**实验一 认识和安装DBMS（2学时）**

成功安装MYSQL5.7.x 数据库（以及安装SQL Server 2014SP），了解命令行，图形界面等工具的使用，（实验主要以mysql运行为主，SQL Server可用来尝试书上的一些SQL的编写等）

1. 创建数据库（db2020）；
2. 使用该数据库；
3. 创建数据表（administrator）

管理员（员工号，姓名，密码， 联系方式）（主码为员工号）

各属性的类型如下：

ano char(8)

name varchar(30)

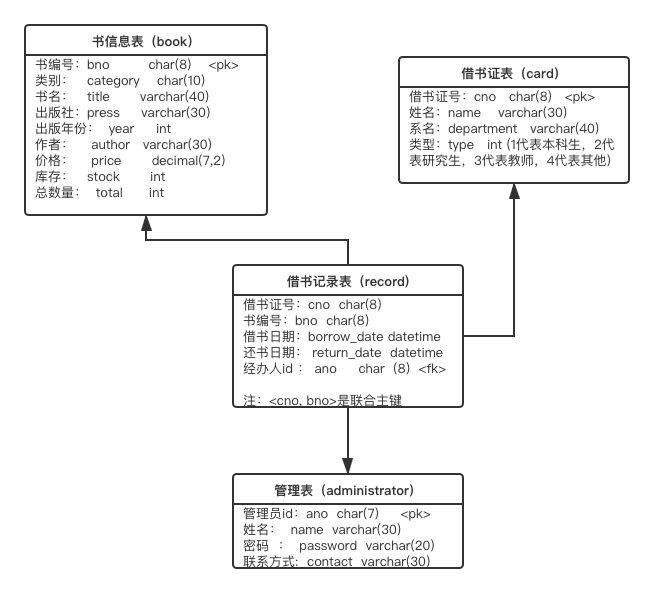
password varchar(20)

contact varchar(30)

1. 往administrator表插入两条数据；
2. 查看插入的数据

**实验二 SQL数据定义和操作（3学时）**

创建数据表，对表属性的修改以及数据的增删改查等(图有点问题，administrator和record中的ano 都设置char(8))

****

1. 根据上图创建这四个表，其中pk指的是主键，fk是外键约束。
2. 表的修改
   1. 修改book表中的category属性的类型为varchar(15)；
   2. 修改card表中的name改为cname；
   3. 修改 record表主键为联合主键（cno, bno, borrow\_date）
3. 数据更新： 用 insert/delete/update命令插入/删除/修改表数据。

book表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bno | category | title | press | year | author | price | total | stock |
| 20200001 | history | 世界历史 | 安徽文艺出版社 | 2009 | 马健 | 46.8 | 30 | 15 |
| 20200030 | computer | 数据库系统原理教程 | 清华大学出版社 | 1998 | 王珊 | 18.5 | 20 | 7 |
| 20201005 | Mathematics | 概率论与数理统计 | 高等教育出版社 | 2010 | 盛骤 | 41.8 | 30 | 20 |
| 20203001 | computer | MYSQL必知必会 | 人民邮电出版社 | 2009 | 刘晓霞 | 39.0 | 25 | 13 |
| 20200304 | music | 勋伯格和声学 | 上海音乐出版社 | 2007 | **阿诺德·勋伯格** | 28.0 | 15 | 6 |

card表

|  |  |  |  |
| --- | --- | --- | --- |
| cno | cname | department | type |
| 1000001 | xiaoli | computer science | 1 |
| 2000005 | xiaozhang | history | 2 |
| 7000302 | Mike | english | 3 |
| 1000420 | xiaowang | mathematics | 1 |
| 2003001 | lihua | computer science | 2 |
| 1000830 | hhh | Computer science | 1 |
| 2200012 | xiaohua | English | 2 |
| 1000330 | xiaozhou | history | 1 |

administrator表

|  |  |  |  |
| --- | --- | --- | --- |
| ano | name | password | contact |
| 100001 | admin | 123456 | 13111111111 |

record表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cno | bno | borrow\_date | return\_date | ano |
| 1000001 | 20200030 | 2019-03-05 09:53:56 | 2019-04-15 15:23:16 | 100001 |
| 1000001 | 20203001 | 2019-04-15 15:25:00 | 2019-05-05 16:10:00 | 100001 |
| 2000005 | 20200001 | 2019-03-29 14:20:00 | 2019-04-12 14:49:00 | 100001 |
| 7000302 | 20200304 | 2019-07-20 11:20:00 | 2019-08-12 10:30:00 | 100001 |
| 1000420 | 20201005 | 2019-09-20 11:20:00 | 2019-10-23 11:30:00 | 100001 |
| 1000830 | 20200030 | 2019-01-10 08:40:00 | 2019-03-01 09:20:00 | 100001 |
| 1000830 | 20200304 | 2019-10-20 09:20:00 | 2019-11-23 12:30:00 | 100001 |

使用insert命令把上面表格数据插入（也可以自己额外再多加点数据）。

使用update命令修改book表中书籍“MYSQL必知必会” 人民邮电出版社的作者改为 Ben Forta，库存改为15.

1. 数据查询： 单表查询，多表查询， 嵌套子查询等。
2. 求总藏书量、藏书总金额，总库存册数。

**select sum(total),sum(price\*total),sum(stock) from book;**

1. 列出藏书在10本以上的书（书名、作者、出版社、年份，库存量）。

select title,author,press,year,stock from book where total>10;

(c) “数据库系统原理教程，王珊编著，清华大学出版社，1998年出版”还有几本？

(d) 查询在图书馆借书次数两次及两次以上的人的cno，姓名，部门以及借书次数,以借书次数从高到低排列。

**实验三 SQL安全性（1学时）**

创建多个用户以及多个表，以及视图，分别赋予不同的权限，尝试登陆不同用户对表进行不同操作，看看这些表，视图对不同用户实际有哪些权限限制。

本实验是开放性实验，主要是考察权限控制，主要的包括对数据的add, delete, update, select权限，对数据表结构的create, alter, drop权限，以及对视图的create view，show view权限。(开放性实验)

* 建立表，考察表的生成者拥有该表的哪些权限（建立不同的用户，考察它们对该表的权限）。
* 学习使用SQL的grant 和revoke命令，对用户进行不同的表/属性权限的操作，理解考察相应命令的作用。
* 建立视图，并把该视图的查询权限授予其他用户，考察通过视图进行权限控制的作用。

**可供参考mysql用户管理资料：**

1. **查看当前已有用户 select user, host from mysql.user;**
2. **创建新用户 (通常可以直接用%或者localhost)**

**#指定ip的xiaowang登陆**

**Create user ‘hhh’@’指定ip’ identified by ‘123456’; ‘hhh’为用户名，‘123456’为密码**

**#任意ip 登陆**

**Create user ‘hhh’@’%’ identified by ‘123456’;**

**# 只可本地登陆**

**Create user ‘hhh’@’localhost’ identified by ‘123456’;**

1. **删除用户**

**Drop user ‘用户名’@’ip地址’;**

1. **修改用户**

**Rename user ‘用户名’@’ip地址’ to ‘新用户名’@’ip地址’；**

1. **修改密码**

**Set password for ‘用户名’@’ip地址’ = password(‘新密码’);**

1. **查看授权**

**Show grants for ‘用户名’@’ip地址’; show grants;**

1. **授权和取消授权**

**Grant all privileges on db2020.test to ‘用户名’@’ip地址’; root才有的命令，全部的权限授予**

**Grant select, update on db2020.test to ‘用户名’@’ip地址’;**

**Rovoke all privileges on \*.\* from ‘用户名’@’ip地址’; 取消该用户对所有数据库中所有表的所有权限；**

**Rovoke all privileges on db2020.\* from 用户名’@’ip地址’; 取消该用户对db2020数据库中所有表的全部权限；**

1. **刷新授权操作：flush privileges;**

**实验四 SQL数据完整性（3学时）**

主要考查主键，外键如何约束完整性，以及考查触发器在数据库中的作用。

1. 了解参照完整性中外键设置的 on delete cascade/set null/no action（默认，同restrict）的原理和区别，on update 也是类似的。
2. 通过实验4提供的sql脚本创建表和插入数据（自己可以看看lab4.sql脚本中的sql语句）q
   1. 考查主键完整性约束， 在employee表中插入一名名为Betty（主码）的员工（通常我们不会把名字作为主键，考虑重名，一般是在前面加一个id字段，这里不作为考虑）

mysql haoge@(none):db2021\_lab4> insert into employee(employee\_name) values("Betty");

(1062, "Duplicate entry 'Betty' for key 'employee.PRIMARY'")

* 1. 考查 on delete no action/ restrict， 删除employee表中员工名字为Betty的数据，发现error（因为带有外键约束不能删除）

mysql haoge@(none):db2021\_lab4> delete from employee where employee\_name = "Betty";

You're about to run a destructive command.

Do you want to proceed? (y/n): y

Your call!

(1451, 'Cannot delete or update a parent row: a foreign key constraint fails (`db2021\_lab4`.`works`, CONSTRAINT `works\_ibfk\_1` FOREIGN KEY (`employee\_name`) REFERENCES `employee` (`employee\_name`))')

* 1. 接下来我们需要创建（或者修改）另外的几个外键（可以通过直接在sql脚本修改以及新添加外键约束的sql语句，然后直接重新运行脚本；也可以在数据库中删除原有的外键，然后把新的外键约束通过alter命令添加）
     1. works表中的employee\_name 参照 父表employee中的employee\_name 在删除时级联删除，更新时级联更新，
     2. works表中的comanpy\_name参照父表company中的company\_name在删除时设置为null，更新时级联更新，
     3. manages表中的employee\_name参照父表employee中的employee\_name 在删除时级联删除，更新时级联更新,
     4. manages表中的manager\_name参照父表employee中的employee\_name 在删除时设置为null，更新时级联更新,
  2. 接下来再删除employee表中员工名字为Betty的数据，查看另外几个表关于Betty的数据有什么变化。

employee:

+---------------+--------+------------+

| employee\_name | street | city |

+---------------+--------+------------+

| Amanda | King | Birmingham |

| Dean | Dover | Sheffield |

| Kevin | King | Birmingham |

| Lily | Regent | Nottingham |

| Mark | Dover | Sheffield |

| Richard | Strand | Cambridge |

| Whitney | Regent | Nottingham |

+---------------+--------+------------+

company:

+--------------+------------+

| company\_name | city |

+--------------+------------+

| Bertelsmann | Cambridge |

| Chloris | Nottingham |

| Eldat GmbH | Birmingham |

| Greenteam | Cambridge |

+--------------+------------+

works:

+---------------+--------------+--------+

| employee\_name | company\_name | salary |

+---------------+--------------+--------+

| Amanda | Greenteam | 2800 |

| Dean | Eldat GmbH | 3100 |

| Kevin | Greenteam | 2700 |

| Lily | Eldat GmbH | 3800 |

| Mark | Chloris | 4000 |

| Richard | Bertelsmann | 3000 |

| Whitney | Chloris | 2500 |

+---------------+--------------+--------+

manages:

+---------------+--------------+

| employee\_name | manager\_name |

+---------------+--------------+

| Richard | <null> |

| Kevin | Amanda |

| Dean | Lily |

| Whitney | Mark |

+---------------+--------------+

* 1. 修改员工表中姓名为Amanda的员工名字为Amy，查看另外几个表原本关于Amanda的数据有什么变化。

employee:

+---------------+--------+------------+

| employee\_name | street | city |

+---------------+--------+------------+

| Amy | King | Birmingham |

| Dean | Dover | Sheffield |

| Kevin | King | Birmingham |

| Lily | Regent | Nottingham |

| Mark | Dover | Sheffield |

| Richard | Strand | Cambridge |

| Whitney | Regent | Nottingham |

+---------------+--------+------------+

company:

+--------------+------------+

| company\_name | city |

+--------------+------------+

| Bertelsmann | Cambridge |

| Chloris | Nottingham |

| Eldat GmbH | Birmingham |

| Greenteam | Cambridge |

+--------------+------------+

works:

+---------------+--------------+--------+

| employee\_name | company\_name | salary |

+---------------+--------------+--------+

| Amy | Greenteam | 2800 |

| Dean | Eldat GmbH | 3100 |

| Kevin | Greenteam | 2700 |

| Lily | Eldat GmbH | 3800 |

| Mark | Chloris | 4000 |

| Richard | Bertelsmann | 3000 |

| Whitney | Chloris | 2500 |

+---------------+--------------+--------+

manages:

+---------------+--------------+

| employee\_name | manager\_name |

+---------------+--------------+

| Richard | <null> |

| Kevin | Amy |

| Dean | Lily |

| Whitney | Mark |

+---------------+--------------+

* 1. 考查触发器，这里我们在company表中增加一条属性tot\_salary, 也就是本公司所以在职的员工总工资，通过写一条更新sql语句更新tot\_salary值，最后创建三个触发器，分别是在insert，update，delete works表数据后，对相应的tot\_salary进行更新操作。（exp：当插入一条A公司的员工数据时候，此员工薪水为2000，则A公司总薪水需要加上2000。）

create view tmp as select company\_name,sum(salary) as s from works group by company\_name;

update company set tot\_salary = (select s from tmp where company.company\_name = tmp.company\_name);

create trigger `Tri\_del`

after insert on `works`

for each row

update company set tot\_salary = tot\_salary + new.salary where(company\_name = new.company\_

name);

create trigger `Tri\_ins`

after insert on `works`

for each row

update company set tot\_salary = tot\_salary + new.salary where(company\_name = new.company\_

name);

create trigger `Tri\_upd`

after update on `works`

for each row

update company set tot\_salary = tot\_salary + new.salary - old.salary where(company\_name =

new.company\_name);

**实验五 数据库索引（1学时）**

考查数据库索引在数据库查询的性能优化。

因为要比较出有无索引的差别，需要大量的数据，我们可以通过执行lab5.sql的脚本生成数据表以及插入数据。（由于主键是唯一性索引，我这里没有定义主键。）

1. 然后通过执行以下三条左连接sql语句观察查询时间：
   1. select \* from my\_test\_table1 a LEFT JOIN my\_test\_table2 b on a.id=b.id;
   2. select \* from my\_test\_table1 a LEFT JOIN my\_test\_table2 b on a.id=b.id left join my\_test\_table1 c on a.id=b.id and b.id=c.id;
   3. select \* from my\_test\_table1 a LEFT JOIN my\_test\_table2 b on a.id=b.id left join my\_test\_table1 c on a.id=b.id and b.id=c.id left join my\_test\_table2 d on a.id=b.id and b.id=c.id and c.id=d.id;
2. 对数据表1和数据表2中的id字段分别建立普通索引；
3. 接下来再执行1中的sql语句查看性能差异

注：本实验考查的只是简单的索引性能实践，想要更深入了解索引底层的原理知识可以去查阅相关书籍。