，非空
项目状态	Pstatus	char	40	非空
开始日期	Pstart	date	8	非空
结束日期	Pend	date	8	

管理员

属性名	字段	类型	长度	约束
管理员 ID	Ano	int		主键，非空
密码	Apass	char	20	非空

■ 可行性分析

1.技术可行性

本次高校人事信息管理系统设计的目的时实现人事信息管理的稳定性、安全性以及快捷性，后端采用数据库配合高级语言 Qt 编程实现，前端拟采用 Qt 实现，配合查阅资料以及教师指导，可以完成本次有关数据库的课程设计实验。

2.应用可行性

本系统面向高校人员，具有较好的计算机基础。本系统设计简明，开发充分考虑了用户的操作体验，界面直观易懂，且附有详细的设计说明以及用户操作指导，配合基本的计算机操作即可使用。

3.经济可行性

现在基础设施较为完备，计算机网络等成本可以忽略，软件对计算机软硬件要求较低，无需购买其他设备。且软件可以减少高校人事管理的人力物力成本，以及减少人事管理失误带来的损失。

■ 软件属性

（1）可用性

本软件也可以通过单步跟踪的操作进行检查处理。

（2）安全性

由于软件运行数据放在数据库中，所以参数不容易被错改、破坏，万一参数受到破坏也不会影响源程序。

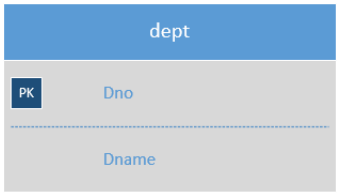
（3）可维护性

本软件利用数据库进行编程，系统结构由程序基本确定，大量的参数及文本内容全部放于数据库中。修改、更新数据只要在数据库进行修改添加，而不需要对系统结构进行修改，这样系统维护性、升级都十分方便。

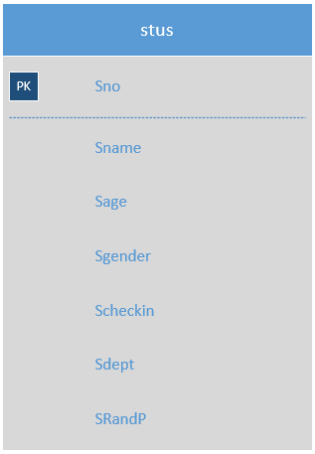
二、概念设计

实体设计

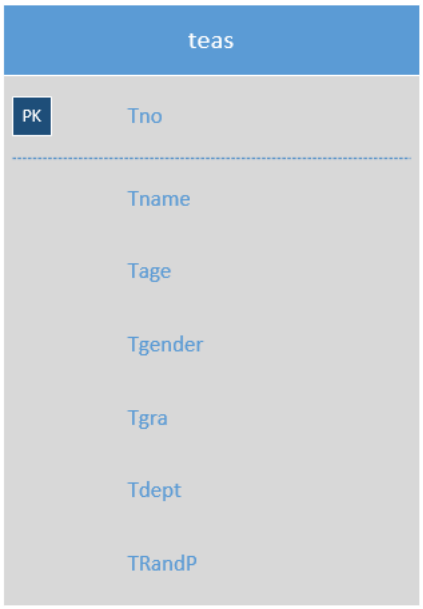
- 部门实体
  - dept（Dno,Dname）
  - 部门编号、部门名称
  - 图示



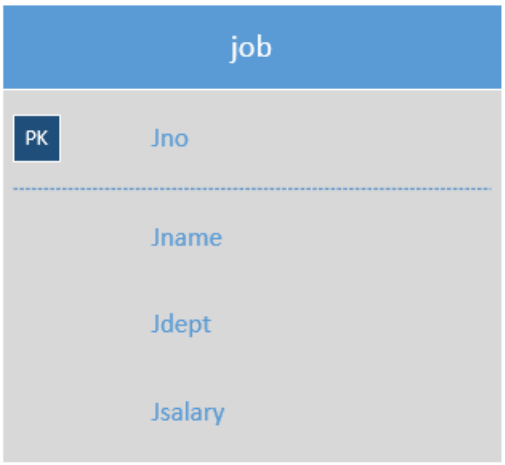
- 学生实体
  - stus(Sname,Sno,Sage,Sgender,Scheckin,Sdept,SRandP)
  - 学生姓名、学生学号、年龄、性别、注册日期、所属部门、奖惩信息
  - 图示



- 教职工实体
  - teas(Tno, Tname, Tage, Tgender, Tdept, TRandP, Tgra)
  - 工号、姓名、年龄、性别、所属部门、奖惩、毕业院校
  - 图示

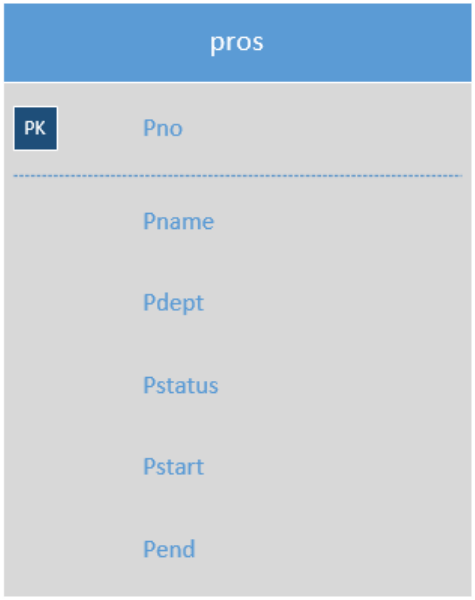


- 职位实体
  - job(Jno, Jdept, Jname, Jsalary)
  - 编号、所属部门、名称、薪资
  - 图示

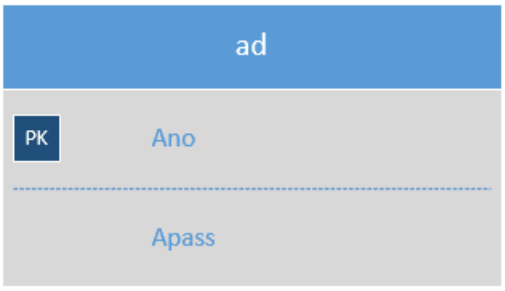


- 课程实体
  - cours(Cno,Cname,Ctime,Ccredit,Cchar,Cdept)
  - 编号、名称、上课时间、学分、性质、所属部门
  - 图示

- 科研实体
  - pros(Pno,Pname,Pdept,Pstatus,Pstart,Pend)
  - 编号、名称、所属部门、状态、开始时间、结束时间
  - 图示

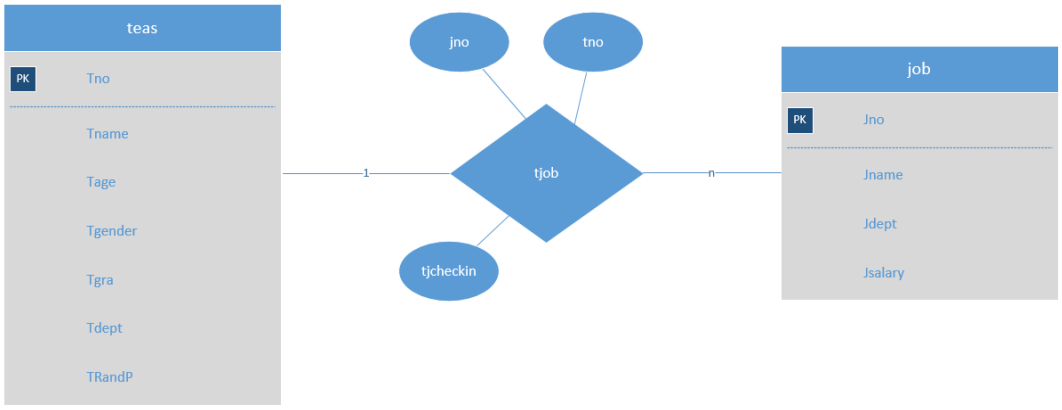


- 管理员实体
  - ad(Ano, Apass)
  - 编号、密码
  - 图示

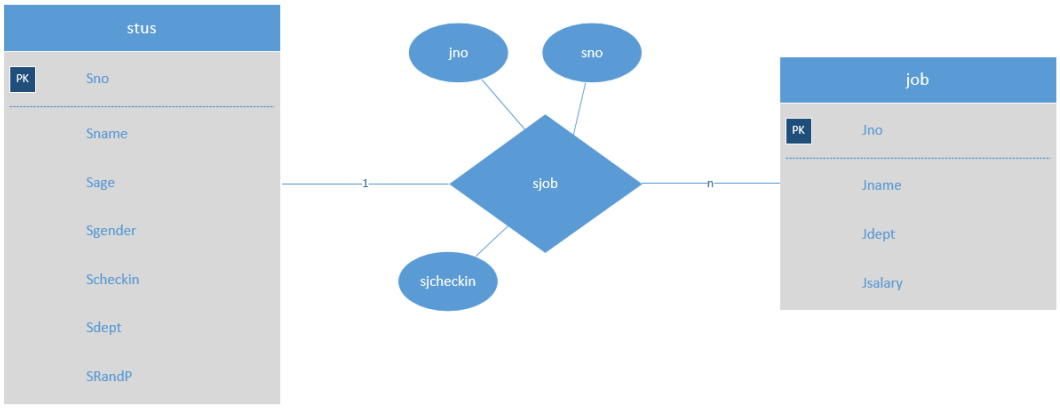


实体之间的联系

- 教职工与职位
  - tjob(tno, jno, tjcheckin)
  - 教师编号、职位编号、入职时间
  - 图示

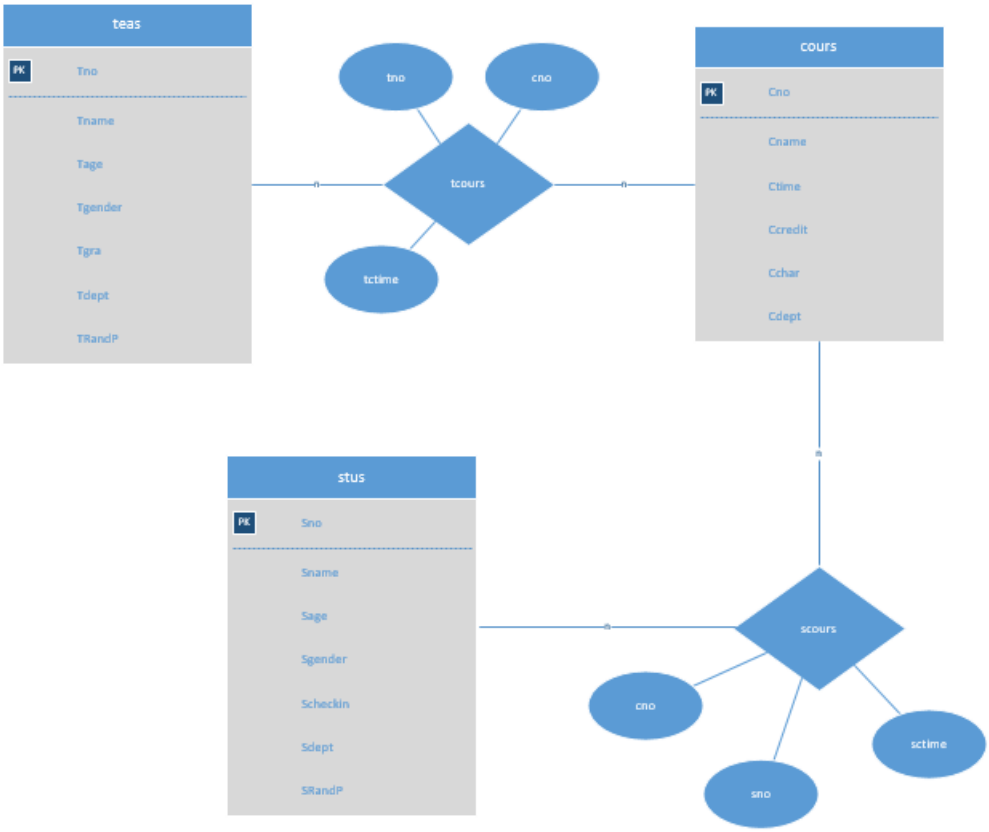


- 学生与职位
  - sjob(sno, jno, sjcheckin)
  - 学生编号、职位编号、入职时间
  - 图示

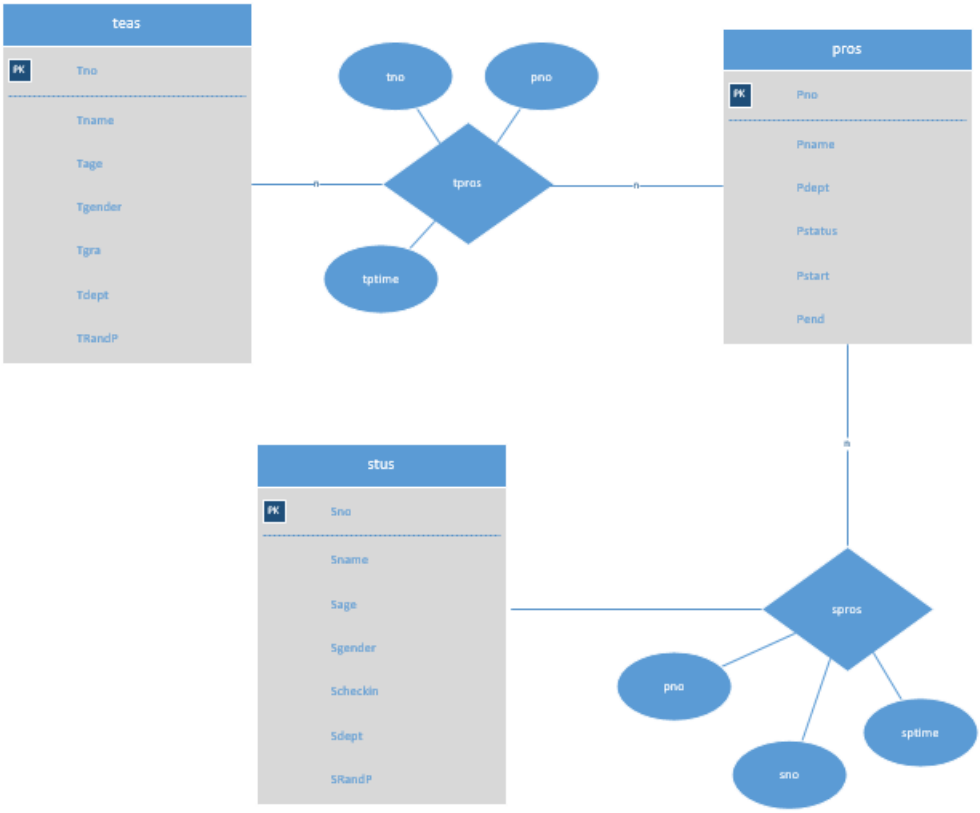


- 教职工、学生与课程
  - tcours(cno, tno, tctime)
  - 课程编号、教师编号、开课时间
  - scours(cno, sno, sctime)
  - 课程编号、学生编号、选课时间

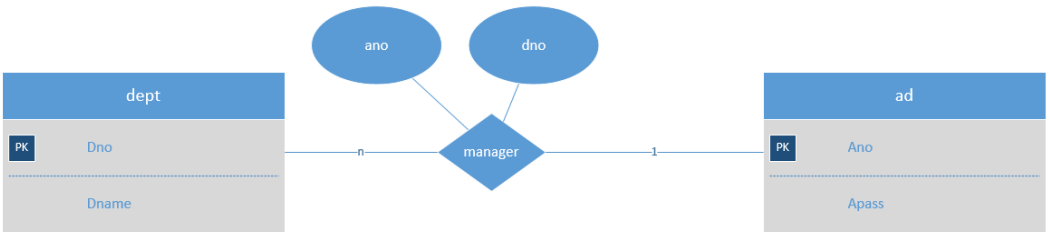
- 图示



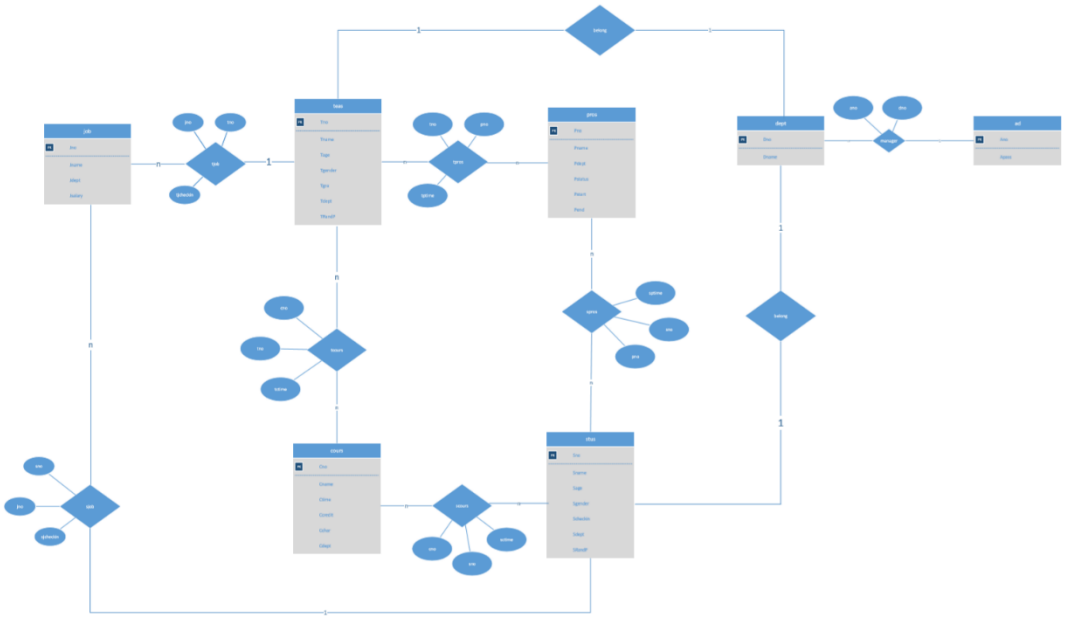
- 教职工、学生与科研
  - tpros(pno, tno, tptime)
  - 项目编号、教师编号、加入时间
  - spros(pno, sno, sptime)
  - 项目编号、学生编号、加入时间
  - 图示



- 管理员与部门
  - manager（ano，dno）
  - 管理员编号、部门编号
  - 图示



全局 E-R 图



三、逻辑设计

- **逻辑设计目标**

  数据库的逻辑结构是一种抽象的概念，主要用来描述各类数据类型的关系与结构。逻辑结构独立于任何一种数据模型，在本次课程设计中，数据库环境为 SQL 关系数据库，因此首先需要将 E-R 图转换为关系模型，然后根据具体 DBMS 的特点和限制转换为特定的 DBMS 支持下的数据模型，最后进行优化。

- **关系模型设计及其规范化**

  - 转换规则：

    1. 实体类转换

       每个实体类型转换为一个关系模式

       实体属性即为关系模式的属性

实体标识符即为关系模式的键

2.联系类转换（二元联系）

1：1 联系：联系两端的实体转换为关系模式，在任意一个关系模式中加入另一个关系模式的键，作为其外键。

1：N 联系：在 N 端实体类型转换得到的关系模式中，加入一端实体类型的键，作为外键。

M：N 联系：联系类型需转换为关系模式，属性为两端实体类型的键，分别作为外键。键为两端实体键的组合。

■ E-R 图转化结果：

部门：{部门名称、部门编号}

学生：{学生姓名、学生学号、年龄、性别、注册日期、所属部门、奖惩信息}

教职工：{工号、姓名、年龄、性别、所属部门、奖惩、毕业院校}

职位：{职位编号、所属部门、职位名称、职位薪资}

课程：{课程编号、名称、上课时间、学分、性质、所属部门}

科研：{科研项目编号、名称、所属部门、状态、开始时间、结束时间}

管理员：{管理员编号、密码}

保证各个关系不存在插入异常、删除异常和数据冗余，且每个关系的属性不可分割。否则，数据库关系模型设计会造成数据库使用和

维护的各种问题，比如存储大量的冗余信息，造成系统资源的浪费。

- 表结构设计

表 3.1 部门表

department

属性名	字段	类型	长度	约束
编号	Dno	int		主键，非空
名称	Dname	char	20	非空

表 3.2 学生表

student

属性名	字段	类型	长度	约束
姓名	Sname	char	20	非空
学号	Sno	int		非空，主键
年龄	Sage	int		非空
性别	Sgender	char	4	female 或 male，非空
入学时间	Scheckin	date	8	非空
所属部门	Sdept	int		非空，外键
奖惩信息	SRandP	char	50	

表 3.3 教职工表

worker

属性名	字段	类型	长度	约束
工号	Tno	int		主键，非空
姓名	Tname	char	20	非空
年龄	Tage	int		非空
性别	Tgender	char	4	female 或 male，非空
所属部门	Tdept	int		外键
奖惩信息	TRandP	char	50	
毕业院校	Tgra	char	50	

表 3.4 课程表

course

属性名	字段	类型	长度	约束
课程号	Cno	int		主键，非空
课程名称	Cname	char	20	非空
授课时间	Ctime	char	20	非空
学分	Ccredit	int		非空
课程性质	Cchar	char	20	required 或

				optional 非空
所属部门	Cdept	int		外键，非空

表 3.5 职位表

job

属性名	字段	类型	长度	约束
职位编号	Jno	int		主键，非空
所属部门	Jdept	int		外键,非空
职位名称	Jname	char	20	非空
薪资	Jsalary	int		非空

表 3.6 科研表

project

属性名	字段	类型	长度	约束
项目编号	Pno	int		主键，非空
项目名称	Pname	char	20	非空
所属部门	Pdept	int		外键，非空
项目状态	Pstatus	char	40	非空
开始日期	Pstart	date	8	非空
结束日期	Pend	date	8	

表 3.7 管理员

ad

属性名	字段	类型	长度	约束
管理员 ID	Ano	int		主键，非空
密码	Apass	char	20	非空

物理设计

一、 表结构

1. Admin

```
mysql> show create table admin;
+-------+-----------------------------------
-------------------------------+
| Table | Create Table
                               |
+-------+-----------------------------------
-------------------------------+
| admin | CREATE TABLE `admin` (
  `id` int(11) NOT NULL,
  `password` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+-------+-----------------------------------
```

2. Course

```
| course | CREATE TABLE `course` (
  `no` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(20) NOT NULL,
  `Time` char(20) NOT NULL,
  `Checkin` date NOT NULL,
  `Credit` double NOT NULL,
  `Cchar` char(20) NOT NULL,
  `Ctest` char(20) NOT NULL,
  `Teach` int(11) NOT NULL,
  `room` char(20) NOT NULL,
  PRIMARY KEY (`no`),
  KEY `course_ibfk_1` (`Teach`),
  CONSTRAINT `course_ibfk_1` FOREIGN KEY (`Teach`) REFERENCES `teacher` (`No`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=114 DEFAULT CHARSET=utf8 |
```

3. Department

```
| department | CREATE TABLE `department` (
  `dno` int(11) NOT NULL,
  `dname` char(20) NOT NULL,
  PRIMARY KEY (`dno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
```

4. select_course

```
| select_course | CREATE TABLE `select_course` (
  `stu_no` int(11) NOT NULL,
  `cou_no` int(11) NOT NULL,
  PRIMARY KEY (`stu_no`,`cou_no`),
  KEY `select_course` (`cou_no`),
  CONSTRAINT `select_course_ibfk_1` FOREIGN KEY (`stu_no`) REFERENCES `student` (`sno`) ON DELETE CASCADE,
  CONSTRAINT `select_course_ibfk_2` FOREIGN KEY (`cou_no`) REFERENCES `course` (`No`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
```

5. select_teacher

```
| select_teacher | CREATE TABLE `select_teacher` (
  `stu_no` int(11) NOT NULL,
  `tea_no` int(11) NOT NULL,
  PRIMARY KEY (`stu_no`,`tea_no`),
  KEY `select_teacher` (`tea_no`),
  CONSTRAINT `select_teacher_ibfk_1` FOREIGN KEY (`stu_no`) REFERENCES `student` (`sno`) ON DELETE CASCADE,
  CONSTRAINT `select_teacher_ibfk_2` FOREIGN KEY (`tea_no`) REFERENCES `teacher` (`No`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
```

6. student

```
| student | CREATE TABLE `student` (
  `sno` int(11) NOT NULL AUTO_INCREMENT,
  `sname` char(20) NOT NULL,
  `sage` int(11) NOT NULL,
  `sgender` char(20) DEFAULT NULL,
  `scheckin` date DEFAULT NULL,
  `sdept` int(11) NOT NULL,
  `password` char(20) NOT NULL,
  `email` char(30) NOT NULL,
  `phonenumber` char(30) NOT NULL,
  `height` int(11) NOT NULL,
  `weight` int(11) NOT NULL,
  `politics_status` char(50) NOT NULL,
  `position` char(50) NOT NULL,
  `address` char(50) NOT NULL,
  PRIMARY KEY (`sno`),
  KEY `student_ibfk_1` (`sdept`),
  CONSTRAINT `student_ibfk_1` FOREIGN KEY (`sdept`) REFERENCES `department` (`dno`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=1235 DEFAULT CHARSET=utf8 |
```

7. teacher

```
| teacher | CREATE TABLE `teacher` (
  `No` int(11) NOT NULL,
  `Name` char(20) NOT NULL,
  `Age` int(11) NOT NULL,
  `Gender` char(10) NOT NULL,
  `Checkin` date NOT NULL,
  `Dept` int(11) NOT NULL,
  `Password` char(20) NOT NULL,
  `Email` char(20) NOT NULL,
  `Phonenumber` char(20) NOT NULL,
  `Height` int(11) NOT NULL,
  `Weight` int(11) NOT NULL,
  `politics_status` char(20) NOT NULL,
  `position` char(20) NOT NULL,
  `address` char(50) NOT NULL,
  PRIMARY KEY (`No`),
  KEY `teacher_ibfk_1` (`Dept`),
  CONSTRAINT `teacher_ibfk_1` FOREIGN KEY (`Dept`) REFERENCES `department` (`dno`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
```

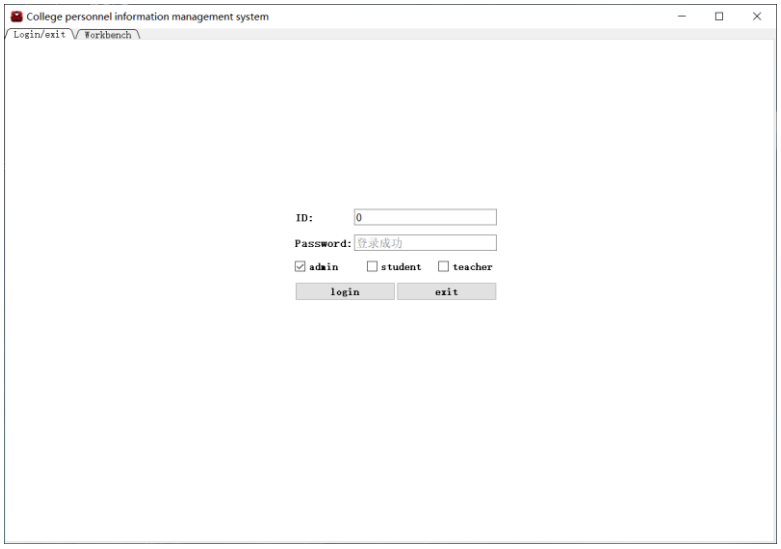
实际设计

二、 实验环境

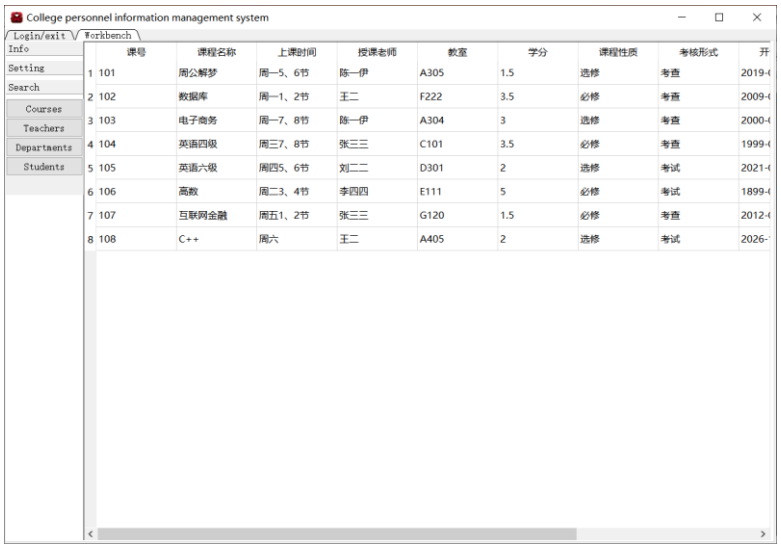
前端：Qt Creator 4.11.1

数据库：mysql-5.7.33

三、 项目成果（详见录屏）



登录登出界面



工作台界面

四、 问题解决

1. 设置自增

   建表时考虑不够周全

   但已经有外键约束

   操作违反了外键约束，破坏了数据库完整性。所以这个修改的行为会被 DBMS 阻止

```
mysql> alter table student change sno sno int  auto_increment;
ERROR 1833 (HY000): Cannot change column 'sno': used in a foreign key constraint
'select_course_ibfk_1' of table 'db1.select_course'
```

   解决如下，先关闭外键约束。

```
mysql> set foreign_key_checks = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> alter table student change sno sno int  auto_increment;
Query OK, 2 rows affected (0.07 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> set foreign_key_checks = 1;
Query OK, 0 rows affected (0.00 sec)
```

2. 级联删除
   建表时考虑不够周全
   解决办法，删除外键，再新建外键

```
alter table xxxxx drop foreign key yyyy
```

无法直接删除外键，需要进行如下操作，找到内置外键名

```
mysql> show create table select_course;
+---------------+----------------------------------------+
| Table         | Create Table                           |
+---------------+----------------------------------------+
| select_course | CREATE TABLE `select_course` (
  `stu_no` int(11) NOT NULL,
  `cou_no` int(11) NOT NULL,
  PRIMARY KEY (`stu_no`,`cou_no`),
  KEY `select_course` (`cou_no`),
  CONSTRAINT `select_course_ibfk_1` FOREIGN KEY (`stu_no`) REFERENCES `student` (`sno`) ON DELETE CASCADE,
  CONSTRAINT `select_course_ibfk_2` FOREIGN KEY (`cou_no`) REFERENCES `course` (`No`) ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 |
+---------------+----------------------------------------+
1 row in set (0.00 sec)
```

```
alter table sd add constraint sd_supervisor_fk foreign key (SupId) references supervisor (SupId) ON DELETE CASCADE;
```

3. 自增不连续

```
db.exec("ALTER TABLE course AUTO_INCREMENT = 1;");//自增连续
```

4. int 类型模糊查询

```
cast(no as char)
```

5. QT connect 函数
   SLOT()中的槽函数如果需要参数，那么这个参数必须来自于 SIGNAL()中信号函数的参数。
   Qt 类 QSignalMapper 建立映射。

```
QSignalMapper *signalMapper = new QSignalMapper(this);
connect(ui->lineEdit_9,SIGNAL(textChanged(QString)),signalMapper,SLOT(map()));
connect(ui->lineEdit_10,SIGNAL(textChanged(QString)),signalMapper,SLOT(map()));
connect(ui->lineEdit_11,SIGNAL(textChanged(QString)),signalMapper,SLOT(map()));
connect(ui->lineEdit_12,SIGNAL(textChanged(QString)),signalMapper,SLOT(map()));
connect(ui->comboBox_2,SIGNAL(currentTextChanged(QString)),signalMapper,SLOT(map()));
connect(ui->doubleSpinBox_2,SIGNAL(valueChanged(double)),signalMapper,SLOT(map()));
connect(ui->checkBox_8,SIGNAL(stateChanged(int)),signalMapper,SLOT(map()));
connect(ui->checkBox_10,SIGNAL(stateChanged(int)),signalMapper,SLOT(map()));
connect(ui->checkBox_9,SIGNAL(stateChanged(int)),signalMapper,SLOT(map()));
connect(ui->checkBox_11,SIGNAL(stateChanged(int)),signalMapper,SLOT(map()));
connect(ui->dateEdit_2,SIGNAL(dateChanged(QDate)),signalMapper,SLOT(map()));
connect(ui->dateEdit_3,SIGNAL(dateChanged(QDate)),signalMapper,SLOT(map()));
signalMapper->setMapping(ui->lineEdit_9,9);
signalMapper->setMapping(ui->lineEdit_10,10);
signalMapper->setMapping(ui->lineEdit_11,11);
signalMapper->setMapping(ui->lineEdit_12,12);
signalMapper->setMapping(ui->comboBox_2,0);//teach
signalMapper->setMapping(ui->doubleSpinBox_2,1);//xuefen
signalMapper->setMapping(ui->checkBox_8,2);//xuan bi
signalMapper->setMapping(ui->checkBox_10,3);//shi cha
signalMapper->setMapping(ui->checkBox_9,2);//xuan bi
signalMapper->setMapping(ui->checkBox_11,3);//shi cha
signalMapper->setMapping(ui->dateEdit_2,4);
signalMapper->setMapping(ui->dateEdit_3,4);
connect(signalMapper, SIGNAL(mapped(int)), this, SLOT(screen(int)));
```

实现实时显示的模糊查询

心得体会

数据库课设是本学期课设中花的时间最多的一个，也在这次课程设计中接触了新的前端工具 qt，虽然不如网页开发色彩鲜艳和多样，但是也比较符合我的课题，信息管理系统。

从上学期就开始的需求分析、各种设计，到现在的完全实现整个系统，成就感还是满满的，也在过程中学到了很多。也让我认识到了一个看似简单的系统，想要顺利的完成并不是一件简单的事情，在前期需求分析时就要有一个清晰的设计，否则在实际实现的过程中会遇到很多困难，多次修改最初的方案，整个过程也是比较迷茫，所以在下手写代码之前一定要慎重，不要轻易写起，以免做了无用功。

这次课设也培养了我的工程思维，ui 界面的审美，还有思维的细节，收获满满，当然还有很多的不足，希望以后可以继续努力，继续进步。