

day23_数据库基本操作

一 内容回顾（列举前一天重点难点内容）

1.1 教学重点:

1. 网络编程基础
2. IP地址、端口、通信协议
3. 相关类
4. TCP编程模型
5. 了解UDP编程

二 教学目标

1. 熟练安装数据库
2. 熟练掌握基本操作DDL, DML
3. 熟练掌握DQL的基本操作

三 教学导读

3.1. 数据库简介(会)

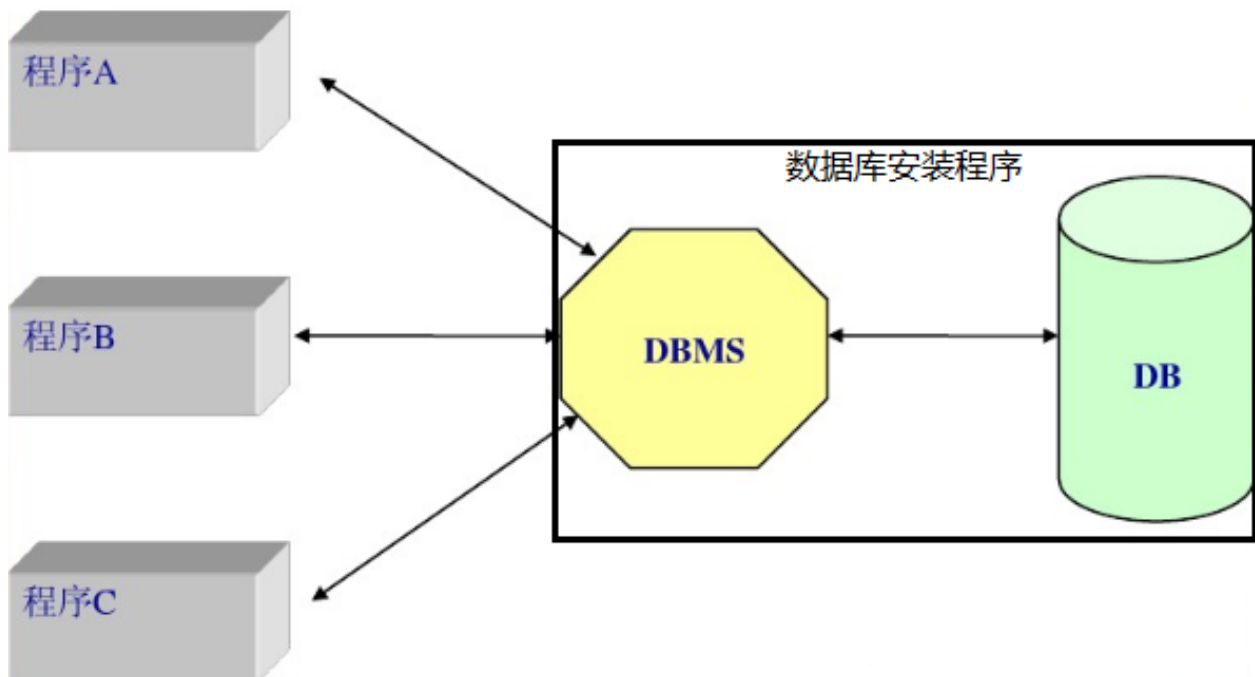
数据库（DataBase，DB）：指长期保存在计算机的存储设备上，按照一定规则组织起来，可以被各种用户或应用共享的数据集合。

数据库管理系统（DataBase Management System，DBMS）：指一种操作和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理和控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中的数据。

数据库软件应该为数据库管理系统，数据库是通过数据库管理系统创建和操作的。

数据库：存储、维护和管理数据的集合。

- DB与DBMS关系



3.2. 数据库分类(会)

- 1.关系型数据库(sql)
- 2.非关系型数据库(nosql)

3.3. 常见关系型数据库

Oracle:是Oracle公司的数据库产品

Mysql: 最早属于瑞典的MysqlAB公司的，后被Sun公司收购，Sun在2009年4月20号被Oracle收购。

SQLServer:微软旗下的数据库产品

Access：微软旗下的数据库产品

DB2:IBM公司旗下的数据库产品

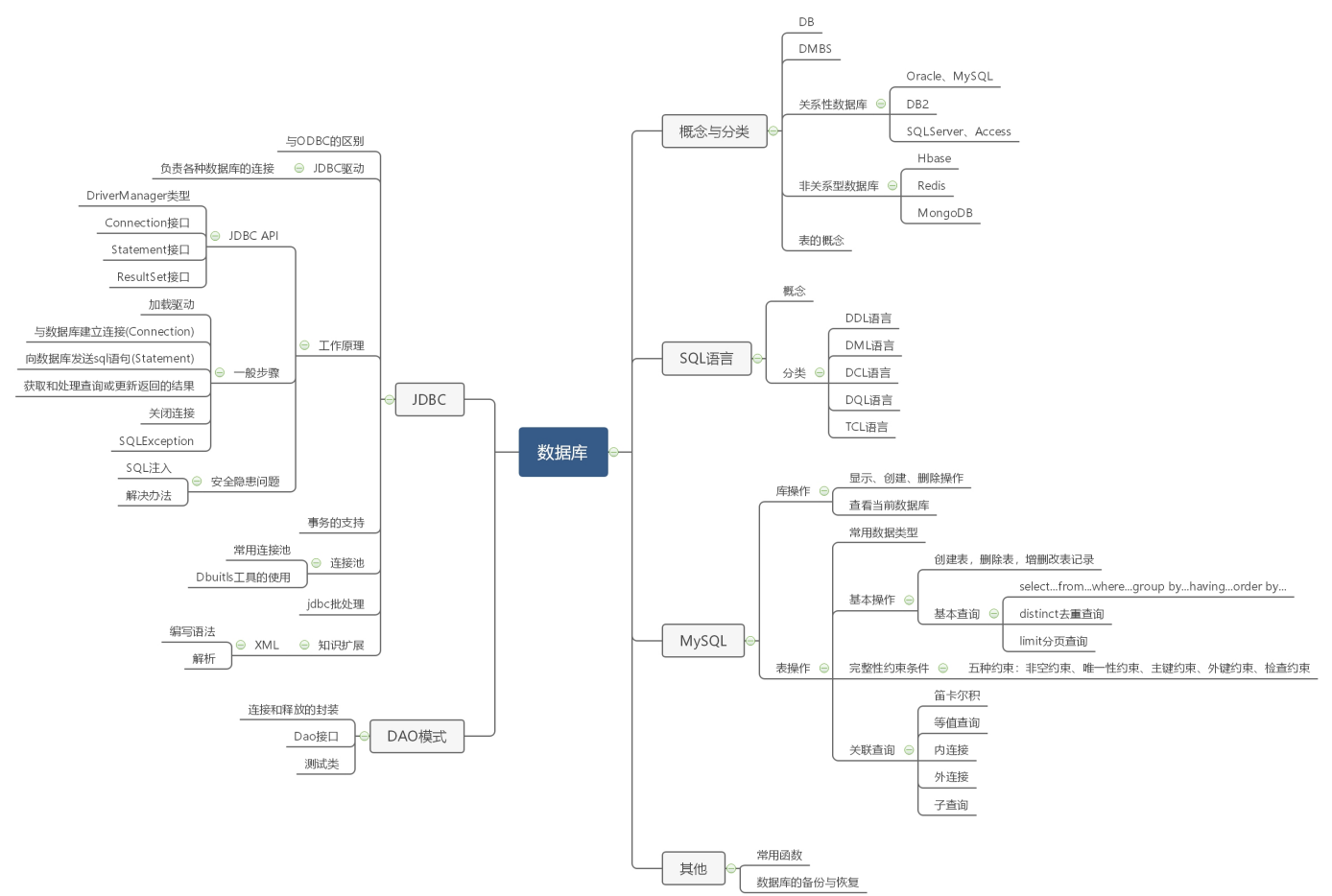
3.4. 常见的非关系型数据库

HBase是一个分布式的、面向列的开源数据库

MongoDB是一个基于分布式文件存储的数据库

Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API

3.5. 数据库思维导图



四 教学内容

4.1. 数据库的安装与配置

4.1.1. 安装

(所有相关的软件文档都已经上传网盘目录中链接: <https://pan.baidu.com/s/1xG-c9wOyTmGeuEYAEh0XZg> 密码: chbg)










- 使用的数据库版本是mysql----mysql-installer-community-5.7.25.0.msi
- 安装过程在文档mysql5.7安装.pdf

注意:首先检查电脑是否安装,Microsoft Visual C++2013运行库_x64,如果没有,要先进行安装.

(点击屏幕左下角按钮---找到设置---找到应用选项,查看下面界面)

— □ ×

应用和功能

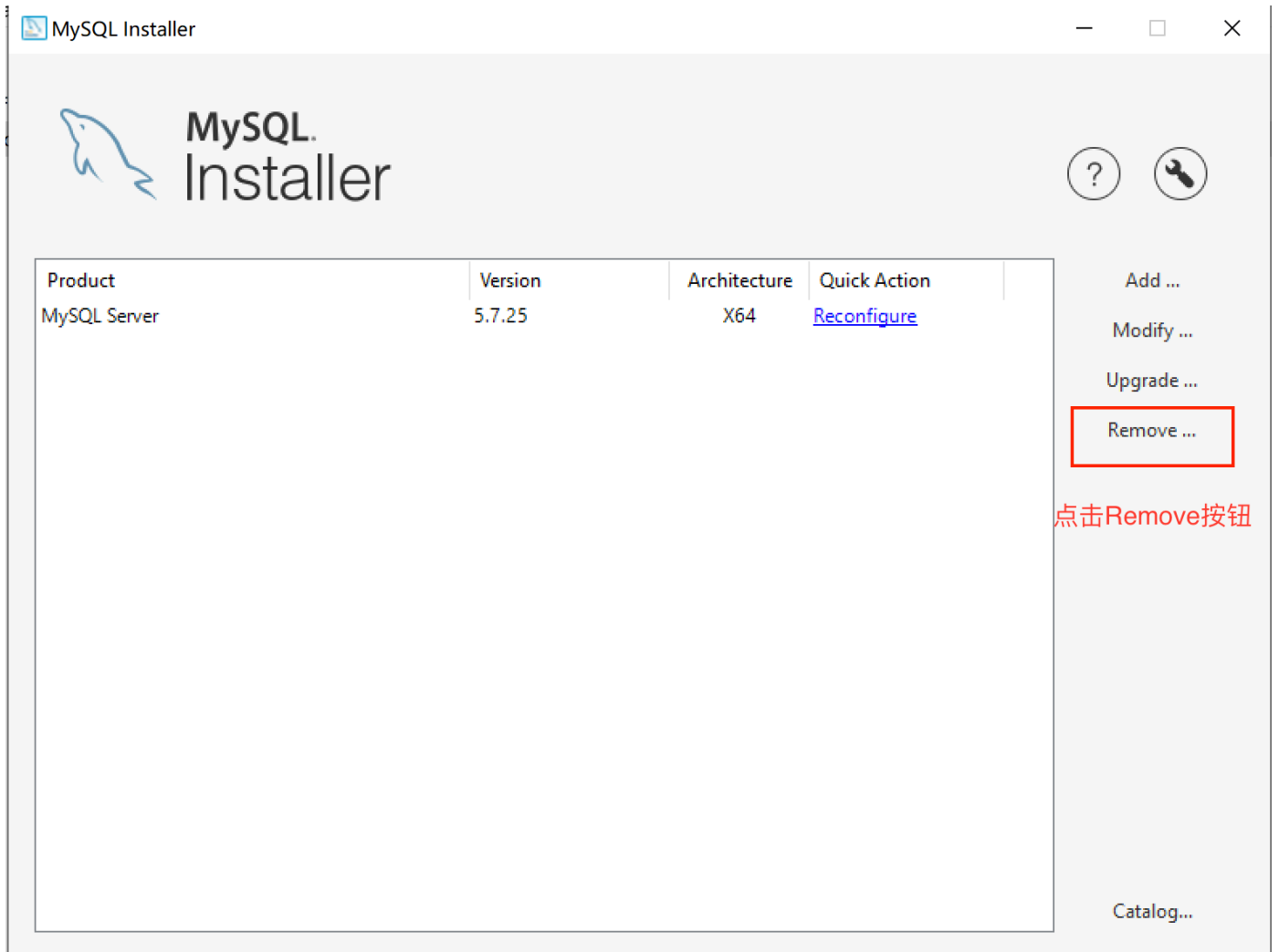
	Microsoft Edge Microsoft Corporation	13.9 MB 2020/8/31
	Microsoft Store Microsoft Corporation	49.8 KB 2021/3/26
	Microsoft Visual C++ 2010 x86 Redistributable - 1...	11.0 MB 2020/9/1
	Microsoft Visual C++ 2013 Redistributable (x64) -...	20.6 MB 2020/9/6
	Microsoft Visual C++ 2015-2019 Redistributable...	23.2 MB 2020/12/27
	Microsoft Visual C++ 2015-2019 Redistributable (...)	20.2 MB 2020/12/11
	Microsoft 照片 Microsoft Corporation	6.62 MB 2021/3/8
	MySQL Server 5.7	289 MB 2020/9/7
	Parallels Tools	22.3 MB 2020/11/18

- 安装失败的可能原因:

- 1.关闭防火墙
- 2.查看系统版本与mysql版本
- 3.检查配置是否正确
- 4.电脑老旧
- 5.尝试以管理员权限安装mysql
- 6.卸载完先关机重启

4.1.2. 卸载

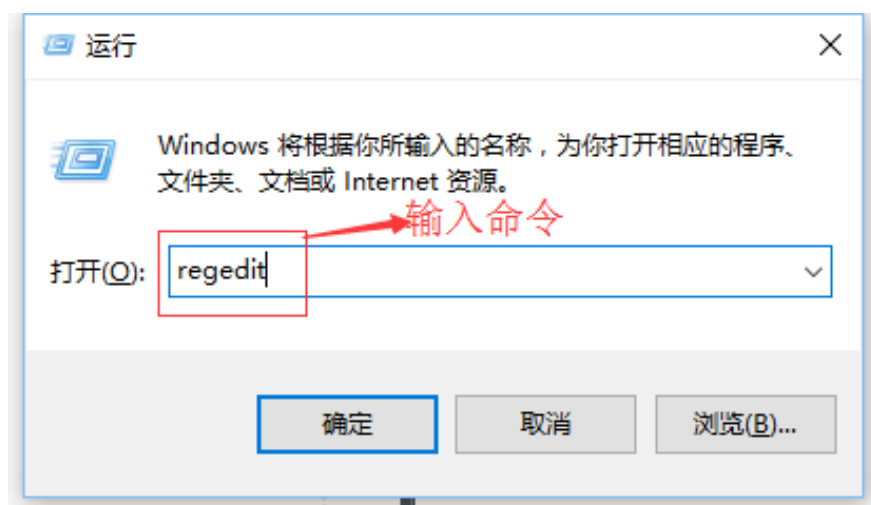
有些人，在卸载某一款软件时，没有使用卸载程序，而是先直接删除安装路径下的文件及其文件夹，这样就会造成注册表的残留，会影响此软件的下一次安装，造成安装失败。现在，市场上的90%的软件安装包内都自带卸载程序，因此，想卸载某款软件时，可以再次双击软件安装包，或者在开始菜单查找卸载程序 uninstall.....，或者在控制面板内卸载软件。避免注册表的残留。MySQL的卸载，只需要双击安装程序，然后出现以下界面:点击Remove即可



点击Remove按钮

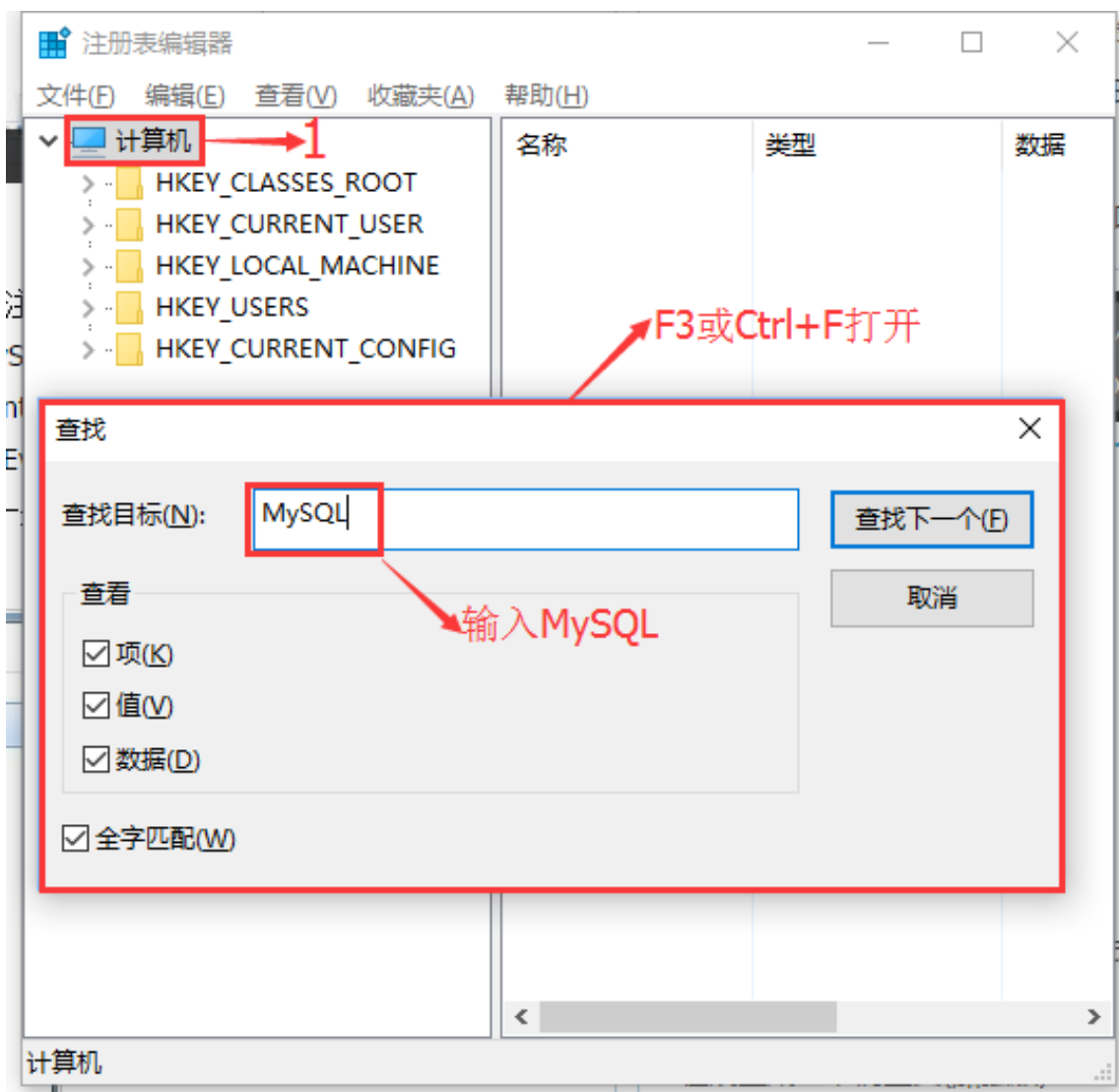
- 注意:上面的正常情况是可以卸载完成的,如果发生卸载步骤不对,产生错误,涉及到了注册表,操作如下

step1: Windows+R-->regedit-->打开注册表



step2: 根据路径打开并删除:

- 1 HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001/Services/Eventlog/Application/MySQL
- 2 HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services/Eventlog/Application/MySQL57
- 3 HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Services一般服务会以相同的名字(名字通常是MySQL)



4.1.3. 运行

安装成功了打开cmd --> mysql -uroot -p直接敲回车

然后输入你的密码

4.1.4. 修改mysql root用户密码

1.停止mysql服务 运行输入services.msc 停止mysql服务 或者 cmd --> net stop mysql

2.在cmd下 输入 mysqld --skip-grant-tables 启动服务器 光标不动 （不要关闭该窗口）

3.新打开cmd 输入mysql -u root -p 不需要密码

```
use mysql; update user set password=password('abc') WHERE User='root';
```

4.关闭两个cmd窗口 在任务管理器结束mysqld 进程

5.在服务管理页面 重启mysql 服务,密码修改完成

4.1.5. 解决编码问题

当我们在mysql5.7数据库中创建默认表的时候默认是latin1 编码集即是iso-8859-1 这样的编码集我们向表中出入数据的时候是无法插入中文的,所以针对此问题提供如下解决方案:

方案一

```
1 我们在创建数据库时候可以指定编码集为gbk/utf8
2 create database mydb4 character set 'utf8';
3
4 或
5
6 create database mydb4 character set 'gbk';
7 以后再在此数据库中创建的表是默认gbk/utf8的编码
```

方案二

数据库已经默认存在了而且编码集是latin1,那么我们可以在创建表的时候指定表的默认编码集,在建表语句的最后添加DEFAULT CHARSET=utf8或gbk

```
1 CREATE TABLE t_1 (  
2     id int(11)  
3 )DEFAULT CHARSET=utf8;  
4  
5 或  
6  
7 CREATE TABLE t_1 (  
8     id int(11)  
9 )DEFAULT CHARSET=gbk;
```

方案三

数据库已经默认存在了,表也默认存在了还是latin1编码集

```
1 修改表的编码集  
2 alter table 表名 convert to character set utf8;  
3 或  
4 alter table 表名 convert to character set gbk;  
5  
6 修改数据库编码集  
7 alter database 数据库名 character set utf8;  
8 或  
9 alter database 数据库名 character set gbk;
```

以上无论是使用哪种方式,一定要完全退出数据库后在重新进入就会发现字符集被修改了

方案四

以上三种方案都是治标不治本的,因为都是修改的其中一部分,但是以后建库或建表的时候还是latin1,这是因为我们在安装数据库的时候,mysql5.7的默认安装编码集是latin1,其实我们可以在cmd模式下查看编码集

show variables like 'character%';

```
mysql> show variables like 'char%';
```

Variable_name	Value
character_set_client	gbk
character_set_connection	gbk
character_set_database	latin1
character_set_filesystem	binary
character_set_results	gbk
character_set_server	latin1
character_set_system	utf8
character_sets_dir	C:\Program Files\MySQL\MySQL Server 5.7\share\charsets\

数据库和服务默认都是 latin1

所以我们要修改mysql安装路径和下的my.ini文件

文件所在路径

C:\ProgramData\MySQL\MySQL Server 5.7

Ps:可能每个文的路径不同,但是安装的位置没有修改前提下大家都是相同的,如果修改了安装路径,那么就在你所安装的盘符下搜索my.ini文件,如果没有那一定是在C盘

使用编译文件软件或是使用记事本打开在下图标注的位置添加这两句话

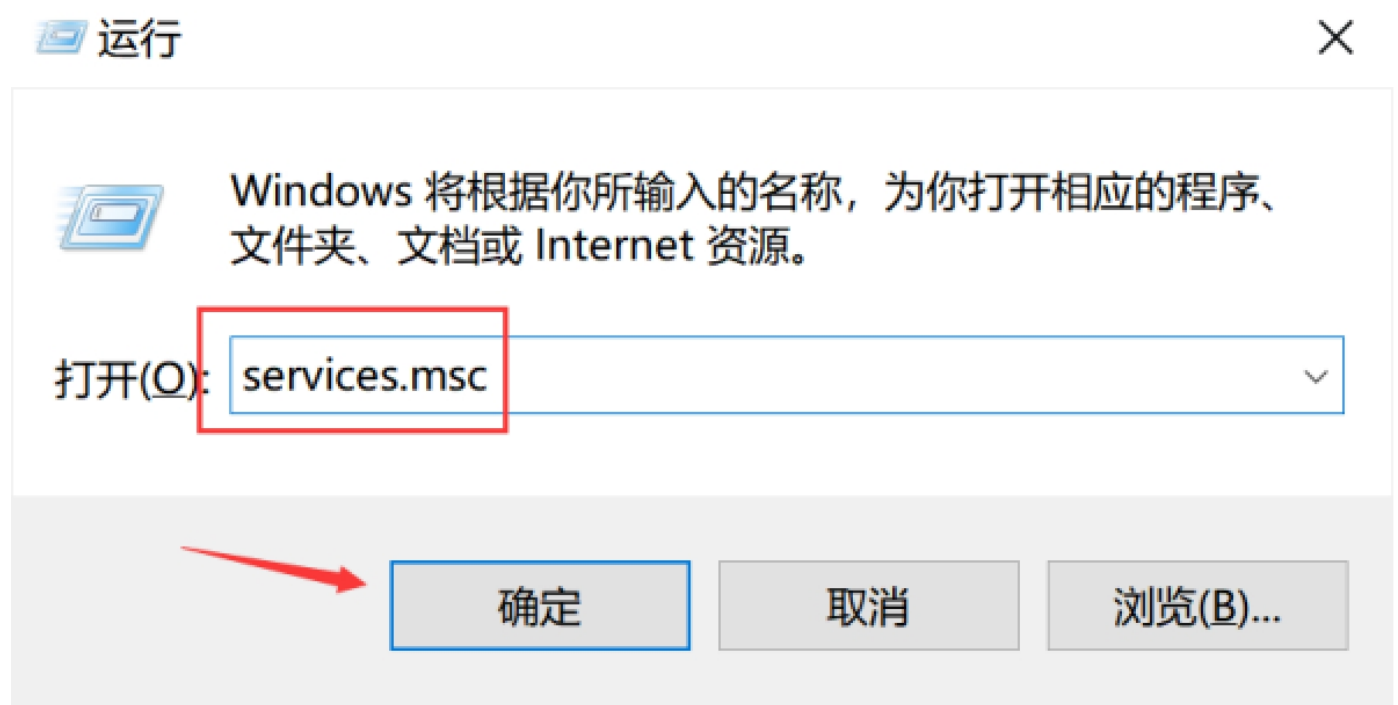
```
[mysql]
no-beep
default-character-set=utf8
# default-character-set=

# SERVER SECTION
# -----
#
# The following options will be read by the MySQL Server
# you have installed the server correctly (see above) so
# file.
#
# server_type=3
[mysqld]
character-set-server=utf8
# The next three options are mutually exclusive to SERVER
```

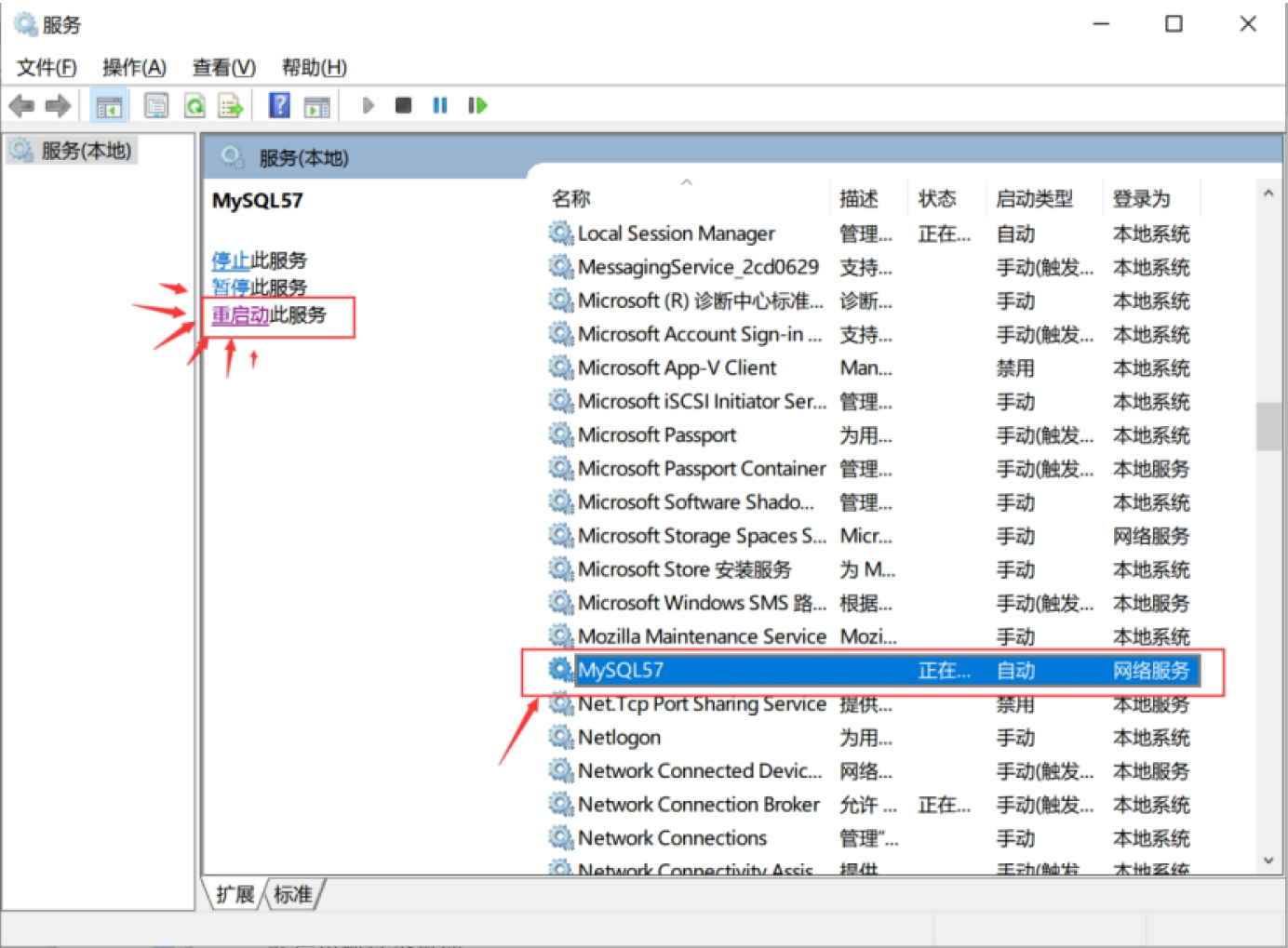
添加这两句

添加完成后保存退出

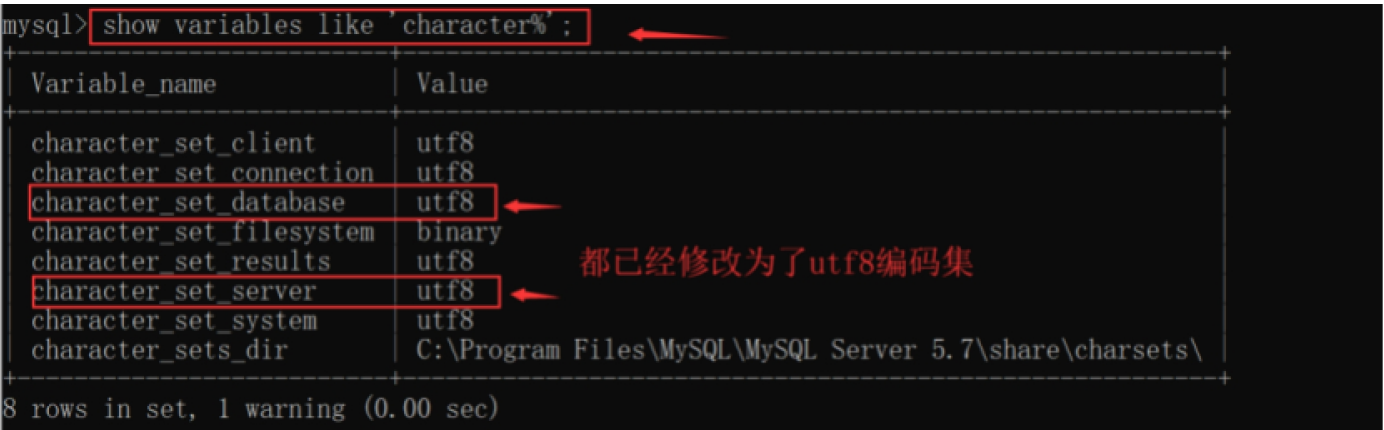
windows键+R 出现运行界面 在打开的后面输入



然后出现这个界面,点击重启此服务



然后在通过cmd界面就到mysql中再次查询编码集



此时在创建库也好,创建表也好utf8编码集了

4.1.6. 可视化工具

1.navicat

2.SQlyog

两个都可以使用,我上课使用SQlyog,建议大家先安装这个.

4.2. sql概述

SQL: Structure Query Language。（结构化查询语言）SQL被美国国家标准局（ANSI）确定为关系型数据库语言的美国标准，后来被国际化标准组织（ISO）采纳为关系数据库语言的国际标准。各数据库厂商都支持ISO的SQL标准。

各数据库厂商在标准的基础上做了自己的扩展。

4.3. Sql的分类

DDL（Data Definition Language）：数据定义语言，用来定义数据库对象：库、表、列等；

DML（Data Manipulation Language）：数据操作语言，用来定义数据库记录（数据）；

DCL（Data Control Language）：数据控制语言，用来定义访问权限和安全级别；

DQL(重要)（Data Query Language）：数据查询语言，用来查询记录（数据）。注意：sql语句以;结尾

4.4. Sql详情(会)

4.4.1. DDL操作

定义:操作数据库、表、列等

使用的关键字: **CREATE**、**ALTER**、**DROP**

4.4.1.1. 操作数据库

- 创建

```
1 Create database mydb1;  
2 Create database mydb2 character set gbk;  
3 Create database mydb3 character set gbk COLLATE  
  gbk_chinese_ci;
```

- 查询

```
1 查看当前数据库服务器中的所有数据库  
2 Show databases;  
3  
4 查看前面创建的mydb2数据库的定义信息  
5 Show create database mydb2;  
6  
7 删除前面创建的mydb3数据库  
8 Drop database mydb3;
```

- 修改

查看服务器中的数据库, 并把mydb2的字符集修改为utf8;

```
1 alter database mydb2 character set utf8;
```

- 删除

```
1 Drop database mydb3;
```

- 其他

查看当前使用的数据库

```
1 Select database();
```

切换数据库

```
1 Use mydb2;
```

4.4.1.2. 操作数据表

- 语法

```
1 create table 表名(  
2     字段1 字段类型,  
3     字段2 字段类型,  
4     ...  
5     字段n 字段类型  
6 );
```

- 常用数据类型

```
1  int: 整型
2  double: 浮点型, 例如double(5,2)表示最多5位, 其中必须有2位小
    数, 即最大值为999.99;
3  char: 固定长度字符串类型;
4  char(10): 'aaa' 占10位
5  varchar: 可变长度字符串类型;
6  varchar(10): 'aaa' 占3为
7  text: 字符串类型;
8  blob: 字节类型;
9  date: 日期类型, 格式为: yyyy-MM-dd;
10 time: 时间类型, 格式为: hh:mm:ss
11 timestamp: 时间戳类型 yyyy-MM-dd hh:mm:ss 会自动赋值
12 datetime: 日期时间类型 yyyy-MM-dd hh:mm:ss
13 boolean: mysql不支持, oracle支持
```

● 常用操作

```
1  查看当前数据库中的所有表
2  SHOW TABLES;
3
4  查看表的字段信息
5  DESC employee;
6
7  在上面员工表的基本上增加一个image列。
8  ALTER TABLE employee ADD image blob;
9
10 修改job列, 使其长度为60。
11 ALTER TABLE employee MODIFY job varchar(60);
12
13 删除image列, 一次只能删一列。
14 ALTER TABLE employee DROP image;
15
16 表名改为user。
```



```
17 RENAME TABLE employee TO user;
18
19 查看表格的创建细节
20 SHOW CREATE TABLE user;
21
22 修改表的字符集为gbk
23 ALTER TABLE user CHARACTER SET gbk;
24
25 列名name修改为username
26 ALTER TABLE user CHANGE name username varchar(100);
27
28 备份表结构和表数据
29 create table tname2 as select * from tname1;
30
31 备份表结构
32 create table tname2 like tname1;
33
34 删除表
35 DROP TABLE user ;
```

4.4.2. DML操作(重要)

DML是对表中的数据进行增、删、改的操作。不要与DDL混淆了。

INSERT、UPDATE、DELETE

小知识：在mysql中，字符串类型和日期类型都要用单引号括起来。空值：null

4.4.2.1. 插入操作

- 语法：

INSERT INTO 表名 (列名1, 列名2 ...) VALUES(列值1, 列值2...);

注意：列名与列值的类型、个数、顺序要一一对应。

可以把列名当做java中的形参，把列值当做实参。值不要超出列定义的长度。如果插入空值，请使用null,插入的日期和字符一样，都使用单引号括起来。

- 课上练习

```
1  create table emp(  
2      id int,  
3      name varchar(100),  
4      gender varchar(10),  
5      birthday date,  
6      salary float(10,2),  
7      entry_date date,  
8      resume text  
9  );  
10  
11  INSERT INTO  
    emp(id,name,gender,birthday,salary,entry_date,resume)  
12  VALUES(1,'zhangsan','female','1990-5-10',10000,'2015-5-  
    5-','good girl');  
13  
14  INSERT INTO  
    emp(id,name,gender,birthday,salary,entry_date,resume)  
15  VALUES(2,'lisi','male','1995-5-10',10000,'2015-5-  
    5','good boy');  
16  
17  INSERT INTO  
    emp(id,name,gender,birthday,salary,entry_date,resume)  
18  VALUES(3,'你好','male','1995-5-10',10000,'2015-5-  
    5','good boy');
```

- 小知识

```
1 查看数据库编码的具体信息 Show variables like 'character%';
2
3 临时更改客户端和服务结果集的编码
4 Set character_set_client=gbk;
5 Set character_set_results=gbk;
```

第二种插入数据的方法 insert into 新表 select 列 from 已有表

```
1 create table emp1(
2     id int,
3     name varchar(100),
4     gender varchar(10),
5     birthday date,
6     salary float(10,2),
7     entry_date date,
8     resume text
9 );
10
11 INSERT INTO emp1 select * from emp 将emp表的所有数据复制到emp1中
```

- 注意:

1.mysql不支持select into

2.在复制时是按照列的顺序依次进行

3.新表的列与原来表的列的名字,类型都可以不一样,照样复制成功.但是会出现数据转换错误.具体的:名字不一样,问题不大.类型不一样,出现错误,比如将int型的数据强制转换成varchar型,会显示0,反之亦然

4.4.2.2. 修改操作

- 语法

UPDATE 表名 SET 列名1=列值1, 列名2=列值2 。。。 WHERE 列名=值

- 课上练习

```
1  将所有员工薪水修改为5000元。
2  UPDATE emp SET salary=5000
3
4  将姓名为'zs'的员工薪水修改为3000元。
5  UPDATE emp SET salary=3000 WHERE name=' zhangsan';
6
7  将姓名为'aaa'的员工薪水修改为4000元, job改为ccc。
8  UPDATE emp SET salary=4000, gender='female' WHERE
   name='lisi';
9
10 将wu的薪水在原有基础上增加1000元。
11 UPDATE emp SET salary=salary+1000 WHERE gender='male';
```

4.4.2.3. 删除操作

- 语法

DELETE FROM 表名 【WHERE 列名=值】

- 课上练习

```
1 删除表中名称为'zs'的记录。
2 DELETE FROM emp WHERE name='zs';
3
4 删除表中所有记录。
5 DELETE FROM emp;
6
7 使用truncate删除表中记录。T
8 TRUNCATE TABLE emp;
```

- truncate和delete的区别

DELETE 删除表中的数据，表结构还在;删除后的数据可以找回 TRUNCATE 删除是把表直接DROP掉，然后再创建一个同样的新表。Truncate删除的数据不能找回。执行速度比DELETE快。

4.4.3. DQL操作

DQL数据查询语言 （重要）

数据库执行DQL语句不会对数据进行改变，而是让数据库发送结果集给客户端。查询返回的结果集是一张虚拟表。

- 语法

SELECT 列名 FROM 表名

【WHERE --> GROUP BY --> HAVING--> ORDER BY LIMIT】

语法子句说明：

SELECT selection_list /要查询的列名称/

FROM table_list /要查询的表名称/

WHERE condition /行条件/

GROUP BY grouping_columns /对结果分组/

HAVING condition /分组后的行条件/

ORDER BY sorting_columns /对结果排序/

LIMIT offset_start, row_count /结果限定/

- 课上练习

创建学生表,部门表,职员表

学生表： stu

字段名称	字段类型	说明
sid	char(6)	学生学号
sname	varchar(50)	学生姓名
age	int	学生年龄
gender	varchar(50)	学生性别

```
1 CREATE TABLE stu (  
2     sid CHAR(6),  
3     sname  VARCHAR(50),  
4     age    INT,  
5     gender VARCHAR(50)  
6 );  
7 INSERT INTO stu VALUES('S_1001', 'liuYi', 35, 'male');  
8 INSERT INTO stu VALUES('S_1002', 'chenEr', 15,  
9     'female');  
9 INSERT INTO stu VALUES('S_1003', 'zhangSan', 95,  
10     'male');  
10 INSERT INTO stu VALUES('S_1004', 'liSi', 65, 'female');
```

```

11 INSERT INTO stu VALUES('S_1005', 'wangWu', 55, 'male');
12 INSERT INTO stu VALUES('S_1006', 'zhaoLiu', 75,
    'female');
13 INSERT INTO stu VALUES('S_1007', 'sunQi', 25, 'male');
14 INSERT INTO stu VALUES('S_1008', 'zhouBa', 45,
    'female');
15 INSERT INTO stu VALUES('S_1009', 'wuJiu', 85, 'male');
16 INSERT INTO stu VALUES('S_1010', 'zhengShi', 5,
    'female');
17 INSERT INTO stu VALUES('S_1011', 'xxx', NULL, NULL);

```

雇员表：emp

字段名称	字段类型	说明
empno	int	员工编号
ename	varchar(50)	员工姓名
job	varchar(50)	员工工作
mgr	int	领导编号
hiredate	date	入职日期
sal	decimal(7,2)	月薪
comm	decimal(7,2)	奖金
deptno	int	部分编号

```

1 CREATE TABLE emp(
2     empno    INT,
3     ename    VARCHAR(50),

```

```
4      job    VARCHAR(50),
5      mgr     INT,
6      hiredate DATE,
7      sal     DECIMAL(7,2),
8      comm     decimal(7,2),
9      deptno   INT
10 ) ;
11 INSERT INTO emp values(7369, 'SMITH', 'CLERK', 7902, '1980-
12-17', 800, NULL, 20);
12 INSERT INTO emp
13 values(7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-
14-20', 1600, 300, 30);
15 INSERT INTO emp
16 values(7521, 'WARD', 'SALESMAN', 7698, '1981-02-
17-22', 1250, 500, 30);
18 INSERT INTO emp
19 values(7566, 'JONES', 'MANAGER', 7839, '1981-04-
20-02', 2975, NULL, 20);
21 INSERT INTO emp
22 values(7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-
23-28', 1250, 1400, 30);
24 INSERT INTO emp
25 values(7698, 'BLAKE', 'MANAGER', 7839, '1981-05-
26-01', 2850, NULL, 30);
27 INSERT INTO emp
28 values(7782, 'CLARK', 'MANAGER', 7839, '1981-06-
29-09', 2450, NULL, 10);
30 INSERT INTO emp
31 values(7788, 'SCOTT', 'ANALYST', 7566, '1987-04-
32-19', 3000, NULL, 20);
33 INSERT INTO emp
34 values(7839, 'KING', 'PRESIDENT', NULL, '1981-11-
35-17', 5000, NULL, 10);
```



```

20 INSERT INTO emp
   values(7844, 'TURNER', 'SALESMAN', 7698, '1981-09-
      08', 1500, 0, 30);
21 INSERT INTO emp values(7876, 'ADAMS', 'CLERK', 7788, '1987-
      05-23', 1100, NULL, 20);
22 INSERT INTO emp values(7900, 'JAMES', 'CLERK', 7698, '1981-
      12-03', 950, NULL, 30);
23 INSERT INTO emp
   values(7902, 'FORD', 'ANALYST', 7566, '1981-12-
      03', 3000, NULL, 20);
24 INSERT INTO emp
   values(7934, 'MILLER', 'CLERK', 7782, '1982-01-
      23', 1300, NULL, 10);

```

部分表: dept

字段名称	字段类型	说明
deptno	int	部分编码
dname	varchar(50)	部分名称
loc	varchar(50)	部分所在地点

```

1 CREATE TABLE dept(
2   deptno    INT,
3   dname     varchar(14),
4   loc       varchar(13)
5 );
6 INSERT INTO dept values(10, 'ACCOUNTING', 'NEW YORK');
7 INSERT INTO dept values(20, 'RESEARCH', 'DALLAS');
8 INSERT INTO dept values(30, 'SALES', 'CHICAGO');
9 INSERT INTO dept values(40, 'OPERATIONS', 'BOSTON');

```

4.4.3.1. 基础查询

1. 查询所有列

```
1 | SELECT * FROM stu;
```

2. 查询指定列

```
1 | SELECT sid, sname, age FROM stu;
```

4.4.3.2. 条件查询

- 语法

条件查询介绍

条件查询就是在查询时给出WHERE子句，在WHERE子句中可以使用如下运算符及关键字：

=、!=、<>、<、<=、>、>=；BETWEEN...AND；IN(set)；IS NULL；AND；OR；NOT,XOR (逻辑异或)；

例如:SELECT 1 XOR 0 返回结果 1

- 课上练习

1. 查询性别为女，并且年龄小于50的记录

```
1 | SELECT * FROM stu
2 | WHERE gender='female' AND age<50;
```

2. 查询学号为S_1001，或者姓名为liSi的记录

```
1 | SELECT * FROM stu WHERE sid ='S_1001' OR sname='liSi';
```

3. 查询学号为S_1001，S_1002，S_1003的记录

```
1 | SELECT * FROM stu WHERE sid IN  
   ('S_1001', 'S_1002', 'S_1003');
```

4.4.3.3. 模糊查询

- 语法

当想查询姓名中包含a字母的学生时就需要使用模糊查询了。模糊查询需要使用关键字LIKE。

通配符:

_ 任意一个字母

%: 任意0~n个字母 '张%'

- 课上练习

1.查询姓名由5个字母构成的学生记录

```
1 | SELECT * FROM stu WHERE sname LIKE '_____';
```

模糊查询必须使用LIKE关键字。其中“_”匹配任意一个字母，5个“_”表示5个任意字母。

2.正则表达式:(了解)

MySQL 同样也支持其他正则表达式的匹配，MySQL中使用 REGEXP 操作符来进行正则表达式匹配。

1	模式	描述
2	<code>^</code>	匹配输入字符串的开始位置。如果设置了 <code>RegExp</code> 对象的 <code>Multiline</code> 属性, <code>^</code> 也匹配 <code>'\n'</code> 或 <code>'\r'</code> 之后的位置。
3	<code>\$</code>	匹配输入字符串的结束位置。如果设置了 <code>RegExp</code> 对象的 <code>Multiline</code> 属性, <code>\$</code> 也匹配 <code>'\n'</code> 或 <code>'\r'</code> 之前的位置。
4	<code>.</code>	匹配除 <code>"\n"</code> 之外的任何单个字符。要匹配包括 <code>'\n'</code> 在内的任何字符, 请使用象 <code>'[\.\n]'</code> 的模式。
5	<code>[...]</code>	字符集合。匹配所包含的任意一个字符。例如, <code>'[abc]'</code> 可以匹配 <code>"plain"</code> 中的 <code>'a'</code> 。
6	<code>[^...]</code>	负值字符集合。匹配未包含的任意字符。例如, <code>'[^abc]'</code> 可以匹配 <code>"plain"</code> 中的 <code>'p'</code> 。
7	<code>p1 p2 p3</code>	匹配 <code>p1</code> 或 <code>p2</code> 或 <code>p3</code> 。例如, <code>'z food'</code> 能匹配 <code>"z"</code> 或 <code>"food"</code> 。 <code>'(z f)ood'</code> 则匹配 <code>"zood"</code> 或 <code>"food"</code> 。
8	<code>*</code>	匹配前面的子表达式零次或多次。例如, <code>zo*</code> 能匹配 <code>"z"</code> 以及 <code>"zoo"</code> 。 <code>*</code> 等价于 <code>{0,}</code> 。
9	<code>+</code>	匹配前面的子表达式一次或多次。例如, <code>'zo+'</code> 能匹配 <code>"zo"</code> 以及 <code>"zoo"</code> , 但不能匹配 <code>"z"</code> 。 <code>+</code> 等价于 <code>{1,}</code> 。
10	<code>{n}</code>	<code>n</code> 是一个非负整数。匹配确定的 <code>n</code> 次。例如, <code>'o{2}'</code> 不能匹配 <code>"Bob"</code> 中的 <code>'o'</code> , 但是能匹配 <code>"food"</code> 中的两个 <code>o</code> 。
11	<code>{n,m}</code>	<code>m</code> 和 <code>n</code> 均为非负整数, 其中 <code>n <= m</code> 。最少匹配 <code>n</code> 次且最多匹配 <code>m</code> 次。

例子:

查询名字以l开头,以i结尾的.

```

1 select * from stu where name regexp '^l|i$'
2
3 SELECT 'hello' REGEXP '^he'          结果:1 表示匹配
4 SELECT 'hello' REGEXP '^hh'          结果:0 表示不匹配

```

4.4.3.4. 字段控制查询

1. 去除重复记录

去除重复记录（两行或两行以上记录中系列的上数据都相同），例如 emp 表中 sal 字段就存在相同的记录。当只查询 emp 表的 sal 字段时，那么会出现重复记录，那么想去除重复记录，需要使用 DISTINCT：

```
1 | SELECT DISTINCT sal FROM emp;
```

2. 查看雇员的月薪与佣金之和

因为 sal 和 comm 两列的类型都是数值类型，所以可以做加运算。如果 sal 或 comm 中有一个字段不是数值类型，那么会出错。

```
1 | SELECT *,sal+comm FROM emp;
```

comm 列有很多记录的值为 NULL，因为任何东西与 NULL 相加结果还是 NULL，所以结算结果可能会出现 NULL。下面使用了把 NULL 转换成数值 0 的函数 IFNULL：

```
1 | SELECT *,sal+IFNULL(comm,0) FROM emp;
```

3. 给列名添加别名

在上面查询中出现列名为 sal+IFNULL(comm,0)，这很不美观，现在我们给这一列给出一个别名，为 total：

```
1 | SELECT *, sal+IFNULL(comm,0) AS total FROM emp;
```

给列起别名时，是可以省略 AS 关键字的：

```
1 | SELECT *,sal+IFNULL(comm,0) total FROM emp;
```

4.4.3.5. 排序

1.查询所有学生记录，按年龄升序排序

```
1 SELECT * FROM stu ORDER BY sage ASC;  
2 或者  
3 SELECT * FROM stu ORDER BY sage;
```

2.查询所有学生记录，按年龄降序排序

```
1 SELECT * FROM stu ORDER BY age DESC;
```

3.查询所有雇员，按月薪降序排序，如果月薪相同时，按编号升序排序

```
1 SELECT * FROM emp ORDER BY sal DESC, empno ASC;
```

4.4.3.6. 聚合函数

sum avg max min count

聚合函数是用来做纵向运算的函数：

COUNT(): 统计指定列不为NULL的记录行数；

MAX(): 计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；

MIN(): 计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；

SUM(): 计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；

AVG(): 计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；

1.COUNT

当需要纵向统计时可以使用COUNT()。

查询emp表中记录数：

```
1 | SELECT COUNT(*) AS cnt FROM emp;
```

查询emp表中有佣金的人数：

```
1 | SELECT COUNT(comm) cnt FROM emp;
```

注意，因为count()函数中给出的是comm列，那么只统计comm列非NULL的行数。

2.SUM和AVG

当需要纵向求和时使用sum()函数。

查询所有雇员月薪和：

```
1 | SELECT SUM(sal) FROM emp;
```

查询所有雇员月薪和，以及所有雇员佣金和：

```
1 | SELECT SUM(sal), SUM(comm) FROM emp;
```

3.MAX和MIN

查询最高工资和最低工资：

```
1 | SELECT MAX(sal), MIN(sal) FROM emp;
```

4.4.3.7. 分组查询

当需要分组查询时需要使用GROUP BY子句，例如查询每个部门的工资和，这说明要使用部门来分组。

注：凡和聚合函数同时出现的列名，则一定要写在group by 之后

1.分组查询

查询每个部门的部门编号和每个部门的工资和：

```
1 SELECT deptno, SUM(sal) FROM emp GROUP BY deptno;
```

2.HAVING子句

查询工资总和大于9000的部门编号以及工资和：

```
1 SELECT deptno, SUM(sal) FROM emp GROUP BY deptno
   HAVING SUM(sal) > 9000;
```

注：having与where的区别：

- 1 1.having是在分组后对数据进行过滤。
- 2 where是在分组前对数据进行过滤
- 3
- 4 2.having后面可以使用分组函数(统计函数)
- 5 where后面不可以使用分组函数。
- 6
- 7 WHERE是对分组前记录的条件，如果某行记录没有满足WHERE子句的条件，那么这行记录不会参加分组；而HAVING是对分组后数据的约束。

4.4.3.8. LIMIT

LIMIT用来限定查询结果的起始行，以及总行数。

1.查询5行记录，起始行从0开始

```
1 | SELECT * FROM emp LIMIT 0, 5; 注意，起始行从0开始，即第一行开始!
```

2.查询10行记录，起始行从3开始

```
1 | SELECT * FROM emp LIMIT 3, 10;
```

3.分页查询

如果一页记录为10条，希望查看第3页记录应该怎么查呢？

第一页记录起始行为0，一共查询10行；

第二页记录起始行为10，一共查询10行；

第三页记录起始行为20，一共查询10行；

currentPage = 1

count = 5;

((currentPage-1)*count,count)

4.4.3.9. 其他知识

- 查询语句书写顺序

select - from - where - group by - having - order by - limit

- 查询语句执行顺序

from - where -group by - having - select - order by-limit

- 运算符的优先级

最高优先级为：！、BINARY、COLLATE。

优先级顺序	运算符
1	:=
2	, OR, XOR
3	&&, AND
4	NOT
5	BETWEEN, CASE, WHEN, THEN, ELSE
6	=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
7	
8	&
9	<<, >>
10	~, +
11	*, /, DIV, %, MOD
12	^
13	-(一元减号), ~(一元比特反转)
14	!