**实验2：空间数据库设计**

一、系统介绍

KPL信息管理系统是一款专门为王者荣耀联赛管理而设计的系统。该系统旨在实现对选手、战队、教练、英雄、经理、联赛等相关信息的全面管理和统计分析。

1.需求分析：

(1)数据流

数据输入：该系统需要能对选手、战队、教练、英雄、经理、联赛等相关信息输入。

数据存储：输入数据需存储在oracle数据库中，以便后续的管理和查询。

数据处理：系统需能够处理各种关联数据、逻辑操作，例如选手加入战队、战队参加联赛等。

数据输出：系统需能够根据用户需求输出相关数据，例如选手信息、战队战绩、联赛积分等。

(2)系统功能

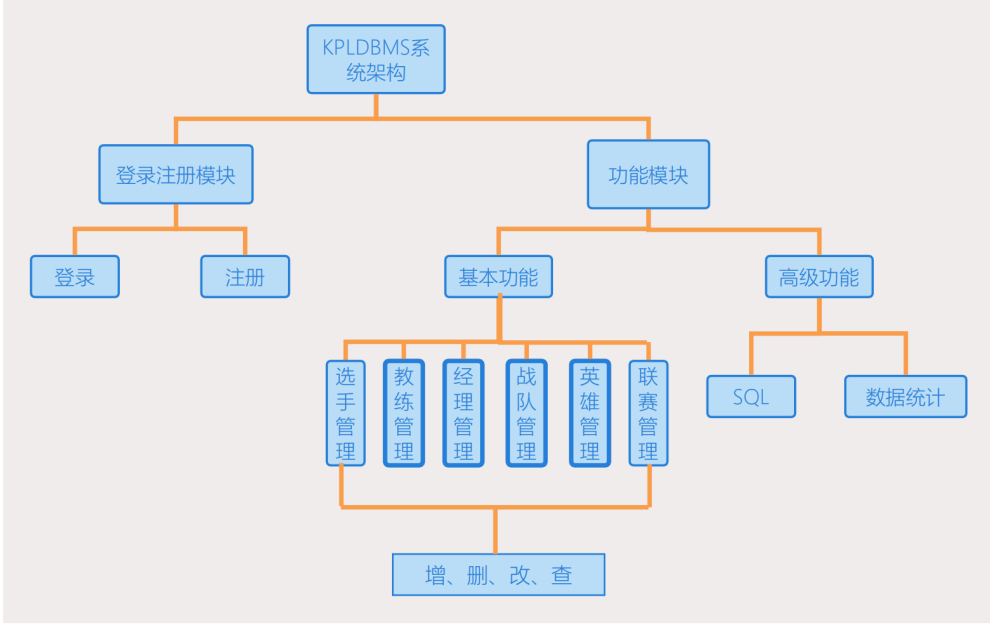
管理功能：系统需要实现对选手、战队、教练、英雄、经理、联赛信息的管理功能，包括增加、删除、修改和查询。

关联功能：系统需要能够建立和维护选手与英雄、战队与联赛、战队与选手等之间的联系。

查询功能：系统提供sql语句查询功能，以便用户能够根据不同条件查询所需的信息。

统计功能：系统需要能够根据用户需求进行数据统计和分析，例如战队战绩统计、选手擅长英雄分析等。

综上所述，这些需求将指导系统的设计和开发，以确保系统能够满足管理员对电竞联赛信息管理的各项需求。



二、数据库设计

1.概念设计

(1)实体和属性

选手Player: id、姓名、年龄、性别、主玩位置、工资；

战队Team：id、队名、主场城市、战队积分；

队长Captain: id、姓名、年龄、性别、主玩位置、工资；

教练Coach：id、姓名、年龄、性别、工资；

英雄Hero：id、英雄名、角色；

经理Manager：id、姓名、年龄、性别；

联赛League：id、联赛名、级别、积分规则；

(2)联系

一对一联系：教练与战队，队长与战队，经理与战队；

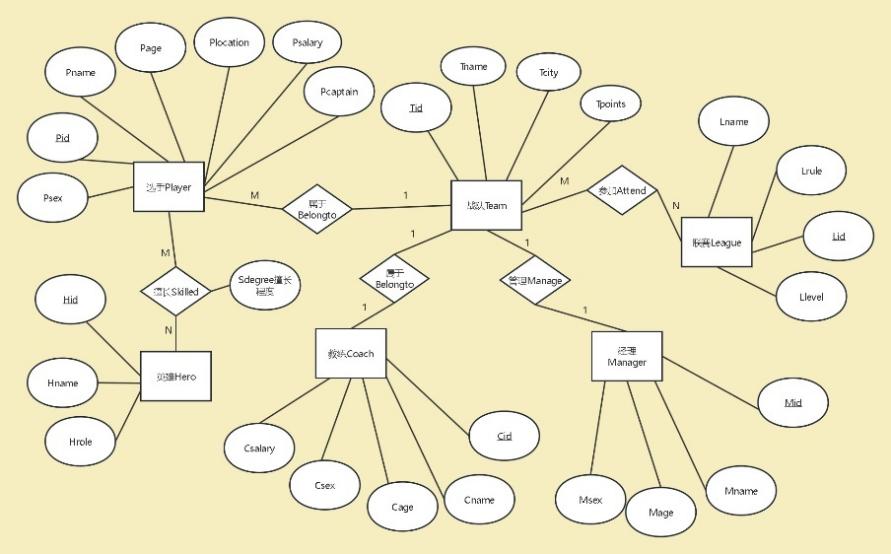
一对多联系：战队与选手；

多对多联系：选手与英雄，战队与联赛；

(3)全局ER图

分析可知，选手实体与队长实体存在语言冲突，将二者合并为如下：

选手Player: id、姓名、年龄、性别、主玩位置、工资、队长？；建立全局ER图如下。



二、逻辑设计

1.概念模型转为逻辑模型

(1)实体转为关系模式

Player (Pid#,Pname,Page,Psex,Plocation,Psalary,Pcaptain)

Hero(Hid#,Hname,Hrole)

Team(Tid#,Tname,Tcity,Tpoints)

Coach(Cid#,Cname,Cage,Csex,Csalary)

Manager(Mid#,Mname,Mage,Msex)

League(Lid#,Lname,Llevel,Lrule)

(2)联系转为关系模式

1）战队与选手间属于联系为一对多，将其转为一个独立关系模式PT(Pid#,Tid)；

2）战队与教练、经理所属联系都为一对一，为减少连接操作，将其与战队关系模式合并，重新得到关系模式，(Tid#,Tname,Tcity,Tpoints,Cid,Mid) 。

3) 选手与英雄之间Skilled联系，战队与联赛之间Attend联系为多对多，将其转换为一个单独的关系模式。Skilled (**Pid#,**Hid#,Sdegree)、Attend(Tid#,Lid#)

2.逻辑模型优化

关系数据模型的优化通常以规范化理论为指导，对每个关系模式优化到第三范式。

第一步，确定关系模式R的所有属性都是不可分的基本数据项，达到1NF。将非规范化表格转换为规范化表格，属性字段间相互独立。分析每个关系模式，可知满足1NF。

第二步，确定关系模式R所有非主属性都完全函数依赖于R的码，达到2NF。消除部分函数依赖，改进增删异常冗余的问题。在选手关系中，Psalary,Pcaptain还依赖于具体战队，不满足完全函数依赖，将其属性分解为(Pid#,Pname,Page,Psex,Plocation)和PT(Pid#,Tid,Psalary,Pcaptain)，使其满足2NF，其它关系同理对其进行优化，分解得到如下关系，CT(Cid#,Tid,Csalary)、Attend(Tid#,Lid#,Tpoints)。

第三步，消除传递函数依赖，达到3NF。分析以上每个关系模式，每个关系模式内非主属性都依赖于主码，不存在传递函数依赖，满足3NF。

经过规范化理论优化，得到最终如下十个关系模式：

Player(**Pid#,**Pname,Page,Psex,Plocation)

PT(**Pid#,**Tid,Psalary,Pcaptain)

Hero(**Hid#,**Hname,Hrole)

Team(**Tid#,**Tname,Tcity,Mid)

Coach(**Cid#,**Cname,Cage,Csex)

CT(**Cid#,**Tid,Csalary)

Manager(**Mid#,**Mname,Mage,Msex)

League(**Lid#,**Lname,Llevel,Lrule)

Skilled(**Pid#,**Hid#,Sdegree)

Attend(**Tid#,**Lid#,Tpoints)

(3)数据表设计

Player表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Pid | CHAR(5) | 是 | 是 | 选手id |  |
| Pname | VARCHAR(6) | 否 | 是 | 选手姓名 |  |
| Page | INT | 否 | 否 | 年龄 |  |
| Psex | CHAR(2) | 否 | 否 | 性别 |  |
| Plocation | VARCHAR(6) | 否 | 否 | 主玩位置 |  |

PT表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Pid | CHAR(5) | 是 | 是 | 选手id |  |
| Tid | CHAR(5) | 否 | 是 | 战队id |  |
| Psalary | INT | 否 | 否 | 工资 |  |
| Pcaptain | INT | 否 | 否 | 是否队长 | 值为0或1 |

Hero表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Hid | CHAR(5) | 是 | 是 | 英雄id |  |
| Hname | VARCHAR(8) | 否 | 是 | 英雄名 |  |
| Hrole | CHAR(4) | 否 | 否 | 角色 |  |

Team表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Tid | CHAR(5) | 是 | 是 | 战队id |  |
| Tname | VARCHAR(12) | 否 | 是 | 战队名 |  |
| Tcity | CHAR(4) | 否 | 否 | 主场城市 |  |
| Mid | CHAR(5) | 否 | 否 | 经理 |  |

CT表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Cid | CHAR(5) | 是 | 是 | 教练id |  |
| Tid | CHAR(5) | 否 | 是 | 战队id |  |
| Csalary | INT | 否 | 否 | 工资 |  |

Manager表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Mid | CHAR(5) | 是 | 是 | 经理id |  |
| Mname | VARCHAR(6) | 否 | 是 | 姓名 |  |
| Mage | INT | 否 | 否 | 年龄 |  |
| Msex | CHAR(2) | 否 | 否 | 性别 |  |

League表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Lid | CHAR(5) | 是 | 是 | 联赛id |  |
| Lname | VARCHAR(12) | 否 | 是 | 联赛名称 |  |
| Llevel | CHAR(4) | 否 | 否 | 联赛级别 |  |
| Lrule | VARCHAR(20) | 否 | 否 | 积分规则 |  |

Skilled表

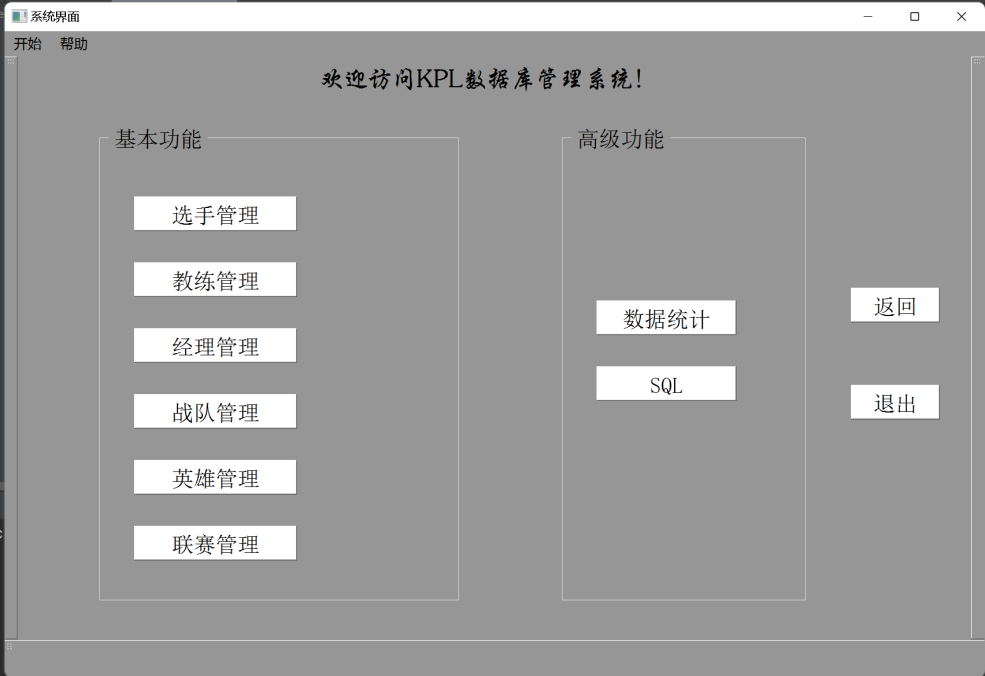
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Pid | CHAR(5) | 是 | 是 | 选手id |  |
| Hid | CHAR(5) | 是 | 是 | 英雄id |  |
| Sdegree | VARCHAR(8) | 否 | 否 | 擅长程度 |  |

Attend表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 主键？ | 非空？ | 备注 | 自定义完整性约束 |
| Tid | CHAR(5) | 是 | 是 | 战队id |  |
| Lid | CHAR(5) | 是 | 是 | 联赛id |  |
| Tpoints | INT | 否 | 否 | 联赛积分 |  |

三、关键功能及算法

（1）访问界面

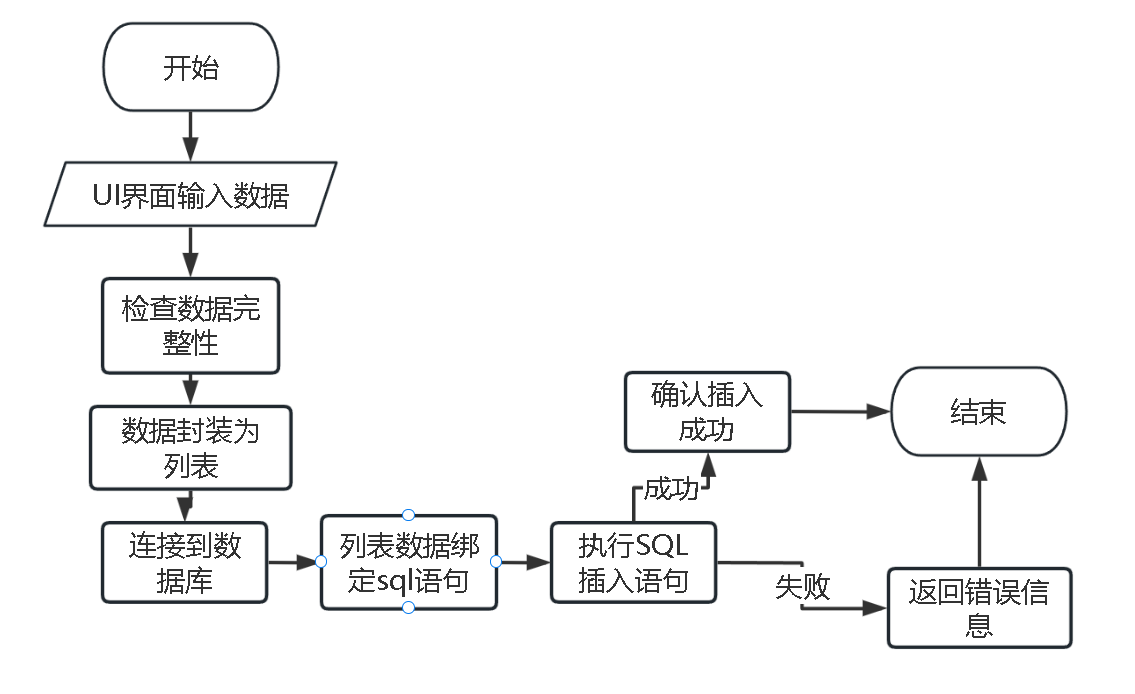


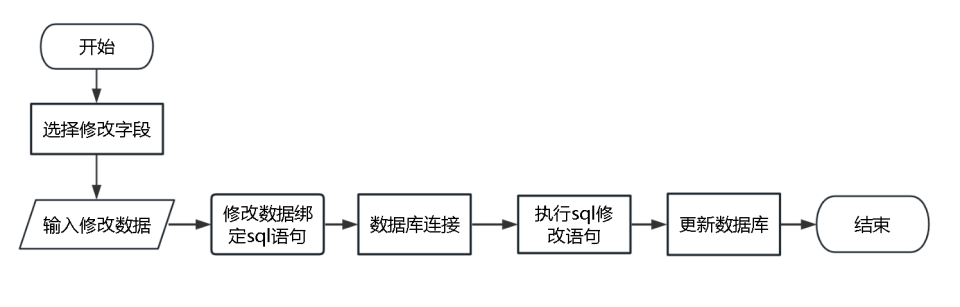
部分界面测试结果

（2）关键功能

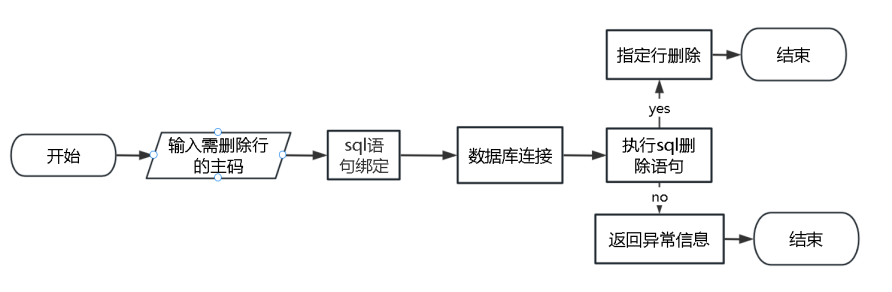
该系统包含基本功能与高级功能2大模块。基本功能模块每个子系统都具有插入、修改、删除、查找、排序功能。每个功能具体算法流程如下图。高级功能模块中有数据统计与sql操作2个功能，数据统计中可以对一些基本表进行统计；sql功能中，用户可直接通过sql命令来操作数据库，实现复杂查询、创建索引、视图等操作。关键功能采用方案1实现。



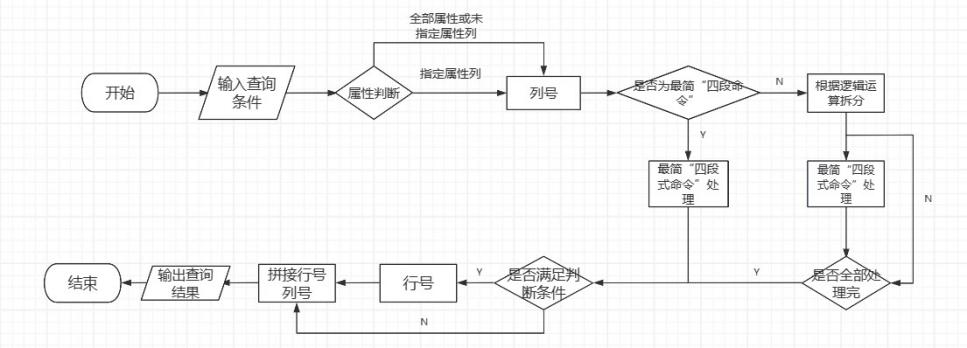
**插入数据流程图**



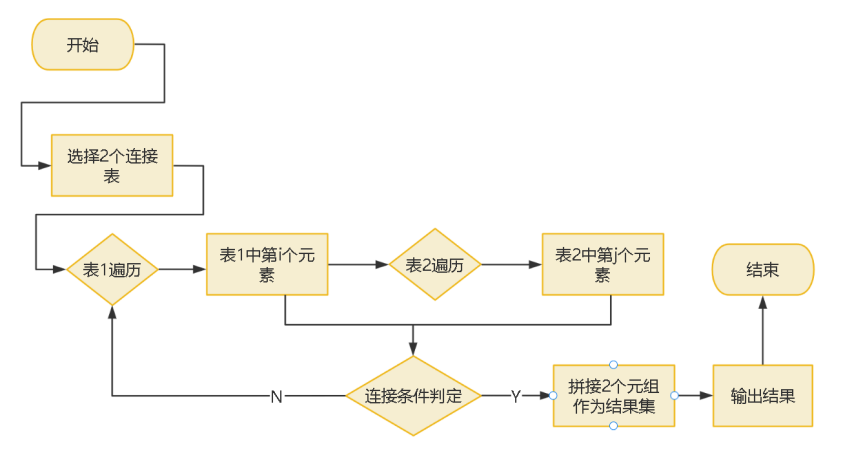
**修改数据流程图**

****

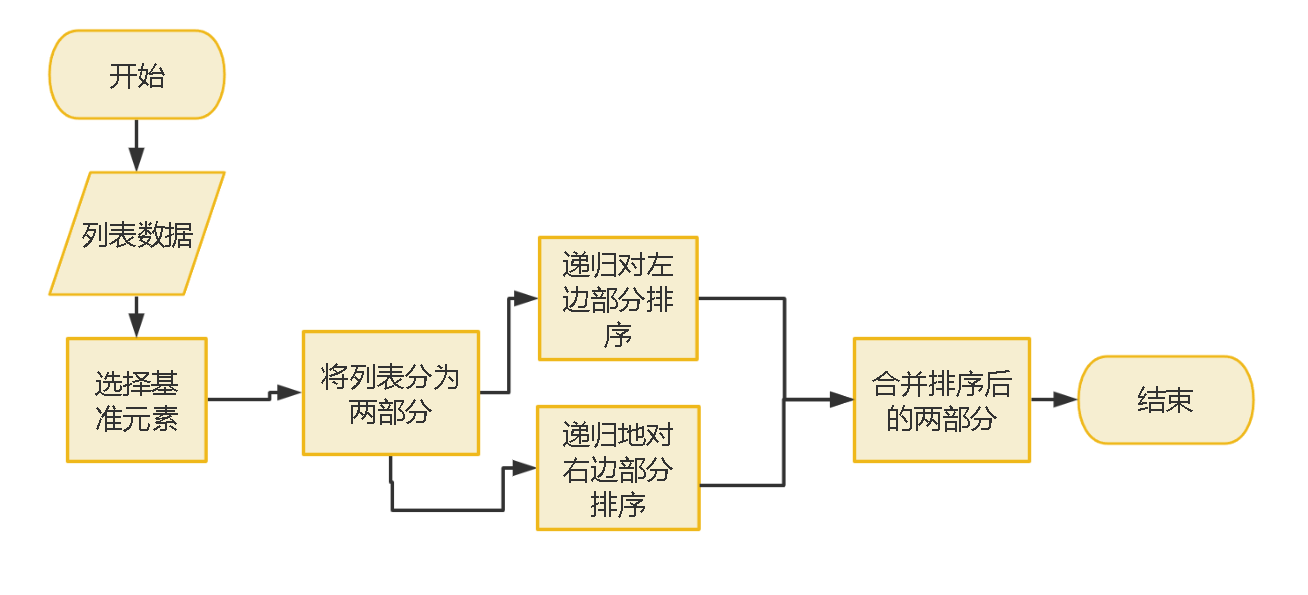
**删除数据流程图**

****

**单表查询流程图**

****

**多表连接查询流程图**

****

**快速排序算法流程图**

**（3）**算法介绍

快速排序（Quick Sort）是一种基于分治思想的排序算法。平均情况下具有很高的性能，时间复杂度为O(n log n)。本系统中，对于多表连接查询得到结果集，依据选择字段，采用快速排序进行排序。

四、总结

通过本次实验，我对数据库设计和实现有了更深入的了解。我掌握了数据库的概念设计、逻辑设计和物理设计的具体实现过程，以及数据库底层实现原理。我还成功地建立并维护了一个基于Oracle的KPL信息管理系统，实现了对空间数据库的设计和操作。

在整个实验过程中，我意识到了算法对于实现底层原理的重要性。在这个算法为王的时代，我认识到了学习算法的重要性，以提升自己的核心竞争力，避免被时代淘汰。因此，我计划在未来的学习中，多学习一些算法知识，并与地理信息专业知识相结合，提升自己的核心竞争力。