学号：3012216083　姓名：王雨朦　日期：2014 年 12 月 24 日　地点：综合实验楼 304

《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。
2. 用 JDBC 编程连接数据库并进行操作。

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 departments_copy SELECT * FROM departments;

将部门表的所有行复制到新表 departments_copy 中。

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

(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)
);
```

Mysql 版本：

```
CREATE TABLE departments_copy_log
(
log_id      int auto_increment,   -- IDENTITY 属性，自增长
```

```
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)
);
```

(3) 使用网络资源学习 MariaDB/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。

```
update departments set dept_name = 'Marketingbbb'
 where dept_no ='d001';
CREATE TRIGGER t_trigger
after update on departments_copy
FOR EACH ROW
insert into
departments_copy_log(login_name,update_date,dept_no,dept_name_o
ld,dept_name_new)
  values(current_user,current_timestamp,new.dept_no,old.dept_na
me,new.dept_name);
```

(5) 执行 UPDATE 语句

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

(5) 执行查询语句

```
SELECT * FROM departments_copy;
```

查看 UPDATE 对于 departments_copy 表的修改。

d005	Developement Dept

(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？

实现了自动记录。如下：

log_id	login_name	update_date	dept_no	dept_name_old	dept_name_new
1	root@localhost	2014-12-24 11:06:33	d001	Marketingaaa	Marketingbbb
2	root@localhost	2015-01-02 10:13:31	d001	Marketingbbb	Marketingbnnbb
3	root@localhost	2015-01-02 11:18:34	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,gender,birth_date,hire_date
from departments_copy natural join dept_emp natural join employees
where dept_name='Finance';
```

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	Miyeon	Macedo	M	1960-06-16	1988-05-17
10173	Shrikanth	Mahmud	M	1962-10-28	1992-03-21

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）

employees (6×1)

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

结果 #1 (3×3)

Query_ID	Duration	Query
1	0.00001950	select *from employeeswhere first_name ='Peternela'an...
2	0.00001900	select *from employeeswhere first_name ='Peternela'an...
3	0.00001875	select *from employeeswhere first_name ='Peternela'an...

(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）

结果 #1 (10×1)

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	employees	ALL	(NULL)	(NULL)	(NULL)	(NULL)	300057	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 empIndex 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）。

4	0.00000775	-- EXPLAINselect *from employeeswhere first_name ='Pe...

(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。

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	employees	ref	empIndex	empIndex	102	const,const	1	Using where

对比得出，两个执行计划在“键”上存在差别。

(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 empIndex on employees;
```

5.22 索引与键。

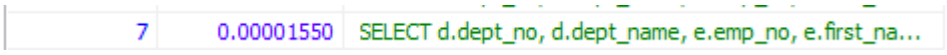
(1) 执行查询：

```
SELECT d.dept_no, d.dept_name, e.emp_no, e.first_name, e.last_name,
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。

6	0.00001425	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_na...

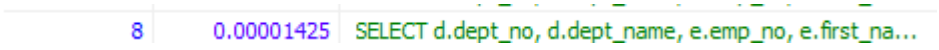
(2) 在员工表 employees 的 first_name 和 last_name 属性列上建立索引。再次执行第(1)步的查询，记录查询执行时间。



(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。



(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。

(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 empIndex ON employees;
```

再次执行第(1)步的查询，记录查询执行时间。

(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。

(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。

12	0.00001600	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_na...

(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。

13	0.00001650	SELECT d.dept_no, d.dept_name, e.emp_no, e.first_na...

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。

```
DELIMITER //
CREATE PROCEDURE calc_avg_salary_for_emp_no(
in emp_no char(50),
out avg_salary real
)
begin
select avg(salary)
into avg_salary
from salaries
where emp_no = emp_no;
end
//
```

(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

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

```
drop procedure calc_avg_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。创建存储函数 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 char(50)) returns boolean
begin
  declare x int(5);
  select count(*) into x
  from dept_manager
  where dept_manager.emp_no = emp_no;
  if (x>0)
  then
    return true;
  else
    return false;
  end if;
end
//
```

(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

(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

(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。）

(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

(4) 使用 DROP PROCEDURE 语句，

删除存储过程 `calc_avg_and_var_salary_for_emp_no`

5.26 JDBC 编程。（选做）

编写 Java 程序，通过 JDBC 连接 MariaDB/MySQL 中的 Employees 数据库，实现对 departments 表中记录的查询（SELECT）、添加（INSERT）、修改（UPDATE）和删除（DELETE）。

(1) JDBC驱动程序：MariaDB Client Library for Java Applications中的mariadb-java-client-1.1.7.jar（课程主页上下载）。

(2) Eclipse项目JDBCEmployees（课程主页上下载）。

(3) 编写代码，实现DepartmentsDAO类中的TODO部分。

四、实验总结

通过此次试验更深刻的理解了数据库的触发器、视图、索引等。

触发器是在对表进行增、删、改时，自动执行的存储过程，它是在原表数据行已经修改完成后再触发。所以，触发器是在约束检查之后才执行。而通过本次实验对 UPDATE 的触发操作，能够更加深刻的理解触发器对数据库进行高级约束的重要作用。

索引是一个具有两面性的数据库功能，对于每一个数据库，是否需要创建索引，要创建多少个索引都是值得我们思考与分析的。创建索引可以大大提高系统的性能，但是创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。此外，索引是建立在数据库表中的某些列的上面。因此，在创建索引的时候，应该仔细考虑在哪些列上可以创建索引，在哪些列上不能创建索引。最后，由修改性能和检索性能的矛盾性来看，增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，我们在建立索引时需要仔细权衡。

由实验 5.22，可得出索引与键有一定的关系。主键和索引都是键，主键是唯一索引。而一个表中可以有多个唯一性索引，但只能有一个主键。当存在主键时，索引对数据库操作时间的影响不大。而主键不存在时，建立索引对数据库的意见就十分明显了。

而通过第一道选做题，我也体会到了用 SQL 语句中函数与分支在数据库存储过程中的魅力。