西安财经大学

《数据库系统概论》课程设计报告

排课管理系统

班级：计本 1801

学号：1831050010

姓名：陈伯硕

目录

1 系统需求分析

1.1 需求概述

1.1.1 课程设计要求

对于排课管理系统, 课程设计的要求如下:

- 实现班级, 课程等基本信息的管理;
- 实现学生, 教师信息的管理;
- 实现班级课程及课程的任课教师和排课管理;
- 创建存储过程检测指定教师, 指定节次是否有课;
- 创建存储过程生成指定班级的课程表;
- 创建存储过程生成指定老师的课程表;
- 建立数据库相关表之间的参照完整性约束.

1.1.2 实体集

即通过数据库自动排课并提供给学生查询, 让学生和老师可以查询具体时间安排. 该系统可以通过以下实体集实现

- 教学楼实体集, 包含楼号和楼名;
- 教室实体集, 包含楼号, 教室号和容量;
- 院系实体集, 包含院系编号和院系名;
- 课程实体集, 包含课程号, 课程名, 课程类型, 开课学院;
- 教师实体集, 包含教师的编号, 姓名, 院系, 职称, 研究方向[1];
- 班级实体集, 班级 ID, 班级名, 人数, 所属院系;

1.2 组织结构分析

本系统适用于教师与学生对课程的管理, 提供给学生, 教师所有表的查看权限, 数据库管理员拥有其他权限.

1.3 管理功能分析

排课是个综合系统, 需要从教务系统中导入数据 (或者由数据库管理员人工导入), 实现课程安排, 即课程管理, 同时将课程的信息分别存储汇总, 部分实现教师管理, 时间管理, 教室管理, 班级管理 (如 图 1).

[1]可能是老师工作的具体院系, 如" 计算机系", 也可能是其他研究所, 如" 基础数学研究所"

图 **1**　排课系统的管理功能

1.4　业务流分析

1.4.1　管理员业务

管理员业务如图图 2

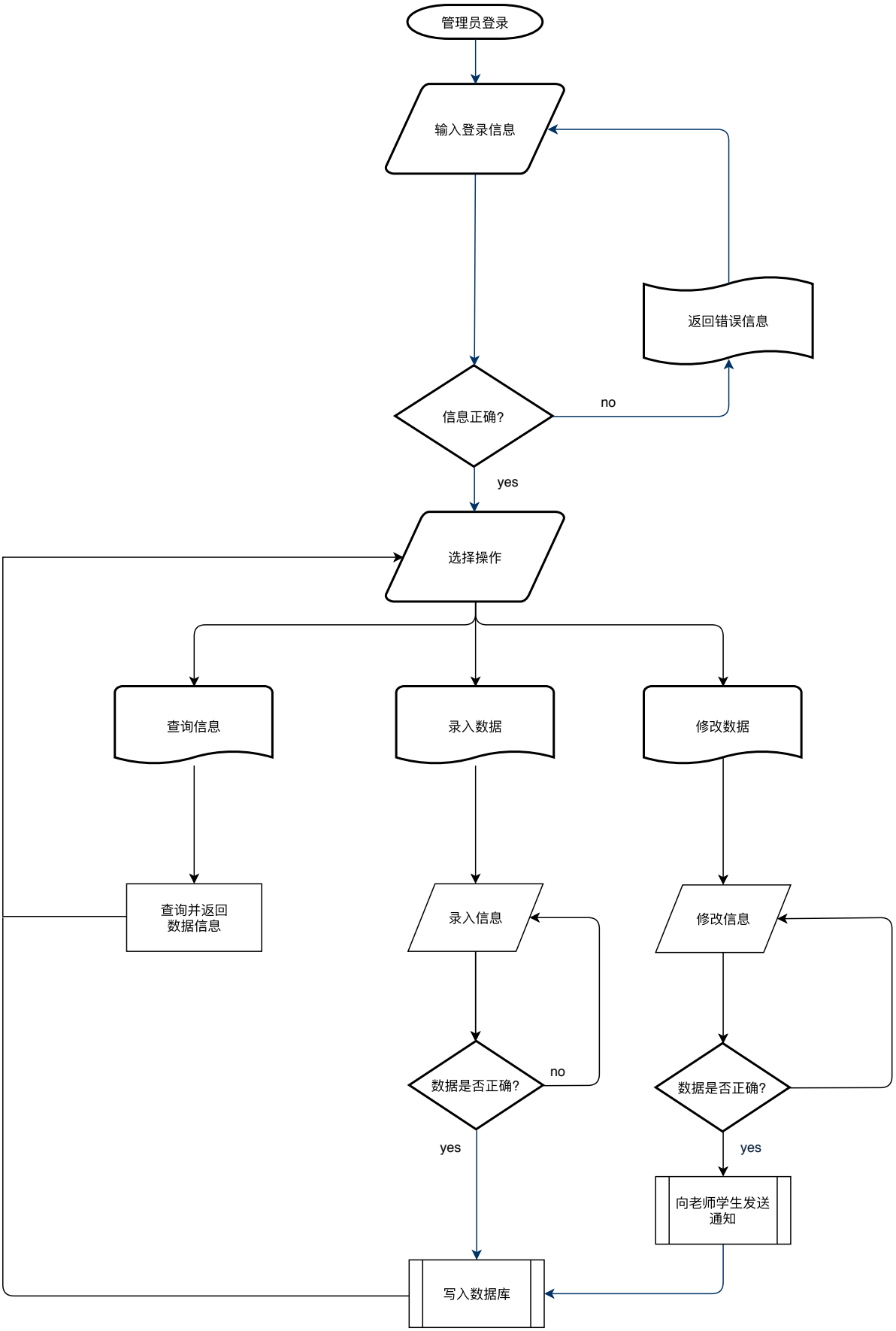
管理员登录
输入登录信息
返回错误信息
信息正确?
no
yes
选择操作
查询信息
录入数据
修改数据
查询并返回
数据信息
录入信息
修改信息
数据是否正确?
no
数据是否正确?
yes
yes
向老师学生发送
通知
写入数据库

图 **2**　管理员的业务流

1.4.2 教师查询

教师的业务如 图 3

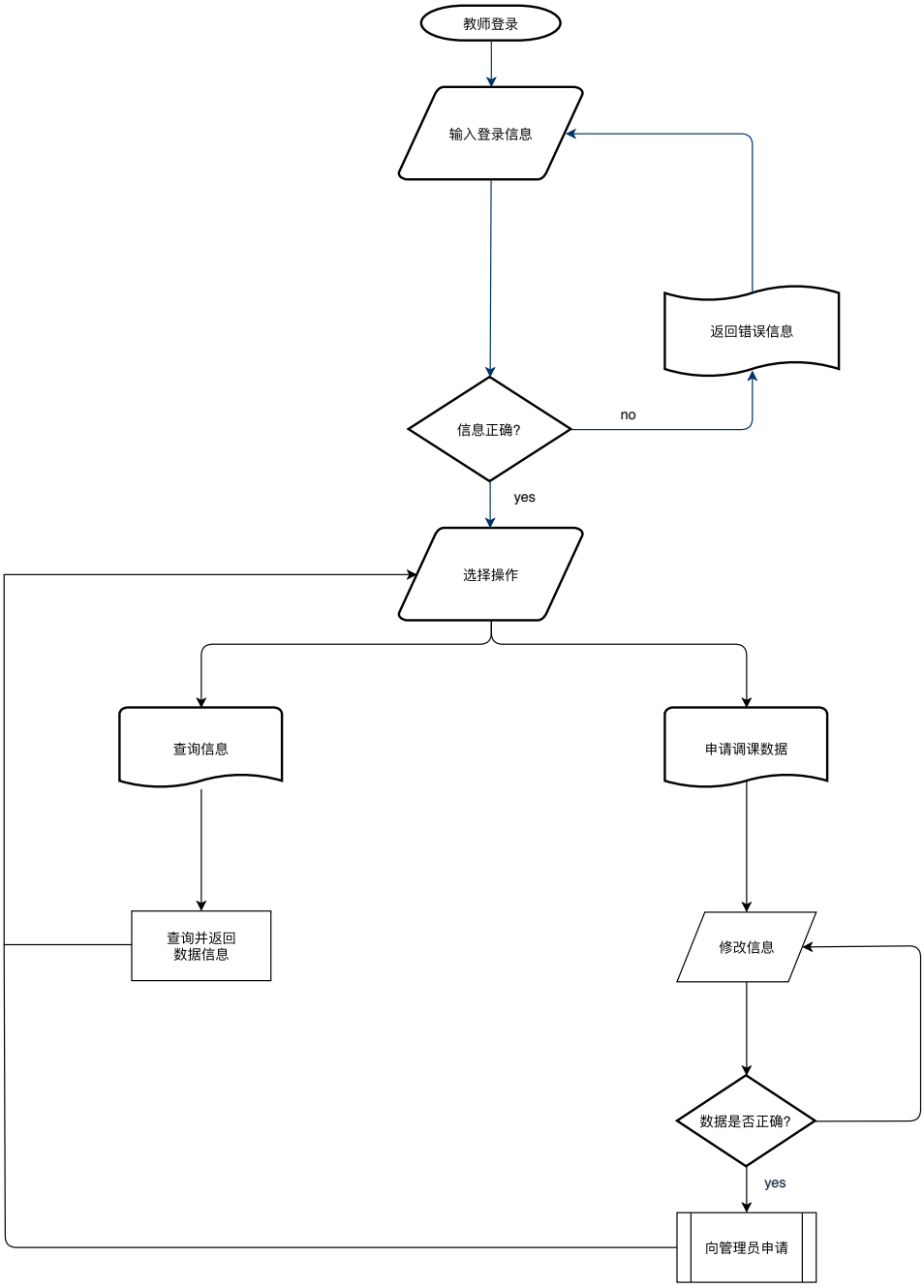

图 **3** 教师业务流程

1.5 数据流分析

系统外部环境图如 图 4

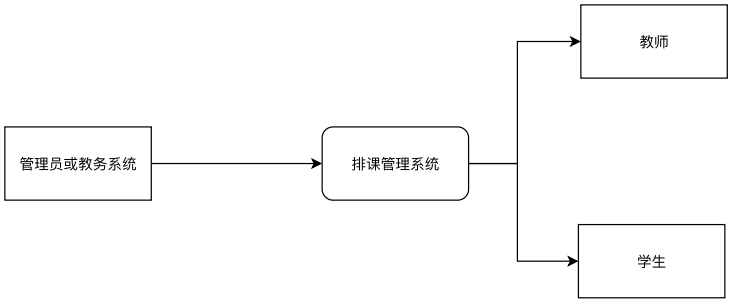

图 **4**　系统外部环境图

接下来将模型细化, 图 5 描绘了系统的主要功能

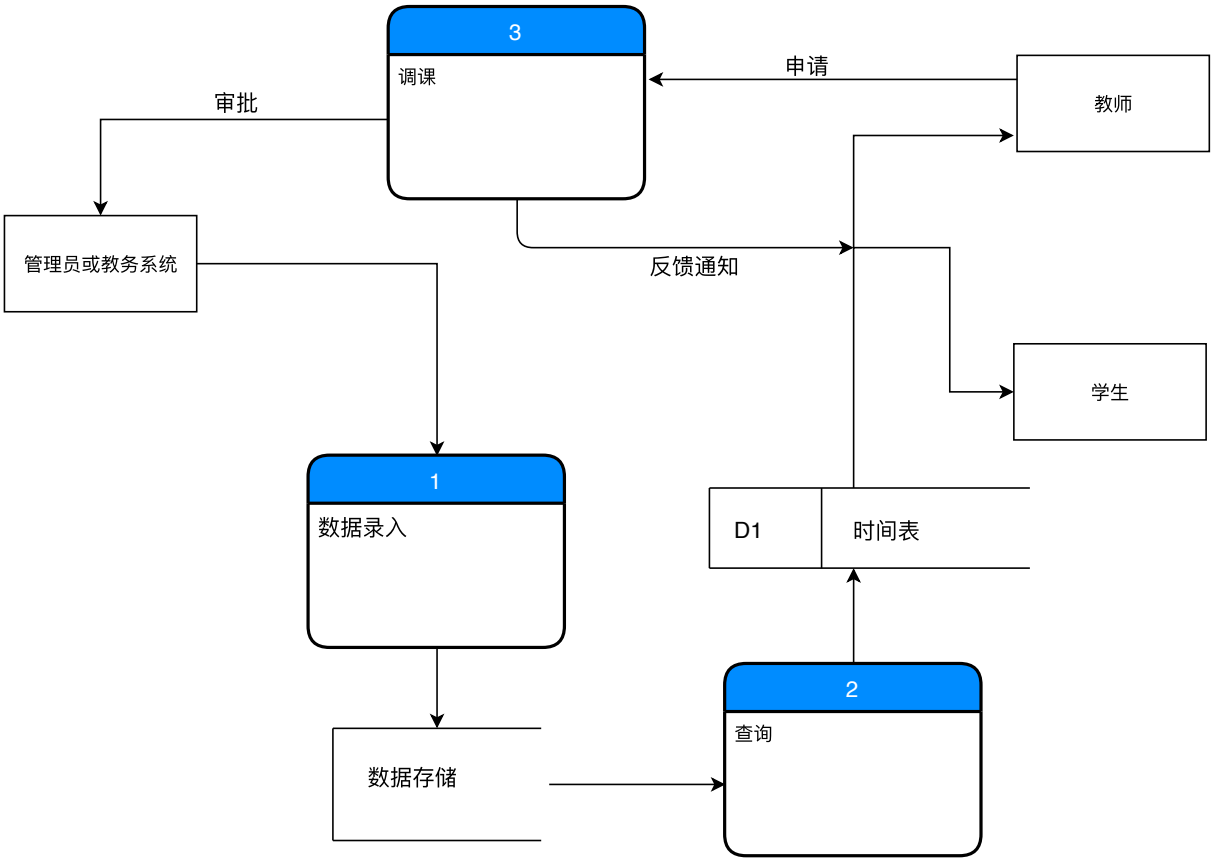

图 **5**　排课系统功能的第一层数据流图

1.6　数据字典

1.6.1　数据字典

根据数据流图中所涉及的信息，并对信息进行相应的分析，确定出所有数据项的描述内容，其主要分为数据项名称、类型、长度和取值范围，如表 1所示[2]

[2]在本系统中, 所有 ID 均由其他信息组合成哈希函数代替 (希望不要碰撞)

表 **1**　数据字典 **(**具体的数据的大小参考 **[10])**

名称	含义	类型	大小	取值范围	备注
楼号	教学楼的编号	tinyint	1 B	0-255	
楼名	教学楼的名称	char(5)	15 B	长度 $\leqslant 5$	
容量	教学楼的容量	tinyint	1 B	0-255	
院系编号	院系的编号	tinyint	1 B	0-255	
院系名	院系名	char(8)	$\leqslant$ 24 B	0-255	
课程号	课程编号	bigint	20 B	20 位	
课程名	课程名	char(10)	$\leqslant$ 30 B	10 位	
类型名	课程的类型	char(10)	$\leqslant$ 30 B		
教师号	教师编号	bigint	20 B	20 位	
教师名	教师的姓名	char(10)	$\leqslant$ 30 B		
职称	教师的职称	char(3)	9 B	助教，讲师，副教授，教授	
班号	班级编号	bigint	20 B	20 位	
班名	班级的全名	char(10)	$\leqslant$ 30 B		
人数	班级人数	tinyint	1 B	0-255	
时间号	上课时间的标识	bigint	8 B	-2^{63} - $2^{63}-1$	
日	星期几	tinyint	1 B	0-255	星期用 1-5 代替
开始时间	上课时间	tinyint	1 B	0-255	假设上课时间都是整点，24 小时制
结束时间	下课时间	tinyint	1 B	0-255	假设下课时间都是整点，24 小时制
开始周	第几周开始	tinyint	1 B	0-255	
结束周	第几周结束	tinyint	1 B	0-255	
is_odd	单周上课	bit	1 B	0,1	默认为 1
is_even	双周上课	bit	1 B	0,1	默认为 1
节号	上课节的标识	bigint	8 B	-2^{63} - $2^{63}-1$	
学期	在上学期或下学期	bit	1 B	0,1	每年第一学期为 0, 第二学期为 1

表 **1**　数据字典 **(**具体的数据的大小参考 **[10])**

名称	含义	类型	大小	取值范围	备注
年	年份	smallint	2 B	$-32,768$-$32,767$	

1.6.2　数据结构

根据数据流图中的信息的分析，在数据项描述的基础上确定所有数据结构的描述，主要有数据结构名称、含义和组成说明。本题数据结构如表 2.

表 **2**　数据结构说明

名称	含义	组成
$build$	教学楼信息	楼号, 楼名
$classroom$	教室信息	楼号, 教室号, 容量
$department$	院系	院系号, 院系名
$course$	课程信息	课程号, 课程名, 课程类型, 开课院系编号
$instructor$	老师信息	编号, 姓名, 职称, 院系编号, 研究方向
$class$	班级	班号, 班级名, 人数, 院系号
$time\_slot$	上课时间信息	时间标识符, 上课日, 上课时间, 下课时间, 开始周, 结束周, 单周上课, 双周上课
$section$	每节课的信息	课程编码, 时间标识, 课程号, 学期, 开课年, 楼号, 教室号.

1.6.3　数据流描述

根据数据流图的数据流向的分析，确定所有数据流的描述，如 表 3

表 **3**　数据流描述

数据流名	数据流来源	数据流去向	说明
课程信息	管理员导入	学生, 教师查询	记录每门课的信息
教师信息	管理员导入	学生, 教师查询	记录教师的信息
教室信息	管理员导入	学生, 教师查询	记录教室的信息
教学楼信息	管理员导入	学生, 教师查询	记录教学楼的信息
调课申请	教师申请	管理员处理	调整上课信息
调课通知	管理员处理	学生, 教师通知	调整上课信息
关注课程	学生, 教师设置	用户自己查询	学生教师关注上课信息
个人课表	数据库	用户查询	可以设置显示关注课程

2　数据库概念结构设计

2.1　实体分析

经需求分析，本系统中包含六个实体，包含教学楼, 教室, 院系, 课程, 教师, 班级

2.2　属性分析

- 教学楼实体集, 包含楼号和楼名;
- 教室实体集, 包含楼号, 教室号和容量;
- 院系实体集, 包含院系编号和院系名;
- 课程实体集, 包含课程号, 课程名, 课程类型, 开课学院;
- 教师实体集, 包含教师的编号, 姓名, 院系, 职称, 研究方向[3];
- 班级实体集, 班级 ID, 班级名, 人数, 所属院系;

2.3　联系分析

- 教室对教学楼有位置联系, 即教室位于特定教学楼;
- 教室, 课程与时间存在联系, 每个课程在特定时间的特定教室上课;
- 课程老师存在联系, 某个课程由特定老师负责;

[3]可能是老师工作的具体院系, 如” 计算机系”, 也可能是其他研究所, 如” 基础数学研究所”

- 班级与学院存在联系, 一个班级属于一个学院;
- 老师与院系存在联系, 老师属于院系;

2.4 概念模型设计

经过需求分析和实体属性分析，以及各实体之间的关系分析，最终得到概念模型如图 6[4]

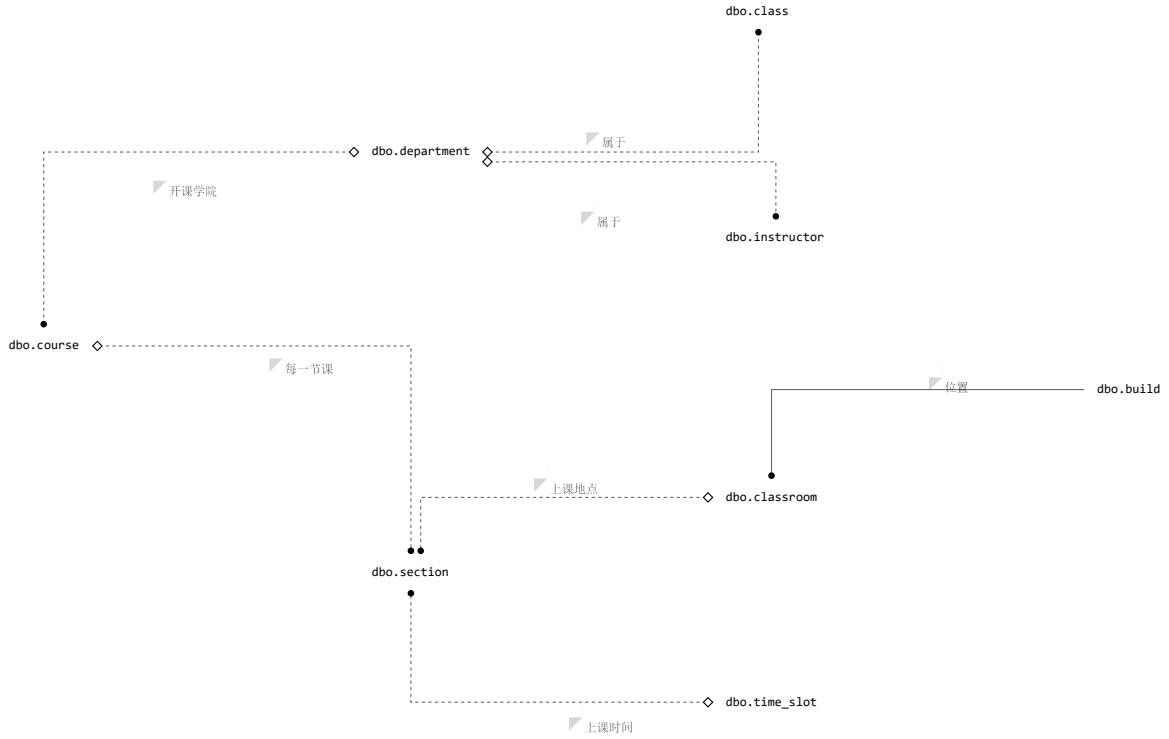

图 **6** 数据库的概念模型 **(**用 **sqlqbm** 制图 **[14])**

3 数据库逻辑结构设计

3.1 概念模型转化为逻辑模型

3.1.1 一对一关系的转化

在数据库系统概念中 [13],1 : 1 关系的定义为

> An entity in A is associated with *at most* one entity in B, and an entity in B is associated with *at most* one entity in A.

根据定义可以看出, 数据库里没有一一对应关系.

[4]访问https://app.sqldbm.com/SQLServer/Share/3gOZcBDE7lI7tlvWVLWffUGFrngIE8md_DYjF4jNYw0可在线查看

3.1.2 一对多关系的转化

一个 $1:n$ 联系有以下两种转换方式

1. 转换为一个独立的关系模式

   关系的属性：与该联系相连的各实体的码以及联系本身的属性.

   关系的码：n 端实体的码.

2. 与 n 端对应的关系模式合并 合并后关系的属性：在 n 端关系中加入 1 端关系的码和联系本身的属性

   合并后关系的码：不变

与 n 端对应的关系模式合并可以减少系统中的关系个数，一般情况下更倾向于采用这种方法.

在本系统中, 院系对班级是 $1:n$ 的关系, 一个院系有多个班级, 一个班级只能属于一个院系, 故班级表主键为班级号, 外键为院系号, 该键作为院系表的主键. 同理课程, 老师对院系也是 $1:n$ 的关系, 为院系号仍作为他们的外键.

此外, 教学楼对教室也是 $1:n$ 关系, 将教学楼号作为教室的外键, 教室, 教学楼主键分别是他们的编号.

3.1.3 多对多关系的转化

一个 $m:n$ 联系可以转换为一个关系模式, 其中:

关系的属性：与该联系相连的各实体的码以及联系本身的属性

关系的码：各实体码的组合

本系统中, 上课为多对多关系, (时间号, 课程号, 教室号, 教师号) 为主键, 分别参照时间表, 课程表, 教室表, 教师表.

3.2 逻辑模型设计

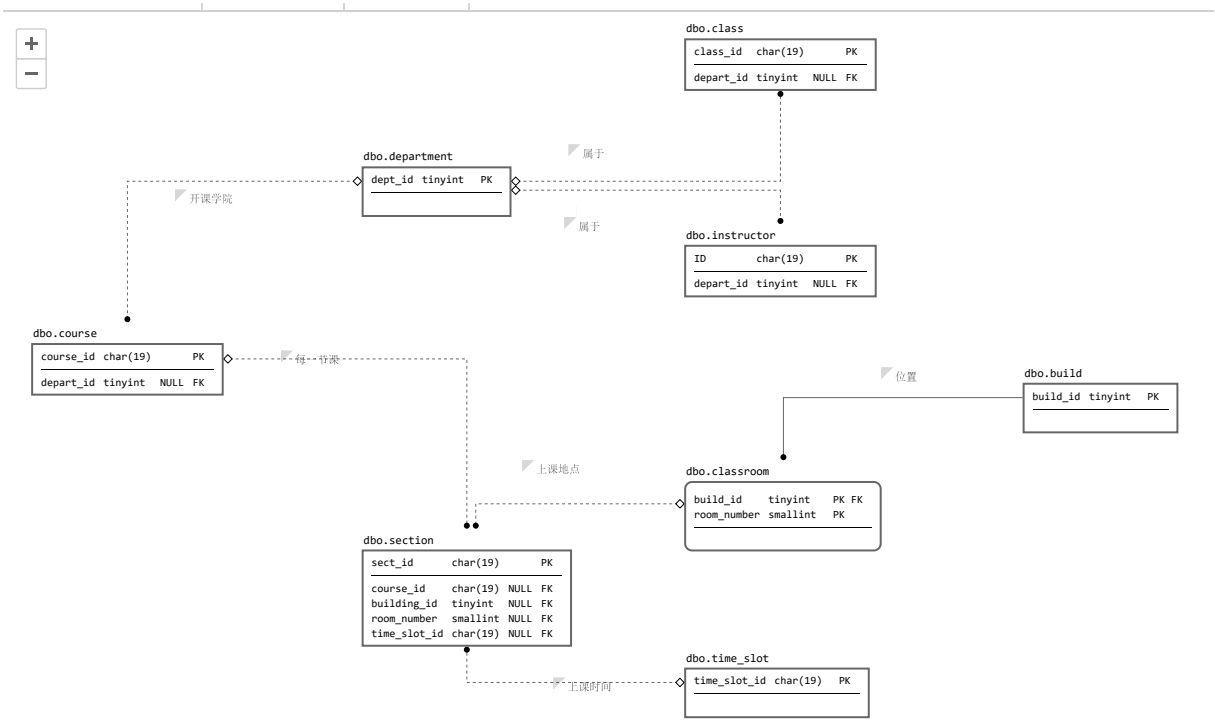

图 **7** 逻辑结构设计

4 数据库的物理实现

4.1 表设计

数据库中各表如图 8

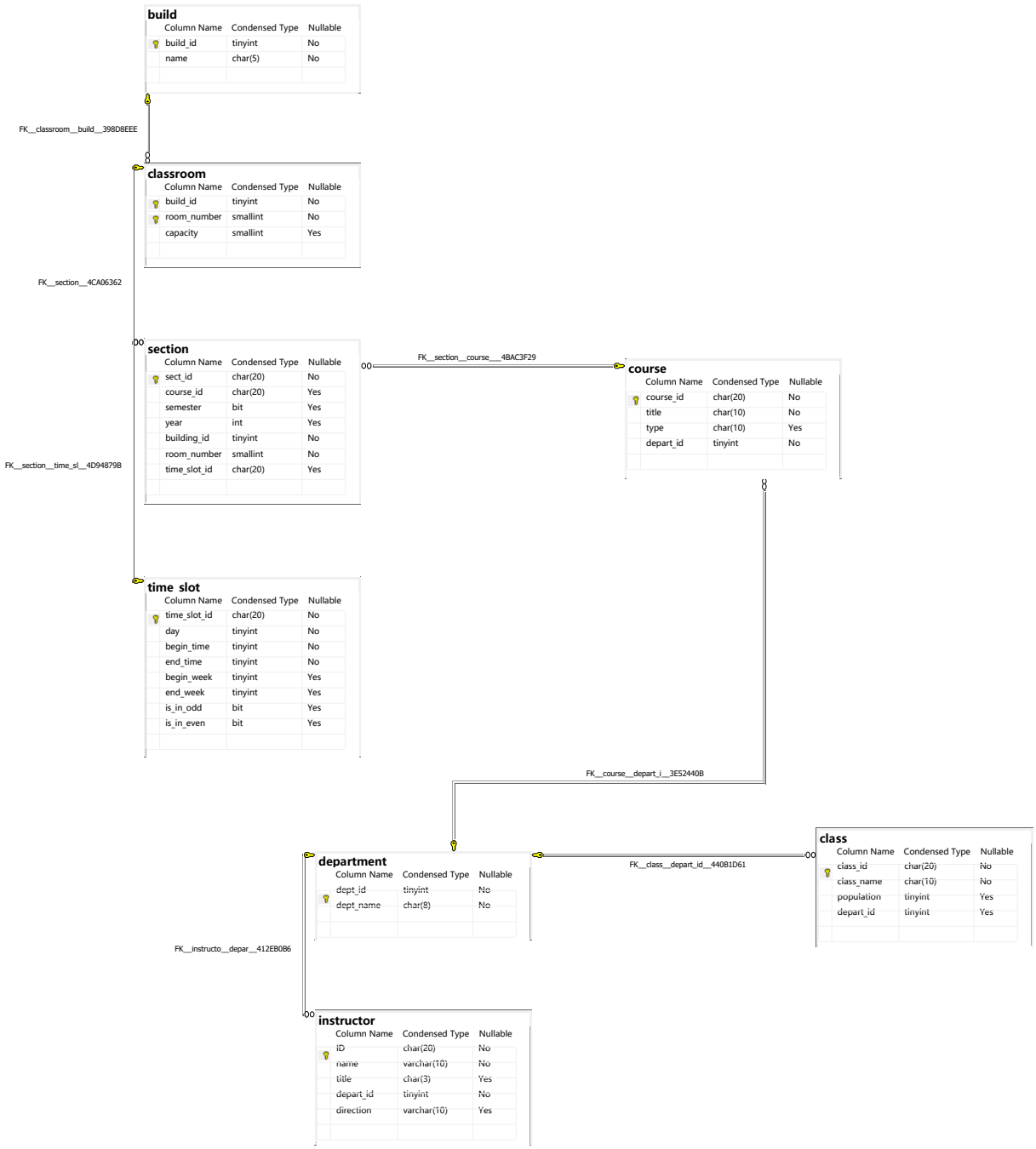

图 **8**　**SQL sever** 中各表设计 [1]

4.2　创建表和完整性约束代码设计

4.2.1　教学楼表

```
create table build(
  build_id tinyint,
  name char(5) not null,
  primary key (build_id),
```

```
);
```

4.2.2 教室表

```
create table classroom(
  build_id tinyint,
  room_number smallint not null,
  capacity smallint,
  primary key (build_id, room_number),
  foreign key (build_id) references build,
);
```

4.2.3 院系表

```
create table department(
  dept_id tinyint,
  dept_name char(8) not null,
  primary key (dept_id),
);
```

4.2.4 课程表

```
create table course(
  course_id bigint,
  title char(10) not null,
  type char(10),
  instructor_id bigint not null,

  primary key (course_id),
  foreign key (instructor_id) references instructor(ID),
);
```

4.2.5 教师表

```
create table instructor(
```

```
  ID bigint,
  name varchar(10) not null,
  title char(3), /* 职称*/
  depart_id tinyint not null,
  direction varchar(10), /* 研究所或者所属系 */

  primary key (ID),
  foreign key (depart_id) references department,
);
```

4.2.6 班级表

```
create table class(
  class_id bigint,
  class_name char(10) not null,
  population tinyint,
  depart_id tinyint,

  primary key (class_id),
  foreign key (depart_id) references department,
);
```

4.2.7 时间表

```
create table time_slot(
  time_slot_id bigint,
  day tinyint not null,
  begin_time tinyint not null,
  end_time tinyint not null,
  begin_week tinyint default 1,
  end_week tinyint,

  is_in_odd bit default 1, /* 单周是否上课 */
  is_in_even bit default 1, /* 双周是否上课 */

  primary key (time_slot_id),
);
```

4.2.8 上课表

```
create table section(
  sect_id bigint,
  course_id bigint,
  class_id bigint,
  semester bit, /* 上半年或下半年 */
  year_ int, /* 开课年份 */
  build_id tinyint not null,
  room_number smallint not null,
  time_slot_id bigint,

  primary key (sect_id),
  foreign key (course_id) references course,
  foreign key (build_id,room_number) references classroom,
  foreign key (time_slot_id) references time_slot,
  foreign key (class_id) references class
);
```

4.3 创建视图、索引、存储过程和触发器

4.3.1 视图: 我的课表

班级课表视图代码如下

```
  use mytimetable;

  go
  create view class_timetable
  as
    select
      build.name as 教学楼,
      room_number as 教室号,
      course.title as 课程名,
      day as 星期,
      begin_time as 上课时间,
      end_time as 下课时间,
      instructor.name as 教师名,
      class_name as 上课班级
    from
      section,
      course,
      time_slot,
      instructor,
      build,
```

```
    class
  where
    section.course_id = course.course_id
    and section.time_slot_id = time_slot.time_slot_id
    and course.instructor_id = instructor.ID
    and section.build_id = build.build_id
    and section.class_id = class.class_id
    and semester = 0 -- 表示第一学期
    and year_ = 2019 -- 2019年
    and 3 between begin_week and end_week
    and is_in_odd = 1
    and class_name = '数学1801'
```

4.3.2 触发器: 检查教室是否被占用

当增加记录时, 检查同时间的教室是否被使用.[9, 3]

```
use mytimetable;
go
create trigger room_check on section
after
insert, update
  as
  if exists(
    select *
    from
      section as s
      join inserted as i
      on i.time_slot_id = s.time_slot_id
    where
      s.build_id = i.build_id
      and s.room_number = i.room_number
  )
  begin
  print 'the room is not empty' ;rollback;
end
```

下面尝试修改, 制造冲突

```
use mytimetable;
update section
set time_slot_id =
            -6176928835455278985,
build_id = 1,
room_number = 404
where course_id = (
```

```
        select course_id
        from course
        where title='数学分析')
```

结果被 sql 终止 (如图 9)

图 **9**　对触发器的测试

4.3.3　索引

根据官方文档的提示 [11]: 设计索引时，应考虑以下数据库准则：

- 对表编制大量索引会影响 INSERT、UPDATE、DELETE 和 MERGE 语句的性能，因为当表中的数据更改时，所有索引都须进行适当的调整。例如，如果某个列在几个索引中使用且您执行修改该列数据的 UPDATE 语句，则必须更新包含该列的每个索引以及基础的基表（堆或聚集索引）中的该列。
- 避免对经常更新的表进行过多的索引，并且索引应保持较窄，就是说，列要尽可能少。
- 使用多个索引可以提高更新少而数据量大的查询的性能。大量索引可以提高不修改数据的查询（例如 SELECT 语句）的性能，因为查询优化器有更多的索引可供选择，从而可以确定最快的访问方法。
- 对小表进行索引可能不会产生优化效果，因为查询优化器在遍历用于搜索数据的索引时，花费的时间可能比执行简单的表扫描还长。因此，小表的索引可能从来不用，但仍必须在表中的数据更改时进行维护。
- 视图包含聚合、表联接或聚合和联接的组合时，视图的索引可以显著地提升性能。若要使查询优化器使用视图，并不一定非要在查询中显式引用该视图。

本数据库规模不大, 删改频繁, 不适合建立索引

4.3.4　存储过程: 按老师查询

存储过程可以执行带参数的命令查询 [12]

```
    use mytimetable
    go

    create procedure teacher_timetable @name varchar(10),
```

```
@semester bit,
/* 上半年或下半年 */
@year_ int
/* 开课年份 */
as
select
  build.name as 教学楼,
  room_number as 教室,
  day as 星期,
  begin_time as 开始时间,
  end_time as 结束时间,
  course.title as 课程,
  instructor.name as 授课教师,
  class_name as 班级,
  dept_name as 开课学院
from
  section
  join course on course.course_id = section.course_id
  join instructor on course.instructor_id = instructor.ID
  join department on department.dept_id = instructor.dept_id
  join build on build.build_id = section.build_id
  join time_slot on time_slot.time_slot_id = section.time_slot_id
  join class on class.class_id = section.class_id
where
  instructor.name = @name
  and semester = @semester
  and year_ = @year_
```

4.3.5 存储过程: 按班级查询

类似地, 利用存储过程, 按班级查课表

```
use mytimetable
go

create procedure class_timetable @class_name varchar(10),
@semester bit,
/* 上半年或下半年 */
@year_ int
/* 开课年份 */
as
select
build.name as 教学楼,
room_number as 教室,
day as 星期,
begin_time as 开始时间,
```

```
    end_time as 结束时间,
    course.title as 课程,
    instructor.name as 授课教师,
    class_name as 班级,
    dept_name as 开课学院
    from
    section
    join course on course.course_id = section.course_id
    join instructor on course.instructor_id = instructor.ID
    join department on department.dept_id = instructor.dept_id
    join build on build.build_id = section.build_id
    join time_slot on time_slot.time_slot_id = section.time_slot_id
    join class on class.class_id = section.class_id
    where
    class.class_name = @class_name
    and semester = @semester
    and year_ = @year_
```

5 数据库功能调试

接下来用 *Transact-SQL* 模拟函数功能

5.1 查询我的课表

直接从视图 *class_timetable* 中查询

```
select * from class_timetable
```

结果如 图 10

```
go
select *
from
class_tim
etable
```

	教学楼	教室号	课程名	星期	上课时间	下课时间	教师名	上课班级
1	实验	211	数学分析	1	8	10	上帝	数学1801

图 **10** 查询我的课表视图

在应用程序中, 用应用程序的变量替换其中的一些信息 (如班级), 检测到设置中班级修改, 重新建立类似的视图, 关注的课表也有类似操作.

5.2 查老师的课表

利用 procedure teacher_timetable, 第一个参数为教师姓名, 第二个为学期, 第三个为学年

```
    exec teacher_timetable '上帝', 0, 2019
```

图 11

```
    @year_

go
exec
teacher_timetable '
上帝', 0, 2019
```

教学楼	教室	星期	开始时间	结束时间	课程	授课教师	班级
实验	211	1	8	10	数学分析	上帝	数学1801

图 **11**　查询随机数据中”上帝”的课

5.3　查询指定班级课表

利用存储过程 class_timetable, 直接查询

```
    exec class_timetable '计科1801', 0, 2019
```

查询结果如 图 12

```
go
exec class_timetable
'计科1801', 0, 2019
```

	教学楼	教室	星期	开始时间	结束时间	课程	授课教师
1	一教	404	1	10	12	密码学	图灵

图 **12**　查询随机数据中”计科 **1801**”课表

6　应用程序设计

- 项目地址: `https://github.com/chenboshuo/learn_database/tree/final_project/final_project`

本项目利用 python 的框架 PyQt5 [2] 实现用户界面.

6.1　登录界面

根据 [8] 的教程, 制作登录界面如 图 13

图 **13**　登录界面

先用模拟的账号尝试登录如 图 14

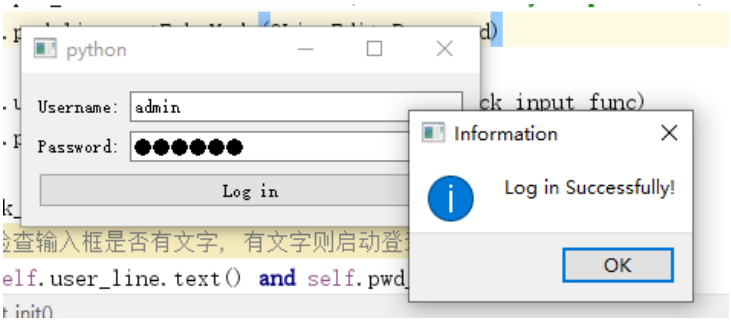

图 **14**　直接设定登录名与密码模拟登录

登录成功.

6.2　链接数据库

根据 [15] 提供的教程, 先用 sql 语句创建测试用户:

```
use mytimetable;

-- 创建一个登录账号TestUser
create login testuser with password = '123456';

-- 3. 创建对应于这个登录账号的数据库用户TestUser
create user testuser from login testuser ;
```

然后用 *QtSql* 模块, 进行链接测试.

图 **15**　测试 **PyQt5** 与 **sqlserver** 的链接

测试通过后, 将该功能封装成函数, 链接为数据库对象的构造函数, 关闭为数据库析构函数, 如 图 16

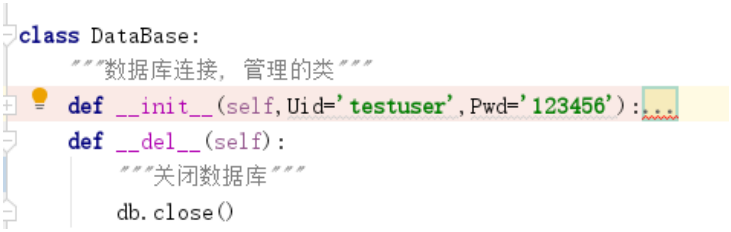

图 **16**　简单的数据库对象图

接下来让数据库连接: 传递输入的变量, 尝试链接数据库, 失败时抛出异常 [17], 登录窗口检测异常, 要求重新输入 (图 18); 若无异常, 链接 (如图 17)

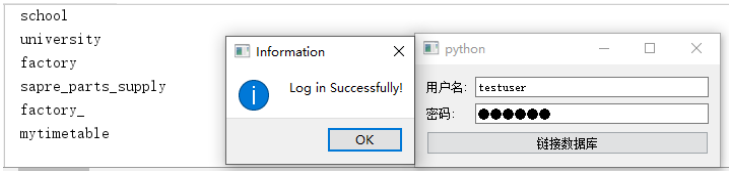

图 **17**　连接成功后控制台打印相关信息

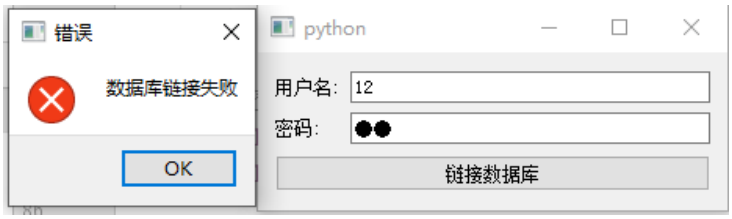

图 **18**　链接失败提示并重新输入

6.3　主界面

根据 [16] 提供的教程和 [6] 提供相关的图标样式, 设计主界面如 图 19

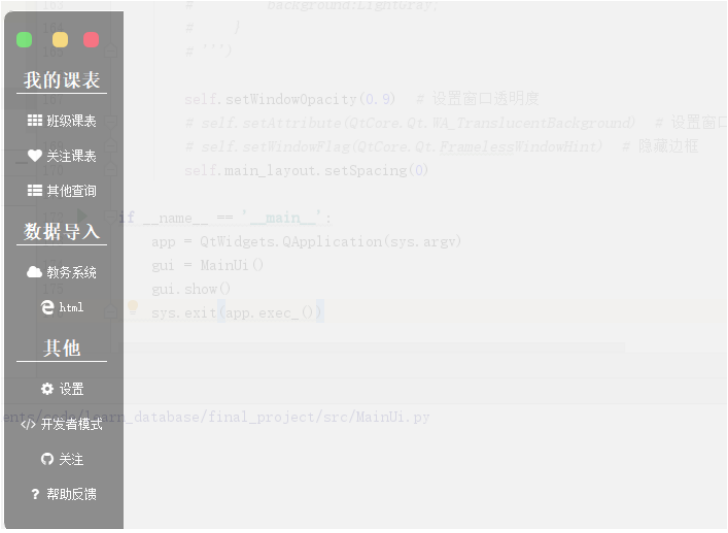

图 **19**　程序主要功能界面

接下来插入表格元件, 放到窗口右侧, 用于显示查询结果 图 20

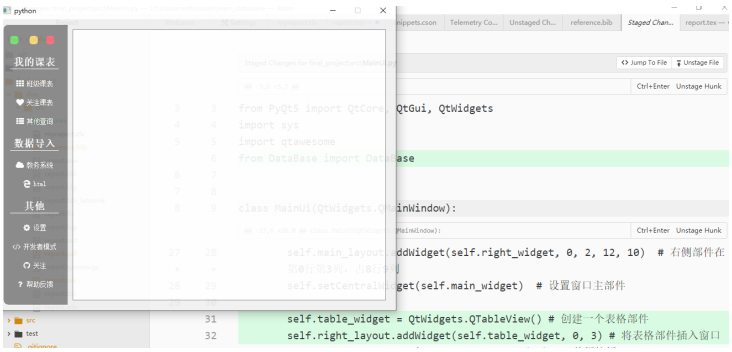

图 **20**　首先在界面中插入表格元件, 用于展示查询结果

7　设计总结

7.1　Tips

- 可以用 *join..on* 语句替代链接条件, 增加可读性
- 触发器可以用于提醒消息, 提醒电子邮件, 强制实现业务规则, 具有数据库范围的 DDL 触发器, 登录触发器等功能
- procedure 可以当作数据库的函数

7.2　应用改进

在开始设计数据库时, 是想把它当作查询课表工具使用, 因为用教务系统查询其他学院课表时会消耗大量时间, 这样的重复工作理应当用程序实现, 但是查询别人课表是

个小众需求, 市场上并没有课程表软件实现查询非自己班级课表的功能, 所以我一直想找机会实现.

由于课程设计要求, 需要设计多角色多过程, 但是, 最终我希望它不是一个用于当作作业提交的系统, 而是一个好的辅助工具, 因此在界面设计是留下一些未完成的功能, 比如 html 转化, 在今后有时间可以将 html 的图表格式通过识别信息储存到数据库中实现查询, 还有爬虫, 可以直接登录教务系统查询结果并保存到本地, 当然这需要大量时间.

7.3 其他感想

老师要求使用的 *powerdesiner* 是不太好用的软件, 现在有在线服务可以将代码转换为示意图 [14], 设计精确到代码也许能保证后面的工作可以正确进行, 且代码的操作比用鼠标操作更高效. 对于其他类型的图片,`draw.io`提供了更美观更自由, 更多样的图片, 并且可以自由导出 [4], 互联网上有大量的优质软件和服务, 作为设计者, 不应该固步自封, 拘泥于某个软件, 有的软件自然好用方便, 但是可能有他不擅长的功能, 选择的时候要目光长远, 不能说能完成, 而是更高效优雅的完成. 比如用鼠标画图生成代码恐怕不是好主意, 诚然, 图片的转化是逻辑转化的过程, 但是图片一旦出错, 难以调整修改, 绘图操作又浪费了大量时间. 迷恋于可视化操作, 当真正需要用程序调用的时候, 又不能很好的完成, 因为没有代码的思维甚至抵触代码, 对我们来说可能是致命的.

对于应用程序, 我选用了面向对象的设计思维, 使用 *python* 的 *PyQt5* 可以较快的完成特别美观的界面, 开发代码也简洁优雅, 并且有强大的界面美化库 [6], 程序可扩充性较强, 并且有简单的测试框架 [7] 可以用 *python* 的强大生态做成实用的工具.

同时, 由于环境版本号比较容易获得, 容易找到问题答案, 例如, 在制作应用程序时, 由于接口不熟悉, 出现窗口切换问题, 通过简单描述, 在 *stackoverflow* 上找到优质答案 [5], 这种方便性是小众软件不能比的.

而老师推荐的方式我们并不熟悉, 并且程序的可移植性很差, 并不能说是好的工具.

此外, 查看了官方文档和数据库系统概念 [13], 里面强调编程风格的重要性, 好的可读性有利于系统维护.

参考文献

[1] Adam Adamowicz. How to create er diagram for existing sql server database with ssms. `https://dataedo.com/kb/tools/ssms/create-database-diagram`.

[2] Riverbank Computing. Pyqt5 reference guide. `https://www.riverbankcomputing.com/static/Docs/PyQt5/`.

[3] and so on craigg msft, MightyPen. Use the inserted and deleted tables. https://docs.microsoft.com/en-us/sql/relational-databases/triggers/use-the-inserted-and-deleted-tables?view=sql-server-ver15.

[4] drawioapp. draw.io. https://www.draw.io/.

[5] ekhumoro. Login dialog pyqt. https://stackoverflow.com/a/11812578.

[6] Dave Gandy. Font awesome 一套绝佳的图标字体库和 css 框架. http://fontawesome.dashgame.com/.

[7] holger krekel and pytest-dev team. pytest: helps you write better programs. https://docs.pytest.org/en/latest/.

[8] la_vie_est_belle. 《快速掌握 PyQt5》第五章完善登录框小程序. https://blog.csdn.net/La_vie_est_belle/article/details/82389163.

[9] and so on MashaMSFT, craigg-msft. Create trigger (transact-sql). https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql.

[10] Microsoft. int,bigint,smallint,and tinyint (transact-sql). https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql.

[11] olprod. Sql server 索引设计指南. https://docs.microsoft.com/zh-cn/sql/2014-toc/sql-server-index-design-guide?view=sql-server-2014&viewFallbackFrom=sqlallproducts-allversions.

[12] OpenLocalizationService olprod. Create procedure (transact-sql). https://docs.microsoft.com/zh-cn/sql/t-sql/statements/create-procedure-transact-sql?view=sql-server-ver15#best-practices.

[13] Abraham Silberschatz, Henry F Korth, Shashank Sudarshan, et al. *Database system concepts*, volume 4. McGraw-Hill New York, 1997.

[14] LLC "We Comes Before Me". Sql database modeler. https://sqldbm.com.

[15] 乐松. [pyqt]pyqt5 连接 sql server 数据库，并读取所有数据库名称. https://www.cnblogs.com/syh6324/p/9491518.html.

[16] 州的先生. Python gui 教程（十六）：在 pyqt5 中美化和装扮图形界面. https://zmister.com/archives/477.html.

[17] 方 naoke. python 自定义异常和主动抛出异常（raise）. https://blog.csdn.net/skullFang/article/details/78820541.