

数据库系统实验教程

物联网与泛在智能研究中心

王扬帆

说明

- 实验都是**课后完成**，实验课检查实验结果，实验报告在实验课一周后命名好发送给本班负责人，班级负责人收齐发送到指定邮箱

18232948487@163.com

- 实验1：**第6周周三**实验课验收，截止日期**第7周周五中午12点**上交报告
 - 不用自己生成表内数据，直接使用“lab1_data”中给定的数据
- 实验2：**第6周周五**实验课验收，截止日期**第7周周五中午12点**上交报告
- 实验3/project2: **二选一**，**第10周**实验课两次课都可以验收，截止日期**第11周周二中午12点**上交代码及报告
 - “实验3源码”中有本次实验中所需要的代码

每个同学实验报告上交pdf，命名“实验x-学号-姓名.pdf”

班级代表汇总上交文件压缩包zip，命名“实验x-班级号.zip”

MySQL 安装

- <https://dev.mysql.com/downloads/windows/installer/8.0.html> 下载
Windows (x86, 32-bit), MSI Installer

1

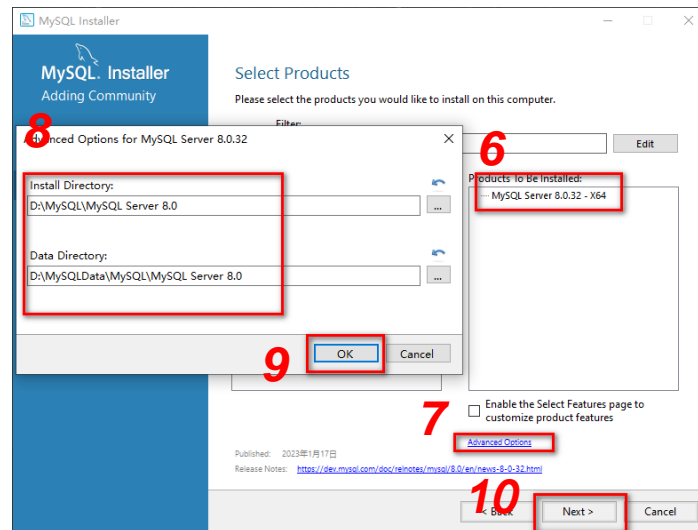
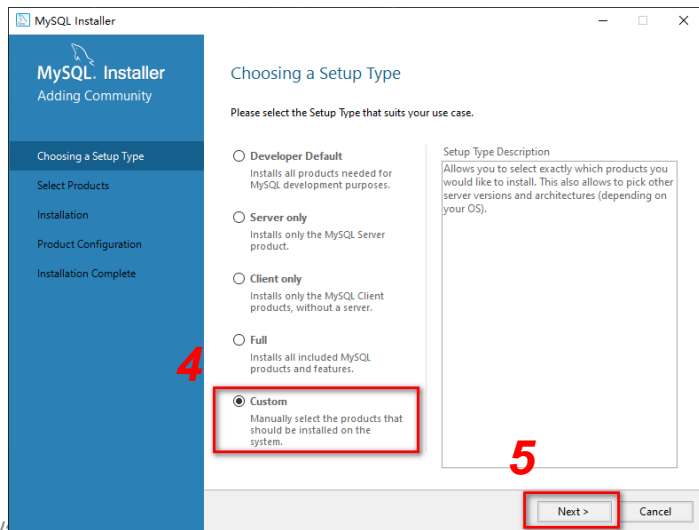
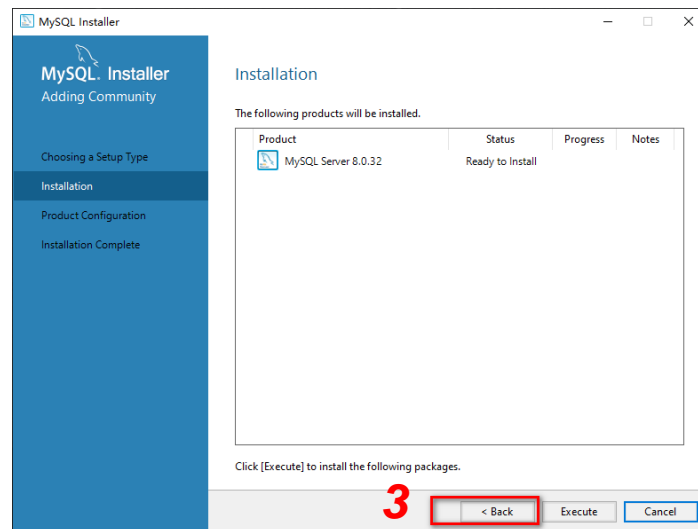
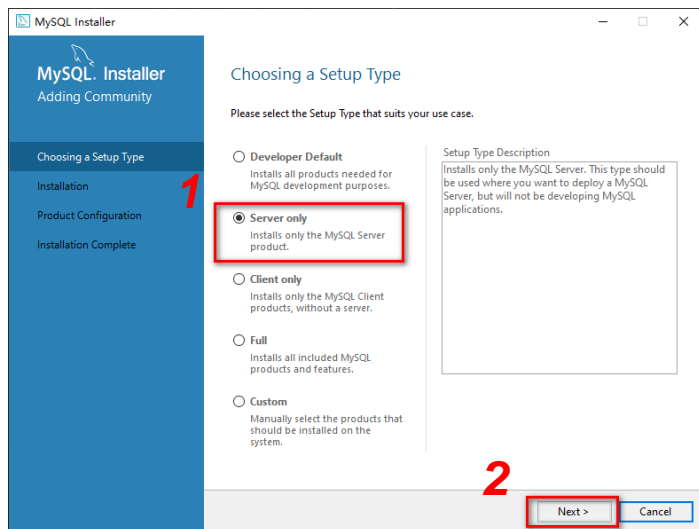
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.32.0.msi)	8.0.32	2.4M	Download
MD5: 0f882590f8338adc614e9dc5cb00ca0b Signature			
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.32.0.msi)	8.0.32	437.3M	Download
MD5: a29b5817cb2c7bc0e0b97e897c2591f Signature			

! We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

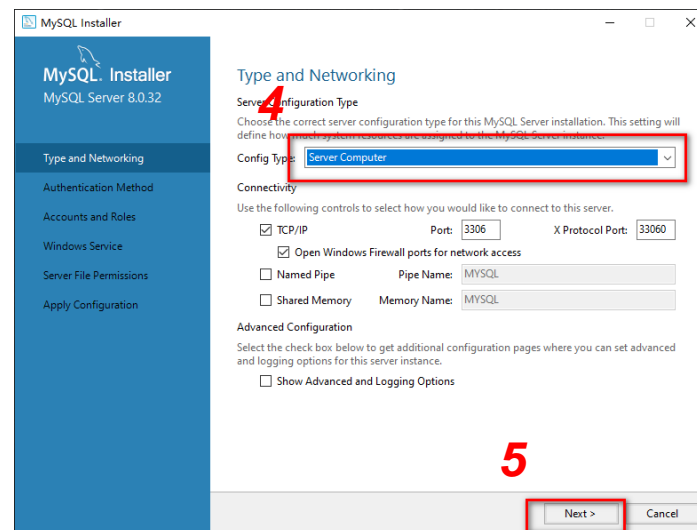
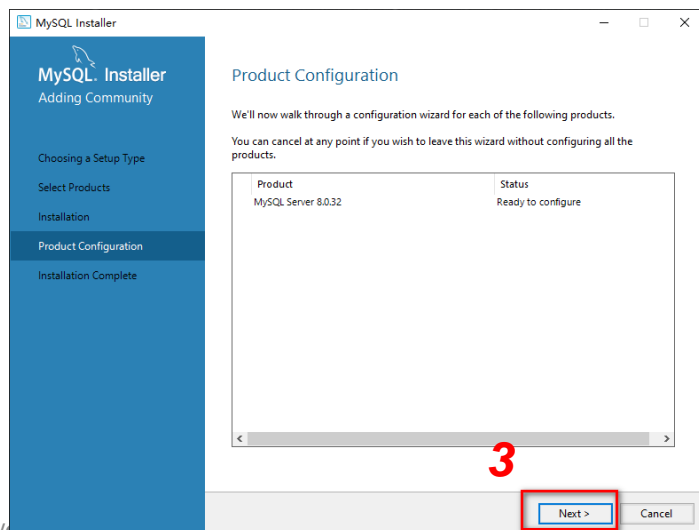
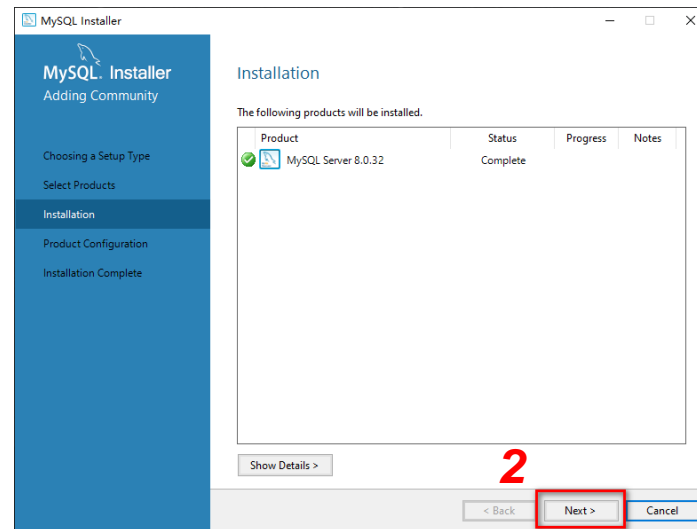
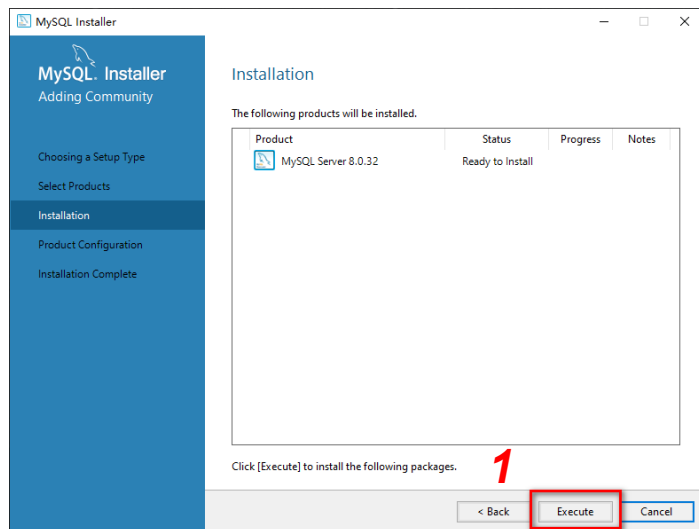
2

[No thanks, just start my download.](#)

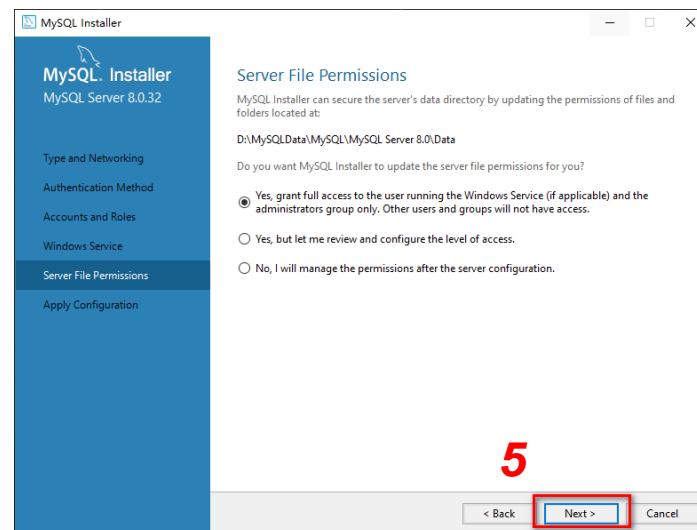
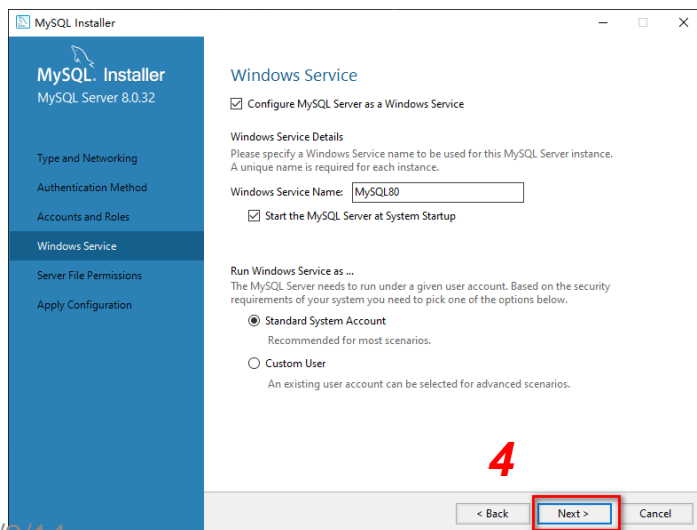
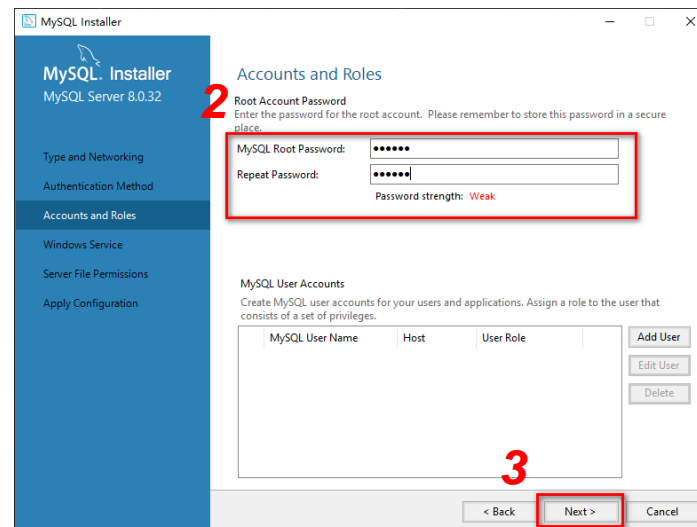
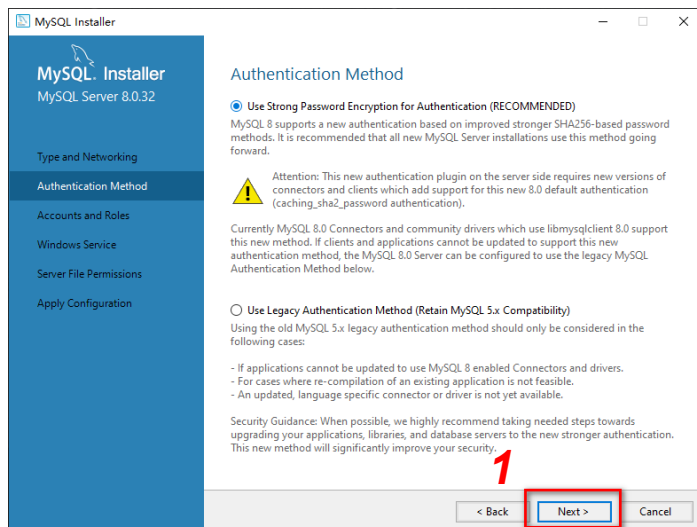
MySQL 安装



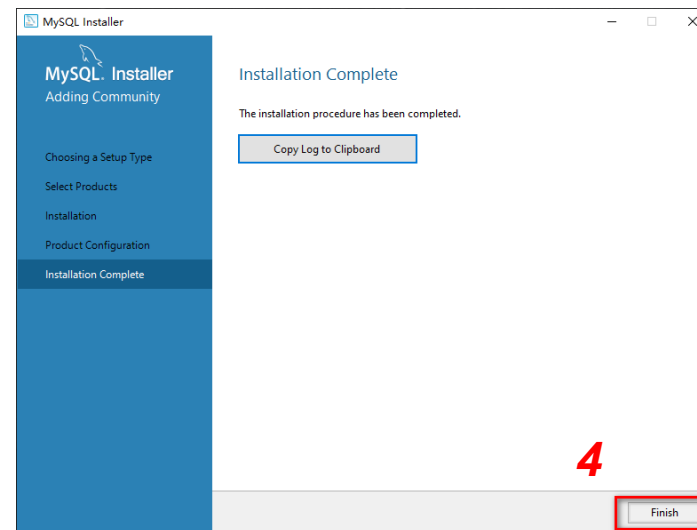
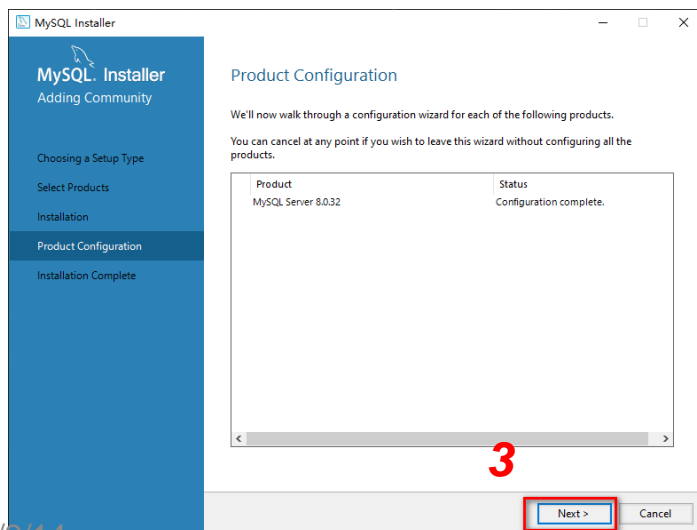
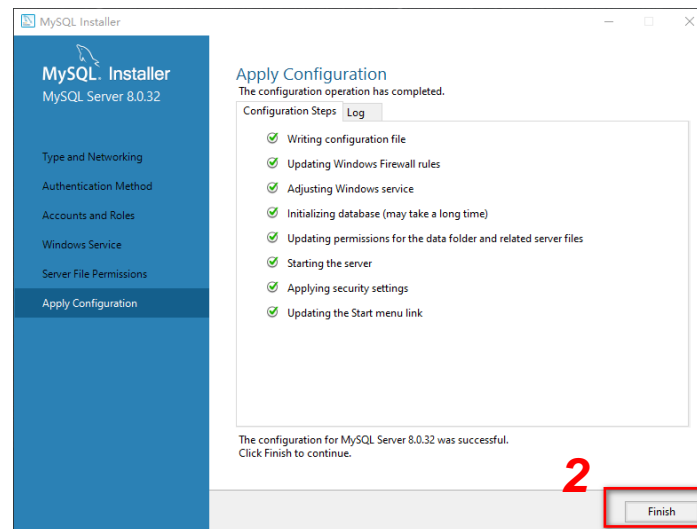
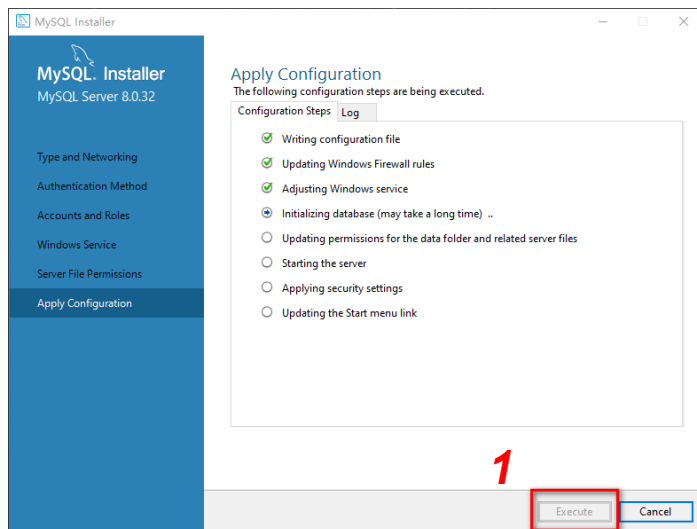
MySQL 安装



MySQL 安装

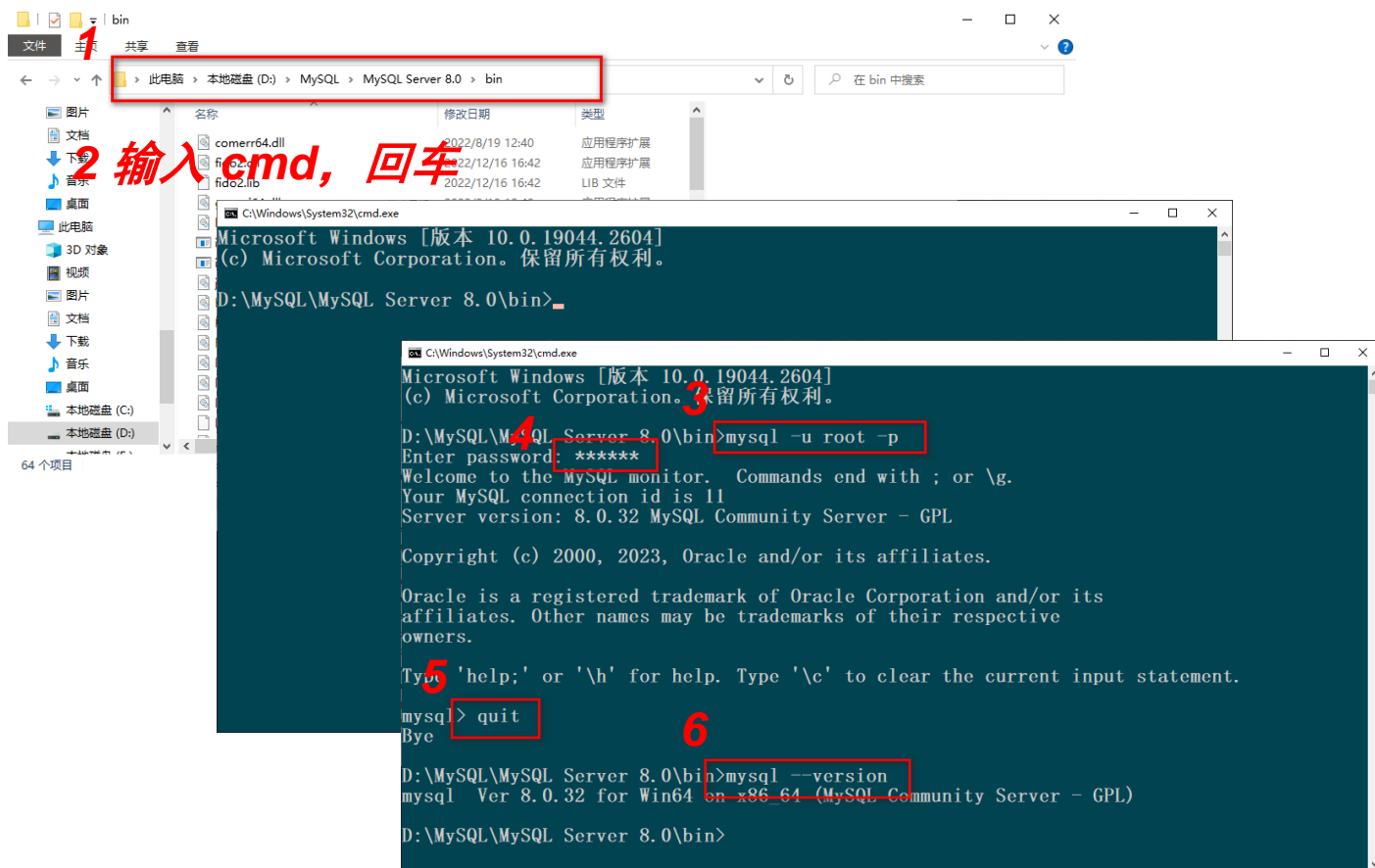


MySQL 安装



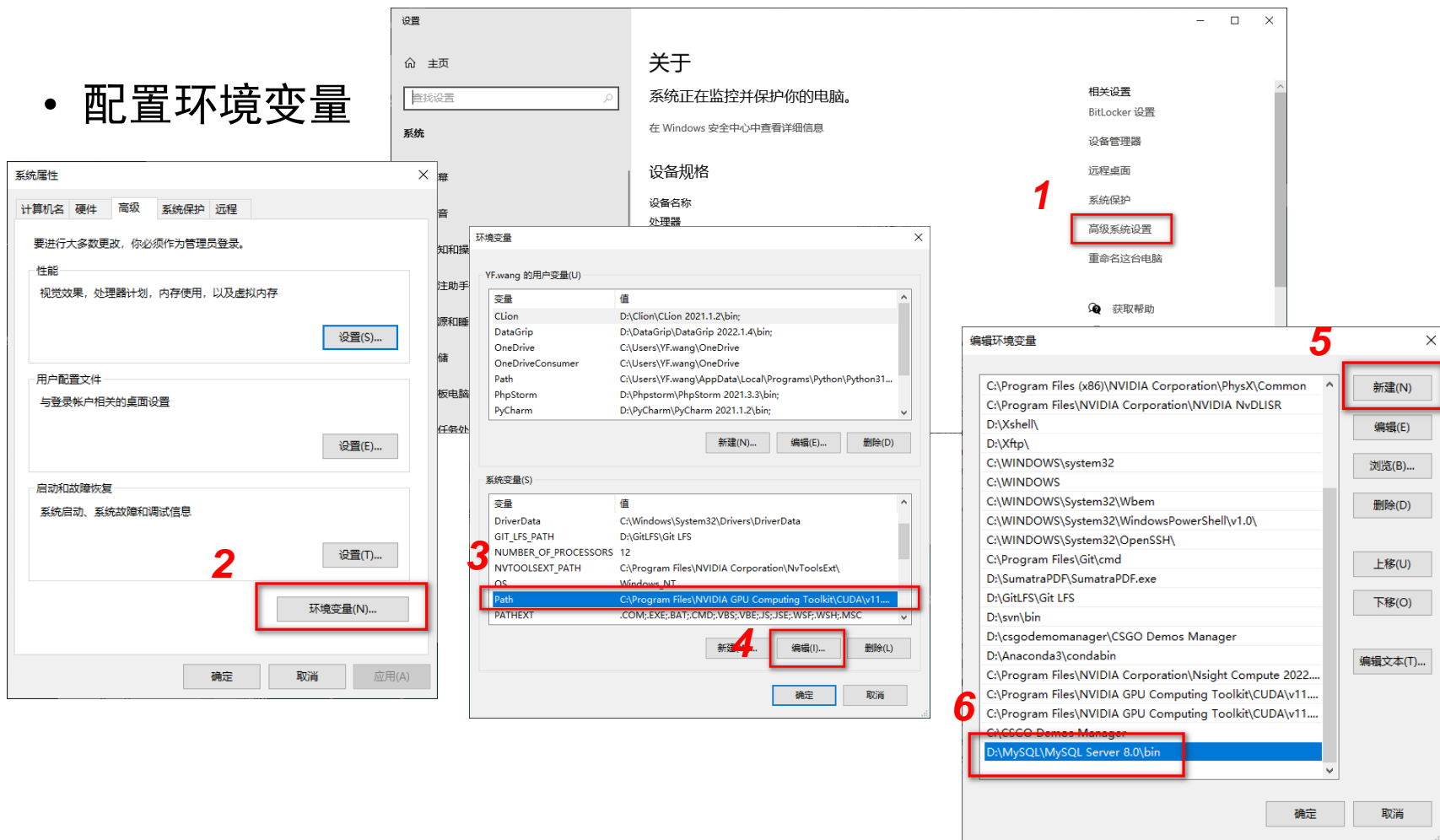
MySQL 安装

- 测试MySQL是否安装成功



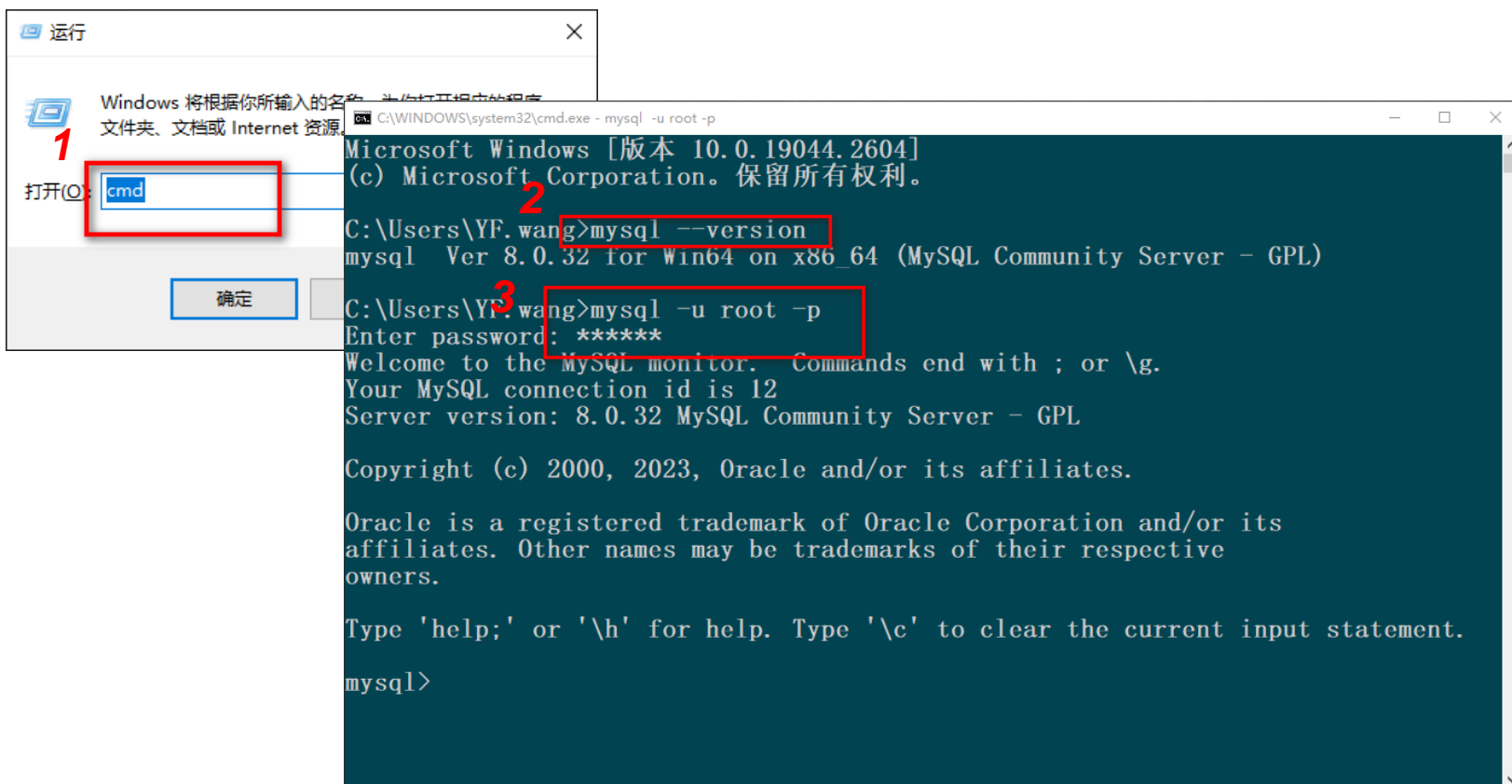
MySQL 安装

- 配置环境变量



MySQL 安装

- 测试环境变量是否配置成功



实验 1

- MySQL 关系数据库管理系统及 SQL 语言的使用

数据准备

- 向创建的数据库COMPANY中添加数据，以备后续查询使用。
- 要求数据库中至少包含50个员工，5个部门，10项工程，并且必须包含“研发部”、编号为P1和P2的项目、名叫张红的员工。

employee.txt

40	刘十	131181199904102143	刘家村	3896	131181199904022143	B02
41	孙大一	131181199905012153	孙村	9000	131181199905032153	C10
42	孙小二	131181199905022153	孙村	8000	131181199905032153	C10
43	孙三	131181199905032153	孙村	10000	131181199905032153	C10
44	孙四	131181199905042153	孙村	2000	131181199905032153	C10
45	孙五	131181199905052153	孙村	2563	131181199905032153	C10
46	孙六	131181199905062153	孙村	2563	131181199905032153	C10
47	孙七	131181199905072153	孙村	6000	131181199905032153	C10
48	孙八	131181199905082153	孙村	6000	131181199905032153	C10
49	孙九	131181199905092153	孙村	3450	131181199905032153	C10
50	孙张红十	131181199905102153	孙村	1100	131181199905032153	C10

department.txt

1	甲类一车间	A01	131181199901012113	2020-10-10
2	甲类二车间	A02	131181199902102123	2023-01-25
3	乙类一车间	B01	131181199903012133	2022-05-06
4	乙类二车间	B02	131181199904022143	2021-09-11
5	研发部	C10	131181199905032153	2022-06-09

project.txt

1	SQL	P1	S市	A01
2	SQL	P1	S市	A02
3	SQL	P1	S市	C10
4	PPA	P2	P市	A01
5	PPA	P2	P市	A02
6	PPA	P2	P市	B01
7	PPA	P2	P市	B02
8	PPA	P2	P市	C10
9	D3T	P3	D市	B01
10	D3T	P3	D市	B02
11	D3T	P3	D市	C10

works_on.txt

100	131181199905102153	PPA	1
101	131181199905012153	D3T	7
102	131181199905022153	D3T	8
103	131181199905032153	D3T	6
104	131181199905042153	D3T	8
105	131181199905052153	D3T	5
106	131181199905062153	D3T	4
107	131181199905072153	D3T	3
108	131181199905082153	D3T	2
109	131181199905092153	D3T	2
110	131181199905102153	D3T	1

创建数据库

- 创建关系数据库 COMPANY

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p

C:\Users\YF.wang>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database company;
Query OK, 1 row affected (0.02 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| company  |
| information_schema |
| menagerie |
| mysql    |
| performance_schema |
| sys      |
+-----+
6 rows in set (0.00 sec)

mysql>
```

创建表（关系模式）

- 创建表，注意数据类型，正确设置主键
- 汉字占3个字符（utf-8） 2个字符（gbk）

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql> use company;
Database changed
mysql> create table employee(
  ->  ename varchar(12),
  ->  essn char(18),
  ->  address text,
  ->  salary int,
  ->  superssn char(18),
  ->  dno char(3),
  ->  primary key(essn));
Query OK, 0 rows affected (0.05 sec)

mysql> show tables;
+-----+
| Tables in company |
+-----+
| employee          |
+-----+
1 row in set (0.00 sec)

mysql> describe employee;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ename  | varchar(12)   | YES  |     | NULL    |       |
| essn   | char(18)      | NO   | PRI | NULL    |       |
| address | text         | YES  |     | NULL    |       |
| salary | int           | YES  |     | NULL    |       |
| superssn | char(18)     | YES  |     | NULL    |       |
| dno    | char(3)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
mysql> create table department(
  ->  dname varchar(15),
  ->  dno char(3),
  ->  mgrssn char(18),
  ->  mgrstartdate date,
  ->  primary key(dno));
Query OK, 0 rows affected (0.08 sec)

mysql> create table project(
  ->  pname varchar(3),
  ->  pno char(2),
  ->  plocation text,
  ->  dno char(3),
  ->  primary key(pno));
Query OK, 0 rows affected (0.08 sec)

mysql> create table works_on(
  ->  essn char(18),
  ->  pno char(2),
  ->  hours int,
  ->  primary key(essn, pno));
Query OK, 0 rows affected (0.07 sec)

mysql> show tables;
+-----+
| Tables in company |
+-----+
| department        |
| employee          |
| project           |
| works_on          |
+-----+
4 rows in set (0.00 sec)
```

导入数据

- *./employee.txt department.txt project.txt works_on.txt* 导入数据

```
C:\WINDOWS\system32\cmd.exe - mysql --local-infile=1 -u root -p

C:\Users\YF.wang>mysql --local-infile=1 -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.32 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current
statement.

mysql> show global variables like "local_infile";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile  | OFF   |
+-----+-----+
1 row in set, 1 warning (0.01 sec)

mysql> set global local_infile=true;
Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like "local_infile";
+-----+-----+
| Variable_name | Value |
+-----+-----+
| local_infile  | ON    |
+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> use company;
Database changed
mysql> delete from employee;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from department;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from project;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from works_on;
Query OK, 0 rows affected (0.00 sec)

mysql> load data local infile "E:/dbms_lab/employee.txt" into table employee;
Query OK, 50 rows affected (0.04 sec)
Records: 50 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile "E:/dbms_lab/department.txt" into table department;
Query OK, 5 rows affected, 4 warnings (0.03 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 4

mysql> load data local infile "E:/dbms_lab/project.txt" into table project;
Query OK, 10 rows affected (0.01 sec)
Records: 10 Deleted: 0 Skipped: 0 Warnings: 0

mysql> load data local infile "E:/dbms_lab/works_on.txt" into table works_on;
Query OK, 109 rows affected (0.02 sec)
Records: 109 Deleted: 0 Skipped: 0 Warnings: 0

mysql>
```

实验任务

- 创建关系数据库COMPANY，使用SQL语言完成如下查询：
 - （1）参加了项目名为“SQL Project”的员工名字；
 - （2）在“Research Department”工作且工资低于3000元的员工名字和地址；
 - （3）没有参加项目编号为P1的项目的员工姓名；
 - （4）由张红领导的工作人员的姓名和所在部门的名称；
 - （5）至少参加了项目编号为P1和P2的项目的员工号；
 - （6）参加了全部项目的员工号码和姓名；
 - （7）员工平均工资低于3000元的部门名称；
 - （8）至少参与了3个项目且工作总时间不超过8小时的员工名字；
 - （9）每个部门的员工小时平均工资；

实验任务

```
select ename
from employee
where essn in (
    select essn
    from project, works_on
    where project.pno=works_on.pno
    and pname="SQL"
    and hours>0);
```

- 参加了项目名为“SQL Project”的员工名字
 - (1) ENAME 在 employee 中；PNAME 在 project 中
 - (2) 每个员工只属于一个部门；员工可以参加其他部门的项目
 - (3) 参加项目的员工的工作时间>0；工作时间=0的没参加项目

```
mysql> select ename from employee
-> where essn in (
-> select essn from project, works_on
-> where project.pno=works_on.pno and pname="SQL" and hours>0);
+-----+
| ename |
+-----+
| 王大  |
| 王小  |
| 王三  |
| 王四  |
| 王五  |
| 王六  |
| 王七  |
| 王十  |
| 孙小  |
+-----+
9 rows in set (0.00 sec)
```

实验任务

```
select ename, address
from employee, department
where employee.dno=department.dno
and dname="研发部"
and salary<3000;
```

- 在“研发部”工作且工资低于3000元的员工名字和地址
 - DNAME 在 department 中； SALARY/ENAME/ADDRESS 在 employee 中

```
mysql> select ename, address from employee, department
-> where employee.dno=department.dno and dname="研发部" and salary<3000;
```

ename	address
孙四	孙村
孙五	孙村
孙六	孙村
孙张红十	孙村

```
4 rows in set (0.00 sec)
```

实验任务

```
select ename
from employee
where essn not in (
    select essn
    from works_on
    where pno="P1"
    and hours>0);
```

- 没有参加项目编号为P1的项目的员工姓名
 - (1) PNO 在 works_on 中; ENAME 在 employee 中
 - (2) 没有参加项目编号 P1 的员工 NOT IN 参加了项目 P1 的员工
 - (3) 参加项目的员工的工作时间>0; 工作时间=0的没参加项目
 - (4) 用主键索引数据记录

```
C:\WINDOWS\system32\cmd.exe - mysql --local-infile=1 -u root -p
mysql> select ename from employee
-> where essn not in (
-> select essn from works_on
-> where pno="P1" and hours>0);
```

ename
王八
王九
张大一
张小二
张三
张四
张五
张六
张七
张八
张九
张红
李十一

刘一
刘二
刘三
刘四
刘五
刘六
刘七
刘八
刘九
刘十
孙十一
孙十二
孙十三
孙十四
孙十五
孙十六
孙十七
孙十八
孙十九
孙二十
孙红

41 rows in set (0.00 sec)

实验任务

```
select ename, dname
from employee, department
where employee.dno=department.dno
and superssn in (
    select essn
    from employee
    where ename="张红");
```

- 由张红领导的工作人员的姓名和所在部门的名字
 - ENAME 在 employee 中；DNAME 在 department 中
 - 用主键索引数据记录

```
mysql> select ename, dname from employee, department
-> where employee.dno=department.dno and superssn in (
-> select essn from employee
-> where ename="张红");
```

ename	dname
张大一	甲类二车间
张小二	甲类二车间
张三	甲类二车间
张四	甲类二车间
张五	甲类二车间
张六	甲类二车间
张七	甲类二车间
张八	甲类二车间
张九	甲类二车间
张红	甲类二车间

```
10 rows in set (0.00 sec)
```

实验任务

```
select essn
from works_on
where pno="P1"
    and hours>0
    and essn in (
        select essn
        from works_on
        where pno="P2"
            and hours>0);
```

- 至少参加了项目编号为P1和P2的项目的员工号
 - PNO/ESSN 在 works_on 中
 - 参加项目的员工的工作时间>0；工作时间=0的没参加项目
 - DISTINCT 去掉重复记录
 - “至少参加……”等价于“既参加……又参加……”
 - 第二种方法要注意列名模糊不清

```
select w1.essn
from works_on as w1,
     works_on as w2
where w1.essn=w2.essn
    and w1.pno="P1"
    and w1.hours>0
    and w2.pno="P2"
    and w2.hours>0;
```

```
mysql> select essn from works_on
-> where pno="P1" and hours>0 and essn in (
-> select essn from works_on
-> where pno="P2" and hours>0);
```

essn
131181199901012113
131181199901022113
131181199901032113
131181199901042113
131181199901052113
131181199901062113
131181199901072113
131181199905022153

8 rows in set (0.00 sec)

```
mysql> select w1.essn from works_on as w1, works_on as w2
-> where w1.essn=w2.essn and w1.pno="P1" and w1.hours>0
    and w2.pno="P2" and w2.hours>0;
```

essn
131181199901012113
131181199901022113
131181199901032113
131181199901042113
131181199901052113
131181199901062113
131181199901072113
131181199905022153

8 rows in set (0.00 sec)

实验任务

```
select essn, ename from employee
where not exists (
    select pno from project
    where not exists (
        select * from works_on
        where works_on.pno=project.pno
        and works_on.essn=employee.essn));
```

- 参加了全部项目的员工号码和姓名
 - ESSN 在 works_on 中; ENAME 在 employee 中
 - “参加全部项目” 等价于 “不存在有一个项目该员工没参加”
 - “参加全部项目” 等价于 “该员工参加的项目为项目总数”

```
mysql> select essn, ename from employee
-> where not exists (
-> select pno from project
-> where not exists (
-> select * from works_on
-> where works_on.pno=project.pno and works_on.essn=employee.essn));
```

essn	ename
131181199905022153	孙小二

1 row in set (0.00 sec)

```
select essn, ename from employee
where essn in (
    select essn from works_on
    group by essn having count(*) in (
        select count(*) from project));
```

```
mysql> select essn, ename from employee
-> where essn in (
-> select essn from works_on
-> group by essn having count(*) in (
-> select count(*) from project));
```

essn	ename
131181199905022153	孙小二

1 row in set (0.00 sec)

实验任务

```
select dname from department
where dno in (
    select dno from employee
    group by dno having avg(salary)<3000);
```

- 员工平均工资低于3000元的部门名称
 - SALARY 在 employee 中；DNAME 在 department 中

```
mysql> select dname from department
-> where dno in (
-> select dno from employee
-> group by dno having avg(salary)<3000);
+-----+
| dname |
+-----+
| 乙类二车间 |
+-----+
1 row in set (0.00 sec)
```

实验任务

```
select ename from employee
where essn in (
    select essn from works_on
    group by essn having count(pno)>=3
    and sum(hours)<=8);
```

- 至少参与了3个项目且工作总时间不超过8小时的员工名字
 - PNO/HOURS 在 works_on 中；ENAME 在 employee 中

```
mysql> select ename from employee
-> where essn in (
-> select essn from works_on
-> group by essn having count(pno)>=3 and sum(hours)<=8);
```

ename
王大一
王小二
王四
王六
王十一
李大一
李七
李八
李十

```
9 rows in set (0.00 sec)
```


实验任务

- 每个部门的员工小时平均工资
 - HOURS 在 works_on 中；DNO/SALARY 在 employee 中
 - 每个部门的员工小时平均工资=部门总工资÷部门总工作时长
 - 注意临时表的使用
 - 注意列名模糊不清

```
select sums.dno, sumsalary/sumhours as hoursavgsalary
from (
    select dno, sum(salary) as sumsalary
    from employee group by dno
) as sums, (
    select dno, sum(hours) as sumhours
    from works_on join employee
        on works_on.essn = employee.essn
    group by dno
) as sumh
where sums.dno=sumh.dno;
```

```
mysql> select sums.dno, sumsalary/sumhours as hoursavgsalary
-> from (select dno, sum(salary) as sumsalary from employee group by dno) as sums, (select dno,
sum(hours) as sumhours from works_on join employee on works_on.essn = employee.essn group by dno)
as sumh
-> where sums.dno=sumh.dno;
```

dno	hoursavgsalary
A01	439.0244
A02	652.1739
B01	685.3933
B02	638.7097
C10	858.9153

```
5 rows in set (0.00 sec)
```

实验 2

- 数据库系统开发

实验内容

- 开发一个数据库系统，可以参考教材的例子
- 要求
 - 该系统的E-R图至少包括8个实体和7个联系（必须有一对一联系、一对多联系、多对一联系）
 - 在设计的关系中需要体现关系完整性约束：主键约束、外键约束，空值约束
 - 对几个常用的查询创建视图、并且在数据库中为常用的属性（非主键）建立索引
 - 该系统功能必须包括：插入、删除、连接查询、嵌套查询、分组查询。其中插入，删除操作需体现关系表的完整性约束，例如插入空值、重复值时需给予提示或警告等
 - 加分项：界面友好、包含事务管理、触发器等功能

实验内容

• 检查点

- （1）检查E-R图，至少包括8个实体和7个联系，必须有一对一联系、一对多联系、多对一联系
- （2）检查创建的数据库是否与E-R图一致，关系模式是否符合范式要求
- （3）对几个常用的查询创建视图、并且在数据库中为常用的属性（非主键）建立索引
- （4）使用SQL语句进行插入、删除操作，体现关系表的完整性约束，插入空值、重复值时需给予提示或警告
- （5）使用SQL语句进行连接查询、嵌套查询、分组查询，查询要体现分组、having语句
- 按时完成、界面友好、包含事务管理、触发器等功能为加分项
- 注：加分项为选做项目，1-5全都没有问题加分项不额外加分

构建概念数据库

- 建立一个工厂管理数据库系统，为构建概念数据库，下分析工厂数据库实体与联系并画出ER图
- 实体：主码加粗标红表示
 - (1) 工厂：编号、厂名、地址
 - (2) 厂长：身份证号、姓名、电话
 - (3) 车间：车间号、地址
 - (4) 车间主任：身份证号、姓名、电话
 - (5) 仓库：仓库号、地址
 - (6) 仓库主任：身份证号、姓名、电话
 - (7) 零件：零件号、重量、价格
 - (8) 产品：产品号、重量、价格

构建概念数据库

- 联系：

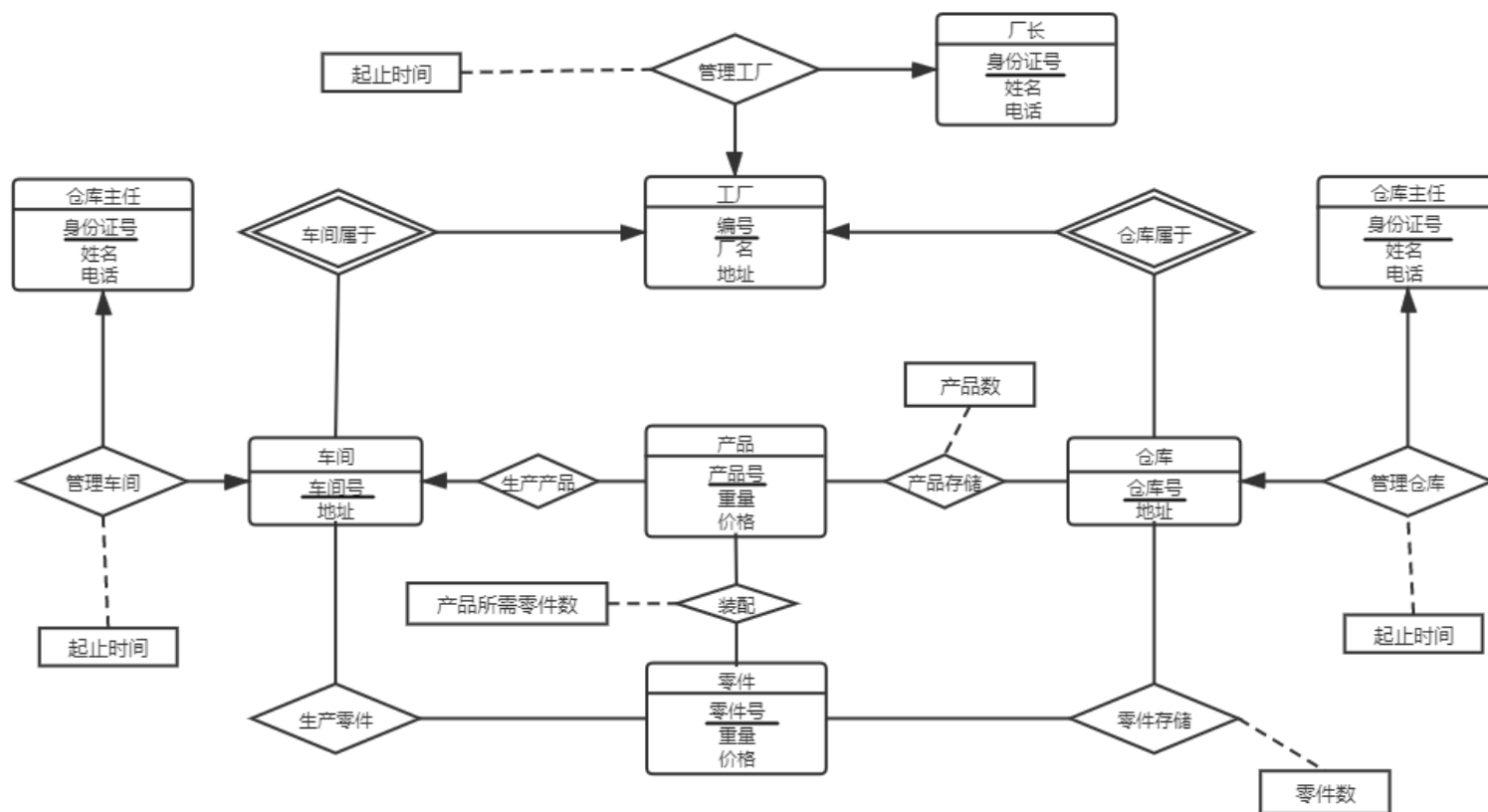
- （1）一个工厂内有多多个车间和多个仓库，一个车间或一个仓库都只能属于一个工厂；
- （2）一个车间生产多种产品，每种产品只能产自一个车间；
- （3）一个车间生产多种零件，一种零件也可能为多个车间所制造；
- （4）一个产品由多种零件组成，一种零件也可装配出多种产品；
- （5）产品和零件均存入仓库；
- （6）一个工厂只能有一个厂长，一个厂长只能管理多个工厂
- （7）一个车间只能有一个车间主任，一个车间主任只能管理多个车间
- （8）一个仓库只能有一个仓库主任，一个仓库主任只能管理多个仓库

构建概念数据库

- 进一步分析弱实体、联系的描述性属性和约束条件：
 - 工厂和车间构成“车间属于”联系，属于1：n联系，
 - 由于车间是工厂的弱实体，该联系采用实体属性表示法，该联系的描述性属性暂时不设
 - 工厂和仓库构成“仓库属于”联系，属于1：n联系，
 - 由于仓库是工厂的弱实体，该联系采用实体属性表示法，该联系的描述性属性暂时不设
 - 车间和产品构成“生产产品”联系，属于1：n联系，
 - 采用联系集表示法，该联系的描述性属性暂时不设
 - 车间和零件构成“生产零件”联系，属于m：n联系，
 - 采用联系集表示法，该联系的描述性属性暂时不设
 - 产品和零件构成“装配”联系，属于m：n联系，
 - 采用联系集表示法，该联系的描述性属性考虑产品所需零件数
 - 产品和仓库构成“产品存储”联系，属于m：n联系，
 - 采用联系集表示法，该联系的描述性考虑产品数
 - 零件和仓库构成“零件存储”联系，属于m：n联系，
 - 采用联系集表示法，该联系的描述性考虑零件数
 - 厂长和工厂构成“管理工厂”联系，属于1：1联系，
 - 采用联系集表示法，描述性属性考虑管理的开始时间和结束时间
 - 车间主任和车间构成“管理车间”联系，属于1：1联系，
 - 采用实体属性表示法，描述性属性考虑管理的开始时间和结束时间
 - 仓库主任和仓库构成“管理仓库”联系，属于1：1联系，
 - 采用实体属性表示法，描述性属性考虑管理的开始时间和结束时间

绘制 E-R 图

- 由此绘制 E-R 图



设计逻辑数据库

- 根据绘制的E-R图，对每个普通实体集构造关系 S_i ，然后分析是否存在弱实体、多值属性、实体间联系：
 - 在语义上不含多值属性
 - 分析实体间联系，采用构造新关系的方式处理联系，对每个联系构造关系 T_i
- 得到初始关系数据库模式：主码加粗标红表示
 - (1) 工厂S0 (**编号G#**, 厂名Gname, 地址Address)
 - (2) 车间S1 (**编号G#**, **车间号A#**, 地址Address)
 - (3) 仓库S2 (**编号G#**, **仓库号B#**, 地址Address)
 - (4) 厂长S3 (**身份证号ID**, 姓名Name, 电话Tel)
 - (5) 车间主任S4 (**身份证号ID**, 姓名Name, 电话Tel)
 - (6) 仓库主任S5 (**身份证号ID**, 姓名Name, 电话Tel)
 - (7) 产品S6 (**产品号C#**, 重量Weight, 价格Price)
 - (8) 零件S7 (**零件号D#**, 重量Weight, 价格Price)
 - (9) 管理工厂T0 (**编号G#**, **身份证号ID**, 开始时间Sdate, 结束时间Edate)
 - (10) 管理车间T1 (**编号G#**, **车间号A#**, **身份证号ID**, 开始时间Sdate, 结束时间Edate)
 - (11) 管理仓库T2 (**编号G#**, **仓库号B#**, **身份证号ID**, 开始时间Sdate, 结束时间Sdate)
 - (12) 生产产品T3 (**编号G#**, **车间号A#**, **产品号C#**)
 - (13) 生产零件T4 (**编号G#**, **车间号A#**, **零件号D#**)
 - (14) 产品存储T5 (**编号G#**, **仓库号B#**, **产品号C#**, 产品数Cnum)
 - (15) 零件存储T6 (**编号G#**, **仓库号B#**, **零件号D#**, 零件数Dnum)
 - (16) 装配T7 (**产品号C#**, **零件号D#**, 产品所需零件数CDnum)

设计逻辑数据库

- 然后确定关系上的函数依赖集

- $S_0\{G\# \rightarrow GName, G\# \rightarrow Address\}$
- $S_1\{\{G\#, A\#\} \rightarrow Address\}$
- $S_2\{\{G\#, B\#\} \rightarrow Address\}$
- $S_3\{ID \rightarrow Name, ID \rightarrow Tel\}$
- $S_4\{ID \rightarrow Name, ID \rightarrow Tel\}$
- $S_5\{ID \rightarrow Name, ID \rightarrow Tel\}$
- $S_6\{C\# \rightarrow Weight, C\# \rightarrow Price\}$
- $S_7\{D\# \rightarrow Weight, D\# \rightarrow Price\}$
- $T_0\{\{G\#, ID\} \rightarrow Sdate, \{G\#, ID\} \rightarrow Edate\}$
- $T_1\{\{G\#, A\#, ID\} \rightarrow Sdate, \{G\#, A\#, ID\} \rightarrow Edate\}$
- $T_2\{\{G\#, B\#, ID\} \rightarrow Sdate, \{G\#, B\#, ID\} \rightarrow Edate\}$
- $T_5\{\{G\#, B\#, C\#\} \rightarrow Cnum\}$
- $T_6\{\{G\#, B\#, D\#\} \rightarrow Dnum\}$
- $T_7\{\{C\#, D\#\} \rightarrow CDnum\}$

- 关系模式规范化

- 以上16个关系满足BCNF，所以这里不用进行规范化

- 关系模式优化

- 这里无需进行关系模式优化

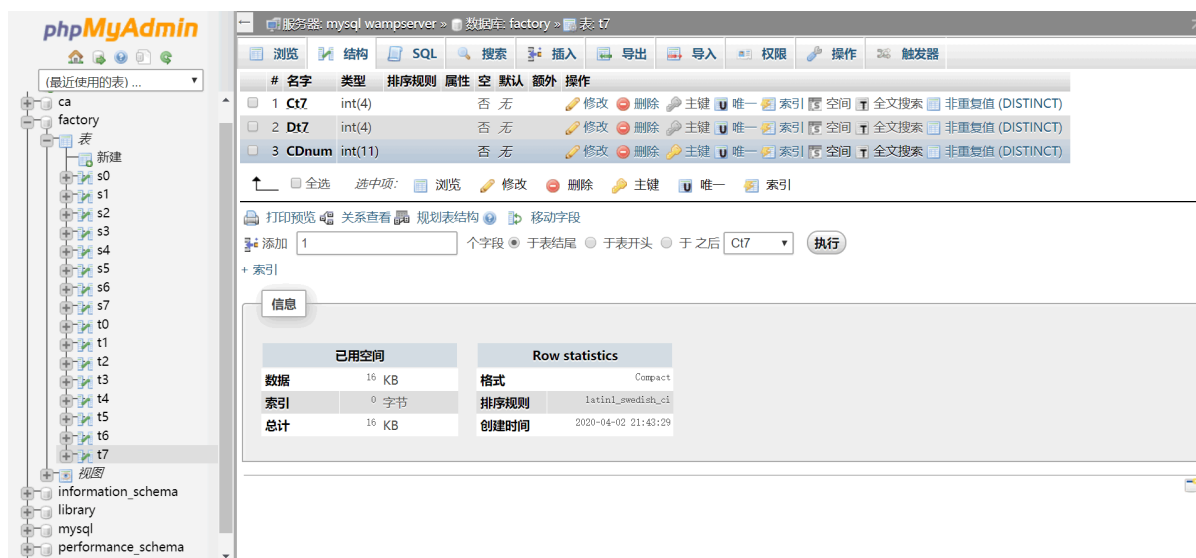
设计逻辑数据库

- 定义关系上的完整性和安全性约束
 - 主码约束：主码不允许重复
 - 外码约束：满足参照完整性
 - 空值约束：本关系数据库系统所有属性值非空
- 子模式定义
 - 为查询厂长管理工厂的信息创建视图，将S0-S3-T0自然连接
 - 为查询车间主任管理车间的信息创建视图，将S1-S4-T1自然连接
 - 为查询仓库主任管理仓库的信息创建视图，将S2-S5-T2自然连接
 - 在产品关系表中为产品的价格建立索引
 - 在零件关系表中为零件的价格建立索引

```
CREATE DATABASE factory;  
USE factory;
```

数据库系统实现

- 数据库实现使用WAMP，即Web、Apache、Mysql、Php实现
- 第一步是建立16个关系表（SQL）
 - 首先使用 mysql 终端创建数据库：
 - 可以通过mysql终端使用CREATE TABLE依据上面分析的逻辑数据库创建16个关系表，为了方便可以使用图形化的方式创建
 - 下面列出创建的关系表的结构：



数据库系统实现

- 第二步是建立常用查询的视图 (SQL)

- 为查询厂长管理工厂的信息创建视图，将S0-S3-T0自然连接，在mysql终端使用如下代码：

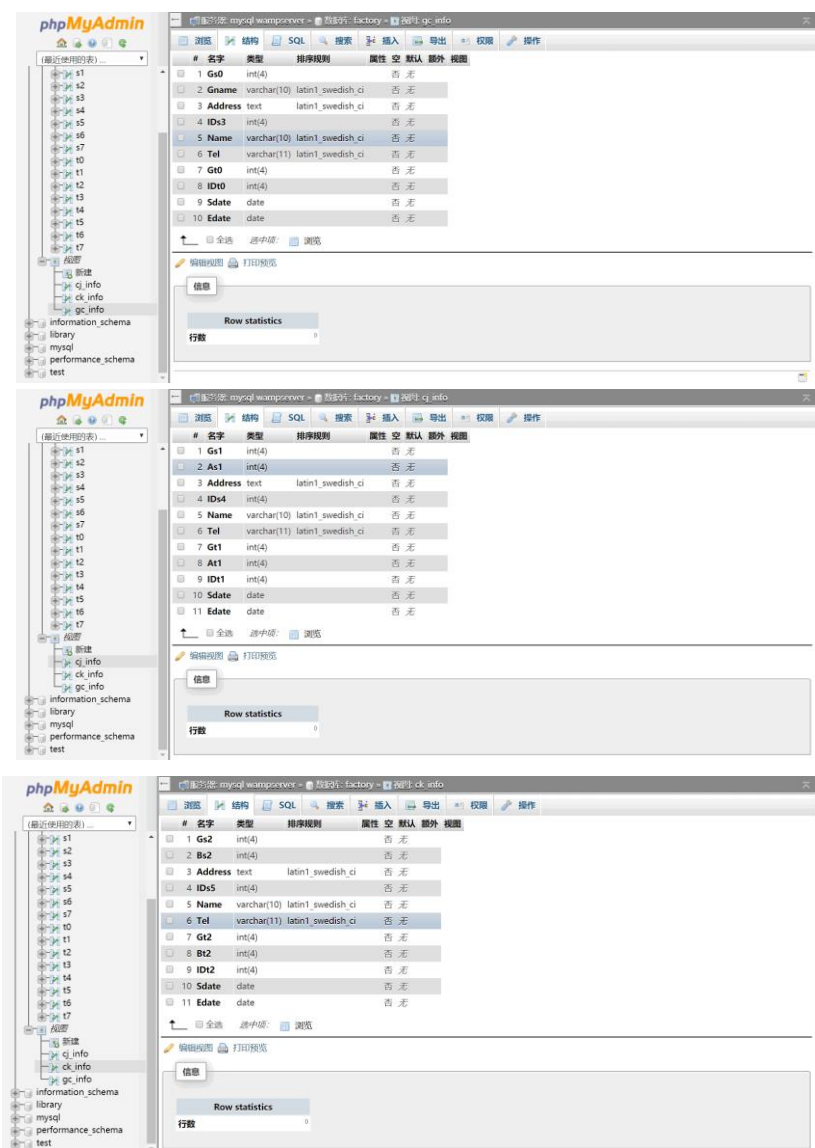
```
CREATE VIEW gc_info AS  
SELECT * FROM s0, s3, t0  
WHERE s0.Gs0=t0.Gt0 AND s3.IDs3=t0.IDt0;
```

- 为查询车间主任管理车间的信息创建视图，将S1-S4-T1自然连接，在mysql终端使用如下代码：

```
CREATE VIEW cj_info AS  
SELECT * FROM s1, s4, t1  
WHERE s1.Cs1=t1.Ct1 AND s4.IDs4=t1.IDt1;
```

- 为查询仓库主任管理仓库的信息创建视图，将S2-S5-T2自然连接，在mysql终端使用如下代码：

```
CREATE VIEW ck_info AS  
SELECT * FROM s2, s5, t2  
WHERE s2.Ds2=t2.Dt2 AND s5.IDs5=t2.IDt2;
```



数据库系统实现

- 第三步是对非键属性建立索引 (SQL)

- 在产品关系表中为产品的价格建立索引, 在mysql终端使用如下代码:

```
CREATE INDEX cp ON s6(Price);
```

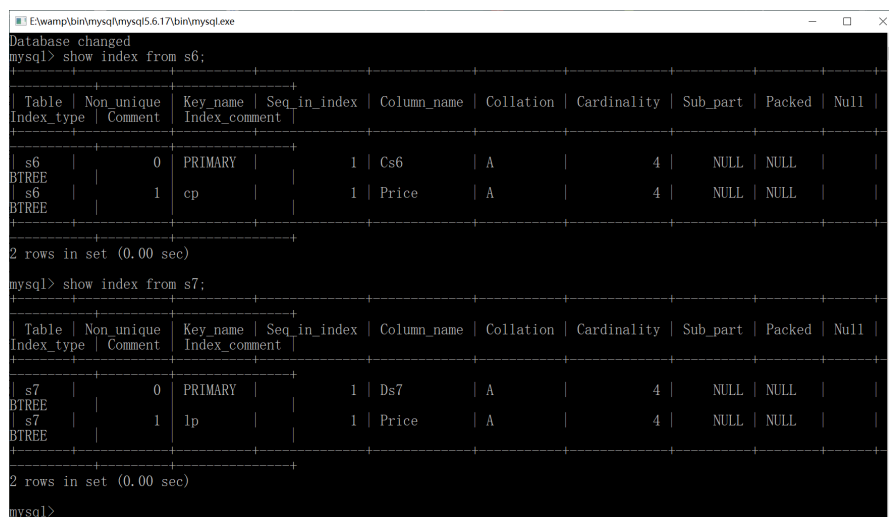
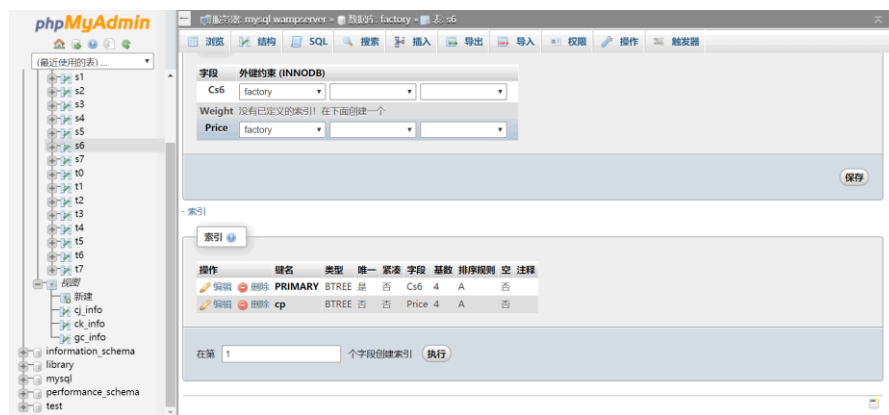
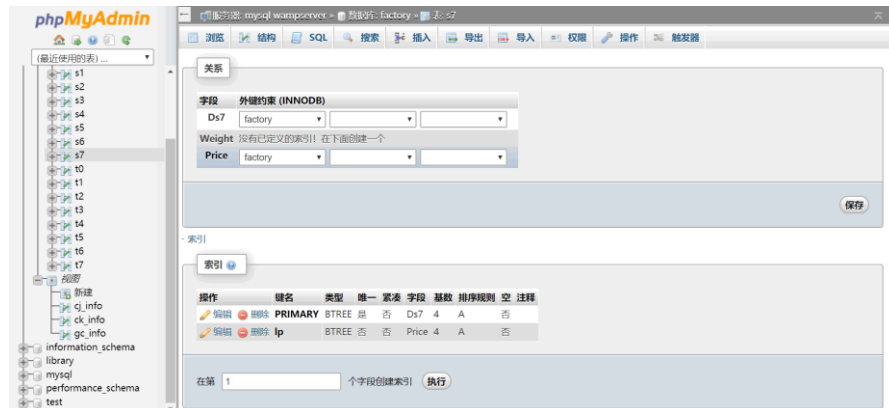
- 在零件关系表中为零件的价格建立索引, 在mysql终端使用如下代码:

```
CREATE INDEX lp ON s7(Price);
```

- 通过终端查看创建的视图:

```
SHOW INDEX FROM s6;
```

```
SHOW INDEX FROM s7;
```



数据库系统实现

- 第四步是设置插入删除触发器，体现完整性约束（PHP）
 - 为工厂管理关系t0设置插入删除触发器，其他关系类似就不再详细介绍

```
CREATE TRIGGER insert_t0 AFTER INSERT ON t0
REFERENCING NEW ROW AS nrow
FOR EACH ROW
WHEN(
    nrow.Gt0 NOT IN (SELECT Gs0 FROM s0) OR
    nrow.IDt0 NOT IN (SELECT IDs3 FROM s3) OR (
        nrow.Gt0 IN (SELECT Gt0 FROM t0) AND
        nrow.IDt0 IN (SELECT IDt0 FROM t0)
    )
)
BEGIN
    ROLLBACK;
END
```

```
CREATE TRIGGER delete_t0 AFTER DELETE ON t0
REFERENCING old row AS orow
FOR EACH ROW
WHEN(
    orow.Gt0 NOT IN (SELECT Gs0 FROM s0) OR
    orow.IDt0 NOT IN (SELECT IDs3 FROM s3) OR
    orow.Gt0 NOT IN (SELECT Gt0 FROM t0) OR
    orow.IDt0 NOT IN (SELECT IDt0 FROM t0)
)
BEGIN
    ROLLBACK;
END
```

数据库系统实现

- 第五步是对数据库进行查询（PHP）

- 以三种场景为例，分别介绍设计的连接查询、嵌套查询和分组查询。
- 连接查询，通过子模式查询工厂管理情况的详细信息：由于我们构建的视图是将工厂管理情况进行连接的，这里直接查询视图即可

```
SELECT * FROM gc_info;
```

- 嵌套查询，查询重量小于5kg且价格大于100元的产品，在产品关系中查询重量小于5kg的产品，并在得到的结果中查询价格大于100元的产品

```
SELECT * FROM (  
    SELECT * FROM s6  
    WHERE Weight<5  
) as in_s6  
WHERE Price>100;
```

- 分组查询（分组、having），查询每个产品用到的零件种数，在装配关系中按照产品进行分组

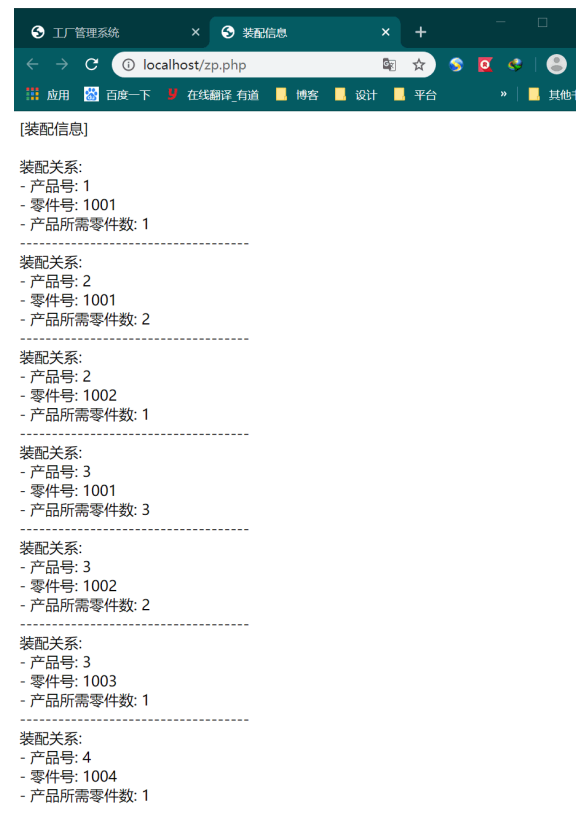
```
SELECT Ct7, count(Dt7) as count_d  
FROM t7  
GROUP BY Ct7 HAVING count(Dt7)>1;
```


数据库系统实现

- 第六步是针对第四步和第五步设计GUI（HTML）定义关系上的完整性和安全性约束
 - 针对第四步和第五步设计实现GUI，采用的是HTML与PHP代码交互的方式，在WEB上实现GUI
 - 为方便使用，为几个常用的场景插入、删除、连接查询、嵌套查询、分组查询设置按钮，为相关的关系表设置查询按钮

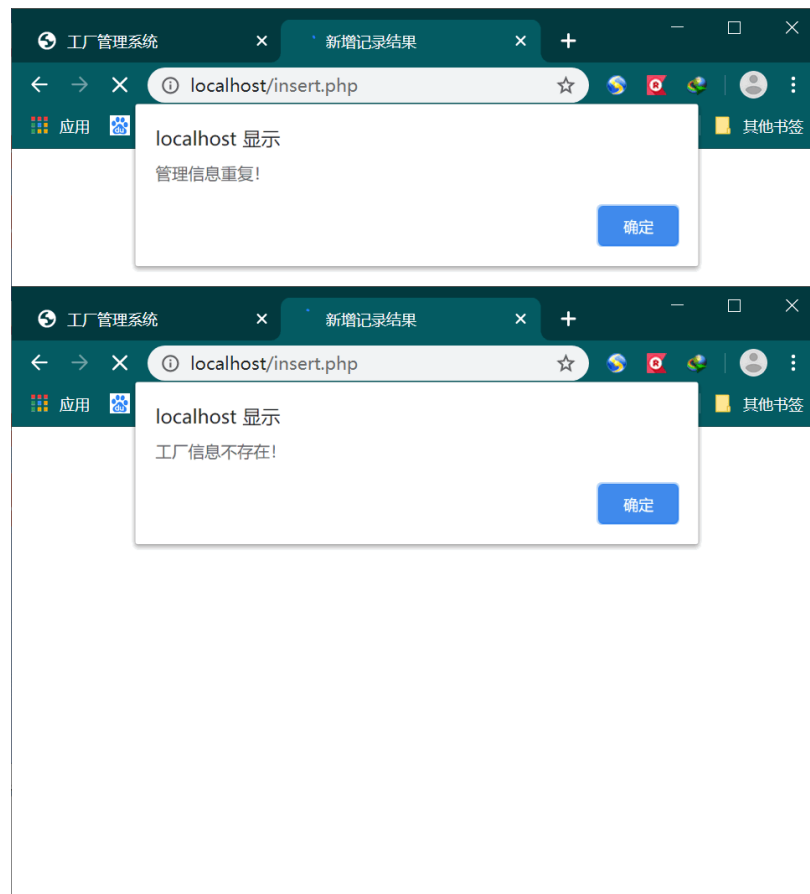
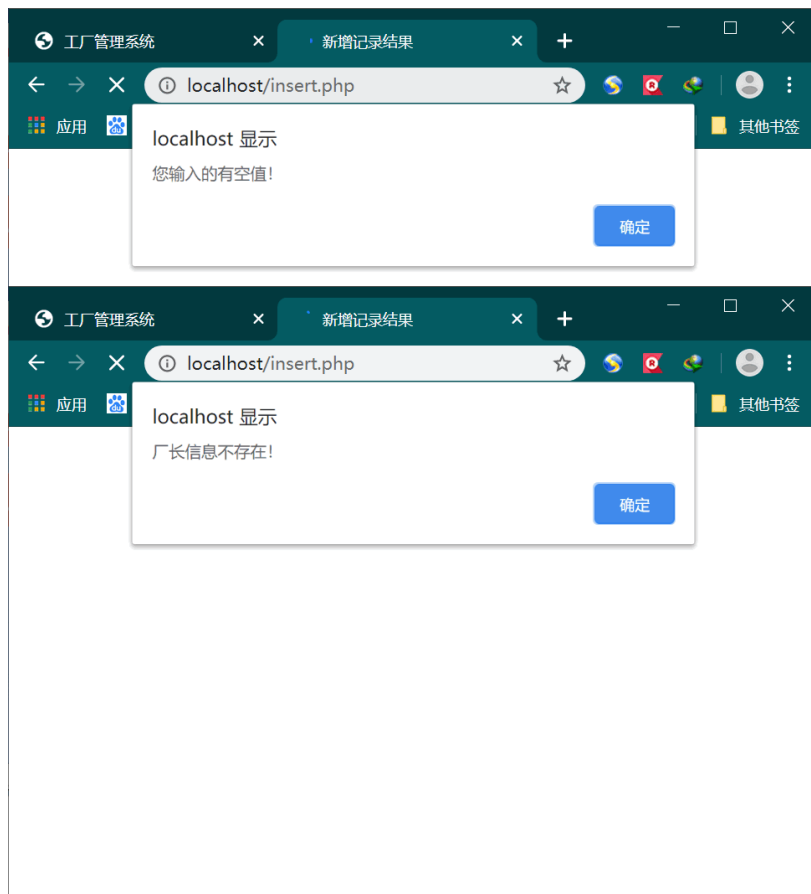


数据库系统实现



在GUI设计过程中也考虑了空值、重复值、不存在等非法情况，设计的代码会放到后面：

数据库系统实现



数据库系统实现

- 插入

- 插入厂长管理工厂的信息要首先判断插入的信息中是否有空，查询该厂长信息是否存在、该工厂信息是否存在，插入的主码信息在管理工厂关系中是否重复，如果条件正常满足则插入，否则警告

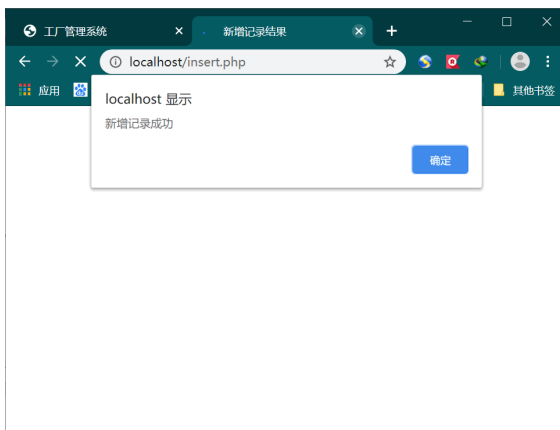
新增工厂管理记录:

工厂编号:

厂长身份证号:

开始时间:

结束时间:



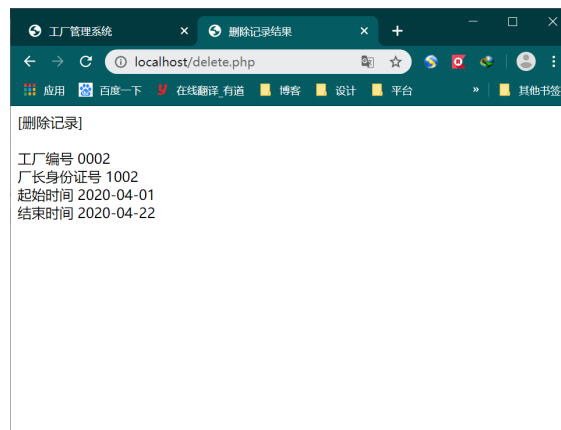
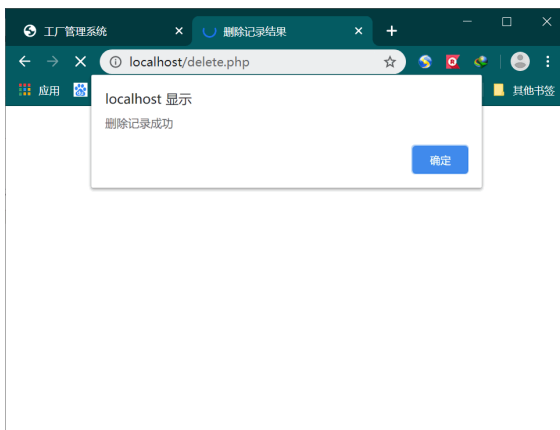
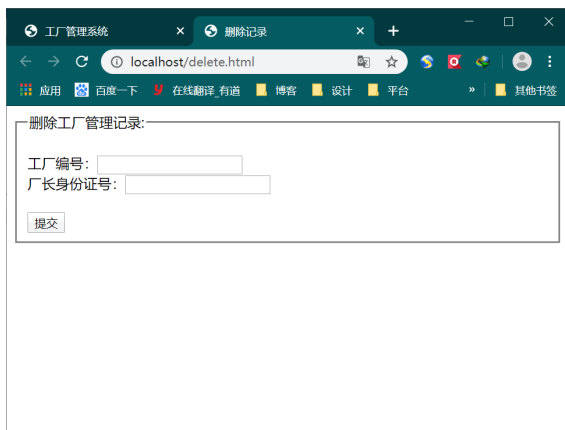
[新增记录]

工厂编号 0002
厂长身份证号 1002
起始时间 2020-04-01
结束时间 2020-04-22

数据库系统实现

- 插入

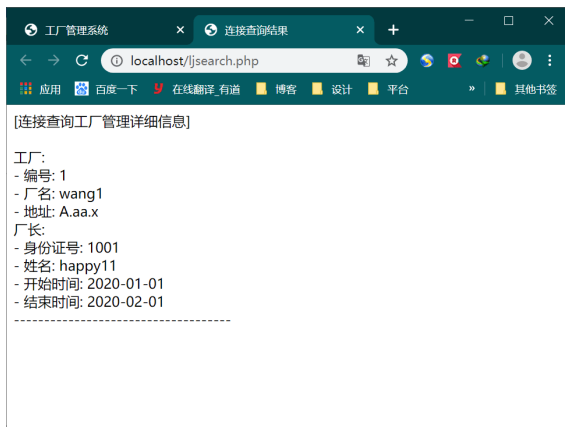
- 删除厂长管理工厂的信息要首先判断删除信息是否有空，查询该厂长信息是否存在、该工厂是否存在、该厂长管理工厂的信息是否存在，如果条件正常满足则插入，否则警告



数据库系统实现

• 查询

- 连接查询：通过子模式查询工厂管理情况的详细信息
- 嵌套查询：查询重量小于5kg且价格大于100元的产品，在产品关系中查询重量小于5kg的产品，并在得到的结果中查询价格大于100元的产品
- 分组查询（分组、having）：查询每个产品用到的零件种数，在装配关系中按照产品进行分组



实验 3

- 存储管理与查询优化算法的设计

任务1：关系连接算法的实现

- 关系R具有两个属性A和B，其中A和B的属性值均为int型（4个字节），A的值域为[1, 40]，B的值域为[1, 1000]。
- 关系S具有两个属性C和D，其中C和D的属性值均为int型（4个字节）。C的值域为[20, 60]，D的值域为[1, 1000]。
 - （1）实现关系选择算法：基于ExtMem程序库，使用高级语言实现关系选择算法，选出R.A=40或S.C=60的元组，并将结果存放在磁盘上。
 - （2）实现关系投影算法：基于ExtMem程序库，使用高级语言实现关系投影算法，对关系R上的A属性进行投影，并将结果存放在磁盘上。
 - （3）实现Nested-Loop Join (NLJ)、hash-join和 sort-merge-join算法：基于ExtMem程序库，使用高级语言实现以上三种join算法，对关系R和S计算R.A连接S.C，并将结果存放在磁盘上。

任务1检查点

- 使用ExtMem程序库建立两个关系R和S的物理存储。关系的物理存储形式为磁盘块序列 B_1, B_2, \dots, B_n ，其中 B_i 的最后4个字节存放 B_{i+1} 的地址。即R和S的每个元组的大小均为8个字节。
- 块的大小设置为64个字节，缓冲区大小设置为 $512+8=520$ 个字节。这样，每块可存放7个元组和1个后继磁盘块地址，缓冲区内可最多存放8个块。
- 编写程序，随机生成关系R和S，使得R中包含 $16 * 7 = 112$ 个元组，S中包含 $32 * 7 = 224$ 个元组。
 - (1) 实现关系选择算法。
 - (2) 实现关系投影算法。
 - (3) 实现Nested-Loop Join (NLJ)、hash-join和 sort-merge-join算法

任务2：查询优化算法的设计

- 任选下列查询语句中的三条，将其转化为对应的查询执行树，并且根据设计的查询优化算法，对生成的查询执行树进行优化。
 - *SELECT [ENAME = 'Mary' & DNAME = 'Research'] (EMPLOYEE JOIN DEPARTMENT)*
 - *PROJECTION [BDATE] (SELECT [ENAME = 'John' & DNAME = 'Research'] (EMPLOYEE JOIN DEPARTMENT))*
 - *SELECT [ESSN = '01'] (PROJECTION [ESSN, PNAME] (WORKS_ON JOIN PROJECT))*
 - *PROJECTION [ENAME] (SELECT [SALARY < 3000] (EMPLOYEE JOIN SELECT [PNO = 'P1'] (WORKS_ON JOIN PROJECT)))*
 - *PROJECTION [DNAME, SALARY] (AVG [SALARY] (SELECT [DNAME = 'Research'] (EMPLOYEE JOIN DEPARTMENT)))*
- 注意：
 - （1）在上述语句中，为了便于设计语法分析器，不同符号与关键词之间均以空格分隔。
 - （2）每人只用完成上述查询语句中的三条，根据完成情况给分。
 - （3）上述语句并非标准sql语句，而是本实验第一部分关系链接操作算法的函数抽象

任务2检查点

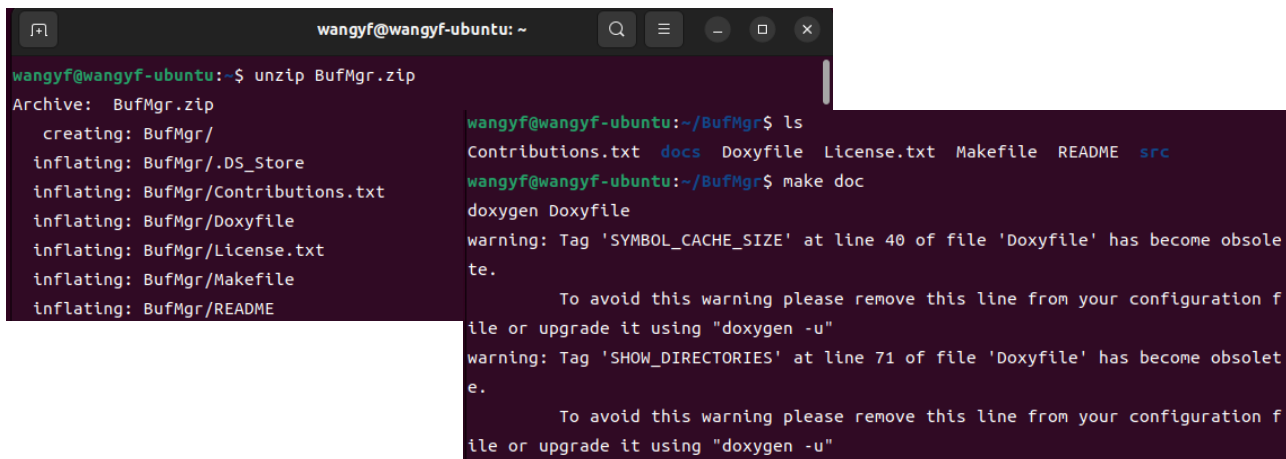
- （1）设计一个语法分析器，能够识别指导书中关系代数语句，并且对其进行解析，生成对应的查询执行树
- （2）可识别查询语句中的三条
- （3）根据本课程所学的查询优化技术，对生成的查询执行树进行优化，并且将最后优化后的查询执行树输出至屏幕

Project 2

- *Buffer Manager*

实验任务

- 该项目需要在提供的存储管理器之上实现缓冲区管理器
- (1) 将 BufMgr.zip 下载并解压
- (2) 使用如下所示的 Doxygen 为您的代码生成文档。在BufMgr目录中，运行命令生成文档文件，在这之前应该先
 - > sudo apt install make
 - > sudo apt install doxygen



```
wangyf@wangyf-ubuntu: ~  
wangyf@wangyf-ubuntu:~$ unzip BufMgr.zip  
Archive:  BufMgr.zip  
  creating:  BufMgr/  
  inflating:  BufMgr/.DS_Store  
  inflating:  BufMgr/Contributions.txt  
  inflating:  BufMgr/Doxyfile  
  inflating:  BufMgr/License.txt  
  inflating:  BufMgr/Makefile  
  inflating:  BufMgr/README  
  
wangyf@wangyf-ubuntu:~/BufMgr$ ls  
Contributions.txt  docs  Doxyfile  License.txt  Makefile  README  src  
wangyf@wangyf-ubuntu:~/BufMgr$ make doc  
doxygen Doxyfile  
warning: Tag 'SYMBOL_CACHE_SIZE' at line 40 of file 'Doxyfile' has become obsolete.  
To avoid this warning please remove this line from your configuration file or upgrade it using "doxygen -u"  
warning: Tag 'SHOW_DIRECTORIES' at line 71 of file 'Doxyfile' has become obsolete.  
To avoid this warning please remove this line from your configuration file or upgrade it using "doxygen -u"
```

检查点

- 收到数据页请求时，是否能返回正确的结果：请求的页在pool中，返回指针，不在，是否正确更新并返回
- Buffer Manager的结构是否正确
- 是否正确访问buffer pool中的页面
- 时钟算法的过程是否正确：时钟指针的检查过程是否正确，页面替换是否正确，是否正确将数据写回磁盘
- 良好的编程风格：面向对象，Doxygen风格的注释