**XX大学计算机科学与工程学院**

**数据库原理 实验报告**

院 系 计算机科学与工程学院

专 业 计算机科学与技术

学 号

姓名

日期 2021年12月1 日

目录

[1. 业务场景 3](#_Toc91237701)

[2. 业务流图 3](#_Toc91237702)

[3. 数据流图 4](#_Toc91237703)

[4. ER图 5](#_Toc91237704)

[5. 数据模式 6](#_Toc91237705)

[6. 核心代码 6](#_Toc91237706)

[6.1 数据表的建立 7](#_Toc91237707)

[6.2 触发器设置 8](#_Toc91237708)

[6.3 查询语句 9](#_Toc91237709)

[7. 实验成效 10](#_Toc91237710)

[7.1 部门管理 10](#_Toc91237711)

[7.2 员工管理 12](#_Toc91237712)

[7.3 项目管理 13](#_Toc91237713)

[7.4 数据统计 14](#_Toc91237714)

[8. 实验总结 15](#_Toc91237715)

# 业务场景

选取的场景为公司管理系统，主要由以下部分构成：

1. 部门管理：支持增删改查部门信息，部门信息包括但不限于：编号、名称、办公地址、部门主管等。
2. 员工管理：支持增删改查员工信息，员工信息包括但不限于：编号、姓名、性别、电话、所属部门、参与项目等。
3. 项目管理：支持增删改查项目信息，项目信息包括但不限于：编号、项目简介、起始时间、截止时间、项目复杂人、参与员工等。
4. 数据统计：能统计员工参与项目情况。

# 业务流图

图示

描述已自动生成

图示

描述已自动生成

# 数据流图

图示

描述已自动生成

图示

描述已自动生成

图示

描述已自动生成

图示

描述已自动生成

# ER图

图示

描述已自动生成

一个员工只能隶属于一个部门，而一个部门可以有多个员工，因此部门和员工间是一对多关系。一个员工可以参与多个项目，一个项目也可有多个员工参与，因此员工和项目间是多对多关系。

# 数据模式

图示

描述已自动生成

由ER图可转化的数据模式如上图所示，各字段的含义以及外键关系已在图中注明。在本实验中业务场景下，我认为应该先有部门，再有员工，且员工不能没有工作部门。此外，有了项目才能添加负责人，因此允许部门主管bossno和项目负责人leaderno字段为NULL，但除了这两个字段的其余字段不允许为空。当部门被删除时，部门里的员工也随之被解雇，因此设置员工表的外键dno的on delete为cascade。当员工或者项目被删除时，项目参与记录也应随之被删除，因此项目参与记录表的外键pno和eno的on delete也设为cascade。此外，当员工被删除时，如果该员工是部门主管或者项目负责人，应该修改对应部门主管或项目负责人外键的值为NULL，因此设置项目表的外键leaderno和部门表的外键bossno的on delete为SET NULL。

# 核心代码

## 数据表的建立

|  |
| --- |
| 1. 员工表   CREATE TABLE if not exists employee  (eno INT PRIMARY KEY,  ename VARCHAR(50) NOT NULL,  gender VARCHAR(50) NOT NULL,  age INT NOT NULL,  phone VARCHAR(20) NOT NULL,  dno INT NOT NULL)   1. 部门表   CREATE TABLE if not exists department  (dno INT PRIMARY KEY,  dname VARCHAR(50) NOT NULL,  address VARCHAR(50) NOT NULL,  bossno INT DEFAULT NULL,  FOREIGN KEY(bossno) REFERENCES employee(eno)  ON DELETE SET NULL)   1. 项目表   CREATE TABLE if not exists project (  pno INT PRIMARY KEY,  dsc VARCHAR(200) DEFAULT '还没有添加描述',  stime DATE NOT NULL,  ftime DATE NOT NULL,  leaderno INT DEFAULT NULL,  FOREIGN KEY(leaderno) REFERENCES employee(eno)  ON DELETE SET NULL)   1. 项目参与表   CREATE TABLE if not exists pro\_emp (  pno INT NOT NULL,  eno INT NOT NULL,  PRIMARY KEY(eno, pno),  FOREIGN KEY(eno) REFERENCES employee(eno)  ON DELETE CASCADE,  FOREIGN KEY(pno) REFERENCES project(pno)  ON DELETE CASCADE)   1. 员工表外键约束补充定义（建立员工表时还没有部门表）   ALTER TABLE employee ADD CONSTRAINT fk\_dno FOREIGN KEY(dno) REFERENCES department(dno) ON DELETE CASCADE |

## 触发器设置

添加了两个触发器：

|  |
| --- |
| 1. 触发器1   create trigger pro\_emp\_delete  after delete on pro\_emp  for each row  begin  if (exists (select \* from project  where leaderno = OLD.eno))  then      update project      set leaderno = NULL      where pno = OLD.pno      and leaderno = OLD.eno;  end if;  end   1. 触发器2   create trigger employee\_department\_change  after update on employee  for each row  begin  if (OLD.dno != NEW.dno     and exists (select \* from department  where bossno = OLD.eno))  then       update department       set bossno = NULL       where bossno = OLD.eno;  end if;  end |

设置第一个触发器的理由：如果某个员工是项目负责人，在将这个员工的参与记录删除时，由于员工信息仍然在employee中，因此项目的负责人字段不会改变。但事实上，这个员工已经不再参与这个项目，故负责人应该设为NULL。

设置第二个触发器的理由：和设置第一个触发器的理由类似，如果某个员工是部门主管，那么更换这个员工工作部门后，这个员工没有被删除，但是部门主管的位置应该是空缺的，应设为NULL。

## 查询语句

仅展示实验中涉及的较复杂的sql查询语句。

|  |
| --- |
| 1. 查询筛选条件flt的部门的部门号，部门名称，办公地址，部门主管编号，部门主管名称。查询结果按部门编号升序排列。   SELECT department.dno, dname, address, bossno, ename  FROM department LEFT JOIN employee  ON department.bossno = employee.eno  where (department.dno REGEXP '${flt}'  or dname REGEXP '${flt}'  or address REGEXP '${flt}'  or ename REGEXP '${flt}')  order by dno asc   1. 查询参与编号为req.query.pno的项目的员工的编号，姓名，部门，部门名称，联系方式。查询结果按员工编号升序排列。   SELECT eno, ename, employee.dno, dname, phone  from employee, department  where employee.dno=department.dno  and eno in (      select eno      from pro\_emp      where pno = ${req.query.pno})  order by eno asc   1. 查询所有参与了项目的员工的参与项目数，员工编号，员工名称，排名。参与项目数多的排名靠前，参与项目数一样的排名相同。   select T.rank, T.eno, T.cnt, ename  from (select  count(\*) as 'rank', X.eno, X.cnt  from (select count(\*) as cnt, eno from pro\_emp group by eno) as X,  (select count(\*) as cnt, eno from pro\_emp group by eno) as Y  where X.cnt < Y.cnt or (X.cnt=Y.cnt and X.eno=Y.eno)  group by X.eno) as T, employee  where employee.eno = T.eno  order by T.cnt desc; |

# 实验成效

## 部门管理

图形用户界面, 应用程序, Teams

描述已自动生成

1. 添加部门：

图形用户界面, 应用程序, Teams

描述已自动生成

表格

中度可信度描述已自动生成

1. 修改部门信息

图形用户界面, 应用程序, Teams

描述已自动生成

图形用户界面, 应用程序, Teams

描述已自动生成

1. 删除部门

图形用户界面, 应用程序

描述已自动生成

图片包含 表格

描述已自动生成

1. 查找部门

图形用户界面, 应用程序

描述已自动生成

## 员工管理

图形用户界面, 应用程序, 表格, Teams

描述已自动生成

员工管理的增删改查和部门管理类似，不再一一展示。

## 项目管理

1. 查看参与员工

图形用户界面, 应用程序

描述已自动生成

表格

描述已自动生成

1. 设置负责人

图形用户界面, 应用程序, Teams

描述已自动生成

图形用户界面, 表格

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

## 数据统计

1. 参与项目数排行榜

图形用户界面, 应用程序

描述已自动生成

1. 参与项目详情

图形用户界面

描述已自动生成

电脑软件截图

描述已自动生成

对应到数据库中：

表格

低可信度描述已自动生成

可以看到编号为2的员工参与了项目3000、3005、3006；编号为4的员工参与了两个项目；其余员工参与了一个项目；没有参与项目的员工不在统计范围内。可见查询返回的结果是正确的。

# 实验总结

本次实验的收获有许多：

1. 加深了对数据库设计的理解，锻炼了需求分析能力，熟悉了业务流图、数据流图、ER图、数据模式的设计，拓展了理论课上没有提到的内容。
2. 学习了一整套WEB应用开发流程，熟悉如何使用React、Node.js、Express、MySQL 等软件，具备了实际设计与应用的能力。
3. 学习了前端（web）与后端数据库的连接和交互，能够初步开发出一个完 整的应用。

本次实验让我深刻意识到数据库设计的重要性，但是智者千虑，必有一失，在设计阶段我们很难保证自己设计的数据库是没有漏洞的；即便没有漏洞，投入实际应用后用户可能会提出新的需求，而这些需求可能会涉及到数据库的根本设计——这无疑是一个难解的问题。不仅数据库后端设计如此，前端设计也是如此，如何让自己的设计更具有鲁棒性、维护起来更方便简单、整体架构更加清晰明了，是值得思考的问题。