**武汉大学计算机学院**

**本科生实验报告**

**数据库系统实现**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：数据库系统实现

团 队 名 称 ：X X X

指 导 教 师 ：王胜

团 队 成 员 一：陈胤良

团 队 成 员 二：杜忠璠

团 队 成 员 三：孔德昱

团 队 成 员 四：王新喆

团 队 成 员 五：王乔林

团 队 成 员 六：胡瑞

二○二三年五月

**郑 重 声 明**

本团队呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

团队成员签名： 日期：

摘 要

本实验的实验目的是基于武汉大学自研的TOTEM数据库和高德地图API在安卓操作系统上实现记录运动轨迹数据并可视化，从而加深学生对数据库系统实现层面的理解以及动手能力。

本实验使用Windows 10作为开发环境，Android Studio作为开发工具，JAVA作为开发语言，Android操作系统作为部署平台。

实验的主要内容包括三个任务。任务一要求设计系统表，实现系统表的读取和保存；任务二要求实现SQL解析中的对象JOIN操作；任务三要求实现更新迁移、跨类查询并实现轨迹数据的JOIN和可视化。

本实验的结论是

本实验的源代码发布在**<https://github.com/cylqqqcyl/whu-tmdb>**，将在实验周期结束后公开。

**关键词：**TOTEMDB；数据库系统实现；JOIN

**目 录**

**1** **实验目的和意义**

1.1 实验目的 ………………………………………………………………………1

1.2 实验意义………………………………………………1

(各章的名称黑体4号，其余宋体小4)

**……**

**……**

**……**

**2** **实验设计**

2.1 概述……………………………………………………………………………… 35

2.2 实验原理………………………………………………………………………… 37

2.3 实验方案………………………………………………………………………… 39

**……**

**……**

**……**

**结论** …………………………………………………………………………………… 57

**参考文献 ………………………………………………………………………………** 59

**1 实验目的和意义**

**1.1** **实验目的**

本实验的目的是通过实践操作，让学生深入理解数据库系统的基本概念和原理，掌握数据库系统的设计、实现、管理和维护技术，培养学生的数据库应用开发能力和解决实际问题的能力。同时，通过实验，学生还能了解数据库系统的应用领域和未来发展趋势。

**1.2实验意义**

该实验是理论知识和动手能力的综合体现。通过本实验，学生可以更加深入地理解数据库系统的基本概念和原理和培养实际操作能力。同时，使用github平台多人协同开发有助于培养学生的团队协作能力。

**2 实验设计**

**2.1 概述**

本实验实现了在Android操作系统上基于TOTEM数据库和高德地图API记录并可视化用户行动轨迹的应用功能开发工具为Android Studio，编程语言为JAVA，开发环境为Windows 10操作系统，应用的部署环境为Android操作系统。

实验步骤主要包括：

1. 设计系统表，实现系统表的读取和保存；
2. 实现SQL解析中的对象JOIN操作；
3. 实现轨迹数据的JOIN和可视化；

**2.2 实验原理**

**2.2.1 TOTEM数据库**

为了管理结构复杂、语义丰富的大数据，武汉大学珞珈图腾数据库实验室在国家863数据库重大专项课题支持下研制了对象代理数据库系统TOTEM[1]。TOTEM数据库基于我们提出的具有原创性的对象代理模型。该模型既具有关系数据模型的柔软性又具有面向对象数据模型表现复杂语义能力，发表在了国际数据库顶级会议IEEE International Conference on Data Engineering 和权威期刊 IEEE Transaction on Knowledge and Data Engineering 上得到学术界认可和引用。相比目前世界上普遍采用的对象关系数据库管理系统，对象代理数据库系统能够提供灵活对象视图、动态分类、跨类查询等先进功能，形成了自主知识产权，获教育部高等学校科技进步二等奖。它是大数据与人工智能时代的应需之作，可广泛应用于非结构、跨媒体、多模态数据管理与分析。

**2.2.2 Android Studio**

Android Studio[2][3]是谷歌Android操作系统的官方集成开发环境（以下简称IDE），基于JetBrains的IntelliJ IDEA软件开发，并专门为Android开发而设计。它可以在Windows、MacOS和Linux操作系统上下载使用，是Eclipse Android开发工具（E-ADT）的替代品，成为本地Android应用程序开发的主要IDE。

Android Studio于2013年5月16日在Google I/O大会上宣布，并从2013年5月的版本0.1开始进入早期预览阶段，随后从2014年6月的版本0.8开始进入Beta阶段。第一个稳定版本从2014年12月的版本1.0开始发布。到2015年底，谷歌停止支持Eclipse ADT，使得Android Studio成为Android开发的唯一官方支持的IDE。

**2.2.3 高德地图API**

高德地图JS API[4]是一套JavaScript语言开发的的地图应用编程接口，移动端、PC端一体化设计，一套 API 兼容众多系统平台。若以非商业目的使用JS API，则可以免费使用。若以商业目的使用，则需事先从高德获取商用授权。

JS API提供了2D、3D地图模式，满足绝大多数开发者对地图展示、地图自定义、图层加载、点标记添加、矢量图形绘制的需求，同时也提供了 POI搜索、路线规划、地理编码、行政区查询、定位等众多开放服务接口。

**2.2.3 JOIN操作**

在SQL中，JOIN语句将一个或多个表的列组合成一个新表。该操作对应于关系代数中的联接操作，即联接将两个表拼接在一起，并将具有匹配字段的记录放在同一行上。JOIN操作包括 INNER、LEFT OUTER、RIGHT OUTER、FULL OUTER和CROSS。

本实验的应用场景是自动驾驶行车轨迹数据和手机轨迹数据用了两张表分别存储，导致了大量的数据冗余。通过对两张表施加JOIN操作，可以消除大量重复数据，便于对数据进行分析和挖掘以及提高数据的准确性和可靠性。

**2.3 实验方案**

本实验主要包括设计系统表、 实现SQL解析中的对象JOIN操作和实现轨迹数据的join和可视化三个任务。

**2.3.1 设计系统表**

系统表（system table）是存储数据库系统元数据（metadata）的表。元数据是描述数据库中各种对象的数据，如表、列、索引、视图、存储过程等。系统表存储这些元数据信息，包括对象的名称、类型、所有者、创建时间、修改时间、大小等信息。其作用包括支持数据库系统的运行和管理，以及提供元数据信息供开发人员和系统管理员使用。

**2.3.2 实现SQL解析中的对象JOIN操作**

针对本实验的应用场景，基于已有的SELECT语句可以实现与JOIN等价的功能。

主要思想是先将多个表做笛卡尔积，然后在需要连接的属性上用WHERE语句判断是否相等已选出目标元组。例如，要将table1和table2分别在column1和column2上连接。SQL语句的实现代码如下所示。

1. **SELECT** \* **FROM** table1, table2 **WHERE** table1.**column** = table2.**column**;

本实验代码中，在Tuple.java文件中给出了元组的定义。每个元组的包含了三个属性，即trajectory\_id、user\_id和trajectory。其中，trajectory是一个字符串，表示相邻两点的经纬度信息，格式为“经度1-维度1-经度2-维度2”。

在TJoinSelect.java文件中给出了TJoinSelect类的实现，关键代码如下图。

1. **public** **class** TJoinSelect extends SelectImpl{

4. **private** MemConnect memConnect;
6. **public** TJoinSelect(MemConnect memConnect) {
7. super(memConnect);
8. **this**.memConnect = memConnect;
9. }
11. **public** TJoinSelect() {
13. }
15. //TODO TMDB
16. //重写select的intersect方法，使其使用trajectory similarity join 进行连接
17. @Override
18. **public** SelectResult intersect(SelectResult left, SelectResult right){
19. LongestCommonSubSequence longestCommonSubSequence=**new** LongestCommonSubSequence();
20. //新建tuplelist，存储两表intersect之后的结果
21. TupleList res=**new** TupleList();
22. //遍历左表的tuple
23. **for** (**int** i = 0; i < left.getTpl().tuplelist.size(); i++) {
24. //获取当前tuple
26. //调用TrajTrans的getTraj方法，将tuple中的String轨迹转换成List<Coordinate>的形式，得到traj1
27. Tuple tuple = left.getTpl().tuplelist.get(i);
28. List<Coordinate> leftTraj = TrajTrans.getTraj((String) tuple.tuple[2]);
29. //遍历右表的每个tuple
30. **for** (**int** j = 0; j < right.getTpl().tuplelist.size(); j++) {
31. //获取当前右表tuple
32. //并通过TrajTrans的getTraj方法得到List<Coordinate>，得到traj2
33. Tuple rightTuple = right.getTpl().tuplelist.get(j);
34. List<Coordinate> rightTraj = TrajTrans.getTraj((String) rightTuple.tuple[2]);
35. //通过longestCommonSubSequence的getCommonSubsequence方法得到traj1和traj2的公共子序列，theta值自设
36. List<Coordinate> commonSubsequence = longestCommonSubSequence.getCommonSubsequence(leftTraj, rightTraj, 3);
37. //通过得到的子序列的长度设置阈值，判定当前子序列是否值得加入结果集合中
38. **if**(commonSubsequence.size()>=1){
39. //如果满足，则新建加入到结果结合中的tuple
40. //此tuple其它的部分与左表的当前tuple全部一致，除了轨迹段改为公共子序列
41. Tuple temp=**new** Tuple();
42. temp.tupleId=tuple.tupleId;
43. temp.tupleIds=tuple.tupleIds;
44. temp.tuple=tuple.tuple;
45. //需要将得到的轨迹子序列，转换成string的形式，然后将tuple中轨迹部分设置为转换后的值
46. String temps=TrajTrans.getString(commonSubsequence);
47. temp.tuple[2]=temps;
48. //在新建的tuplelist中加入当前tuple
49. res.tuplelist.add(temp);
50. **break**;
51. }
52. }
53. }
54. //将左表的selectResult 也就是left的tuplelist设置为新的结果集
55. left.setTpl(res);
56. //返回新的selectrResult
57. **return** left;
58. }
59. }

通过重写intersect函数可以实现trajectory之间的连接操作。遍历左表和右表，对于左表中的元组leftTuple和右表中的元组rightTuple，首先使用TrajTrans.getTraj()方法将元组中的trajectory转换成坐标格式；其次调用getCommonSubsequence()方法获取两个trajectory的最长公共子串。其作用是衡量两个trajectory之间的相似度。对于最长公共字串长度达到阈值，即轨迹trajectory足够相似的两个元组，将其加入到连接的结果之中。遍历完左表和右表后就可以得到最终的连接结果。

一个使用该连接方法的SQL语句如下所示。该语句将watch\_traj表和mobile\_traj通过intersect的方式连接成tj表。

1. **create** tjoindeputy tj **as** **select** \* **from** watch\_traj **intersect** **select** \* **from** mobile\_phone\_traj;

2.3.3 实现轨迹数据的JOIN和可视化

6条路线需要将mod改为6。

调用高德地图API。

**结论**

**参考文献**

[1] 珞珈图腾数据库实验室. TOTEM数据[DB/OL]. <http://totemdb.whu.edu.cn/aspx/main/zhanshi_show.aspx?id=1>

[2] Android Studio开发者. 探索Android Studio[DB/OL]. <https://developer.android.google.cn/studio/intro?hl=zh-cn>

[3] Wikipedia. Android Studio[DB/OL]. <https://en.wikipedia.org/wiki/Android_Studio>

[4] 高德开放平台. 地图 JS API[DB/OL]. <https://lbs.amap.com/api/javascript-api/summary/>

|  |
| --- |
| 【结论】： |
| 【小结】： |
| 指导老师评语及成绩 |
| 【评语】：  成 绩：  指导老师签名：  批阅日期：2022.8.20 |