**数据库原理与应用**

**课程实习报告**

课题名称： **企业员工考勤管理系统**

指导教师： 谭琳

专业班级： 计科四班

学 号： 20203773

姓 名： 林靖洋

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **指导教师评定情况：** | | | | | | | |
| **考核内容** | | **等级** | | | | | **综合评语及成绩** |
| 优 | 良 | 中 | 合格 | 不合格 |
| 数据库  设计 | 40 |  |  |  |  |  |  |
| 答辩 | 10 |  |  |  |  |  |
| 设计报告文档 | 30 |  |  |  |  |  |
| 考勤 | 20 |  |  |  |  |  |
| 其他 |  |  |  |  |  |  |
| 签名： 年 月 日 | | | | | | | |

完成日期： 年 月 日

**完成日期： 年 月 日**

1

目 录

[1 问题的提出与定义 3](#_Toc18160)

[2 需求分析 3](#_Toc27622)

[2.1 系统目的 3](#_Toc21497)

[2.2 用户需求 3](#_Toc5645)

[2.3 业务流程图 3](#_Toc9046)

[2.4 数据流图 4](#_Toc31128)

[2.5 数据字典 5](#_Toc21101)

[3 概念设计 5](#_Toc29521)

[4 逻辑设计 6](#_Toc5101)

[5 物理设计 6](#_Toc22512)

[6 数据库实现 6](#_Toc19281)

[6.1 表的创建 6](#_Toc24609)

[6.2 自动作业创建 7](#_Toc3399)

[6.3 触发器创建 7](#_Toc22981)

[6.3.1触发器—自动将员工表中的数据插入数据总表 7](#_Toc29807)

[6.3.2触发器—将数据总表中已删除的数据从员工表中删除 8](#_Toc28720)

[6.3.3触发器—自动计算考勤天数并赋值给列“考勤天数” 8](#_Toc29981)

[6.4 存储过程创建 8](#_Toc11128)

[6.4.1 存储过程—考勤打卡 8](#_Toc28659)

[6.4.2 存储过程—考勤天数计算 9](#_Toc1804)

[7 应用程序实现 9](#_Toc661)

[7.1 应用程序设计 9](#_Toc16938)

[7.2 应用程序实现 10](#_Toc20172)

[7.2.1 登录窗体设计 10](#_Toc24820)

[7.2.2 管理窗体设计 11](#_Toc29636)

[7.2 应用程序效果展示 15](#_Toc19244)

[8 结论 15](#_Toc4693)

[9 参考文献 16](#_Toc7657)

# 1 问题的提出与定义

企业员工考勤管理系统，是每一个社会企业单位所不可或缺的一个信息化建设部分。该系统对于管理员工考勤情况以及季末绩效评估都发挥着不可替代的作用。因此，本系统的设计着重于让领导和员工能更加方便、快捷、全面地查看与管理考勤数据，再配合多种管理模块，能极大简化企业内部运作机制，加强内部审查管理，优化内部管理结构。

# 2 需求分析

基于上文所提出的设计理念，第二章将针对上述问题对数据库系统形成基本设计思路以及流程设计。

## 2.1 系统目的

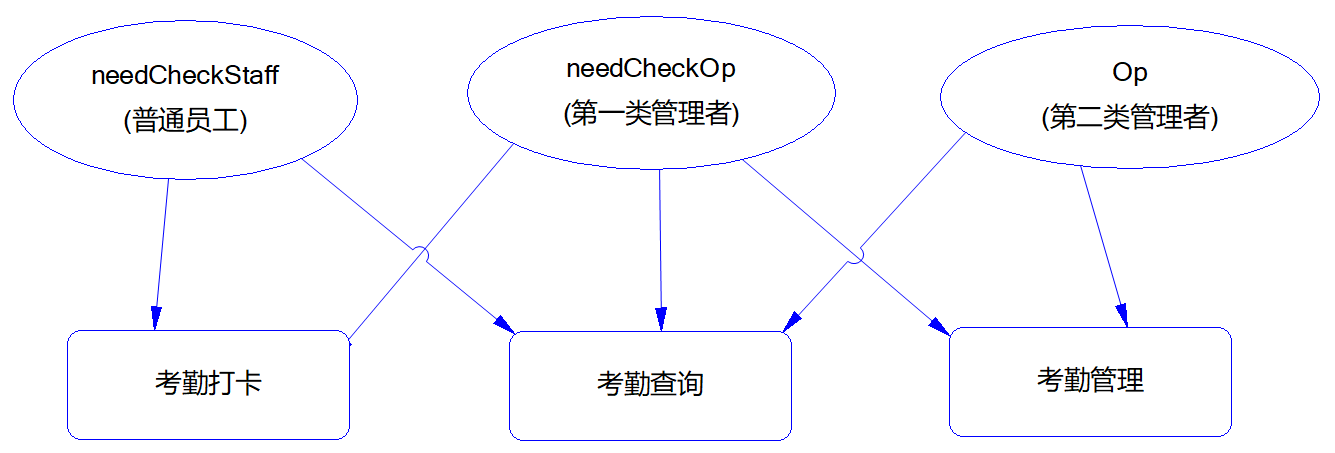
本系统数据库主体为随时间自动增长的员工考出勤情况表，基于本表将衍生出很多企业管理模块以实现对企业员工信息的全方位管理，如考勤率计算，某一时间段考勤情况反馈，以及考勤情况对工资影响计算等等方面。

## 2.2 用户需求

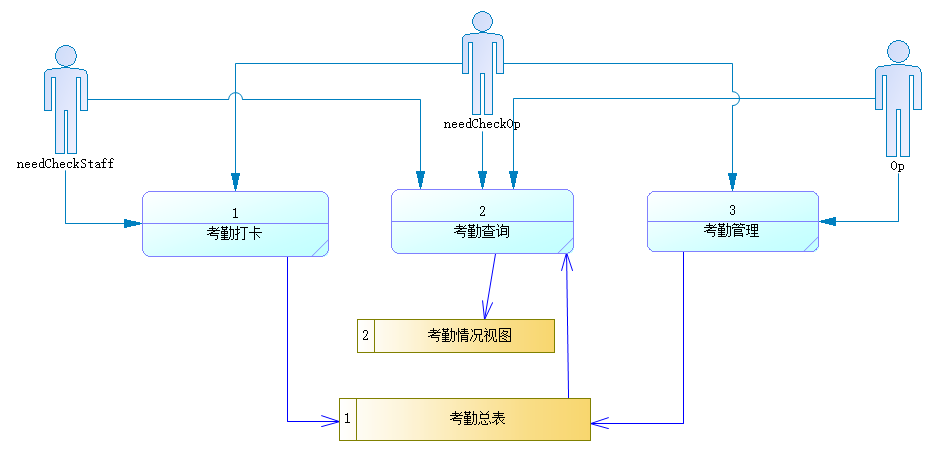
考虑到使用用户身份不同，员工的应用场景主要为考勤打卡和考勤情况查询，而管理者则分为两类，一类是本身需要考勤打卡的管理者，这类管理者的应用场景为考勤打卡以及查询，并且还有考勤情况查看管理以及统计权限，但是该类管理者却不能修改自身考勤情况。而第二类则是本身不需要考勤打卡的管理者，这类管理者由于特定原因没有考勤要求，因此该类管理者除了不需要考勤打卡以外，还拥有第一类管理者所有权限，并且能够管理、修改第一类管理者的考勤情况。

## 2.3 业务流程图

基于上文所描述的设计思路，下面给出本系统的基本业务流程图。我们将上文中提到的员工命名为“needCheckStaff”，将第一类管理者命名为“needCheckOp”，以及将第二类管理者命名为“Op”。具体角色定义请参见2.2 用户需求。



## 2.4 数据流图

至此可由设计思路以及业务流程图绘出基本数据流图。

“needCheckStaff”没有“数据总表”的修改、查看 权限，只能通过预先设置好的功能按钮或存储过程，限制性地对“考勤总表”进行修改，即“考勤打卡”功能。这位角色还能通过功能按钮或存储过程使“考勤总表”生成个人的考勤情况视图。

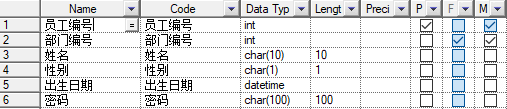
“needCheckOp”拥有大部分“考勤总表”的权限，除“考勤打卡”外，其还拥有修改“考勤总表”的权限，但是无法修改自身数据。并且其可以直接查看“考勤总表”

“Op”没有“考勤打卡”功能，并且拥有“needCheckOp”的所有权限以及修改他们数据的权限

## 2.5 数据字典

在此列出本系统所需要使用到的数据字典。

员工表数据字典



部门表数据字典

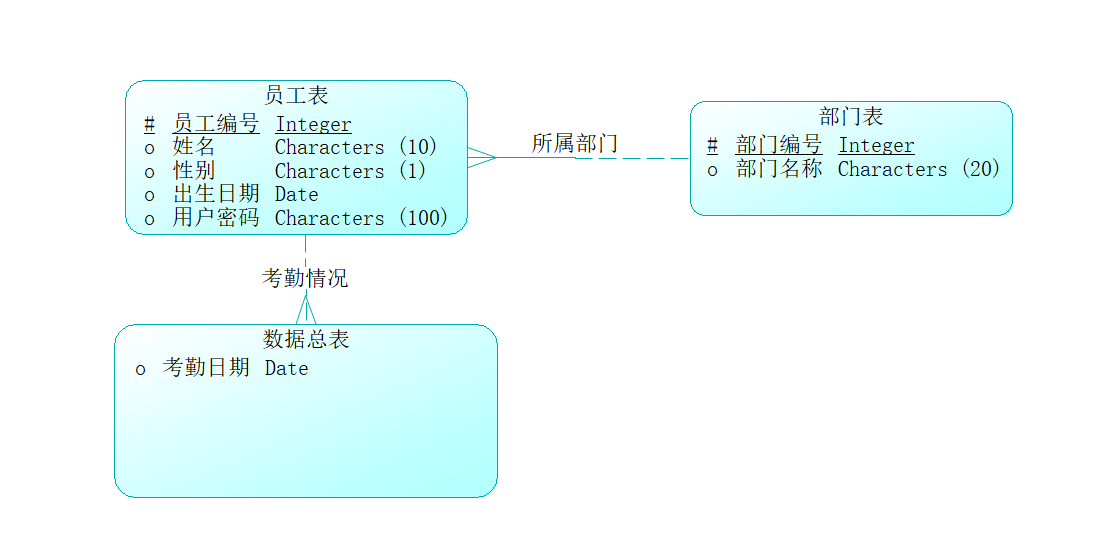


考勤表数据字典

E:\GitHub\DataBase_Design\Pics\Date_3.pngDate_3

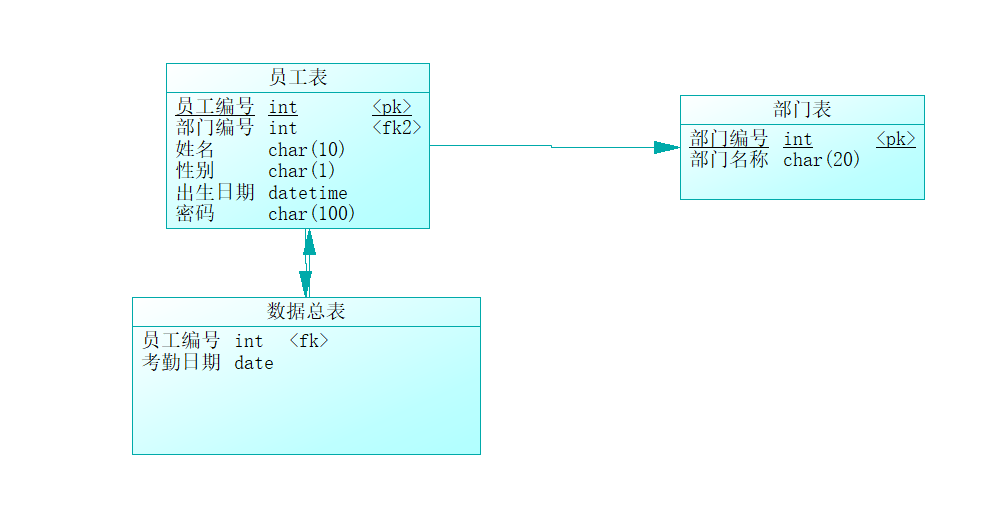
# 3 概念设计

下面给出概念设计的cdm图（E-R图）



# 4 逻辑设计

由上面的cdm图可以直接使用 Powerdesigner 自动生成pdm图



# 5 物理设计

由前文已经可以直观的展现出本系统所需要创建的表结构。目前需要创建三个表以及一个可以自由生成的视图，基于“SQL Server 2019”给出表的结构：员工表（员工编号（int，主键），部门编号（int，外键），姓名（char（10），不可为空），性别（char（1），不可为空），出生日期（datetime）），部门表（部门编号（int，主键），部门名称（char（20），不可为空）），数据总表（员工编号（int，外键），考勤天数（int）），数据总表会随时间推移而增加考勤属性，以及可由员工生成的视图。

# 6 数据库实现

## 6.1 表的创建

Use DatabaseDesign

GO

Create Table 部门表(

部门编号 int Primary key Not null,

部门名称 nvarchar(20) Not null

)

Create Table 员工表(

员工编号 int Primary key Not null,

部门编号 int Not null,

姓名 nchar(10) Not null,

性别 nvarchar(1) Not null check(性别 = '男' OR 性别 = '女'),

出生日期 datetime Not null,

Foreign key(部门编号) References 部门表(部门编号)

)

Create Table 数据总表(

员工编号 int Not null,

考勤天数 int Not null Default(0),

Foreign key(员工编号) References 员工表(员工编号)

)

## 6.2 自动作业创建

由于该系统需要每日新增一列以时间为名称的考勤情况，因此使用自动作业来在每天00：00：01时对数据总表进行操作，具体自动作业代码如下：

Declare @nowDate varchar(60)

set @nowDate = CONVERT(varchar(8),GETDATE(),112)

if not exists(SELECT \* FROM syscolumns WHERE id = (SELECT id FROM sysobjects WHERE name = '数据总表') AND name = @nowDate)

begin

Alter table 数据总表 add a bit Default(0)

EXEC sp\_rename '数据总表.[a]' , @nowDate

end

## 6.3 触发器创建

### 6.3.1触发器—自动将员工表中的数据插入数据总表

Create TRIGGER dbo.Autoinsert

ON [dbo].[员工表]

AFTER INSERT

AS

BEGIN

SET NOCOUNT ON;

DECLARE @uid int

SET @uid = (select 员工编号 from inserted)

if LEFT(@uid, 1) <> '3'

begin

INSERT INTO dbo.数据总表(

员工编号

)

SELECT

i.员工编号

FROM

inserted i

end

END

### 6.3.2触发器—将数据总表中已删除的数据从员工表中删除

Create TRIGGER dbo.AutoDelete

ON dbo.员工表

AFTER delete

AS

BEGIN

SET NOCOUNT ON;

Delete from dbo.数据总表

where 员工编号 = (select 员工编号 from deleted)

END

### 6.3.3触发器—自动计算考勤天数并赋值给列“考勤天数”

Create Trigger dbo.AutoCount

ON dbo.数据总表

AFTER UPDATE

AS

BEGIN

declare @sql varchar(200)

set @sql=''

select @sql = @sql +'","'+ name from syscolumns where id=object\_id('数据总表') and name not in( '员工编号' , '考勤天数' )

set @sql = @sql + '"'

set @sql=stuff(@sql,1,2,'')

EXEC('EXEC p\_count ''数据总表'', ''' + @sql + ''' , ''1'' , '','' , ' + '''考勤天数''')

END

注：该触发器调用了存储过程“p\_count” 详情见 “6.4.2 存储过程—考勤天数计算”

## 6.4 存储过程创建

本系统需要限制不同角色所拥有的权限，因此通过预设好的存储过程进行角色与数据库的交互，这样既能方便使用者操作数据库，也能保障信息安全性。

### 6.4.1 存储过程—考勤打卡

该存储过程用于使Java直接调用存储过程进行快速考勤打卡。使用角色：普通员工，需打卡管理员。

ALTER proc [dbo].[proc\_check]

as

begin

declare @nowdate varchar(60)

set @nowdate = CONVERT(varchar(8),GETDATE(),112)

EXEC('update 数据总表 set "' + @nowdate + '" = 1 where 员工编号 = SUSER\_NAME()')

End

### 6.4.2 存储过程—考勤天数计算

将考勤情况统计并赋值给列“考勤天数”

Create proc [dbo].[p\_count]

@table sysname, @columns nvarchar(max), @value nvarchar(max), @separator char(1) = ',', @updateto sysname = null

as

declare @sql nvarchar(max)

set @sql =

case when @updateto is null then 'select' else 'update x set ' + quotename(@updateto) + '=' end +

' case when ' + replace(@columns, @separator, '=' + QuoteName(@value,'''') +

' then 1 else 0 end + case when ') + '=' + QuoteName(@value,'''') + ' then 1 else 0 end

from ' + quotename(@table) + ' x'

print @sql

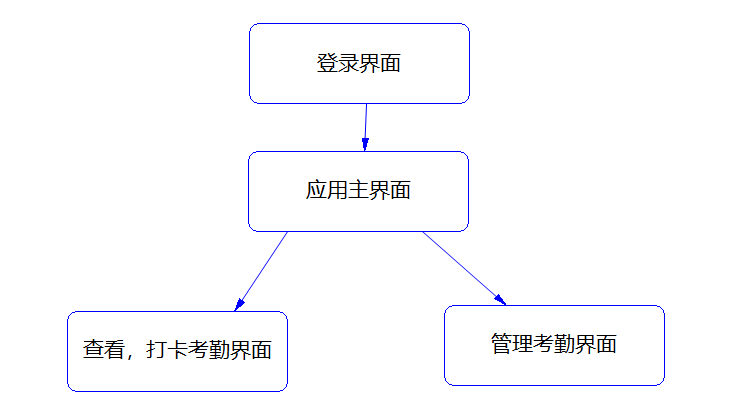
exec (@sql)

GO

# 7 应用程序实现

## 7.1 应用程序设计

基于本数据库，下面实现了该数据库外接应用程序，本应用程序采用Java实现所有功能，通过JDBC实现应用程序与数据库之间的连接。用户可以根据自己的身份方便快捷的在自身权限内查看、操作数据库。下图为本应用程序业务流程图。



## 7.2 应用程序实现

### 7.2.1 登录窗体设计

设计简洁，实现基本登录功能，效果图如下所示：



实现代码如下：

void loginPanel(){

setLayout(null);

setVisible(true);

//输入框

user = new JTextField();

paswd = new JTextField();

user.addFocusListener(new JTextFieldHintListener(user, "请输入用户名"));

paswd.addFocusListener(new JTextFieldHintListener(paswd, "请输入密码"));

user.setBounds(300,170,200,40);

paswd.setBounds(300,220,200,40);

add(user); add(paswd);

//主体文字

JLabel loginText = new JLabel("用户登录");

loginText.setFont(new Font("宋体",Font.BOLD,30));

loginText.setBounds(340,50,200,100);

add(loginText);

//提示文字

JLabel tipText = new JLabel("");

tipText.setBounds(350, 300, 200, 40);

add(tipText);

//提交按钮功能实现

submitButton = new JButton("登录");

submitButton.setBounds(300,270,200,40);

submitButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

boolean flag = Program.connection(user.getText(), paswd.getText());

if(flag == true){

Main.loginFrame.dispose();

Main.mainFrame = new JFrame();

Main.mainFrameInit(Main.mainFrame);

}

else{

tipText.setText("用户名或密码错误");

}

//Main.loginFrame.dispose();

}

});

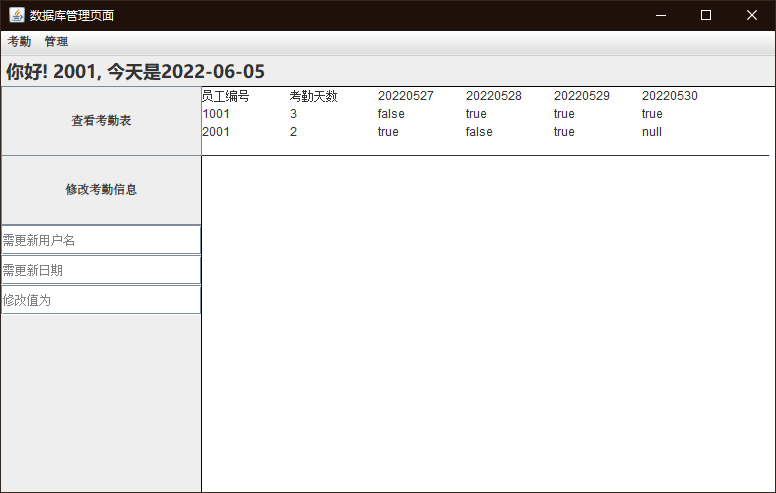
add(submitButton);

}

### 7.2.2 管理窗体设计

在用户成功登录数据库后，界面分为打卡界面与管理界面,打卡界面允许用户进行每日打卡，并且还可以查看考勤情况。管理界面允许管理员查看全体用户的考勤情况并对具体用户具体日期的考勤情况进行修改。下面两张图分别为打卡界面和管理界面。





代码如下：

void checkPanel(){

setVisible(true);

setLayout(null);

WelcomeText welcomeText = new WelcomeText();

welcomeText.welcomeText();

add(welcomeText);

JButton checkButton = new JButton("一键打卡");

checkButton.setBounds(0,30,201,70);

checkButton.setContentAreaFilled(false);

checkButton.setFocusPainted(false);

add(checkButton);

JButton queryButton = new JButton("考勤查询");

queryButton.setBounds(0,99,201,70);

queryButton.setContentAreaFilled(false);

queryButton.setFocusPainted(false);

add(queryButton);

displayArea = new JTextArea();

displayArea.setBorder(BorderFactory.createLineBorder(Color.BLACK,1));

displayArea.setBounds(200,30,590,470);

displayArea.setEditable(false);

add(displayArea);

checkButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(Program.fCheck()){

displayArea.append("每日打卡成功！\n");

Panel.displayArea.append("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

else{

displayArea.append("每日打卡失败！\n");

Panel.displayArea.append("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

//displayArea.append("每日打卡成功！\n");

}

});

queryButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

Program.fQuery();

}

});

}

void ctrlPanel(){

setVisible(true);

setLayout(null);

WelcomeText welcomeText = new WelcomeText();

welcomeText.welcomeText();

add(welcomeText);

JButton checkButtonForCtrl = new JButton("查看考勤表");

checkButtonForCtrl.setBounds(0,30,201,70);

checkButtonForCtrl.setContentAreaFilled(false);

checkButtonForCtrl.setFocusPainted(false);

add(checkButtonForCtrl);

JButton updateButton = new JButton("修改考勤信息");

updateButton.setBounds(0,99,201,70);

updateButton.setContentAreaFilled(false);

updateButton.setFocusPainted(false);

add(updateButton);

displayAreaForCtrl = new JTextArea();

displayAreaForCtrl.setBorder(BorderFactory.createLineBorder(Color.BLACK,1));

displayAreaForCtrl.setBounds(200,30,590,470);

displayAreaForCtrl.setEditable(false);

add(displayAreaForCtrl);

JTextField upName,upDate,upTo;

upName = new JTextField();

upName.addFocusListener(new JTextFieldHintListener(upName, "需更新用户名"));

upDate = new JTextField();

upDate.addFocusListener(new JTextFieldHintListener(upDate, "需更新日期"));

upTo = new JTextField();

upTo.addFocusListener(new JTextFieldHintListener(upTo, "修改值为"));

upName.setBounds(0,169,201,30);

upDate.setBounds(0,199,201,30);

upTo.setBounds(0,229,201,30);

add(upName);

add(upDate);

add(upTo);

checkButtonForCtrl.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

Program.fQueryAll();

}

});

updateButton.addActionListener(new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

Program.fUpdate(upName.getText(), upDate.getText(), upTo.getText());

}

});

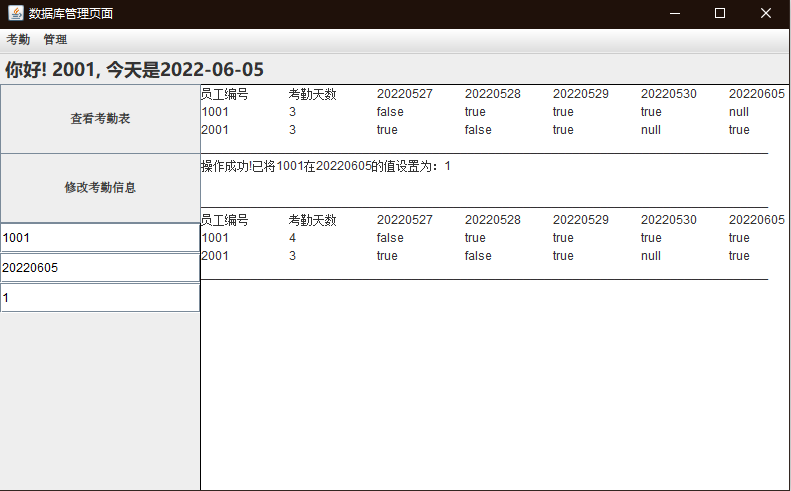
}

## 7.2 应用程序效果展示

下面展示普通用户--查看考勤--每日打卡--查看考勤流程。



此处展示管理员--查看考勤表--修改考勤信息--查看考勤表流程。



# 8 结论

此次数据库课程设计基本完成了课题要求，完成基本框架，对往后模块添加以及数据库维护具有较好操作性，实现了基本需求及功能。应用程序是对数据库系统的充分扩展，极大地完善了用户功能，使普通用户不需要直接操作数据库即可完成对数据库的修改。

本次课设花费4周的时间，而构思制图占了非常大的一部分。完成数据库时遇到了挺多的困难，例如在自动作业中如何以纯数字作为列名新建列等等，网络上对于部分问题根本没有答案，我只能自己慢慢摸索。之后制作Java应用程序也花费了很多的心思，学习了很多之前未曾知晓的新技术。

虽然说本次数据库课设还有很多可以完善的地方，但是无论完善与否，在现有技术下能做出来的事情，并不是说一定要全部体现，我相信此次数据库课设已经展现出了我自身基本的学科素养。

最后，虽然我特别“讨厌”T-SQL语言，因为它和我所学的无论是C还是Java，都有非常多不同的地方，错误提示也特别耐人寻思，但是我依然非常努力的在学习它，因为我知道，数据库在当今计算机中不可替代的地位。

# 9 参考文献

[1]m0\_37314542.使用PowerDesigner画图详细教程.[使用PowerDesigner](https://blog.csdn.net/m0_37314542/article/details/79823603)

[2]王珊、萨师煊.数据库系统概论（第五版）：高等教育出版社

[3]Fanzongshen.java连接数据库实现查询：[java连接数据库实现查询](https://blog.csdn.net/Fanzongshen/article/details/117296541)

[4]Microsoft.JDBC文档：[JDBC Driver for SQL Server | Microsoft Docs](https://docs.microsoft.com/zh-cn/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15)