《数据库系统》实验报告

实验题目	GaussDB 云数据库应用编程

一、实验目的

1. 掌握数据库建模工具 PowerDesigner 的使用，利用 PowerDesigner 工具建立具体需求的概念数据模型和物理数据模型，并转换成数据库对象。
2. 学习通过应用程序连接 GaussDB 云数据库，进行数据的操作。主要考查学生以数据库知识为主线，结合“程序设计技术”、“软件工程”等多门课程知识的综合应用。

二、实验项目内容

要求设计并编程实现一个小型的企业员工管理系统。

具体要求：

（1）对企业员工管理系统进行**需求分析**，使用 PowerDesigner **设计 E-R 模型**，详细描述实体的属性和实体之间的联系，消除不必要的冗余；实现 E-R 图向**关系模型**的转换，定义主键、外键，同时设计出系统**功能模块图**。

（2）基于 GaussDB 云数据库搭建企业员工管理系统数据库。

通过 java、python 等程序设计语言编程实现企业员工管理系统，重点是实现数据库的连接和各种 CRUD 操作；必须有对数据的增、删、改、查功能。界面自选，可以是命令行形式，web 页面形式或者 GUI。

三、实验过程或算法（源程序）

1. 需求分析

本实验要构建一个小型企业员工管理系统，实现数据库的连接和各种 CRUD 操作，以及增、删、改、查功能。通过设计数据库实现对企业员工的基本信息、职位、工资等数据进行存储、查询和管理。

数据库需要存储**员工**和**部门**的基本信息：员工信息有**员工编号、姓名、性别、部门、职位、工资、联系方式、工龄、办公室**等信息，部门信息有**部门编号、部门名称、部门职责、办公地点**等。一个员工只能在某个部门下面承担一个职位，一个部门有多个员工，员工的信息可以进行增删改查。**本实验所设计的企业员工管理系统是一个供企业内部管理人员操作的系统，可以增删查改员工的信息，但部门信息是提前设定好的，不能随意改动。**

2. ER 模型设计

根据需求分析可以提取出两个实体：**部门实体**和**员工实体**，他们的关系为**一对多**，即一个员工只能属于一个部门，但是一个部门可以有多个员工。

报告创建时间：2023.4.25

部门的主键为部门编号，员工的主键为员工编号，同时员工有一个外键为部门，用于约束员工表和部门表之间的关系。在 PowerDesign 中创建 Employee 和 Department 表如下：

Entity Properties - 员工 (员工)

General | Attributes | Identifiers | Subtypes | Notes | Rules

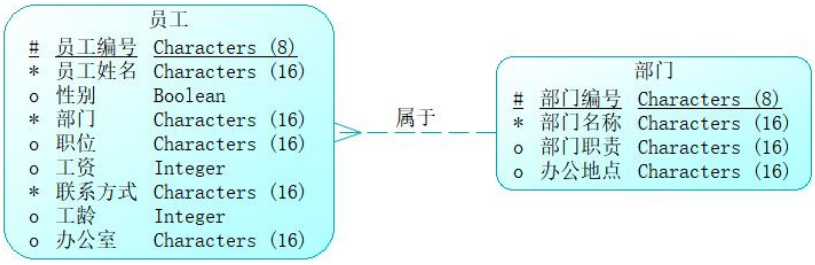
	Name	Code	Data Type	Lengt	Preci	M	P	D	Doma
1	员工编号	employee_id	Characters (8)	8		☑	☑	☑	<None
2	员工姓名	employee_name	Characters (16)	16		☑	☐	☑	<None
3	性别	sex	Boolean			☐	☐	☑	<None
→	部门	department	Characters (8)	8		☑	☐	☑	<None
5	职位	job	Characters (16)	16		☐	☐	☑	<None
6	工资	salary	Integer			☐	☐	☑	<None
7	联系方式	telephone	Characters (16)	16		☐	☐	☑	<None
8	工龄	seniority	Integer			☐	☐	☑	<None
9	办公室	office	Characters (16)	16		☐	☐	☑	<None

Entity Properties - 部门 (部门)

General | Attributes | Identifiers | Subtypes | Notes | Rules

	Name	Code	Data Type	Lengt	Preci	M	P	D	Doma
1	部门编号	department_id	Characters (8)	8		☑	☑	☑	<None
2	部门名称	department_name	Characters (16)	16		☑	☐	☑	<None
3	部门职责	duty	Characters (16)	16		☐	☐	☑	<None
4	办公地点	building	Characters (16)	16		☐	☐	☑	<None

绘制出 ER 图如下：



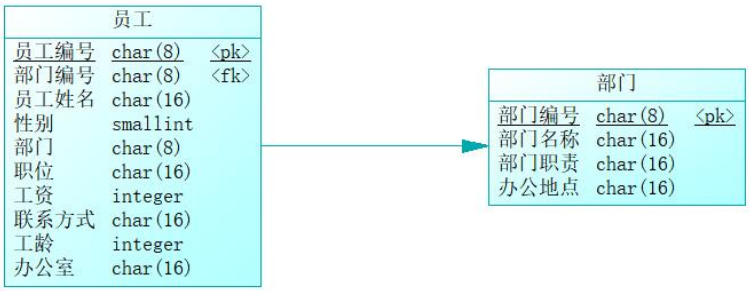
3. 关系模型

打开“Tools”，然后选择“Generate Physical Data Model”，可以将 ER 图转换为关系模型：

Output

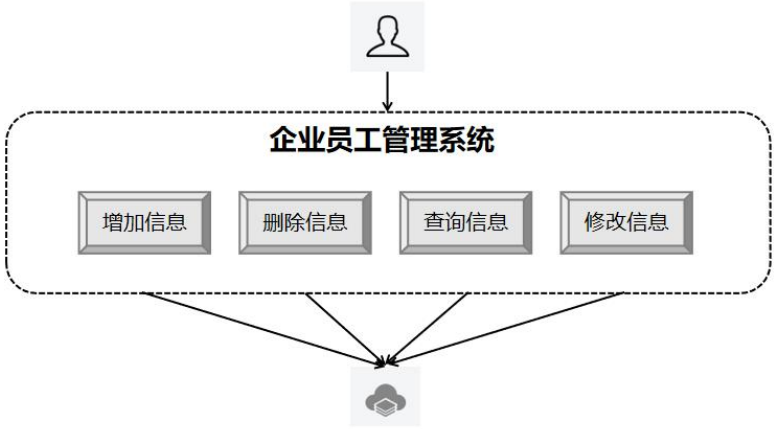
```
Generating model EnterpriseEmploreeMS...

 - Generating classes...
 - Generating relationships...
 - Migrating columns...
 - Generating diagrams and graphical symbols...

The model has been successfully generated...
Rebuild indexes in progress...
Rebuild indexes completed.
```

转换后的关系模型如下：



4. 功能模块图

本实验设计的数据库系统功能模块图如下：



5. 云数据库环境搭建

（1）使用 IAM 账号登录华为云 DAS，直接使用 Lab1 中创建的数据库实例即可。为本地 python 环境安装 psycopg2 包：pip install --trusted-host pypi.tuna.tsinghua.edu.cn -i https://pypi.tuna.tsinghua.edu.cn/simple/ -U psycopg2

```
PS C:\Users\Jackson1125> pip install --trusted-host pypi.tuna.tsinghua.edu.cn -i https://pypi.tuna.tsinghua.edu.cn/simple/ -U psycopg2
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple/
Collecting psycopg2
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/c9/37/f49a95fb00ca9f4678475284e60cb341aea84201ac45faa9f32ad88c02e8/psycopg2-2.9.6-cp311-cp311-win_amd64.whl (1.2 MB)
     ---------------------------------------- 1.2/1.2 MB 3.4 MB/s eta 0:00:00
Installing collected packages: psycopg2
Successfully installed psycopg2-2.9.6
```

6. 编程实现企业员工管理系统

（1）创建连接对象和指针对象

```
# 创建连接对象
conn=psycopg2.connect(database="db_2020_01",user="db_user2020_27",
password="db_user@123",host="116.205.157.173",port=8000)
# 创建指针对象
cur=conn.cursor()
```

（2）创建 Department 表和 Employee 表，因为程序可能会多次执行，后一次执行时可能出现表已经创建的情况，因此加入 IF NOT EXISTS 判断：

```
# 创建 Department 表和 Employee 表
cur.execute("CREATE TABLE IF NOT EXISTS Department(department_id VARCHAR(8) PRIMARY KEY,department_name VARCHAR(16) NOT NULL,duty VARCHAR(16),building VARCHAR(16));")
cur.execute("CREATE TABLE IF NOT EXISTS Employee(employee_id VARCHAR(8) PRIMARY KEY,employee_name VARCHAR(16) NOT NULL,sex BOOLEAN,department VARCHAR(8) NOT NULL,job VARCHAR(16),salary INTEGER,telephone VARCHAR(16) NOT NULL,seniority INTEGER,office VARCHAR(16),FOREIGN KEY (department) REFERENCES Department(department_id));")
```

创建成功的空表如下（course、student、sc、teacher 是 Lab1 建立的表，没有删除，因此还存储在实例中）：

（3）因为部门表的内容是指定的，不能改动，因此要提前向部门表中添加数据：

```
# 向部门表中添加数据
cur.execute("INSERT INTO Department (department_id,department_name,duty,building) VALUES(%s,%s,%s,%s)",('CQU001','行政部门','行政','主教 A'))
cur.execute("INSERT INTO Department (department_id,department_name,duty,building) VALUES(%s,%s,%s,%s)",('CQU002','财务部门','财政','主教 B'))
cur.execute("INSERT INTO Department (department_id,department_name,duty,building) VALUES(%s,%s,%s,%s)",('CQU003','技术部门','技术','主教 C'))
cur.execute("INSERT INTO Department (department_id,department_name,duty,building) VALUES(%s,%s,%s,%s)",('CQU004','客服部门','客服','主教 D'))
```

（4）进入企业员工管理系统主菜单，可以对员工信息进行增删查改：

```
# 企业员工管理系统主菜单
print("----------欢迎使用 CQU 企业员工管理系统----------")
print("    增加员工信息请输入 1")
print("    删除员工信息请输入 2")
print("    查询员工信息请输入 3")
print("    修改员工信息请输入 4")
print("    退出请输入 Q")
while True:
```

```
    print("------------------------------")
    choice = input("请根据需求选择序号进行相应操作：")
    if(choice == '1'):
        Creat_Employee(cur)
    elif(choice == '2'):
        Delet_Employee(cur)
    elif(choice == '3'):
        Query_Employee(cur)
    elif(choice == '4'):
        Update_Employee(cur)
    print("------------------------------")
    if(input("输入 Q 退出，其余任意键继续") == 'Q'):
        break
print("---------------成功退出 CQU 企业员工管理系统---------------")

# 关闭数据库连接，释放资源
conn.close()
```

（5）增加员工信息

```
# 增加员工信息
def Creat_Employee(cur):
    print("--------增加员工信息--------")
    employee_id = input("请输入员工编号：")
    employee_name = input("请输入员工姓名：")
    sex = input("请输入员工性别（未定按回车继续）：")
    department = input("请输入员工部门：")
    job = input("请输入员工职位（未定按回车继续）：")
    salary = input("请输入员工工资（未定按回车继续）：")
    telephone = input("请输入员工电话：")
    seniority = input("请输入员工工龄（未定按回车继续）：")
    office = input("请输入员工办公室（未定按回车继续）：")

    if(sex == ""):
        sex = "0"
    if(job == ""):
        job = "0"
    if(salary == ""):
        salary = "0"
    if(seniority == ""):
        seniority = "0"
    if(office == ""):
        office = "000"

    try:
        cur.execute("INSERT INTO Employee VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);",
                    (employee_id,employee_name,sex,department,job,salary,telephone,seni
```

```
ority,office))
        conn.commit()
        print("添加成功！")
    except Exception as e:
        print("添加失败，请检查！")
        print("An exception occurred:", e)
        print(traceback.format_exc())
    return
```

（6）删除员工信息

```
# 删除员工信息
def Delet_Employee(cur):
    print("--------删除员工信息--------")
    employee_id = input("请输入要删除的员工编号：")

    try:
        cur.execute("DELETE FROM Employee WHERE employee_id=%s;", (employee_id,))
        conn.commit()
        print("删除成功！")
    except Exception as e:
        print("删除失败，请检查！")
        print("An exception occurred:", e)
        print(traceback.format_exc())
    return
```

（7）查询员工信息

```
# 查询员工信息
def Query_Employee(cur):
    print("--------查询员工信息--------")
    print("1.按员工编号查询员工信息")
    print("2.按部门编号查询员工信息")
    print("3.查询企业所有员工信息")
    choice = input("请输入查询员工的查询方式（只输入序号）：")
    if(choice == '1'):
        employee_id = input("请输入待查询的员工编号：")
        try:
            cur.execute("SELECT * FROM Employee WHERE employee_id=%s;",
(employee_id,))
            print("查询成功！")
            results = cur.fetchall()
            table = PrettyTable()    # 调用 API 创建 PrettyTable 对象
            table.add_column('编号', [Employee[0] for Employee in results])
            table.add_column('姓名', [Employee[1] for Employee in results])
            table.add_column('性别', [Employee[2] for Employee in results])
            table.add_column('部门', [Employee[3] for Employee in results])
            table.add_column('职位', [Employee[4] for Employee in results])
```

```
            table.add_column('工资', [Employee[5] for Employee in results])
            table.add_column('电话', [Employee[6] for Employee in results])
            table.add_column('工龄', [Employee[7] for Employee in results])
            table.add_column('办公室', [Employee[8] for Employee in results])
            print(table)
        except Exception as e:
            print("查询失败，请检查！")
            print("An exception occurred: ", e)
            print(traceback.format_exc())
    elif(choice == '2'):
        department = input("请输入待查询的部门编号：")
        try:
            cur.execute("SELECT COUNT(*) FROM Employee WHERE department=%s",
(department,))
            print("查询成功！")
            results = cur.fetchall()
            print("%s 共有%d 个员工，他们的相关信息见下表" % (department,
results[0][0]))
            cur.execute("SELECT * FROM Employee WHERE department=%s", (department,))
            results = cur.fetchall()
            table = PrettyTable()
            table.add_column('编号', [Employee[0] for Employee in results])
            table.add_column('姓名', [Employee[1] for Employee in results])
            table.add_column('电话', [Employee[6] for Employee in results])
            print(table)
        except Exception as e:
            print("查询失败，请检查！")
            print("An exception occurred: ", e)
            print(traceback.format_exc())
    elif(choice == '3'):
        cur.execute("SELECT COUNT(*) FROM Employee")
        results = cur.fetchall()
        print("企业共有%d 个员工，他们的相关信息见下表" % (results[0][0]))
        cur.execute("SELECT * FROM Employee")
        results = cur.fetchall()
        table = PrettyTable()
        table.add_column('编号', [Employee[0] for Employee in results])
        table.add_column('姓名', [Employee[1] for Employee in results])
        table.add_column('部门', [Employee[3] for Employee in results])
        table.add_column('电话', [Employee[6] for Employee in results])
        print(table)
    else:
        print("查询失败，请检查！")
    return
```

（8）修改员工信息

```
# 修改员工信息
def Update_Employee(cur):
    print("--------修改员工信息--------")
    try:
        employee_id = input("请输入待修改的员工编号：")
        if(input("是否修改该员工姓名(Y/N)：") == 'Y'):
            employee_name = input("请输入该员工新姓名：")
            cur.execute("UPDATE Employee SET employee_name=%s WHERE
employee_id=%s;",(employee_name, employee_id))
            conn.commit()
        if(input("是否修改该员工性别(Y/N)：") == 'Y'):
            sex = input("请输入该员工新性别：")
            cur.execute("UPDATE Employee SET sex=%s WHERE employee_id=%s;",(sex,
employee_id))
            conn.commit()
        if(input("是否修改该员工部门(Y/N)：") == 'Y'):
            department = input("请输入该员工新部门：")
            try:
                cur.execute("UPDATE Employee SET department=%s WHERE
employee_id=%s;", (department, employee_id))
                conn.commit()
            except Exception:
                print("该部门不存在，请检查")
        if(input("是否修改该员工职位(Y/N)：") == 'Y'):
            job = input("请输入该员工新职位：")
            cur.execute("UPDATE Employee SET job=%s WHERE employee_id=%s;", (job,
employee_id))
            conn.commit()
        if(input("是否修改该员工工资(Y/N)：") == 'Y'):
            salary = input("请输入该员工新工资：")
            cur.execute("UPDATE Employee SET salary=%s WHERE employee_id=%s;",(salary,
employee_id))
            conn.commit()
        if(input("是否修改该员工电话(Y/N)：") == 'Y'):
            telephone = input("请输入该员工新电话：")
            cur.execute("UPDATE Employee SET telephone=%s WHERE
employee_id=%s;",(telephone, employee_id))
            conn.commit()
        if(input("是否修改该员工工龄(Y/N)：") == 'Y'):
            seniority = input("请输入该员工新工龄：")
            cur.execute("UPDATE Employee SET seniority=%s WHERE
employee_id=%s;",(seniority, employee_id))
            conn.commit()
        if(input("是否修改该员工办公室(Y/N)：") == 'Y'):
            office = input("请输入该员工新办公室：")
            cur.execute("UPDATE Employee SET office=%s WHERE employee_id=%s;",(office,
```

```
employee_id))
            conn.commit()
        print("修改成功！")
    except Exception as e:
        print("修改失败，请检查！")
        print("An exception occurred:", e)
        print(traceback.format_exc())
    return
```

四、实验结果及分析和（或）源程序调试过程

1. 实验结果

（1）运行文件，企业员工管理系统主页面如下：

```
----------------欢迎使用CQU企业员工管理系统----------------
    增加员工信息请输入1
    删除员工信息请输入2
    查询员工信息请输入3
    修改员工信息请输入4
    退出请输入Q
---------------------------------
请根据需求选择序号进行相应操作：
```

增删查改操作结束后，可以键入“Q”推出系统：

```
---------------------------------
输入Q退出，其余任意键继续Q
----------------成功退出CQU企业员工管理系统----------------
```

（2）增加员工信息

```
--------增加员工信息--------
请输入员工编号：20170001
请输入员工姓名：李老师
请输入员工性别（未定按回车继续）：1
请输入员工部门：CQU001
请输入员工职位（未定按回车继续）：行政主管
请输入员工工资（未定按回车继续）：6000
请输入员工电话：10086
请输入员工工龄（未定按回车继续）：5
请输入员工办公室（未定按回车继续）：201
添加成功！
------------------------------
输入Q退出，其余任意键继续
------------------------------
```

```
--------增加员工信息--------
请输入员工编号：20160031
请输入员工姓名：王老师
请输入员工性别（未定按回车继续）：0
请输入员工部门：CQU002
请输入员工职位（未定按回车继续）：财务主管
请输入员工工资（未定按回车继续）：5000
请输入员工电话：10088
请输入员工工龄（未定按回车继续）：3
请输入员工办公室（未定按回车继续）：
添加成功！
------------------------------
输入Q退出，其余任意键继续
------------------------------
```

```
--------增加员工信息--------
请输入员工编号：20200045
请输入员工姓名：张老师
请输入员工性别（未定按回车继续）：
请输入员工部门：CQU003
请输入员工职位（未定按回车继续）：技术工程师
请输入员工工资（未定按回车继续）：8000
请输入员工电话：10087
请输入员工工龄（未定按回车继续）：6
请输入员工办公室（未定按回车继续）：212
添加成功！
--------------------------------
输入Q退出，其余任意键继续
--------------------------------
```

```
--------增加员工信息--------
请输入员工编号：20190076
请输入员工姓名：孙老师
请输入员工性别（未定按回车继续）：0
请输入员工部门：CQU004
请输入员工职位（未定按回车继续）：后勤主任
请输入员工工资（未定按回车继续）：7000
请输入员工电话：10089
请输入员工工龄（未定按回车继续）：4
请输入员工办公室（未定按回车继续）：113
添加成功！
--------------------------------
输入Q退出，其余任意键继续
--------------------------------
```

```
请根据需求选择序号进行相应操作：1
--------增加员工信息--------
请输入员工编号：20180082
请输入员工姓名：李老师
请输入员工性别（未定按回车继续）：1
请输入员工部门：CQU001
请输入员工职位（未定按回车继续）：行政科员
请输入员工工资（未定按回车继续）：4000
请输入员工电话：10085
请输入员工工龄（未定按回车继续）：1
请输入员工办公室（未定按回车继续）：408
添加成功！
--------------------------------
```

执行操作后查看后台数据库中的 Employee 表，员工信息被成功加入：

以下是SELECT * FROM employee的执行结果集　　① 该表不可编辑。　　复制行　复制列　列设置

	employee_id	employee_name	sex	department	job	salary	telephone	seniority	office
1	20170001	李老师	t	CQU001	行政主管	6000	10086	5	201
2	20160031	王老师	f	CQU002	财务主管	5000	10088	3	000
3	20200045	张老师	f	CQU003	技术工程师	8000	10087	6	212
4	20190076	孙老师	f	CQU004	后勤主任	7000	10089	4	113
5	20180082	李老师	t	CQU001	行政科员	4000	10085	1	408

（3）删除员工信息

```
请根据需求选择序号进行相应操作：2
--------删除员工信息--------
请输入要删除的员工编号：20180082
删除成功！
--------------------------------
```

执行操作后查看后台数据库中的 Employee 表，员工信息被成功删除：

以下是SELECT * FROM employee的执行结果集　　① 该表不可编辑。　　复制行　复制列　列设置

	employee_id	employee_name	sex	department	job	salary	telephone	seniority	office
1	20170001	李老师	t	CQU001	行政主管	6000	10086	5	201
2	20160031	王老师	f	CQU002	财务主管	5000	10088	3	000
3	20200045	张老师	f	CQU003	技术工程师	8000	10087	6	212
4	20190076	孙老师	f	CQU004	后勤主任	7000	10089	4	113

（4）查询员工信息

按员工编号查询：

```
--------查询员工信息--------
1.按员工编号查询员工信息
2.按部门编号查询员工信息
3.查询企业所有员工信息
请输入查询员工的查询方式（只输入序号）：1
请输入待查询的员工编号：20190076
+----------+--------+-------+--------+----------+------+-------+------+--------+
|   编号   | 姓名   | 性别  | 部门   |   职位   | 工资 |  电话 | 工龄 | 办公室 |
+----------+--------+-------+--------+----------+------+-------+------+--------+
| 20190076 | 孙老师 | False | CQU004 | 后勤主任 | 7000 | 10089 |  4   |  113   |
+----------+--------+-------+--------+----------+------+-------+------+--------+
```

按部门编号查询：

```
请根据需求选择序号进行相应操作：3
--------查询员工信息--------
1.按员工编号查询员工信息
2.按部门编号查询员工信息
3.查询企业所有员工信息
请输入查询员工的查询方式（只输入序号）：2
请输入待查询的部门编号：CQU002
CQU002共有1个员工，他们的相关信息见下表
+----------+--------+-------+
|   编号   | 姓名   |  电话 |
+----------+--------+-------+
| 20160031 | 王老师 | 10088 |
+----------+--------+-------+
```

查询企业所有员工信息：

```
请根据需求选择序号进行相应操作：3
--------查询员工信息--------
1.按员工编号查询员工信息
2.按部门编号查询员工信息
3.查询企业所有员工信息
请输入查询员工的查询方式（只输入序号）：3
企业共有4个员工，他们的相关信息见下表
+----------+--------+--------+-------+
|   编号   | 姓名   |  部门  |  电话 |
+----------+--------+--------+-------+
| 20170001 | 李老师 | CQU001 | 10086 |
| 20160031 | 王老师 | CQU002 | 10088 |
| 20200045 | 张老师 | CQU003 | 10087 |
| 20190076 | 孙老师 | CQU004 | 10089 |
+----------+--------+--------+-------+
```

（5）修改员工信息

```
请根据需求选择序号进行相应操作：4
--------修改员工信息--------
请输入待修改的员工编号：20190076
是否修改该员工姓名(Y/N)：Y
请输入该员工新姓名：钟老师
是否修改该员工性别(Y/N)：N
是否修改该员工部门(Y/N)：N
是否修改该员工职位(Y/N)：Y
请输入该员工新职位：后勤人员
是否修改该员工工资(Y/N)：N
是否修改该员工电话(Y/N)：N
是否修改该员工工龄(Y/N)：N
是否修改该员工办公室(Y/N)：N
修改成功！
--------------------------------
```

执行操作后查看后台数据库中的 Employee 表，员工信息被成功修改：

以下是SELECT * FROM employee的执行结果集　　该表不可编辑。　　复制行　复制列　列设置

	employee_id	employee_name	sex	department	job	salary	telephone	seniority	office
1	20170001	李老师	t	CQU001	行政主管	6000	10086	5	201
2	20160031	王老师	f	CQU002	财务主管	5000	10088	3	000
3	20200045	张老师	f	CQU003	技术工程师	8000	10087	6	212
4	20190076	钟老师	f	CQU004	后勤人员	7000	10089	4	113

2. 调试过程

（1）错误：主程序中的建表语句在第二次执行时会报“表已存在”的错：

```
PS D:\PyWorkspace> python -u "d:\PyWorkspace\SQL.py"
Traceback (most recent call last):
  File "d:\PyWorkspace\SQL.py", line 172, in <module>
    cur.execute("CREATE TABLE Department(department_id VARCHAR(8) PRIMARY KEY,department_name VARCHAR(16) NO
T NULL,duty VARCHAR(16),building VARCHAR(16));")
psycopg2.errors.DuplicateTable: relation "department" already exists in schema "db_user2020_27"
DETAIL:  creating new table with existing name in the same schema
```

解决方法：将 CREATE TABLE Department 语句改为 CREATE TABLE IF NOT EXISTS Department 即可；

（2）错误：主程序中向部门表中添加数据的语句在第二次执行时会报“数据已存在”的错：

```
PS D:\PyWorkspace> python -u "d:\PyWorkspace\SQL.py"
Traceback (most recent call last):
  File "d:\PyWorkspace\SQL.py", line 176, in <module>
    cur.execute("INSERT INTO Department (department_id,department_name,duty,building) VALUES(%s,%s,%s,%s)",(
'CQU001','行政部门','行政','主教A'))
psycopg2.errors.UniqueViolation: duplicate key value violates unique constraint "department_pkey"
DETAIL:  Key (department_id)=(CQU001) already exists.
```

解决方法：由于 SQL 中添加数据语句不存在 IF NOT EXISTS 判断，实际应用中重复添加也会出现异常，因此设计系统时没有将其再处理，后续执行时注释掉即可；

（3）错误：员工信息管理系统中重复加入主键相同的数据项会报错：

```
--------增加员工信息--------
请输入员工编号：20190076
请输入员工姓名：钟老师
请输入员工性别（未定按回车继续）：
Traceback (most recent call last):
  File "d:\PyWorkspace\SQL.py", line 48, in Creat_Employee
    cur.execute("INSERT INTO Employee VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);",
psycopg2.errors.UniqueViolation: duplicate key value violates unique constraint "employee_pkey"
DETAIL:  Key (employee_id)=(20190076) already exists.
```

解决方法：加入 try…except…防错机制来捕捉异常；

（4）错误：之前的错误事项导致当前事务不能执行：

```
An exception occurred:  current transaction is aborted, commands ignored until end of transaction block, fir
stChar[Q]

Traceback (most recent call last):
  File "d:\PyWorkspace\SQL.py", line 48, in Creat_Employee
    cur.execute("INSERT INTO Employee VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s);",
psycopg2.errors.InFailedSqlTransaction: current transaction is aborted, commands ignored until end of transa
ction block, firstChar[Q]
```

解决方法：每执行完一个操作就进行提交，并加入防错机制捕捉异常；

3. 实验总结

本实验构建了一个小型企业员工管理系统，有助于对员工数量增多，信息量增大，以及员工部门分配，工资发放等问题实现现代化网络化管理，能够提高企业管理效率，提高准确度，节约企业成本，提高生产效率。该实验通过 E-R 图设计数据库的流程，以及 python 操作云数据库增删改查的方式，也进一步巩固了数据库的设计规则和设计原理，以及对数据库进行多种逻辑查询。

在后续设计数据库系统的过程中，我也不断调整先前的需求分析与实体提取，这让我更加深刻地认识到关系模型设计和减少冗余的重要性。为了让这个系统更加好用，我通过命令行设计了人机交互界面，将提示指令通过命令行传递给用户，并加入防错机制捕捉异常。为了使查询结果便于查看，我还调用了 PrettyTable API 创建表格，将查询信息规范化打印出来。