MySQL基础回顾

WEB前端 + 后端 + 爬虫 + 数据分析 + 人工智能

1、数据库概念

数据库

- 存储数据的仓库（逻辑概念，并未真实存在）

数据库软件

- 真实软件，用来实现数据库这个逻辑概念

数据仓库

- 数据量更加庞大，更加侧重数据分析和数据挖掘，供企业决策分析之用，主要是数据查询，修改和删除很少

2、MySQL的特点

- 关系型数据库
- 跨平台
- 支持多种编程语言（python、java、php）

- 基于磁盘存储，数据是以文件形式存放在数据库目录/var/lib/mysql下

3、启动连接

- 服务端启动

```
sudo /etc/init.d/mysql
start|stop|restart|status
sudo service mysql start|stop|restart|status
```

- 客户端连接

```
mysql -hIP地址 -u用户名 -p密码
本地连接可省略 -h 选项
```

4、基本SQL命令

库管理

1、查看已有库；

2、创建库并指定字符集；

3、查看当前所在库；

4、切换库；

5、查看库中已有表；

6、删除库；

表管理

1、创建表并指定字符集；

2、查看创建表的语句 （字符集、存储引擎)；

3、查看表结构；

4、删除表；

表记录管理

1、增 ：
2、删 ：
3、改 ：
4、查 ：

表字段管理（alter table 表名)

```
1、增 ：
2、删 ：
3、改 ：
4、表重命名：
```

5、数据类型

四大数据类型

- 数值类型

```

```

- 字符类型

```

```

- 枚举类型

```

```

- 日期时间类型

```

```

日期时间函数

```

```

日期时间运算

```
select * from 表名 where 字段名 运算符(NOW()-interval 间隔);
间隔单位: 1 day | 3 month | 2 year
eg1:查询1年以前的用户充值信息
```

6、MySQL运算符

- **数值比较**

```
> >= < <= = !=
eg1 : 查询成绩不及格的学生

eg2 : 删除成绩不及格的学生

eg3 : 把id为3的学生的姓名改为 周芷若
```

- **逻辑比较**

```
and  or
eg1 : 查询成绩不及格的男生

eg2 : 查询成绩在60-70之间的学生
```

- **范围内比较**

```
between 值1 and 值2 、in() 、not in()
eg1 ： 查询不及格的学生姓名及成绩

eg2 ： 查询AID1905和AID1903班的学生姓名及成绩
```

- **模糊比较（like)**

```
where 字段名 like 表达式(%_)
eg1 ： 查询北京的姓赵的学生信息
```

- **NULL判断**

```
is NULL 、is not NULL
eg1 ： 查询姓名字段值为NULL的学生信息
```

7、查询

- **order by**

给查询的结果进行排序(永远放在SQL命令的倒数第二的位置写)

```
order by 字段名 ASC/DESC
eg1 ： 查询成绩从高到低排列
```

- **limit**

限制显示查询记录的条数（永远放在SQL命令的最后写）

```
limit n ：显示前n条
limit m,n ：从第(m+1)条记录开始，显示n条
分页：每页显示10条，显示第6页的内容
```

MySQL高级-Day01

MySQL基础巩固

- **创建库： country（指定字符编码为utf8)**
- **创建表： sanguo 字段：id 、name、attack、defense、gender、country**
  要求： id设置为主键,并设置自增长属性

  id int primary key auto_increment,
- **插入5条表记录（id 1-5,name-诸葛亮、司马懿、貂蝉、张飞、赵云），攻击>100,防御<100)**
- **查找所有蜀国人的信息**

```

```

- **将赵云的攻击力设置为360,防御力设置为68**

```

```

- **将吴国英雄中攻击值为110的英雄的攻击值改为100,防御力改为60**

- **找出攻击值高于200的蜀国英雄的名字、攻击力**

- **将蜀国英雄按攻击值从高到低排序**

- **魏蜀两国英雄中名字为三个字的按防御值升序排列**

- **在蜀国英雄中,查找攻击值前3名且名字不为 NULL 的英雄的姓名、攻击值和国家**

MySQL普通查询

```
    3、select ...聚合函数 from 表名
    1、where ...
    2、group by ...
    4、having ...
    5、order by ...
    6、limit ...;
```

- **聚合函数**

方法	功能
avg(字段名)	该字段的平均值
max(字段名)	该字段的最大值
min(字段名)	该字段的最小值
sum(字段名)	该字段所有记录的和
count(字段名)	统计该字段记录的个数

eg1 : 找出表中的最大攻击力的值?

```

```

eg2 : 表中共有多少个英雄?

```

```

eg3 : 蜀国英雄中攻击值大于200的英雄的数量

```

```

- **group by**

给查询的结果进行分组
eg1 : 计算每个国家的平均攻击力

```

```

eg2 : 所有国家的男英雄中 英雄数量最多的前2名的 国家名称及英雄数量

```

```

==group by后字段名必须要为select后的字段==

==查询字段和group by后字段不一致,则必须对该字段进行聚合处理(聚合函数)==

- **having语句**

对分组聚合后的结果进行进一步筛选

```
eg1 : 找出平均攻击力大于105的国家的前2名,显示国家名称和平均攻击力
```

注意

```
having语句通常与group by联合使用
having语句存在弥补了where关键字不能与聚合函数联合使用的不足,where只能操作表中实际存在的字段,having操作的是聚合函数生成的显示列
```

- **distinct语句**

不显示字段重复值

```
eg1 : 表中都有哪些国家

eg2 : 计算一共有多少个国家
```

注意

```
distinct和from之间所有字段都相同才会去重
distinct不能对任何字段做聚合处理
```

- **查询表记录时做数学运算**

运算符： + - * / % **

```
eg1: 查询时显示攻击力翻倍

eg2: 更新蜀国所有英雄攻击力 * 2
```

索引概述

- **定义**

对数据库表的一列或多列的值进行排序的一种结构(Btree方式)

- **优点**

加快数据检索速度

- **缺点**

```
占用物理存储空间(/var/lib/mysql)
当对表中数据更新时,索引需要动态维护,降低数据维护速度
```

- **索引示例**

```
# cursor.executemany(SQL,[data1,data2,data3])
# 以此IO执行多条表记录操作，效率高，节省资源
1、开启运行时间检测
  mysql>show variables like '%pro%';
```

```
  mysql>set profiling=1;
2、执行查询语句(无索引)
  select name from students where
name='Tom99999';
3、查看执行时间
  show profiles;
4、在name字段创建索引
  create index name on students(name);
5、再执行查询语句
  select name from students where
name='Tom88888';
6、查看执行时间
  show profiles;
```

索引分类

普通(MUL) and 唯一(UNI)

- **使用规则**

```
1、可设置多个字段
2、普通索引 ： 字段值无约束,KEY标志为 MUL
3、唯一索引(unique) ： 字段值不允许重复,但可为 NULL
                        KEY标志为 UNI
4、哪些字段创建索引:经常用来查询的字段、where条件判断
字段、order by排序字段
```

- **创建普通索引and唯一索引**

创建表时

```
create table 表名(
字段名 数据类型,
字段名 数据类型,
index(字段名),
index(字段名),
unique(字段名)
);
```

已有表中创建

```
create [unique] index 索引名 on 表名(字段名);
```

- **查看索引**

```
1、desc 表名;  --> KEY标志为：MUL 、UNI
2、show index from 表名\G;
```

- **删除索引**

```
drop index 索引名 on 表名;
```

主键(PRI)and自增长(auto_increment)

- **使用规则**

```
1、只能有一个主键字段
2、所带约束 ：不允许重复,且不能为NULL
3、KEY标志(primary) ：PRI
4、通常设置记录编号字段id,能唯一锁定一条记录
```

- **创建**

创建表添加主键

```
create table student(
id int auto_increment,
name varchar(20),
primary key(id)
)charset=utf8,auto_increment=10000;##设置自增长起始值
```

已有表添加主键

```
alter table 表名 add primary key(id);
```

已有表操作自增长属性

```
1、已有表添加自增长属性
  alter table 表名 modify id int auto_increment;
2、已有表重新指定起始值：
  alter table 表名 auto_increment=20000;
```

- **删除**

```
1、删除自增长属性(modify)
  alter table 表名 modify id int;
2、删除主键索引
  alter table 表名 drop primary key;
```

今日作业

- **1、把今天所有的课堂练习重新做一遍**

- **2、面试题**

有一张文章评论表comment如下

comment_id	article_id	user_id	date
1	10000	10000	2018-01-30 09:00:00
2	10001	10001	… …
3	10002	10000	… …
4	10003	10015	… …
5	10004	10006	… …
6	10025	10006	… …
7	10009	10000	… …

以上是一个应用的comment表格的一部分，请使用SQL语句找出在本站发表的所有评论数量最多的10位用户及评论数，并按评论数从高到低排序

备注：comment_id为评论id

article_id为被评论文章的id

user_id 指用户id

- **3、操作题**

综述：两张表，一张顾客信息表customers，一张订单表orders

表1：顾客信息表，完成后插入3条表记录

c_id 类型为整型，设置为主键，并设置为自增长属性
c_name 字符类型，变长，宽度为20
c_age 微小整型，取值范围为0~255(无符号)
c_sex 枚举类型，要求只能在('M','F')中选择一个值
c_city 字符类型，变长，宽度为20
c_salary 浮点类型，要求整数部分最大为10位，小数部分为2位

表2：顾客订单表（在表中插入5条记录）

o_id 整型
o_name 字符类型，变长，宽度为30
o_price 浮点类型，整数最大为10位，小数部分为2位
设置此表中的o_id字段为customers表中c_id字段的外键，更新删除同步

```
insert into orders values(1,"iphone",5288),
(1,"ipad",3299),(3,"mate9",3688),
(2,"iwatch",2222),(2,"r11",4400);
```

增删改查题

1、返回customers表中，工资大于4000元，或者年龄小于29岁，满足这样条件的前2条记录
2、把customers表中，年龄大于等于25岁，并且地址是北京或者上海，这样的人的工资上调15%
3、把customers表中，城市为北京的顾客，按照工资降序排列，并且只返回结果中的第一条记录
4、选择工资c_salary最少的顾客的信息
5、找到工资大于5000的顾客都买过哪些产品的记录明细
6、删除外键限制
7、删除customers主键限制
8、增加customers主键限制c_id