

## 《数据库原理》课程 实验报告

### 上机实验：Employees 数据库（2）

#### 一、实验目的

1. 通过上机练习掌握关系数据库编程技术。

#### 二、实验原理

1. 用 SQL 进行高级数据库操作。

#### 三、实验内容

5.19 触发器实验：实现自动审计日志。

修改部门名称时，将数据库用户登录名、修改时间、部门编号、部门名称的旧值、部门名称的新值记录到 departments\_copy\_log 表中。

(1) 准备

执行语句

```
CREATE TABLE employees (  
    emp_no      INT          NOT NULL,  
    birth_date  DATE          NOT NULL,  
    first_name  VARCHAR(14)   NOT NULL,  
    last_name   VARCHAR(16)   NOT NULL,  
    gender      CHAR(1)       NOT NULL,  
    hire_date   DATE          NOT NULL,  
    CONSTRAINT pk_employees PRIMARY KEY (emp_no)  
);
```

```
CREATE TABLE departments (  
    dept_no     CHAR(4)       NOT NULL,  
    dept_name   VARCHAR(40)   NOT NULL,  
    CONSTRAINT pk_departments PRIMARY KEY (dept_no)  
);
```

```
CREATE TABLE dept_emp (  
    emp_no      INT          NOT NULL,  
    dept_no     CHAR(4)       NOT NULL,  
    emp_start   DATE          NOT NULL,  
    emp_end     DATE          NOT NULL,  
    CONSTRAINT pk_dept_emp PRIMARY KEY (emp_no, dept_no)
```

学号： 3020205015 姓名： 石云天 日期： 2022 年 12 月 3 日 地点： 郑东图书馆

```
emp_no      INT          NOT NULL,
dept_no     CHAR(4)      NOT NULL,
from_date   DATE         NOT NULL,
to_date     DATE         NOT NULL,
CONSTRAINT pk_dept_emp PRIMARY KEY (emp_no, dept_no),
CONSTRAINT fk_dept_emp_employees FOREIGN KEY (emp_no)
REFERENCES employees(emp_no),
CONSTRAINT fk_dept_emp_departments FOREIGN KEY (dept_no)
REFERENCES departments(dept_no)
);
```

```
CREATE TABLE dept_manager (
dept_no     CHAR(4)      NOT NULL,
emp_no      INT          NOT NULL,
from_date   DATE         NOT NULL,
to_date     DATE         NOT NULL,
CONSTRAINT pk_dept_manager PRIMARY KEY (emp_no, dept_no),
CONSTRAINT fk_dept_manager_employees FOREIGN KEY (emp_no)
REFERENCES employees(emp_no),
CONSTRAINT fk_dept_manager_departments FOREIGN KEY (dept_no)
REFERENCES departments(dept_no)
);
```

```
CREATE TABLE titles (
emp_no      INT          NOT NULL,
title       VARCHAR(50)  NOT NULL,
from_date   DATE         NOT NULL,
to_date     DATE,
CONSTRAINT pk_titles PRIMARY KEY (emp_no, title, from_date),
CONSTRAINT fk_titles_employees FOREIGN KEY (emp_no)
REFERENCES employees (emp_no)
);
```

```
CREATE TABLE salaries (
```

学号: 3020205015 姓名: 石云天 日期: 2022 年 12 月 3 日 地点: 郑东图书馆

```
emp_no      INT          NOT NULL,
salary      INT          NOT NULL,
from_date   DATE         NOT NULL,
to_date     DATE         NOT NULL,
CONSTRAINT pk_salaries PRIMARY KEY (emp_no, from_date),
CONSTRAINT fk_salaries_employees FOREIGN KEY (emp_no)
REFERENCES employees(emp_no)
);
```

将提供的示例数据导入到已创建的表中。

数据文件说明:

data_employees.txt	员工数据
data_departments.txt	部门数据
data_dept_emp.txt	部门员工关系数据
data_dept_manager.txt	部门经理关系数据
data_salaries.txt	工资数据
data_titles.txt	职称数据

使用 PostgreSQL 提供的批量导入数据的语句 COPY...FROM...或使用 MySQL 提供的批量导入数据的语句 LOAD DATA INFILE。

(关于 COPY...FROM...语法解释, 请自己查询 MySQL 文档; 关于 LOAD DATA INFILE 语法解释, 请自己查询 MySQL 文档)

导入之后的结果:

employees 表	300024	行数据
departments 表	9	行数据
dept_emp 表	331603	行数据
dept_manager 表	24	行数据
titles 表	443308	行数据
salaries 表	2844047	行数据

```
CREATE TABLE departments_copy SELECT * FROM departments;
```

将部门表的所有行复制到新表 departments\_copy 中。

学号: 3020205015 姓名: 石云天 日期: 2022 年 12 月 3 日 地点: 郑东图书馆

结果如下图所示:

1 queries executed, 1 success, 0 errors, 0 warnings

查询: CREATE TABLE departments\_copy SELECT \* FROM departments

共 9 行受到影响

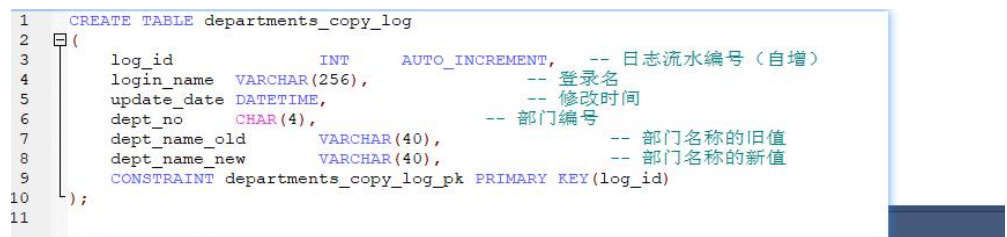
执行耗时 : 0.014 sec  
传送时间 : 1.007 sec  
总耗时 : 1.022 sec

## (2) 建立 departments\_copy\_log 表

执行语句

```
CREATE TABLE departments_copy_log
(
log_id      INT AUTO_INCREMENT,    -- 日志流水编号 (自增)
login_name  VARCHAR(256),          -- 登录名
update_date DATETIME,              -- 修改时间
dept_no     CHAR(4),               -- 部门编号
dept_name_old VARCHAR(40),          -- 部门名称的旧值
dept_name_new VARCHAR(40),          -- 部门名称的新值
CONSTRAINT departments_copy_log_pk PRIMARY KEY(log_id)
);
```

结果如下图所示:



The screenshot shows the SQL command to create the table and its execution details. The command is: `CREATE TABLE departments_copy_log ( log_id INT AUTO_INCREMENT, login_name VARCHAR(256), update_date DATETIME, dept_no CHAR(4), dept_name_old VARCHAR(40), dept_name_new VARCHAR(40), CONSTRAINT departments_copy_log_pk PRIMARY KEY(log_id) );`. The execution status is "1 queries executed, 1 success, 0 errors, 0 warnings". The execution time is 0.009 sec, transfer time is 1.003 sec, and total time is 1.013 sec.

1 queries executed, 1 success, 0 errors, 0 warnings

查询: CREATE TABLE departments\_copy\_log ( log\_id INT AUTO\_INCREMENT, login\_name VARCHAR(256), update\_date DATETIME, dept\_no CHAR(4), dept\_name\_old VARCHAR(40), dept\_name\_new VARCHAR(40), CONSTRAINT departments\_copy\_log\_pk PRIMARY KEY(log\_id) );

共 0 行受到影响

执行耗时 : 0.009 sec  
传送时间 : 1.003 sec  
总耗时 : 1.013 sec

## (3) 使用网络资源学习 MySQL 的触发器语法。

(4) 编写触发器, 实现修改部门名称时, 将数据库用户登录名、修改时间、部门编号、部门名称的旧值、部门名称的新值记录到 departments\_copy\_log 表中。

### 1.CREATE TRIGGER t\_insert\_dep

## 2.AFTER UPDATE ON departments\_copy

## 3.FOR EACH ROW

```
4.INSERT INTO departments_copy_log (login_name, update_date,
dept_no, dept_name_old, dept_name_new) VALUES(CURRENT_USER(), NOW(),
NEW.dept_no, OLD.dept_name,NEW. dept_name);
```

### (5) 执行 UPDATE 语句

```
UPDATE departments_copy
SET dept_name = CONCAT(dept_name, ' Dept')
WHERE dept_no = 'd005';
```

1 queries executed, 1 success, 0 errors, 0 warnings

查询: UPDATE departments\_copy SET dept\_name = CONCAT(dept\_name, ' Dept') WHERE dept\_no = 'd005'

共 1 行受到影响

执行耗时 : 0.002 sec  
传送时间 : 0 sec  
总耗时 : 0.003 sec

### (5) 执行查询语句

```
SELECT * FROM departments_copy;
```

查看 UPDATE 对于 departments\_copy 表的修改。

结果见下图:

dept no	dept_name
d001	Marketing
d002	Finance
d003	Human Resources
d004	Production
d005	Development Dept
d006	Quality Management
d007	Sales
d008	Research
d009	Customer Service

### (6) 执行查询语句

```
SELECT * FROM departments_copy_log;
```

查看触发器的作用, 是否实现了题目要求的审计日志的自动记录?

结果见下图:

log_id	login_name	update_date	dept_no	dept_name_old	dept_name_new
1	root@localhost	2022-11-27 11:06:53	d005	Development	Development Dept

5.20 建立财务部门（名称为 Finance）的员工视图 `finance_employees_view`，要求包括员工编号、员工姓名、性别、出生日期和入职日期。

```
CREATE VIEW finance_employees_view AS SELECT
emp_no,first_name,last_name,birth_date,hire_date FROM employees;
```

执行语句：

```
SELECT * FROM finance_employees_view LIMIT 10;
```

查询视图 `finance_employees_view` 的前 10 行，返回结果为：

emp_no	first_name	last_name	gender	birth_date	hire_date
10042	Magy	Stamatiou	F	1956-02-26	1993-03-21
10050	Yinghua	Dredge	M	1958-05-21	1990-12-25
10059	Alejandro	McAlpine	F	1953-09-19	1991-06-26
10080	Premal	Baek	M	1957-12-03	1985-11-19
10132	Ayakannu	Skrikant	M	1956-12-15	1994-10-30
10144	Marla	Brendel	M	1959-06-17	1985-10-14
10146	Chenyi	Syang	M	1959-01-12	1988-06-28
10147	Kazuhito	Encarnacion	M	1964-10-13	1986-08-21
10165	Miyeeon	Macedo	M	1960-06-16	1988-05-17
10173	Shrikanth	Mahmud	M	1962-10-28	1992-03-21

结果见下图：

emp_no	first_name	last_name	gender	birth_date	hire_date
10001	Georgi	Facello	M	1953-09-02	1986-06-26
10002	Bezalel	Simmel	F	1964-06-02	1985-11-21
10003	Parto	Bamford	M	1959-12-03	1986-08-28
10004	Chirstian	Koblick	M	1954-05-01	1986-12-01
10005	Kyoichi	Maliniak	M	1955-01-21	1989-09-12
10006	Anneke	Preusig	F	1953-04-20	1989-06-02
10007	Tzvetan	Zielinski	F	1957-05-23	1989-02-10
10008	Saniya	Kalloufi	M	1958-02-19	1994-09-15
10009	Sumant	Peac	F	1952-04-19	1985-02-18
10010	Duangkaew	Piveteau	F	1963-06-01	1989-08-24

5.21 索引的作用。

(1) 查询员工 “Peternela Anick” 的全部属性，记录查询执行时间。（提示：通过学习 MariaDB/MySQL 文档，使用 `SET profiling = 1;` 语句在查询之前打开记录时间功能，使用 `SET profiling = 0;` 语句在查询之后关闭记录时间功能，使用 `SHOW PROFILES;` 语句查看查询执行时间）

```
SET profiling = 1;
SELECT *
FROM employees
WHERE first_name = 'Peternela' AND last_name = 'Anick';
SET profiling = 0;
SHOW PROFILES;
```

结果见下图：

学号: 3020205015 姓名: 石云天 日期: 2022 年 12 月 3 日 地点: 郑东图书馆

emp_no	birth_date	first_name	last_name	gender	hire_date
234348	1961-07-16	Peternela	Anick	M	1991-01-15

Query_ID	Duration	Query
1	0.15586200	Select *from employeeswhere first_name = 'Peternela' and last_name

(2) 使用 **EXPLAIN** 语句查看第(1)步中查询的查询执行计划。(提示: 通过 MariaDB/MySQL 文档学习 **EXPLAIN** 语法)

**EXPLAIN SELECT \***

**FROM employees**

**WHERE first\_name = 'Peternela' AND last\_name = 'Anick';**

结果见下图所示:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	(N...	OK ALL	(NULL)	(NULL)	(NULL)	(NULL)	259607	1.00	Using where

(3) 在员工表 employees 的 first\_name 和 last\_name 属性列上建立索引。(提示: 通过 MariaDB/MySQL 文档学习 **CREATE INDEX** 语法)

**CREATE INDEX personindex**

**ON employees(first\_name, last\_name);**

(4) 再次执行第(1)步的查询, 记录查询执行时间(提示: 使用 **SHOW PROFILES;** 语句查看查询执行时间)。

从下图结果可见, 时间明显下降:

Query_ID	Duration	Query
1	0.15586200	Select *from employeeswhere first_name = 'Peternela' and last_name
2	0.00092350	Select *from employeeswhere first_name = 'Peternela' and last_name

(5) 使用 **EXPLAIN** 语句查看第(1)步中查询的查询执行计划, 与第(2)步给出的查询执行计划进行对比。

**EXPLAIN**

**SELECT\***

**FROM employees**

**WHERE first\_name = 'Peternela' AND last\_name = 'Anick';**

结果见下图所示:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	employees	(N...	OK ref	personindex	personindex	406	const,const	1	100.00	(NULL)

(6) 删除在 first\_name 和 last\_name 属性列上建立索引(提示: 用 **DROP INDEX** 语句)。

**DROP INDEX personindex ON employees;**

## 5.22 索引与键。

(1) 执行查询:

**SELECT d.dept\_no, d.dept\_name, e.emp\_no, e.first\_name, e.last\_name,**



学号: 3020205015 姓名: 石云天 日期: 2022 年 12 月 3 日 地点: 郑东图书馆

```
s.salary
FROM departments AS d
    INNER JOIN dept_emp AS de ON d.dept_no=de.dept_no
    INNER JOIN employees AS e ON de.emp_no=e.emp_no
    INNER JOIN salaries AS s ON e.emp_no=s.emp_no
WHERE e.first_name='Peternela' AND e.last_name='Anick';
```

记录查询执行时间。

dept no	dept name	emp no	first name	last name	salary
d009	Customer Service	234348	Peternela	Anick	40000
d009	Customer Service	234348	Peternela	Anick	42701
d009	Customer Service	234348	Peternela	Anick	44499
d009	Customer Service	234348	Peternela	Anick	47187
d009	Customer Service	234348	Peternela	Anick	48581
d009	Customer Service	234348	Peternela	Anick	50241
d009	Customer Service	234348	Peternela	Anick	50788
d009	Customer Service	234348	Peternela	Anick	54208
d009	Customer Service	234348	Peternela	Anick	58528
d009	Customer Service	234348	Peternela	Anick	62439
d009	Customer Service	234348	Peternela	Anick	66088
d009	Customer Service	234348	Peternela	Anick	70435

(2) 在员工表 employees 的 first\_name 和 last\_name 属性列上建立索引。再次执行第(1)步的查询，记录查询执行时间。

```
CREATE INDEX emp
ON employees(first_name,last_name);
```

结果见下图（第六行与第八行对比）：

6	0.80865425	create index emp on employees(first_name,last_name)
7	0.00018725	emp_no=s.emp_no WHERE e.first_name='Peternela' AND e.last_name='Ani
8	0.00240300	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,

(3) 执行下列语句，删除外键：

```
-- drop foreign keys
ALTER TABLE salaries DROP FOREIGN KEY fk_salaries_employees;
ALTER TABLE titles DROP FOREIGN KEY fk_titles_employees;
ALTER TABLE dept_emp DROP FOREIGN KEY fk_dept_emp_employees;
ALTER TABLE dept_emp DROP FOREIGN KEY fk_dept_emp_departments;
ALTER TABLE dept_manager DROP FOREIGN KEY fk_dept_manager_employees;
ALTER TABLE dept_manager DROP FOREIGN KEY fk_dept_manager_departments;
```

再次执行第(1)步的查询，记录查询执行时间。

18	0.00093500	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
----	------------	---------------------------------------------------------------------

(4) 执行下列语句，删除主键：

```
-- drop primary keys
ALTER TABLE employees DROP PRIMARY KEY;
ALTER TABLE departments DROP PRIMARY KEY;
ALTER TABLE dept_emp DROP PRIMARY KEY;
ALTER TABLE salaries DROP PRIMARY KEY;
```

再次执行第(1)步的查询，记录查询执行时间。



23	0.00068575	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
----	------------	---------------------------------------------------------------------

(5) 执行下列语句, 删除员工表 employees 的 first\_name 和 last\_name 属性列上建立的索引:

```
-- drop index
```

```
DROP INDEX employees_name_index ON employees;
```

再次执行第(1)步的查询, 记录查询执行时间。

26	0.00144350	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
27	0.00072600	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,

(6) 执行下列语句恢复主键:

```
-- add primary keys
```

```
ALTER TABLE employees ADD CONSTRAINT pk_employees PRIMARY KEY(emp_no);
```

```
ALTER TABLE departments ADD CONSTRAINT pk_departments PRIMARY  
KEY(dept_no);
```

```
ALTER TABLE dept_emp ADD CONSTRAINT pk_dept_emp PRIMARY KEY(emp_no,  
dept_no);
```

```
ALTER TABLE salaries ADD CONSTRAINT pk_salaries PRIMARY KEY(emp_no,  
from_date);
```

再次执行第(1)步的查询, 记录查询执行时间。

32	0.00137475	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
----	------------	---------------------------------------------------------------------

(7) 执行下列语句恢复外键:

```
-- add foreign keys
```

```
ALTER TABLE salaries ADD CONSTRAINT fk_salaries_employees FOREIGN KEY  
(emp_no) REFERENCES employees(emp_no);
```

```
ALTER TABLE titles ADD CONSTRAINT fk_titles_employees FOREIGN KEY  
(emp_no) REFERENCES employees (emp_no);
```

```
ALTER TABLE dept_emp ADD CONSTRAINT fk_dept_emp_employees FOREIGN KEY  
(emp_no) REFERENCES employees(emp_no);
```

```
ALTER TABLE dept_emp ADD CONSTRAINT fk_dept_emp_departments FOREIGN KEY  
(dept_no) REFERENCES departments(dept_no);
```

```
ALTER TABLE dept_manager ADD CONSTRAINT fk_dept_manager_employees  
FOREIGN KEY (emp_no) REFERENCES employees(emp_no);
```

```
ALTER TABLE dept_manager ADD CONSTRAINT fk_dept_manager_departments  
FOREIGN KEY (dept_no) REFERENCES departments(dept_no);
```

再次执行第(1)步的查询, 记录查询执行时间。

39	0.00313875	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
----	------------	---------------------------------------------------------------------

(8) 在员工表 employees 的 first\_name 和 last\_name 属性列上建立索引。再次执行第(1)步的查询, 记录查询执行时间。

```
CREATE INDEX emp
```

```
ON employees(first_name,last_name);
```

40	0.83738200	CREATE INDEX empON employees(first_name,last_name)
41	0.00220500	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
42	0.00070425	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,

### 5.23 存储过程。

(1) 通过 MariaDB/MySQL 文档，学习 `CREATE PROCEDURE` 语句的语法。

(2) 创建存储过程 `calc_avg_salary_for_emp_no`,

其有两个参数：`emp_no_in` 输入型，指定员工编号；

`avg` 输出型，保存工资平均值。

该存储过程的功能是：计算编号为 `emp_no_in` 的员工在工资表 `salaries` 中的工资数额的平均值，并将该平均值保存到 `@avg` 中。

```
CREATE PROCEDURE clac_avg_aslary_for_emp_no
(IN emp_no INT(11),
OUT avge INT(11))
SELECT AVG(salary) INTO avge
FROM salaries
WHERE salaries.`emp_no` = emp_no;
```

(3) 执行存储过程 `calc_avg_salary_for_emp_no`，执行下列语句，调用存储过程并输出结果：

```
CALL calc_avg_salary_for_emp_no(10002, @avg_salary);
SELECT @avg_salary;
```

执行结果为：

```
@avg_salary
68854.5
```

```
SET @avge = 0;
CALL clac_avg_aslary_for_emp_no(10002,@avge);
SELECT @avge;
```

(4) 使用 `DROP PROCEDURE` 语句，删除存储过程 `calc_avg_salary_for_emp_no`  
`DROP PROCEDURE calc_vag_salary_for_emp_no;`

### 5.24 存储过程（函数、分支）。

（选做）

(1) 通过 MariaDB/MySQL 文档，学习 `CREATE FUNCTION` 语句的语法。创建存储过程 `is_manager`,

其有一个参数：`emp_no_in`，指定员工编号；

返回值，`BOOL` 类型，返回 1 表示 `emp_no_in` 编号的员工是经理（`manager`），返回 0 表示 `emp_no_in` 编号的员工不是经理。

该函数的功能是：查询编号为 `emp_no_in` 的员工是否为经理（在 `dept_manager`

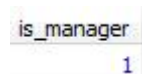
表中查询)。

(提示：使用 IF...ELSE 语句实现分支判断。)

```
DELIMITER &&
CREATE FUNCTION is_manager(emp_no_in INT(11))
RETURNS TINYINT UNSIGNED
BEGIN
    DECLARE str TINYINT UNSIGNED;
    SELECT COUNT(*)
    INTO str
    FROM dept_manager
    WHERE dept_manager.emp_no = emp_no_in;
    RETURN str;
END &&
DELIMITER;
```

(2) 执行存储过程 `is_manager`，执行下列语句，调用函数并输出返回结果：  
`SELECT is_manager(110022) AS is_manager;`

执行结果为：




is_manager
1

```
SELECT is_manager(110022) AS is_manager;
```

(3) 执行存储过程 `is_manager`，执行下列语句，调用函数并输出返回结果：  
`SELECT is_manager(100002) AS is_manager;`

执行结果为：



is_manager
0

```
SELECT is_manager(100002) AS is_manager;
```

(4) 使用 `DROP FUNCTION` 语句，删除函数 `is_manager`  
`DROP FUNCTION is_manager;`

## 5.25 存储过程（游标、循环）。

(选做)

(1) 创建存储过程 `calc_avg_and_var_salary_for_emp_no`，  
其有三个参数：`emp_no_in`，输入型，指定员工编号；

`avg`，REAL 类型，输出型，返回 `emp_no_in` 指定的员工的工资数额的平均值。

`var`，REAL 类型，输出型，返回 `emp_no_in` 指定的员工的工

资数额的方差。

该函数的功能是：计算编号为 emp\_no\_in 的员工的工资数额的平均值和方差。

（提示：使用游标获取指定 emp\_no\_in 的员工的每条工资记录中的工资数额。使用 WHILE 语句编写循环。）

```
CREATE PROCEDURE calc_avg_and_var_salary_for_emp_no
(IN emp_no_in INT(11),
OUT avge REAL ,
OUT var REAL)
BEGIN
DECLARE sume INTEGER DEFAULT 0;
DECLARE s INT DEFAULT 0;
DECLARE salcount INTEGER DEFAULT 0;
DECLARE newsal INTEGER DEFAULT 0.0;
DECLARE EmployeesCurosr CURSOR FOR
    SELECT salary FROM salaries WHERE salaries.emp_no = emp_no_in;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET s=1;
OPEN EmployeesCurosr;
SET avge = 0.0;
SET var = 0.0; .
    FETCH EmployeesCurosr INTO newsal;
    WHILE s<> 1 DO
        SET SUM = SUM + newsal;
        SET salcount = salcount + 1;
        FETCH EmployeesCurosr INTO newsal;
    END WHILE;
```

(3) 执行存储过程 calc\_avg\_and\_var\_salary\_for\_emp\_no，执行下列语句，调用存储过程并输出结果：

```
CALL calc_avg_and_var_salary_for_emp_no(10002 , @avg_salary,
@var_salary);
SELECT @avg_salary AS avg_salary, @var_salary AS var_salary;
```

执行结果为：

avg_salary	var_salary
68854.5	7165175.583333015

```
CALL calc_avg_and_var_salary_for_emp_no(10002 , @avg_salary,
@var_salary);
SELECT @avg_salary AS avg_salary, @var_salary AS var_salary;
```

(4) 使用 DROP PROCEDURE 语句，删除存储过程 calc\_avg\_and\_var\_salary\_for\_emp\_no

学号： 3020205015 姓名： 石云天 日期： 2022 年 12 月 3 日 地点： 郑东图书馆

**DROP PROCEDURE calc\_avg\_and\_var\_salary\_for\_emp\_no;**

#### 四、实验总结

(300 字以上)

通过这次试验，我深切体会到了数据库在数据管理方面的强大功能，同时也发现了自己对其的掌握不够不够深入透彻，在实验过程中需要完成某些特定目标时，我不能很快的想出其对应的指令和语法，需要课下继续不断练习以熟能生巧，还可以多查阅一些资料以开阔自己的思路。在实验过程中，通过不断尝试使用图形化界面的不同功能，对其使用方法掌握的更加透彻。在整个实验的过程中需要认真阅读实验要求，按照要求一步一步慢慢操作，避免自己盲目实验而出现错误，甚至南辕北辙。通过对数据的批量装载，我掌握了 mysql 中 LOAD DATA INFILE 语句的用法，它可以帮助我们将巨量的数据轻松无损的导入进数据库中。在利用数据库完成实验要求的查询操作时，我进一步加深了对数据库语言的掌握，可以利用视图、索引等帮助我们优化查询，缩短查询所需时间。通过对存储过程中函数、分支、游标等环节的练习，可以更好地把握整个存储过程，加深相关知识的理解。在实验过程中，对于一些特定的语法如：JOIN...ON 连接、NOT EXISTS 连接以及子查询的方法也有了更加深刻的理解，针对一些复杂的问题，可以采取分而治之的观点进行看待，利用数据库中不同操作逐个击破，再统一整合解决问题。对于实验而言，课内所学的知识点是重中之重，只有完全理解掌握了课内所学的知识点，把每个细节都弄明白，才能更加出色的完成整个实验任务，事半功倍，相辅相成。