**实习题目**

### 1、需求分析

(1).背景：

项目管理系统，是反映公司员工资料，项目负责部门和项目进度，设备采购软件系统，是管理项目的有效工具。

面向用户：对项目进度跟踪，公司人员信息管理和设备管理的公司。只能由公司内部查看数据库中的数据。要求使用本系统的用户管理系统的使用有所了解，知道基本的操作和使用过程的注意事项。

(2).系统目标：

系统对外部封闭，不允许外部人员访问公司项目管理系统中的数据库。能够安全的访问系统独立的数据库。程序实现数据库数据的直观显示，保证数据库的能够为公司提供较为方便和基础的项目管理服务，能够为公司管理层提供监督渠道。

数据分析：员工的基本信息，部门的基本信息，项目的基本信息，设备基本信息

(3). 功能分析：

项目管理系统应该能够提供以下功能：管理员登陆、员工信息管理、部门信息管理、项目信息管理、设备信息管理。

1.管理员登录：项目管理系统采用Oracle数据库连接的服务名，用户名和口令(密码)验证模式，进入项目管理系统前项目管理员必须在登陆界面输入验证信息方可进入项目管理系统的主界面进行相应的操作。

2.学生信息管理：学生信息管理包含五个模块：学生信息的浏览、添加、删除、查询、修改。

部门信息管理：包含一个模块：部门经理信息查询。

项目信息管理：包含两个模块：根据项目经理的姓名查询项目的完成情况、查询某个项目的设备购买情况。

设备信息管理：包含三个模块：设备费用查询、设备供应商查询、添加设备。

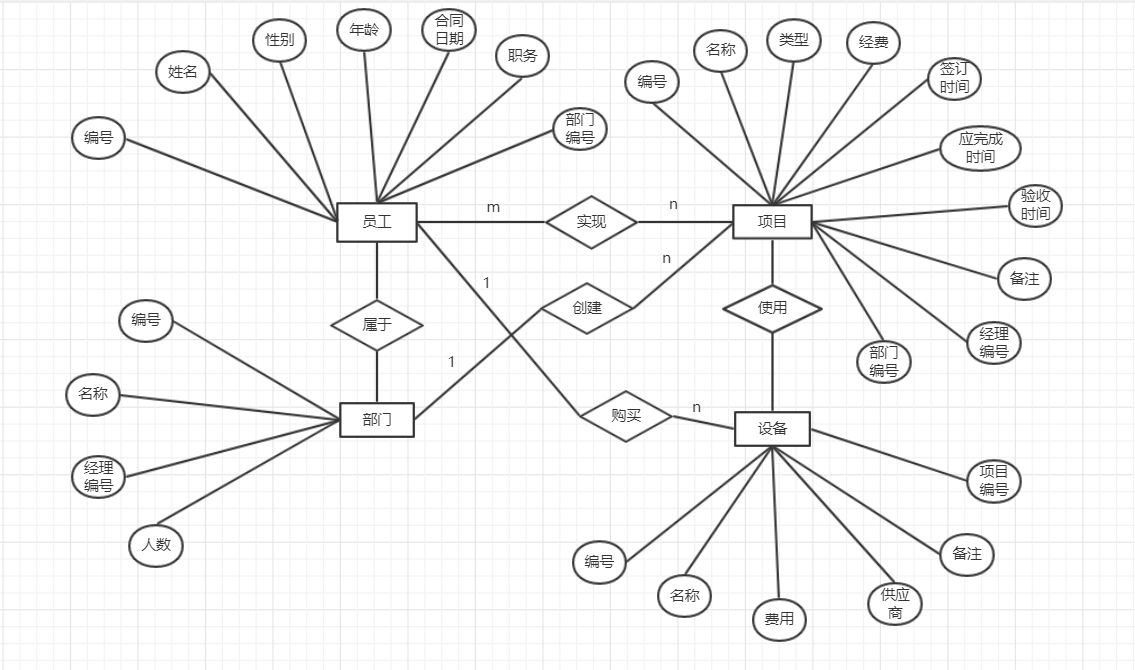
(4).性能需求分析：

系统易操作性：项目管理系统应该做到操作简单，界面友好，使得用户可以快速上手使用，不受到专业知识的限制。

系统可维护性：由于系统涉及的信息比较多，数据库中的数据需定期修改，系统可利用的空间及性能也随之下降，为了使系统更好地运转，用户可以对系统数据及一些简单的功能进行独立的维护及调整。

### 2、概念设计

（1）**概念模型（E-R图）**：



（2）**数据字典**：

1. 数据字典：

数据字典包括的项目有数据项、数据结构、数据流、数据存储、加工逻辑和外部实体。可使用一些符号来表示数据结构、数据流和数据存储的组成。

1）.员工表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| w\_id | VARCHAR2 | 4 | 员工编号 |
| w\_name | VARCHAR2 | 16 | 员工姓名 |
| sex | CHAR | 2 | 员工性别 |
| age | CHAR | 20 | 员工年龄 |
| contract\_date | DATE |  | 合同日期 |
| d\_id | VARCHAR2 | 4 | 部门编号 |
| d\_name | VARCHAR2 | 12 | 部门名称 |
| post | VARCHAR2 | 6 | 员工职务 |

2）.部门表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| d\_id | VARCHAR2 | 4 | 部门编号 |
| d\_name | VARCHAR2 | 12 | 部门名称 |
| m\_id | VARCHAR2 | 4 | 部门经理编号 |
| m\_name | VARCHAR2 | 8 | 部门经理姓名 |
| w\_num | NUMBER |  | 员工人数 |

3）.项目表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| p\_id | VARCHAR2 | 4 | 项目编号 |
| p\_name | VARCHAR2 | 16 | 项目名称 |
| d\_id | VARCHAR2 | 4 | 部门名称 |
| w\_id | VARCHAR2 | 4 | 项目经理编号 |
| fund | FLOAT | 126 | 项目经费 |
| type | VARCHAR2 | 4 | 项目类型 |
| signing\_time | DATE |  | 签订时间 |
| complete\_time | DATE |  | 应完成时间 |
| check\_time | DATE |  | 验收时间 |
| remarks | VARCHAR2 | 8 | 备注 |

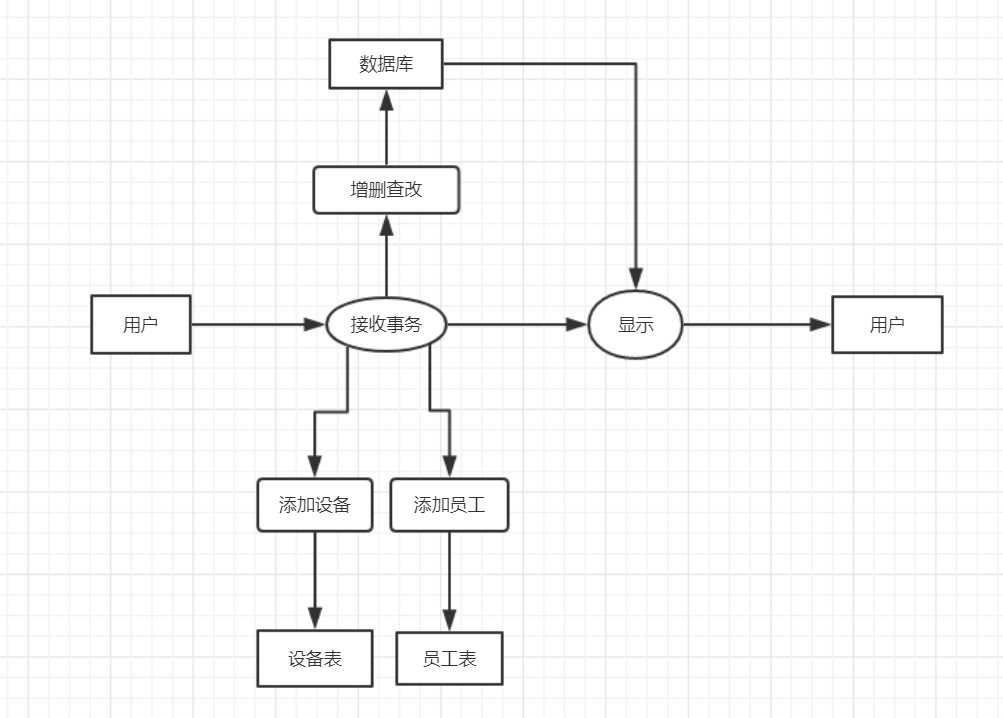
4）.设备表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| e\_id | VARCHAR2 | 4 | 设备编号 |
| e\_name | VARCHAR2 | 12 | 设备名称 |
| fee | FLOAT |  | 设备费用 |
| supplier | VARCHAR2 | 12 | 供应商 |
| p\_id | VARCHAR2 | 4 | 项目编号 |
| remarks | VARCHAR2 | 8 | 备注 |

2. 数据流图

数据流图(Data Flow Diagram)是一种图形化技术，它描绘信息流和数据从输入到输出的过程中所经受的变换。根据数据流图，可以分析出程序所需的模块和模块之间的调用关系。

如下图所示，在项目管理系统，在用户界面捕捉用户的操作，接受事务后数据流流向不同的模块。



1. 关系数据库设计：
2. 员工表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 职工号 | 姓名 | 性别 | 年龄 | 合同日期 | 职务 | 部门编号 |

1. 项目表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项目号 | 项目名 | 合同经费 | 类型 | 签订时间 | 应完成时间 | 验收时间 | 项目经理编号 | 备注 |

1. 设备采购表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目号 | 设备名称 | 费用 | 供货商 | 项目编号 | 备注 |

1. 部门表

|  |  |  |
| --- | --- | --- |
| 部门编号 | 部门名称 | 经理编号 |

### 3、逻辑结构设计

1).关系描述：

员工与部门的关系：n ：1一个员工只能属于一个部门，一个部门可以有多个员工；

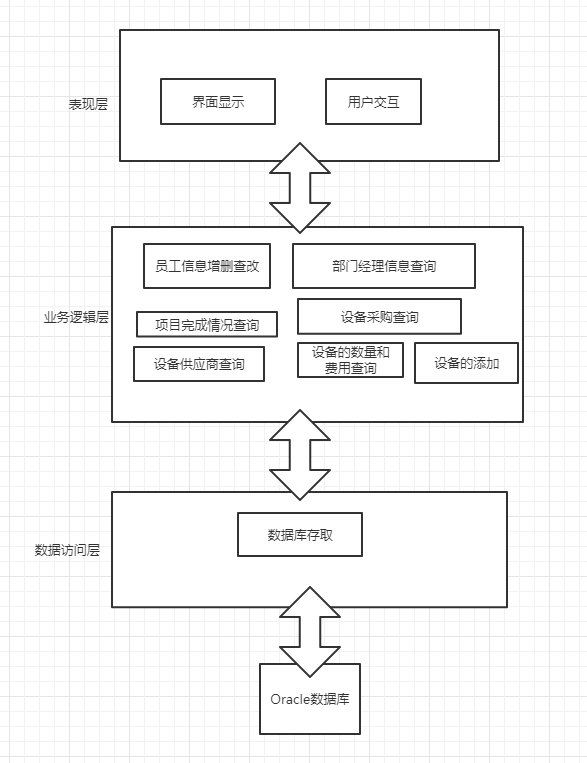
部门与项目的关系：1：n一个部门可以负责多个项目，一个项目只能由一个部门负责；

员工与项目的关系：m:n一个员工可以实现多个项目，一个项目可以由多个员工实现。

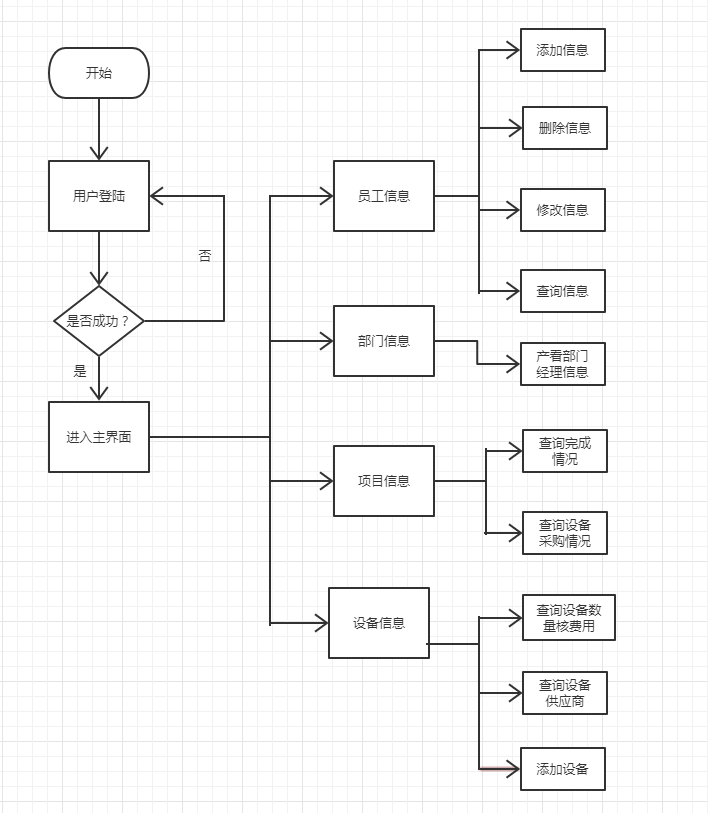
员工与设备的关系：1:n一个员工可以采购多个设备，一个设备只能由一个员工采购。

项目与设备的关系：1:n一个项目可以使用多个设备，一个设备只能由一个部门使用。

2).系统结构图：



流程图：



### 4、物理设计

存储安排及路径：将数据的稳定性将数据分开存放以提高系统性能，并对数据的存取频率对数据进行分开存放。方法选择：数据库管理系统自动会在每个表的主码上建立索引。在员工表上w\_id列上建立PK\_WORKER索引，这样可以多次进行对员工查询。同样对部门表，项目表，设备表也有这样的操作。

表格的结构：

1）.员工表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| w\_id | VARCHAR2 | 4 | 员工编号 |
| w\_name | VARCHAR2 | 16 | 员工姓名 |
| sex | CHAR | 2 | 员工性别 |
| age | CHAR | 20 | 员工年龄 |
| contract\_date | DATE |  | 合同日期 |
| d\_id | VARCHAR2 | 4 | 部门编号 |
| d\_name | VARCHAR2 | 12 | 部门名称 |
| post | VARCHAR2 | 6 | 员工职务 |

2）.部门表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| d\_id | VARCHAR2 | 4 | 部门编号 |
| d\_name | VARCHAR2 | 12 | 部门名称 |
| m\_id | VARCHAR2 | 4 | 部门经理编号 |
| m\_name | VARCHAR2 | 8 | 部门经理姓名 |
| w\_num | NUMBER |  | 员工人数 |

3）.项目表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| p\_id | VARCHAR2 | 4 | 项目编号 |
| p\_name | VARCHAR2 | 16 | 项目名称 |
| d\_id | VARCHAR2 | 4 | 部门名称 |
| w\_id | VARCHAR2 | 4 | 项目经理编号 |
| fund | FLOAT | 126 | 项目经费 |
| type | VARCHAR2 | 4 | 项目类型 |
| signing\_time | DATE |  | 签订时间 |
| complete\_time | DATE |  | 应完成时间 |
| check\_time | DATE |  | 验收时间 |
| remarks | VARCHAR2 | 8 | 备注 |

4）.设备表

|  |  |  |  |
| --- | --- | --- | --- |
| 数据元素 | 数据类型 | 数据长度 | 数据描述 |
| e\_id | VARCHAR2 | 4 | 设备编号 |
| e\_name | VARCHAR2 | 12 | 设备名称 |
| fee | FLOAT |  | 设备费用 |
| supplier | VARCHAR2 | 12 | 供应商 |
| p\_id | VARCHAR2 | 4 | 项目编号 |
| remarks | VARCHAR2 | 8 | 备注 |

索引：

对每个表的主键建立索引：

CREATE UNIQUE INDEX "CHAN"."PK\_WORKER" ON "CHAN"."worker" ("w\_id")

CREATE UNIQUE INDEX "CHAN"."PK\_DEPARTMENT" ON "CHAN"."department" ("d\_id")

CREATE UNIQUE INDEX "CHAN"."PK\_PROJECT" ON "CHAN"."project" ("p\_id")

CREATE UNIQUE INDEX "CHAN"."PK\_EQUIPMENT" ON "CHAN"."equipment" ("e\_id")

对project的”p\_name”字段表：

CREATE INDEX "CHAN"."I\_PROJECT\_NAME" ON "CHAN"."project" ("p\_name")

视图：

1).部门经理信息视图：

create or REPLACE view v\_d\_m

AS

select "worker"."w\_id","worker"."w\_name","worker"."sex","worker"."age","worker"."contract\_ date","worker"."post","department"."d\_name"

from "worker","department"

where "worker"."w\_id"="department"."m\_id"

2).项目完成情况视图：

create or REPLACE view v\_m\_p

AS

select "worker"."w\_name","project"."p\_name","project"."signing\_time","project"."complete\_time"

from "worker","project"

where "worker"."w\_id"="project"."w\_id"

### 5、系统实施

数据库初始化：

建表语句：

/\*==============================================================\*/

/\* Table: "worker" \*/

/\*==============================================================\*/

create table "worker"

(

"w\_id" VARCHAR2(4) not null,

"w\_name" VARCHAR2(8),

"sex" CHAR(2),

"age" CHAR(2),

"contract\_date" DATE,

"post" VARCHAR2(8),

"d\_id" VARCHAR2(4),

constraint PK\_WORKER primary key ("w\_id")

);

/\*==============================================================\*/

/\* Table: "department" \*/

/\*==============================================================\*/

create table "department"

(

"d\_id" VARCHAR2(4) not null,

"d\_name" VARCHAR2(16),

"m\_id" VARCHAR2(4),

constraint PK\_DEPARTMENT primary key ("d\_id")

);

/\*==============================================================\*/

/\* Table: "project" \*/

/\*==============================================================\*/

create table "project"

(

"p\_id" VARCHAR2(4) not null,

"p\_name" VARCHAR2(16),

"fund" FLOAT(126),

"type" VARCHAR2(4),

"signing\_time" DATE,

"complete\_time" DATE,

"check\_time" DATE,

"m\_id" VARCHAR2(4),

"remarks" VARCHAR2(256),

constraint PK\_PROJECT primary key ("p\_id")

);

/\*==============================================================\*/

/\* Table: "equipment" \*/

/\*==============================================================\*/

create table "equipment"

(

"e\_id" VARCHAR2(4) not null,

"e\_name" VARCHAR2(16),

"fee" FLOAT(126),

"supplier" VARCHAR2(16),

"p\_id" VARCHAR2(4),

"remarks" VARCHAR2(256),

constraint PK\_EQUIPMENT primary key ("e\_id")

);

插入数据：

worker表：

INSERT INTO "worker" VALUES ('1008', '马化腾', '男', '29', TO\_DATE('20171229201245', 'YYYYMMDDHH24MISS'), '经理', '11');

INSERT INTO "worker" VALUES ('1011', '雷军', '男', '21', TO\_DATE('20171220085418', 'YYYYMMDDHH24MISS'), '职员', '11');

INSERT INTO "worker" VALUES ('1006', '司马懿', '男', '56', TO\_DATE('20171229202106', 'YYYYMMDDHH24MISS'), '职员', '13');

INSERT INTO "worker" VALUES ('1005', '王超', '男', '38', TO\_DATE('20171018161809', 'YYYYMMDDHH24MISS'), '经理', '14');

INSERT INTO "worker" VALUES ('1001', '张三', '男', '26', TO\_DATE('20171026143550', 'YYYYMMDDHH24MISS'), '职员', '11');

INSERT INTO "worker" VALUES ('1002', '李四', '男', '25', TO\_DATE('20171026143733', 'YYYYMMDDHH24MISS'), '职员', '12');

INSERT INTO "worker" VALUES ('1003', '李玲', '女', '27', TO\_DATE('20171026144030', 'YYYYMMDDHH24MISS'), '经理', '13');

INSERT INTO "worker" VALUES ('1004', '王五', '男', '25', TO\_DATE('20171026144412', 'YYYYMMDDHH24MISS'), '经理', '14');

department表：

INSERT INTO "department" VALUES ('11', '采购部', '1008');

INSERT INTO "department" VALUES ('12', '销售部', '1002');

INSERT INTO "department" VALUES ('13', '策划部', '1003');

INSERT INTO "department" VALUES ('14', '人事部', '1004');

project表：

INSERT INTO "project" VALUES ('004', '跨海大桥', 300000000, '桥梁', TO\_DATE('20160226150155', 'YYYYMMDDHH24MISS'), TO\_DATE('20170326150206', 'YYYYMMDDHH24MISS'), TO\_DATE('20171001150213', 'YYYYMMDDHH24MISS'), '1004', NULL);

INSERT INTO "project" VALUES ('001', '京沪高速', 300000, '建筑', TO\_DATE('20170901145036', 'YYYYMMDDHH24MISS'), TO\_DATE('20171026145042', 'YYYYMMDDHH24MISS'), TO\_DATE('20171027145048', 'YYYYMMDDHH24MISS'), '1005', NULL);

INSERT INTO "project" VALUES ('002', '青藏铁路', 1500000, '建筑', TO\_DATE('20170701145439', 'YYYYMMDDHH24MISS'), TO\_DATE('20170901145459', 'YYYYMMDDHH24MISS'), TO\_DATE('20171026145505', 'YYYYMMDDHH24MISS'), '1008', NULL);

INSERT INTO "project" VALUES ('003', '鸟巢', 30000000, '工程', TO\_DATE('20170801145900', 'YYYYMMDDHH24MISS'), TO\_DATE('20170901145906', 'YYYYMMDDHH24MISS'), TO\_DATE('20171004145911', 'YYYYMMDDHH24MISS'), '1003', NULL);

equipment表：

INSERT INTO "equipment" VALUES ('101', '挖掘机', 100000, '徐工', '001', NULL);

INSERT INTO "equipment" VALUES ('102', '玻璃', 10000, '通用', '003', NULL);

INSERT INTO "equipment" VALUES ('103', '铝合金', 20000, '上汽', '002', NULL);

INSERT INTO "equipment" VALUES ('104', '液晶', 50000, '京东方', '004', NULL);

INSERT INTO "equipment" VALUES ('105', '台式机', 100000, '清华同方', '003', '办公专用');

INSERT INTO "equipment" VALUES ('108', 'A4纸', 200, '华润', '003', NULL);

INSERT INTO "equipment" VALUES ('109', '键鼠套装', 1000, '达尔优', '003', NULL);

### 6、运行维护

转储：定期进行静态转储，动态转储，海量转储

恢复数据库：

数据库中可能发生各种各样的故障，大致可以分为以下几类：

1．事务故障的恢复策略主要是：

反向扫描日志文件，查找该事物的更新操作；对该事物的更新操作执行逆操作；继续反向扫描日志文件，查找该事物的其他更新操作，并做同样处理；如此处理下去，直至读到此事物的开始标记，事物故障恢复就完成了。

2．系统故障的恢复策略主要是：正向扫描日志文件，找出在故障发生前已经提交的事务，将其事务标识记入重做队列。同时找出故障发生时尚未完成的事务，将其事务标识记入撤消队列；对撤消队列的各个事务进行撤消处理；对重做队列的各个事务进行重做处理。

3．介质故障的恢复策略主要是：装入最新的数据库后备副本，使数据库恢复到最近一次转储的一致性状态。装入相应的日志文件副本，重做已完成的事物。

### 7、用户手册

(1).系统信息

名称：项目管理系统

作者：陈均永

时间：2017.12.20

(2).安装及配置

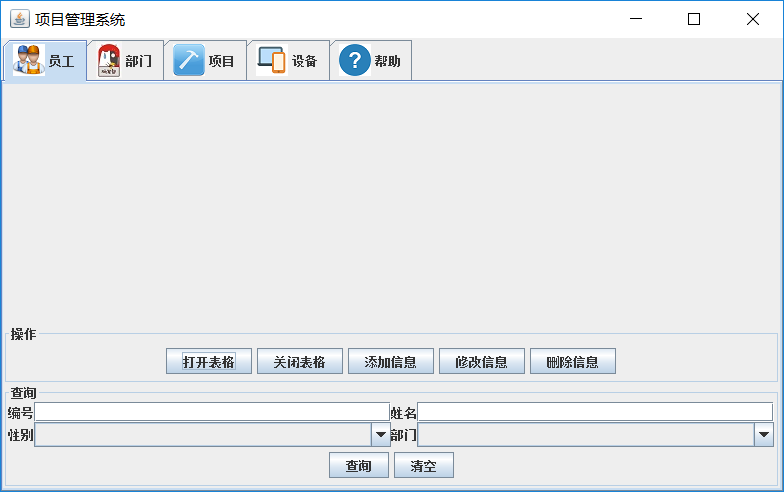
本系统基于Java开发，要使用本系统在本机上必须安装有Java开发环境。数据库使用Oracle。

(3).使用方法

登录界面：



主界面：



1).基本操作

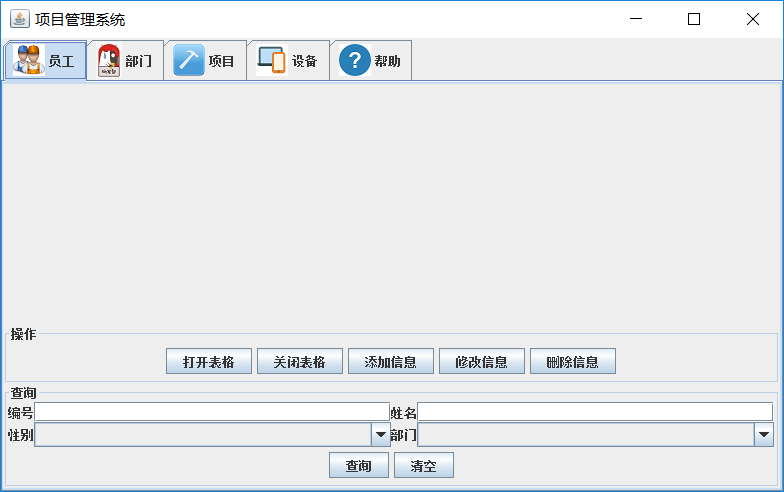
登录：

运行程序之后首先进入登陆界面

在登录界面中输入服务名，用户名(数据库的用户名，比如SCOTT)，密码。例如我的数据库的服务名是orcl,用户名是CHAN。经过正确的连接即可进入到程序的主界面。

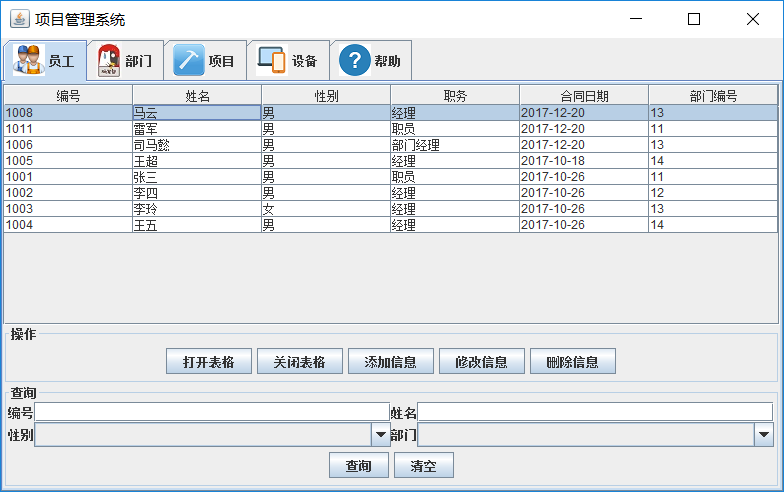


如下图所示，应用程序界面包括 “切换卡 (用来切换页面)”，“显示区 (显示基本信息)”，“查询区 (用来进行数据查询)”，“控制按钮 (用来进行基本操作)”。

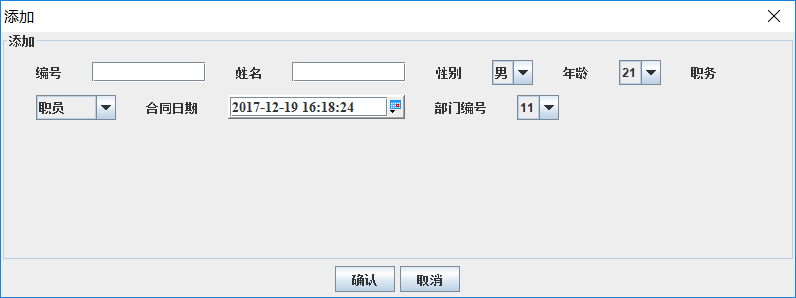


2).员工页面

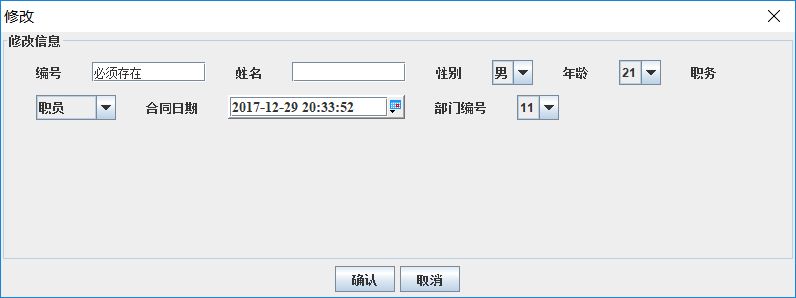
点击打开表格可以查看所有员工的基本信息，关闭表格可以清空页面上的表格显示。



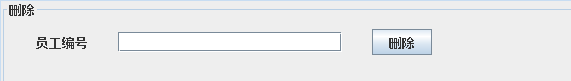
点击添加信息可以添加某个员工的信息：



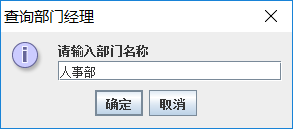
点击修改可以修改某个员工的信息（注意：员工的编号在数据库中是唯一的，被修改的员工的编号在数据库中必须存在）

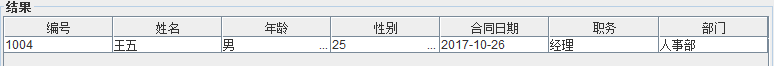


点击删除信息可以删除某个员工的信息（同样编号必须存在）

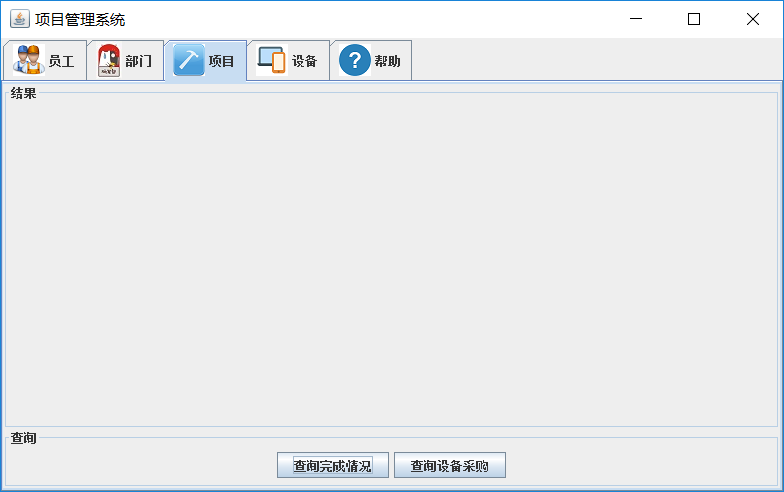


在部门页面可以通过部门名称查询部门经理的信息：

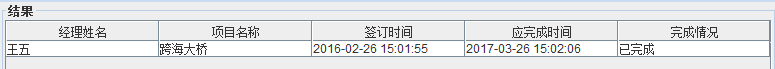




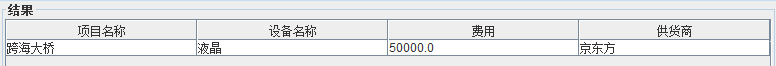
在项目页面



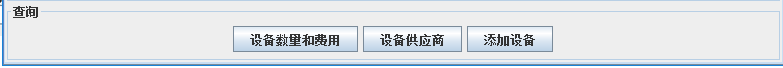
点击查询完成情况可以查询某项目经理（姓名）完成合同的情况



点击查询设备采购可以某个项目设备采购情况



在设备页面有一下几个查询：



### 8、附录

对客户端进行响应的逻辑处理函数大部分通过在客户端中调用数据库的存储过程完成。

客户端程序逻辑处理的核心代码如下(界面代码在提交的工程中)：

**import** com.connectdb.ConnectDB;  
**import** oracle.jdbc.OracleTypes;  
**import** java.sql.\*;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Vector;  
**import** java.util.Date;  
  
**public class** ProjectFunction {  
 */\*\*  
 \* 添加员工信息  
 \** ***@param*** *\*/* **public static boolean** addWorker(String wId,String dId,String wName,String sex,String age,String date,String dName,String post){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 System.***out***.println(wId+dId+wName+sex+age+date+dName+post);  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *p\_addworker***(?,?,?,?,?,?,?)}"**);  
 c.setString(1,wId);  
 c.setString(2,wName);  
 c.setString(3,sex);  
 c.setString(4,age);  
 System.***out***.println(**"传入的时间是 "**+date);  
 c.setString(5,date);  
 c.setString(6,post);  
 c.setString(7,dId);  
 *//c.setDate(6,Date.valueOf(date));  
 //执行Oracle存储过程* c.execute();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return true**;  
 }  
  
 */\*\*  
 \* 删除员工信息  
 \** ***@param*** *\*/* **public static void** deleteWorker(String wId){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *p\_delworker***(?)}"**);  
 c.setString(1,wId);  
 *//执行Oracle存储过程* c.execute();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 }  
  
 */\*\*  
 \* 查询员工信息  
 \** ***@param*** *\*/* **public static** Vector<Vector<String>> queryWorker(String sql){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 ResultSet rs=**null**;  
 Vector<Vector<String>> list=**new** Vector<Vector<String>>();  
 **int** i=0;  
 **try** {  
 Statement st = conn.createStatement();  
 rs = st.executeQuery(sql);  
 System.***out***.println(**"执行查询"**);  
 *// 判断结果集是否有效* **while**(rs.next()){  
 *// 数据库存在，返回结果集* System.***out***.println(rs.getString(**"w\_id"**));  
 System.***out***.println(rs.getString(**"w\_name"**));  
 System.***out***.println(rs.getString(**"sex"**));  
 System.***out***.println(rs.getString(**"post"**));  
 Vector<String> result=**new** Vector<String>();  
 result.add(rs.getString(**"w\_id"**));  
 result.add(rs.getString(**"w\_name"**));  
 result.add(rs.getString(**"sex"**));  
 result.add(rs.getString(**"d\_name"**));  
 result.add(rs.getString(**"post"**));  
 result.add(rs.getDate(**"contract\_date"**).toString());  
 list.add(i,result);  
 i++;  
 }  
 *// 释放此 ResultSet 对象的数据库和 JDBC 资源* rs.close();  
 *// 释放此 PreparedStatement 对象的数据库和 JDBC 资源* st.close();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** list;  
 }  
  
 */\*\*  
 \* 修改员工信息  
 \** ***@param*** *\*/* **public static boolean** updateWorker(String wId,String dId,String wName,String sex,String age,String date,String dName,String post){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 System.***out***.println(wId+dId+wName+sex+age+date+dName+post);  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *p\_updateworker***(?,?,?,?,?,?,?)}"**);  
 c.setString(1,wId);  
 c.setString(2,wName);  
 c.setString(3,sex);  
 c.setString(4,age);  
 System.***out***.println(**"传入的时间是 "**+date);  
 c.setString(5,date);  
 c.setString(6,post);  
 c.setString(7,dId);  
 *//c.setDate(6,Date.valueOf(date));  
 //执行Oracle存储过程* c.execute();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return true**;  
 }  
  
 */\*\*  
 \* 获取员工列表  
 \** ***@param*** *\*/* **public static** Vector<Vector<String>> getWorkerList(){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 ResultSet rs=**null**;  
 Vector<Vector<String>> list=**new** Vector<Vector<String>>();  
 **int** i=0;  
 **try** {  
 Statement st = conn.createStatement();  
 String sql=**"SELECT** *\** **FROM \"worker\""**;  
 rs = st.executeQuery(sql);  
 *// 判断结果集是否有效* **while**(rs.next()){  
 *// 数据库存在，返回结果集* Vector<String> result=**new** Vector<String>();  
 result.add(rs.getString(**"w\_id"**));  
 result.add(rs.getString(**"w\_name"**));  
 result.add(rs.getString(**"sex"**));  
 result.add(rs.getString(**"post"**));  
 result.add(rs.getDate(**"contract\_date"**).toString());  
 result.add(rs.getString(**"d\_id"**));  
 list.add(i,result);  
 i++;  
 }  
 System.***out***.println(**"长度 "**+list.size());  
 rs.close();  
 *// 释放此 PreparedStatement 对象的数据库和 JDBC 资源* st.close();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** list;  
 }  
  
 */\*\*  
 \* 查询部门经理  
 \** ***@param*** *\*/* **public static** Vector<String> queryManager(String dName){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 String wId=**""**,wName=**""**,sex=**""**,age=**""**,post=**""**,contractTime=**""**;  
 Vector<String> list=**new** Vector<String>();  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_FDMANAGER***(?,?,?,?,?,?,?)}"**);  
 c.setString(1,dName);  
 c.registerOutParameter(2, OracleTypes.***VARCHAR***);  
 c.registerOutParameter(3,OracleTypes.***VARCHAR***);  
 c.registerOutParameter(4,OracleTypes.***CHAR***);  
 c.registerOutParameter(5,OracleTypes.***CHAR***);  
 c.registerOutParameter(6,OracleTypes.***DATE***);  
 c.registerOutParameter(7,OracleTypes.***VARCHAR***);  
 *//执行Oracle存储过程* c.execute();  
 wId=c.getString(2);  
 wName=c.getString(3);  
 sex=c.getString(4);  
 age=c.getString(5);  
 contractTime=c.getDate(6).toString();  
 post=c.getString(7);  
 System.***out***.println(**"执行查询"**);  
 list.add(wId);  
 list.add(wName);  
 list.add(sex);  
 list.add(age);  
 list.add(contractTime);  
 list.add(post);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** list;  
 }  
  
 */\*\*  
 \* 查询项目设备采购信息  
 \** ***@param*** *\*/* **public static** Vector<String> getEqubyPro(String pName){  
 Vector<String> result=**new** Vector<String>();  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 String eNmae=**""**,eSupplier=**""**;  
 **float** eFee;  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_GETEBYP***(?,?,?,?)}"**);  
 c.setString(1,pName);  
 c.registerOutParameter(2, OracleTypes.***VARCHAR***);  
 c.registerOutParameter(3,OracleTypes.***FLOAT***);  
 c.registerOutParameter(4,OracleTypes.***VARCHAR***);  
 *//执行Oracle存储过程* c.execute();  
 eNmae=c.getString(2);  
 eFee=c.getFloat(3);  
 eSupplier=c.getString(4);  
 result.add(eNmae);  
 result.add(String.*valueOf*(eFee));  
 result.add(eSupplier);  
 System.***out***.println(**"查询设备 "**+eNmae+eFee+eSupplier);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** result;  
 }  
  
 */\*\*  
 \* 查询设备采购费用信息  
 \** ***@param*** *\*/* **public static float** getEquFee(String eName){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 **float** eFee=0;  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_GETEFEE***(?,?)}"**);  
 c.setString(1,eName);  
 c.registerOutParameter(2,OracleTypes.***FLOAT***);  
 *//执行Oracle存储过程* c.execute();  
 eFee=c.getFloat(2);  
 System.***out***.println(**"查询设备 "**+eName+eFee+**"----"**);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** eFee;  
 }  
  
 */\*\*  
 \* 添加设备  
 \** ***@param*** *\*/* **public static void** addEqu(String eId,String eName,**float** fee,String supplier,String remarks,String pId){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 System.***out***.println(eId+eName+fee+supplier+remarks+pId);  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *p\_addequ***(?,?,?,?,?,?)}"**);  
 c.setString(1,eId);  
 c.setString(2,eName);  
 c.setFloat(3,fee);  
 c.setString(4,supplier);  
 c.setString(5,pId);  
 c.setString(6,remarks);  
 *//执行Oracle存储过程* c.execute();  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 }  
  
 */\*\*  
 \* 查询设备供应商信息  
 \** ***@param*** *\*/* **public static** String getEquSup(String eName){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 String eSupplier=**""**;  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_GETESUP***(?,?)}"**);  
 c.setString(1,eName);  
 c.registerOutParameter(2,OracleTypes.***VARCHAR***);  
 *//执行Oracle存储过程* c.execute();  
 eSupplier=c.getString(2);  
 System.***out***.println(**"查询设备 "**+eName+eSupplier);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** eSupplier;  
 }  
  
 */\*\*  
 \* 查询项目员工信息  
 \** ***@param*** *\*/* **public static void** getWerbyPro(String pName){  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 String wNmae=**""**,wId=**""**;  
 **float** eFee;  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_GETEBYP***(?,?,?,?)}"**);  
 c.setString(1,pName);  
 c.registerOutParameter(2, OracleTypes.***VARCHAR***);  
 c.registerOutParameter(3,OracleTypes.***FLOAT***);  
 c.registerOutParameter(4,OracleTypes.***VARCHAR***);  
 *//执行Oracle存储过程* c.execute();  
 wNmae=c.getString(2);  
 wId=c.getString(4);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 }  
  
 */\*\*  
 \* 查询项目完成情况  
 \** ***@param*** *\*/* **public static** Vector<String> queryComplete(String ManagerName){  
 Vector<String> result=**new** Vector<String>();  
 *// 获取数据库连接Connection对象* Connection conn = ConnectDB.*getConnection*();  
 String date1=**""**,date2=**""**,pName=**""**,isComplete=**""**;  
 **try** {  
 CallableStatement c = conn.prepareCall(**"{call** *P\_ISFINISHED***(?,?,?,?)}"**);  
 c.setString(1,ManagerName);  
 c.registerOutParameter(2, OracleTypes.***DATE***);  
 c.registerOutParameter(3, OracleTypes.***DATE***);  
 c.registerOutParameter(4, OracleTypes.***VARCHAR***);  
 *//执行Oracle存储过程* c.execute();  
 date1=c.getString(2).toString();  
 date2=c.getString(3).toString();  
 pName=c.getString(4);  
 System.***out***.println(**"签订时间是："**+date2);  
 System.***out***.println(**"应完成时间是："**+date1);  
 *//pName=c.getString(4);* Date date;  
 date=c.getDate(2);  
 System.***out***.println(**"应完成时间是 "**+date1);  
 System.***out***.println(**"时间格式 "**+date);  
 Date now = **new** Date();  
 SimpleDateFormat dateFormat = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm:ss"**);*//可以方便地修改日期格式* String hehe = dateFormat.format( now );  
 System.***out***.println(hehe);  
 **if**(date1.compareTo(hehe)<0){  
 isComplete=**"已完成"**;  
 System.***out***.println(**"已经完成"**);  
 } **else** {  
 isComplete=**"未完成"**;  
 }  
 result.add(pName);  
 result.add(date2);  
 result.add(date1);  
 result.add(isComplete);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }**finally**{  
 *// 关闭数据库连接* ConnectDB.*closeConnection*(conn);  
 }  
 **return** result;  
 }  
}

### 9、总结

遇到的问题：

1. Oracle数据库与之前接触过的数据库SQL语句有些不一样，除了常用的增删查改与其他主关系型数据库相似，其他的语句还是有一些区别的。在实习中经常因为在SQL语句中少个双引号或者将双引号写成单引号而出错，所以在刚开始写存储过程中用了不少时间。可见Oracle数据库对语法的要求比较严格。

存在的问题：

1. 在这次实习中，我发现自己的数据库设计能力还稍有不足，对项目管理系统的几个表的设计不是很合理，表中有些字段的类型和长度可能与实际应用不相符。在进行数据库设计时应该考虑实际项目中的应用，这样才能设计出满足实际需求的数据库管理系统。
2. 客户端程序中的程序设计时没有很好地采用架构来实现。使用三层逻辑架构时没有做到完全的层次分离，在表现层中还存在一些业务逻辑层的代码。数据库中每个表都应该用一个类来封装，然后对类中的成员变量也就是数据库表中的每个字段分别设置set()和get()方法。
3. 程序尚存在一些已知和未知的BUG，反映出自己在程序设计时对问题的考虑不够全面，对于在实际中可能存在的情况没完全把握。

收获：

1. 通过完成数据库课程设计，加深了我对数据库理论知识和在实际应用方面的认识，在实践中发现了自己在数据库设计和程序设计上的不足，对于提高自己的编程能力很有帮助。实习中的数据库设计部分可以巩固在实用数据库中的知识，让我学会在实际场景中更好地使用数据库。
2. 这次实习让我学习了在项目中经常使用到的主流关系型数据库Oracle，掌握Oracle的特征和基本使用方法。在项目中使用Oracle可以实现对SQL语句的封装，不仅提高程序的执行效率还提高了数据的安全性。当对数据库进行复杂操作时(如对多个表进行 Update,Insert,Query,Delete 时），可将此复杂操作用存储过程封装起来与数据库提供的事务处理结合一起使用。这些操作，如果用程序来完成，就变成了一条条的 SQL 语句，可能要多次连接数据库。而换成存储，只需要连接一次数据库就可以了。