

CS307 Project 2

Presentation

王奕童 11910104

张彤 11911831

Content

- Step 0.小组分工与贡献
- Step 1.重新评测Part1 (skipped)
- Step 2. searchCourse接口实现介绍
- Step 3. 先修关系与enrollCourse接口实现介绍
- Step 4. Additional Works (Divided into Several Parts)
- Step 5. Conclusion

1.部分设计参考于Project1

2.代码可见于：<https://github.com/YeeTone/CS307Project2-2021Spring>

Step 0.

小组分工与贡献

By YeeTone WANG

Step 0.小组分工与贡献

- 小组成员数：2

姓名	SID	专业	贡献比
王奕童	11910104	计算机科学与技术	50
张彤	11911831	计算机科学与技术	50

Step 1.

重新评测Part1

Just skipped

Step 2.

searchCourse接口实现介绍

By YeeTone WANG

Step 2.1 searchCourse接口简览

```
List<CourseSearchEntry> searchCourse  
(int studentId, int semesterId, @Nullable String searchCid,  
@Nullable String searchName, @Nullable String searchInstructor,  
@Nullable DayOfWeek searchDayOfWeek, @Nullable Short searchClassTime,  
@Nullable List<String> searchClassLocations,  
CourseType searchCourseType,  
boolean ignoreFull, boolean ignoreConflict,  
boolean ignorePassed, boolean ignoreMissingPrerequisites,  
int pageSize, int pageIndex);
```

CourseSearchEntry		
f	course	Course
f	section	CourseSection
f	sectionClasses	Set<CourseSectionClass>
f	conflictCourseNames	List<String>

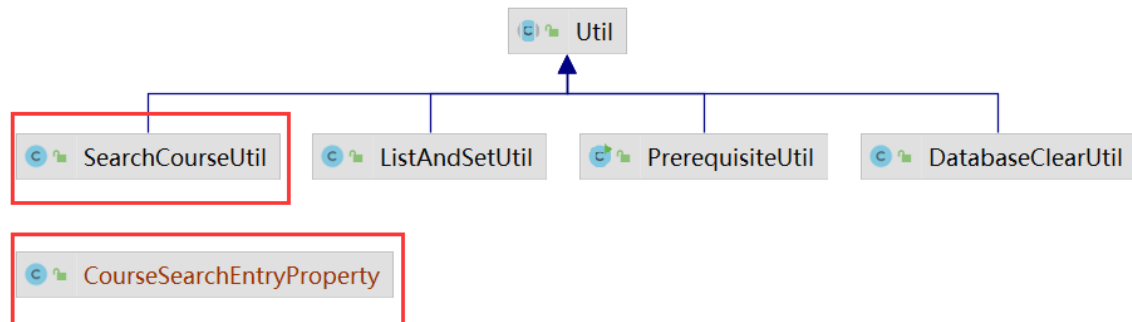
CourseSection		
f	id	int
f	name	String
f	totalCapacity	int
f	leftCapacity	int

Course		
f	id	String
f	name	String
f	credit	int
f	classHour	int
f	grading	CourseGrading

CourseSectionClass		
f	id	int
f	instructor	Instructor
f	dayOfWeek	DayOfWeek
f	weekList	Set<Short>
f	classBegin	short
f	classEnd	short
f	location	String

- 要求：根据传入的参数类型，返回课程查询的相关结果
- 实现特点：
代码量大+多种情况判断+类之间关联情况复杂
- 本地用例通过情况：
search1: 1000/1000
search2: 972/1000

Step 2.2 searchCourse接口总体实现文件



```
try{
    return SearchCourseUtil.searchCourse(
        studentId, semesterId, searchCid,
        searchName, searchInstructor,
        searchDayOfWeek, searchClassTime,
        searchClassLocations, searchCourseType,
        ignoreFull, ignoreConflict, ignorePassed,
        ignoreMissingPrerequisites,
        pageSize, pageIndex);
}catch (Exception e){
    e.printStackTrace();
    throw new IntegrityViolationException();
}
```

- 核心处理类：

reference.util.SearchCourseUtil

- 辅助工具类：

reference.util.

CourseSearchEntryProperty

- 设计目的：

独立用一个类实现功能，减少代码修改犯错的可能性

- 总体Java层面代码量(该接口实现)：

241+414=655 lines

Step 2.3 searchCourse接口实现介绍

- Problem1: 接口中多个nullable参数

```
@Nullable String searchCid, @Nullable String searchName,  
@Nullable String searchInstructor, @Nullable DayOfWeek searchDayOfWeek,  
@Nullable Short searchClassTime, @Nullable List<String> searchClassLocations,
```

- Fact about PostgreSQL :

在多数情况下，对null进行判断后结果仍然是null

Nulls?

- Difficulty in the Implementataion

如果使用=判断null，则会出现筛选结果的错误

known + unknown = unknown

Step2.3.searchCourse接口实现介绍

- Problem1: 接口中多个nullable参数

```
@Nullable String searchCid, @Nullable String searchName,  
@Nullable String searchInstructor, @Nullable DayOfWeek searchDayOfWeek,  
@Nullable Short searchClassTime, @Nullable List<String> searchClassLocations,
```

- Solution1: Brute Force

暴力枚举 $2^6=64$ 种可能的sql语句

- 容易出错+代码冗余+实现不优雅



实验2班 王子勤

给个小提示:

```
select * from table  
where ? is null or col1 = ?
```

- Solution2: or 额外条件判断

- Example:

```
"where (c.courseid like('%'||?||'%') or ? is null) " +  
"and (c.coursename||'['||cs.sectionname||']' like '%'||?||'%') or ? is null)" +  
"and (? is null or i.firstname||i.lastname like('%'||?||'%') " +  
"or(i.firstname||' '||i.lastname like('%'||?||'%')))" +  
"or i.firstname like('%'||?||'%') or i.lastname like('%'||?||'%'))" +  
"and (? is null or c2.dayofweek=?) " +  
"and (? is null or ? between c2.classbegin and c2.classend)" +  
"and (cs.semesterid=?) " +
```

Step 2.3 searchCourse接口实现介绍

- Problem 2: 建立CourseSearchEntry中规定的映射关系
(Course+ CourseSection 映射到Set)
- Solution: 设计一个封装Course和CourseSection对象的类，然后使用Map数据结构完成映射
- Difficulty: 多次查询时leftCapacity可能变化，导致Map映射错误
- Solution: hashCode中不加入capacity相关元素即可

CourseSearchEntry		
f	course	Course
f	section	CourseSection
f	sectionClasses	Set<CourseSectionClass>
f	conflictCourseNames	List<String>

```
@Override
public int hashCode() {
    return Objects.hash(c, cs.id, cs.name);
}
```

Step 2.3 searchCourse接口实现介绍

- Problem 3: 参数列表中对结果集的筛选：
CourseType参数和多个boolean类型的ignore参数

- Solution: 设计

CourseSearchEntryProperty类，用于标记每个返回结果的状态，然后通过Map数据结构完成相应的映射，最后再进行结果的筛选操作即可

```
public void checkAll(StudentService.CourseType type){  
    checkIsFull();  
    checkIsHasConflict();  
    checkIsHasPassed();  
    checkIsPrerequisitesMissing();  
    checkIsAcceptable4CourseType(type);  
}
```

```
CourseType searchCourseType,  
boolean ignoreFull, boolean ignoreConflict,  
boolean ignorePassed, boolean ignoreMissingPrerequisites,
```

CourseSearchEntryProperty		
f	studentId	int
f	isFull	boolean
f	isHasConflict	boolean
f	isHasPassed	boolean
f	isPrerequisitesMissing	boolean
f	isAcceptable4CourseType	boolean
f	entry	CourseSearchEntry

```
Map<CourseSearchEntry, CourseSearchEntryProperty>  
    entryPropertyMap=new HashMap<>();
```

Step 2.3 searchCourse接口实现介绍

- Problem 4: 数据库大小写和编码问题对参数列表中要求对结果集的排序结果的影响。

- Solution1: Brute Force by Java

对于符合条件的元素，统一转换大小写后使用Java排序，可自动解决编码问题

- Solution2: Better by PostgreSQL

首先使数据库符合UTF-8编码格式，再使用order by和limit+offset的语句做排序筛选

关于searchCourse的测试数据的疑问 #101

Closed

YeeTone opened this issue 7 days ago · 2 comments



YeeTone commented 7 days ago

请问这条测试用例里面，为什么字典序大的MA102a反而排在字典序小的MA102B之前？



NewbieOrange commented 3 days ago

创建符合需求的数据库的方法：

```
create database "project2_test_utf8"
with encoding = 'UTF8'
lc_collate = 'en_US.UTF8'
template = template0;
```

Step 2.4 未能通过search2的原因分析

- search2 Accepted: 972/1000
- Problems:
- 在测试中未能通过的结果集中，Course、Section和Classes和标答保持一致，但是conflictCourseName出现了多余的情况
- Analysis: 当自己和自己冲突时，expected的结果集中没有放入自己的对应名称，但actual的结果集中有自己的名称
- 可参考Issue #151

<https://github.com/NewbieOrange/SUSTech-SQL-Project2-Public/issues/151>

Expected:

```
7 = {ArrayList@3458} size = 10
> 0 = {CourseSearchEntry@3745} "CourseSearchEntry{course=Course{id='BIO102B', name='生命科学概论', cre
> 1 = {CourseSearchEntry@3746} "CourseSearchEntry{course=Course{id='BIO302', name='现代生物技术', cre
> 2 = {CourseSearchEntry@3747} "CourseSearchEntry{course=Course{id='BIO303', name='遗传学实验', credit
> 3 = {CourseSearchEntry@3748} "CourseSearchEntry{course=Course{id='BIO303', name='遗传学实验', credit
> 4 = {CourseSearchEntry@3749} "CourseSearchEntry{course=Course{id='CH102-17', name='化学原理实验A',
> sectionClasses = {ImmutableCollections$Set12@3767} size = 1
> course = {Course@3765} "Course{id='CH102-17', name='化学原理实验A', credit=1, classHour=48, gradir
> section = {CourseSection@3766} "CourseSection{id=101, name='中英双语1班 (缺课不补课, 缺课当次成绩
> conflictCourseNames = {ArrayList@3768} size = 1
> 0 = "化学原理实验A[中英双语2班 (缺课不补课, 缺课当次成绩记为0分)]"
```

Actually:

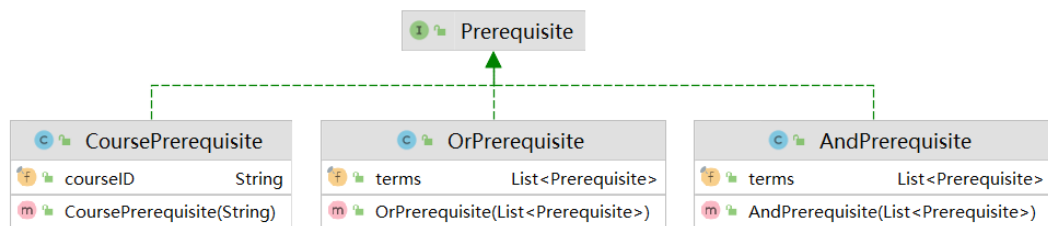
```
7 = {ArrayList@3651} size = 10
> 0 = {CourseSearchEntry@3772} "CourseSearchEntry{course=Course{id='BIO102B', name='生命科学概论', c
> 1 = {CourseSearchEntry@3773} "CourseSearchEntry{course=Course{id='BIO302', name='现代生物技术', cr
> 2 = {CourseSearchEntry@3774} "CourseSearchEntry{course=Course{id='BIO303', name='遗传学实验', cred
> 3 = {CourseSearchEntry@3775} "CourseSearchEntry{course=Course{id='BIO303', name='遗传学实验', cred
> 4 = {CourseSearchEntry@3776} "CourseSearchEntry{course=Course{id='CH102-17', name='化学原理实验A'
> sectionClasses = {HashSet@3786} size = 1
> course = {Course@3784} "Course{id='CH102-17', name='化学原理实验A', credit=1, classHour=48, grac
> section = {CourseSection@3785} "CourseSection{id=101, name='中英双语1班 (缺课不补课, 缺课当次成
> conflictCourseNames = {ArrayList@3787} size = 2
> 0 = "化学原理实验A[中英双语1班 (缺课不补课, 缺课当次成绩记为0分)]"
> 1 = "化学原理实验A[中英双语2班 (缺课不补课, 缺课当次成绩记为0分)]"
```

Step 3.

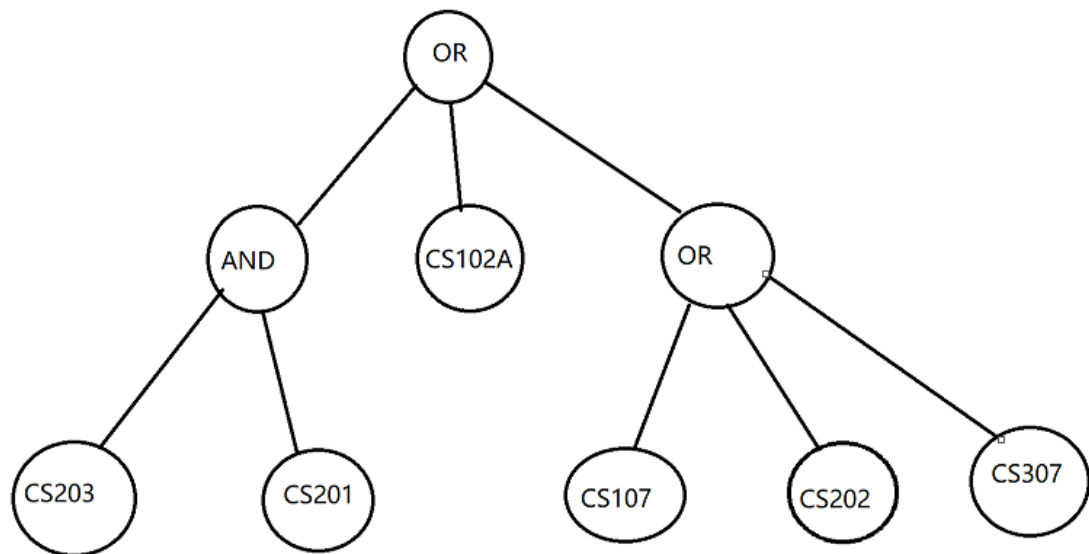
先修关系与enrollCourse接口 实现介绍

By YeeTone WANG

Step 3.1.1 先修关系数据结构解读



- dto数据结构：多叉布尔树
- 特点：中间节点与或类型不定+中间节点孩子数不定+树高度不定



New ALGORITHM is required !

- Solution inProject1 :
二叉树DFS搜索+中间节点类型确定
group分组

- However, **NOT** Work in Project2!

原因分析：数据结构可能不符合要求，现为多叉树且不保证符合project1的分组条件

Step 3.1.2 先修关系算法设计

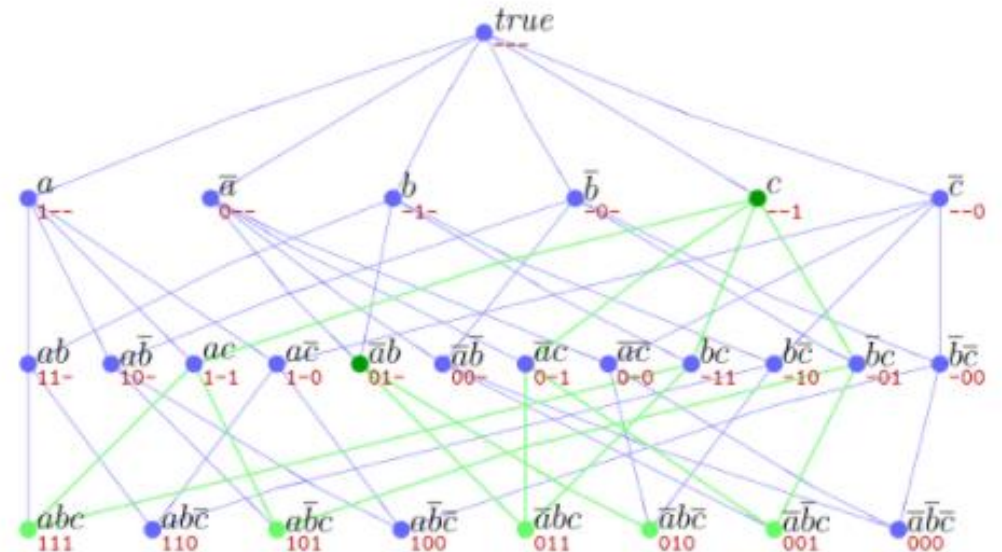
- 算法：Quine–McCluskey算法
- 功能：
 1. 以增加多叉树孩子节点数为代价，强制降低表达式多叉树的高度至 ≤ 2
 2. 将布尔表达式树转化为最小项之和（如数字逻辑课程中定义）
- 时间复杂度：(n为叶子结点个数)

• $T(n) = \frac{3^n}{\ln n}$ 但由于测试数据规模较小，因此仍然在可接受范围内

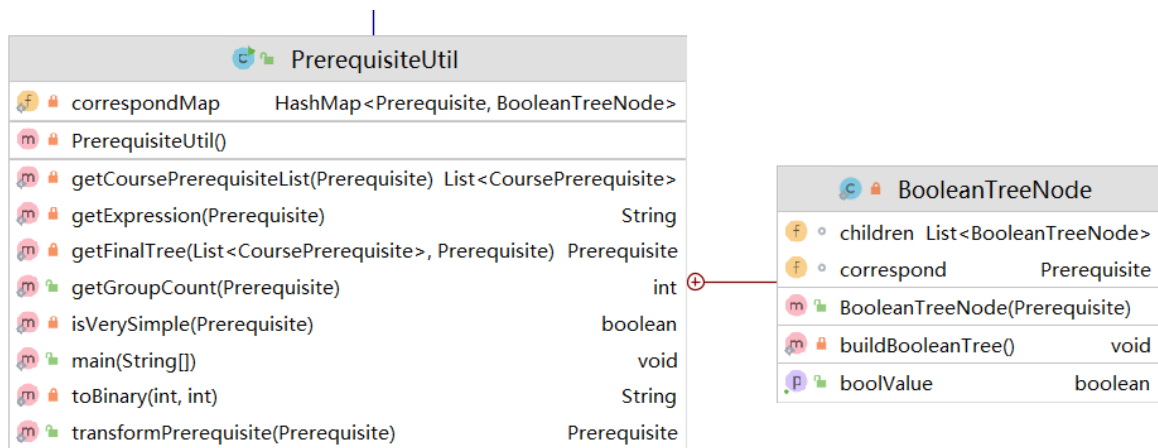


你需要实现以下组件：

1. 一个能求解任意布尔表达式的组件
2. 一个列真值表的组件
3. 一个根据真值表取出最大/最小项的组件 (trivial)



Step 3.1.3 先修关系算法实现



```
public static Prerequisite transformPrerequisite(Prerequisite originRoot){  
    List<CoursePrerequisite> allCourseList= getCoursePrerequisiteList(originRoot);  
    BooleanTreeNode.buildBooleanTree();  
  
    return getFinalTree(allCourseList,originRoot);  
}
```

- 核心处理类：

reference.util.PrerequisiteUtil

- 辅助工具类：

reference.util.PrerequisiteUtil.

BooleanTreeNode

- 设计预期：

实现将传入的Prerequisite进行化简，化简为最小项之和形式

Step 3.1.4 先修课程数据库存储

- 主体函数：ReferenceCourseService类中的insertPrerequisite

```
private synchronized void insertPrerequisite  
(String courseId, @Nullable Prerequisite simplified)
```

- 实现思路：

先利用上述算法实现分组，然后再将分组的结果一一存入数据库中，并在course中存储最大可能的group数值

id	cours...	pre...	group_id
1	CH102-17	RD267	1
2	MSE201	RD267	1
3	MSE305	MSE201	1
4	MSE202	RD267	1
5	MSE202	MA102B	1
6	MSE202	RD267	2
7	MSE202	MA102a	2
8	MSE202	RD267	3
9	MSE202	MA102a	3
10	MSE202	MA102B	3

courseid	coursename	maxprerequisitegroupcount
71	MSE201	1
72	MSE202	3

Step 3.1.5 先修课程数据库检查

- 主体函数：
 - F isprerequisitefullfilled (integer, integer): boolean
 - F isprerequisitefullfilledbycourse (integer, varchar): boolean

- 实现思路：

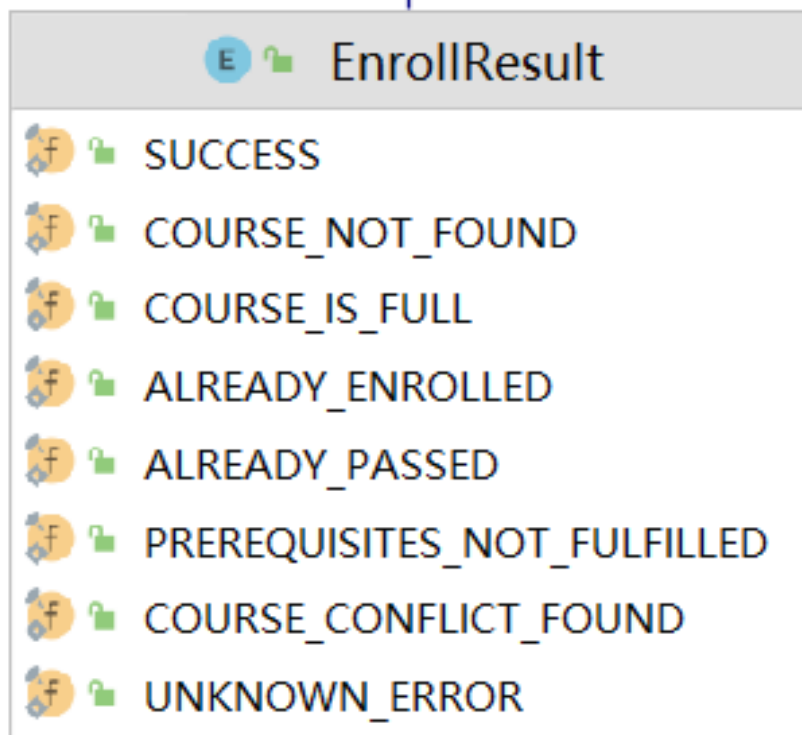
针对Course当中的每一个group，检查该学生的通过情况。对于所有group来说，只需要有一组能够全部满足即可通过（与最小项之和的判定规则保持一致）

```
[2021-06-13 00:35:22] [00000] not fulfilled in the group 1
[2021-06-13 00:35:22] [00000] not fulfilled in the group 2
[2021-06-13 00:35:22] [00000] not fulfilled in the group 3
[2021-06-13 00:35:22] 1 row retrieved starting from 1 in 168 ms (execution: 117 ms, fetching: 51 ms)
```

Step 3.2 enrollCourse接口简览

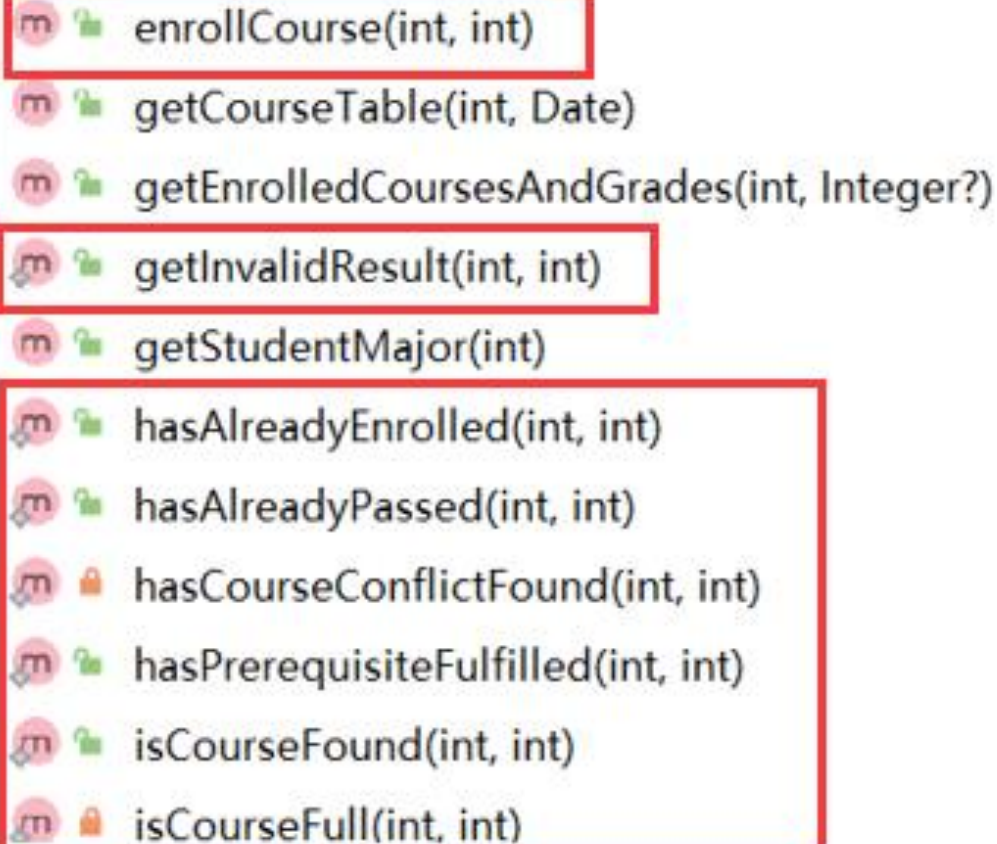
```
EnrollResult enrollCourse(int studentId, int sectionId);
```

- 要求：传入学生和课程，返回学生的八种选课结果之一



- 实现特点：
代码量中等+多种复杂情况判断+返回结果简单
- 本地测试结果：
enroll1: 1000/1000
enroll2: 924/1000

Step 3.3 enrollCourse接口总体实现文件



```
m enrollCourse(int, int)
m getCourseTable(int, Date)
m getEnrolledCoursesAndGrades(int, Integer?)
m getInvalidResult(int, int)
m getStudentMajor(int)
m hasAlreadyEnrolled(int, int)
m hasAlreadyPassed(int, int)
m hasCourseConflictFound(int, int)
m hasPrerequisiteFulfilled(int, int)
m isCourseFound(int, int)
m isCourseFull(int, int)
```

- 处理类：

reference.service.

ReferenceStudentService

- 代码量：

- 8 methods, 282lines

- 其中getInvalidResult方法：
优化效率，大大缩短运行时间
(使得2000+s缩短为1s左右)

Step 3.4 enrollCourse接口实现介绍

```
try(Connection conn=SQLDataSource.getInstance().getSQLConnection()){
    if(!isCourseFound(studentId,sectionId)){
        return EnrollResult.COURSE_NOT_FOUND;
    }

    if(hasAlreadyEnrolled(studentId,sectionId)){
        return EnrollResult.ALREADY_ENROLLED;
    }

    if(hasAlreadyPassed(studentId,sectionId)){
        return EnrollResult.ALREADY_PASSED;
    }

    if(!hasPrerequisiteFulfilled(studentId,sectionId)){
        return EnrollResult.PREREQUISITES_NOT_FULFILLED;
    }
}
```

Test enroll course 1: 1000

Test enroll course 1 time: 3113.54s

- Problem1: 如何返回对应要求的结果？
- Solution: 根据传入的参数列表，做逐一的检查
- Difficulty:
 - 1.部分非法情况的判定的sql语句较为复杂，容易出现错误
 - 2.速度非常慢（第一版对于1000条数据的运行时间在3000s以上）

Step 3.4 enrollCourse接口实现介绍

- 优化前：

```
Test enroll course 1: 1000
```

```
Test enroll course 1 time: 3113.54s
```

- 优化后：

```
Test enroll course 1: 1000
```

```
Test enroll course 1 time: 0.94s
```

```
EnrollResult er1=getInvalidResult(studentId,sectionId);  
if(er1!=null){  
    return er1;  
}
```

- Problem2: enrollCourse的运行速度非常慢

- Solution:

- 1.增加数据库列索引提升查询速度；
- 2.合并多种情况的判断的sql语句，大幅减少Java与数据库的通讯次数；
- 3.关闭自动提交，将多条语句同时批量化执行

Step 4.

Additional Works

By YeeTone WANG and Tong ZHANG

Step 4.1 提升效率

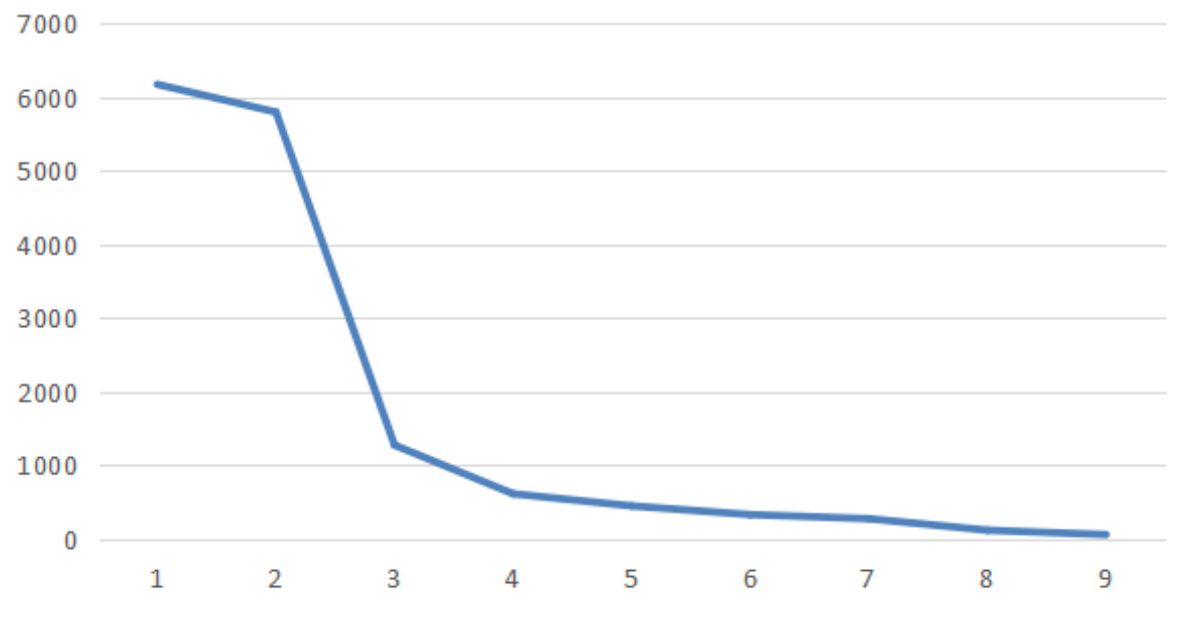
```
Test search course 1: 1000
Test search course 1 time: 237.00s
Test enroll course 1: 1000
Test enroll course 1 time: 3214.07s
Test drop enrolled course 1: 797
Test drop enrolled course 1 time: 3.05s
Import student courses
Import student courses time: 8.70s
Test drop course: 88423
Test drop course time: 2.27s
Test course table 2: 1000
Test course table 2 time: 3.40s
Test search course 2: 972
Test search course 2 time: 334.09s
Test enroll course 2: 924
Test enroll course 2 time: 2367.40s
```

- 最初版本实现方法：
多次连接+暴力查询+暴力筛选
- 单次测试用时：
6000+s , 约为1.5h
- Not Acceptable !



Step 4.1 提升效率

优化次数-benchmark运行时间折线图



- 操作类型：运行全部benchmark
- 数据规模：不多于10W
- 数据来源：本地测试的结果中的运行时间
- 最初版本运行时长：6170.43s
- 最终版本运行时长：23.95s
- 效率提升百分比：

25662%



Step 4.1 提升效率

优化次数	优化策略	运行时间(s)	效率提升百分比
0	无	6170.43	0%
1	将Statement更换为PreparedStatement	5794.37	+6.5%
2	合并查询语句，减少通讯次数	776.62	+646%
3	合并插入/修改语句，批量执行	317.11	+144%
4	增加数据库索引列	252.66	+25.8%
5	调整sql语句逻辑	174.92	+44.8%
6	将触发器更改为约束	129.63	+34.9%
7	减少java与数据库传输中的结果返回数量	43.34	+297%
8	笔记本电脑连接电源	23.95	+80%

Step 4.1 提升效率

操作类型	优化前 运行时间	优化后 运行时间
数据导入	3.56	1.37
数据修改	10.75	7.32
数据访问	6152.43	15.26

	无缓存命中	有缓存命中
访问时间1	20ms+157ms =177ms	19ms+16ms= 35ms
访问时间2	14ms+43ms= 57ms	15ms+10ms= 25ms

• 结论：

- 1.减少Java与数据库的通讯次数和结果返回数量大大有利于运行的效率提升
- 2.SQL语句的运行逻辑对效率也有较大影响
- 3.数据库触发器对并发情况下的性能有较大损失
- 4.硬件设施的设备条件对查询时间也有重要影响，且随着运行时间增加，影响也会相应变大
- 5.数据库对于刚访问的行列及其附近元素，访问速度会明显加快（表现为execution基本不变，fetch时间有所减少）

（与project1结论保持一致）

Step 4.2 资源消耗

	table_name	table_size
1	student100course	4920 kB
2	students	1560 kB
3	coursesectionclass	344 kB
4	instructor	176 kB
5	studenttpfcourse	136 kB
6	coursesection	64 kB
7	studentcourseselection	40 kB
8	course	16 kB
9	major	8192 bytes
10	department	8192 bytes
11	majorcourse	8192 bytes
12	semester	8192 bytes
13	prerequisite	8192 bytes

	pg_size_pretty
1	41 MB

- 监测数据库内存使用情况
- 数据库大小：41MB
- 各表大小之和：7296kb=7.125MB
- 数据库大小>>各表大小之和！

But Why?

Step 4.2 资源消耗

	indexrelname	pg_size_pretty
1	s100_uq	24 MB
2	s100c_index	24 MB
3	students_pkey	624 kB
4	instructor_pkey	16 kB
5	spf_uq	288 kB
6	spf_index	288 kB
7	coursesection_pkey	56 kB
8	student_course_selection_index	208 kB
9	studentcourseselection_index2	208 kB
10	course_pkey	16 kB
11	department_departmentname_uq	32 kB
12	major_majorname_uq	32 kB
13	prerequisite_pkey	16 kB
14	pk	16 kB
15	major_pkey	32 kB
16	department_pkey	32 kB
17	semester_pkey	16 kB

- 查询数据库索引大小

- 数据库索引大小之和：
51032kB=49.8MB>41MB

- 结论：

- 1.数据库数据索引占用内存高于数据本身所占用的内存

- 2.数据库索引内存之和有可能大于数据库本身占用的内存

Step4.3 多线程-高并发

```
11910112 1 0  
SUCCESS  
11910104 1 1  
SUCCESS  
11911831 1 2  
SUCCESS  
11911832 1 3  
SUCCESS  
11911833 1 4  
COURSE_IS_FULL  
11911834 1 5  
COURSE_IS_FULL  
11910112 1 6  
ALREADY_ENROLLED  
11910104 1 7  
ALREADY_ENROLLED  
11911831 1 8  
ALREADY_ENROLLED
```

一、实现方式：同时采用**1000**个线程调用**Enroll**方法，继承**Runnable**接口，重写**run（）**方法，其中该方法在**start（）**方法中被调用。

二、结果：符合预期，测试样例中只有**section1**，容量为**4**，结果只有四条数据插入**SUCCESS**其他都失败

Step4.3 多线程-高并发

去除Join()方法后的结果

```
11911832 1 3
SUCCESS
11910104 1 1
SUCCESS
11910112 1 6
SUCCESS
11911832 1 33
ALREADY_ENROLLED
11911834 1 5
SUCCESS
java.sql.SQLTransientConnectionException Create breakpoint : HikariPool-1 - Connection is not available, request timed out after 30008ms.
    at com.zaxxer.hikari.pool.HikariPool.createTimeoutException(HikariPool.java:696)
    at com.zaxxer.hikari.pool.HikariPool.getConnection(HikariPool.java:197)
    at com.zaxxer.hikari.pool.HikariPool.getConnection(HikariPool.java:162)
    at com.zaxxer.hikari.HikariDataSource.getConnection(HikariDataSource.java:128)
    at cn.edu.sustech.cs307.database.SQLDataSource.getSQLConnection(SQLDataSource.java:34)
    at reference.service.ReferenceStudentService.enrollCourse(ReferenceStudentService.java:66)
    at reference.test.ConcurrencyTest$MyPostgreSQLThread.run(ConcurrencyTest.java:79) <1 internal line>
```

Step4.3 线程安全

- synchronized 控制线程同步

```
private synchronized void adjustSequenceVal(Connection conn){...}
```

- Lock 可操作性

```
private void lockTest1 (Thread thread) {  
    lock.lock();  
    try {  
        System.out.println("Index " + thread.getName() + " onLock");  
    } catch (Exception e) {  
        e.printStackTrace();  
    } finally {  
        System.out.println("Index " + thread.getName() + " offLock");  
        lock.unlock();  
    }  
}
```

Index t1 onLock

Index t1 offLock

Index t2 onLock

Index t2 offLock

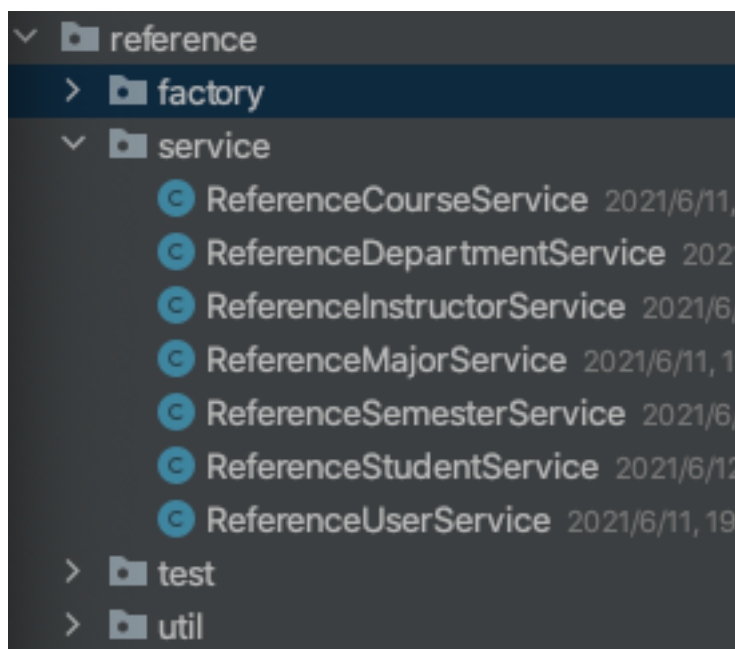
```
private void lockTest2(Thread thread) {  
    if (lock.tryLock()) {  
        try {  
            System.out.println("Index " + thread.getName() + " onLock");  
        } catch (Exception e) {  
            e.printStackTrace();  
        } finally {  
            System.out.println("Index " + thread.getName() + " offLock");  
            lock.unlock();  
        }  
    }  
}
```

Index t1 onLock

Index t1 offLock

Step4.4 Correctness

- 接口实现：



- 测试

- CourseServiceTest
- StudentServiceTest
- DepartmentServiceTest
- MajorServiceTest
- InstructorServiceTest
- SemesterServiceTest

- 其他

- DatabaseClearUtil
- PrerequisiteUtil
- ListAndSetUtil
- SearchCourseUtil
- CourseSearchEntryProperty

代码总量2000余行
满足ACID

Step5 Conclusion

- 接口的实现与样例测试
- Java与数据库通讯效率的提升
- 数据库高并发访问
- 多线程中锁的使用
- 线程安全分析及解决方案
- 事务管理ACID原则
- 使用索引的优劣及内存分析
- 数据库系统效率与不同硬件系统的关系
- 项目的迭代思维与对比Project1->Project2
- 面对bug和报错信息冷静耐心的处理
- 团队之间的共同协作与指导
-



对于深入理解整合CS307课程所学，乃至未来在CS专业的学习
都是大有裨益！