

性能方面可采用的策略

1、采用 ajax 技术

在本项目没有使用 ajax 技术，例如在登录页面中，当账号和密码输入错误导致服务器发回错误信息时，会附带整个登录页面发回，无疑增加无用的数据量，浪费了服务器的带宽，在目前的整个应用中，这种情况还有确认密码修改，确认选课，以及当前主要的页面切换。基于这种考虑可以采用 ajax 来只传输不同的数据。

2、镜像服务器

在目前的项目中 web 服务器和中间层，数据库服务器都只有一台，当数据访问量和访问人数的增多，将导致服务器负担大，因此可对各个服务器或其中负担较大的服务器进行镜像，来分流数据量和访问人数，从而增强性能需求

3、优化数据库

① 采用存储过程

由于存储过程是预先编译存放在数据库的可调用函数，它比客户端传送 sql 语句给服务器来的效率高，单条的 sql 每次传送到数据库都要编译和优化，而存储过程只在第一次的运行时优化，当有大量相同格式的 sql 传送，可采用存储过程来增强效率，在本项目中由于查询占多数，如登录查询，个人信息查询，成绩查询

② 创建数据库索引

由于在本项目中查询占多数，因此可对数据库表创建索引，来促进数据库的检索效率，如在如下几个表中创建索引

在 login 表中，可对登录名两个字段建立索引增加检索效率

```
MySQL 5.6 Command Line Client - Unicode

mysql> show tables;
+-----+
| Tables_in_xdem |
+-----+
| courselist      |
| electiveinfo    |
| electivelist    |
| grade          |
| login          |
| personinformation |
+-----+
6 rows in set (0.00 sec)

mysql> desc login;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| loginName  | char(10)  | NO   | PRI |          |       |
| password   | char(32)  | NO   |     | NULL    |       |
| type       | char(6)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

个人信息表中的学号建立索引

```
MySQL 5.6 Command Line Client - Unicode

3 rows in set (0.01 sec)

mysql> desc personinformation;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| 学号       | char(10)  | YES  | MUL | NULL    |       |
| 姓名       | char(20)  | NO   |     | NULL    |       |
| 班级       | char(20)  | NO   |     | NULL    |       |
| 性别       | char(4)   | NO   |     | NULL    |       |
| 民族       | char(20)  | NO   |     | NULL    |       |
| 政治面貌   | char(20)  | NO   |     | NULL    |       |
| 专业       | char(20)  | NO   |     | NULL    |       |
| 身份证号   | char(18)  | NO   |     | NULL    |       |
| 联系电话   | char(15)  | YES  |     | NULL    |       |
| 备注       | char(255) | YES  |     | NULL    |       |
| email      | char(30)  | YES  |     | NULL    |       |
| 入学年份   | year(4)   | YES  |     | NULL    |       |
| 出生日期   | date      | YES  |     | NULL    |       |
| 学院       | varchar(20) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
14 rows in set (0.08 sec)

mysql>
```

在课程列表中课程 id 和学生 id 的建立索引

```
MySQL 5.6 Command Line Client - Unicode
+----+-----+-----+-----+-----+-----+
| 入学年份 | year(4) | YES | | NULL | |
| 出生日期 | date | YES | | NULL | |
| 学院 | varchar(20) | YES | | NULL | |
+----+-----+-----+-----+-----+-----+
14 rows in set (0.08 sec)

mysql> desc courselist;
+----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+----+-----+-----+-----+-----+-----+
| courseid | varchar(20) | NO | PRI | NULL | |
| coursetype | varchar(20) | NO | | NULL | |
| coursename | varchar(20) | NO | | NULL | |
| grade | varchar(10) | YES | | NULL | |
| credit | float | NO | | NULL | |
| studentid | varchar(20) | NO | PRI | NULL | |
| orderInsert | int(11) | NO | | NULL | |
+----+-----+-----+-----+-----+-----+
7 rows in set (0.01 sec)

mysql> _
```

③ 使用触发器

触发器是一种特殊的存储过程，它可以再一个表上针对增、删、改、查建立触发器，可在触发器上进行一些计算和多表插入等，例如在本项目中设置了三个触发器用于辅助工作，

触发器一：

```
MySQL 5.6 Command Line Client - Unicode
ITUTION | root@::1 | utf8 | utf8_general_ci | utf8_general_ci
ci |
| InsertGrade | INSERT | personinformation | begin
insert into grade values(NEW.学号,NEW.姓名,NEW.学院,NEW.专业,0,0,0);
end
```

当管理员插入一个新的学生信息时，该触发器将向个人信息中插入基本的信息。

触发器二：

该触发器建立在 courselist 上，当教师更新学生的成绩时会触发，如果成绩是数字，计算平均成绩

```

MySQL 5.6 Command Line Client - Unicode

| Timing | Created | sql_mode
| Definer | character_set_client | collation_co
nnection | Database Collation |
+-----+-----+-----+-----+
| updateInsert | UPDATE | courselist | begin
| update grade set grade.credit = grade.credit+old.credit where studentID = o
ld.studentid;
if new.grade>0 then
update grade set grade.grade = grade.grade+new.grade,grade.gradenum=grade.grade
num+1 where studentID = old.studentid;
end if;
end

```

触发器三：

该触发器建立在选课表上，当学生选课后触发，向学生的课程列表中加入选课信息。

```

MySQL 5.6 Command Line Client - Unicode

| updateInsert | UPDATE | courselist | begin
| update grade set grade.credit = grade.credit+old.credit where studentID = o
ld.studentid;
if new.grade>0 then
update grade set grade.grade = grade.grade+new.grade,grade.gradenum=grade.grade
num+1 where studentID = old.studentid;
end if;
end

| AFTER | NULL | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | root@:1 | utf8
| utf8_general_ci | utf8_general_ci |
| updatecourse | INSERT | electiveinfo | begin
| set @num = (select count(*) from courselist where courselist.studen
tid = new.studentid)+1;
| set @id = (select courseid from electivelist where orderShow = new.
orderShow);
| set @type = (select coursetype from electivelist where orderShow =
new.orderShow);
| set @name = (select coursename from electivelist where orderShow =
new.orderShow);
| set @credit = (select credit from electivelist where orderShow = ne
w.orderShow);
| insert into courselist values(@id,@type,@name,',',@credit,new.stude
ntid,@num);
end | AFTER | NULL | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBST
ITUTION | root@:1 | utf8 | utf8_general_ci | utf8_general_
ci |
| InsertGrade | INSERT | personinformation | begin
| insert into grade values(NEW.学号,NEW.姓名,NEW.学院,NEW.专业,0,0,0);
end

| AFTER | NULL
| STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION | root@:1
| utf8 | utf8_general_ci | utf8_general_ci |
+-----+-----+-----+-----+

```

使用触发器可以减少发送的 sql 语句的大小,更多的功能在数据库触发器中实现特殊功能,如多表处理的功能,从而减少 sql 的大小和复杂性。

4、数据库配置

如可开启 mysql 数据库的查询高速缓存,修改并发数目。

可用性分析

1、易学性和易记性

目前由于功能较少，并且多数的功能集中在查询上，因此，满足易学性和易记性，可从如下两个页面反映出来



2、有效性

从以上的两个展示页面，可以有效的检索到数据，因此满足有效性的要求

3、效率

目前的项目更能较少，因此用户可以很快的完成任务，满足效率性

4、容错度

目前项目的没有过多错误处理，因此容错度还不足