MySQL基础回顾

王伟超 wangweichao@tedu.cn

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;
  show databases;
2、创建库并指定字符集;
  create database 库名 charset utf8;
  create database 库名 character set utf8;
3、查看当前所在库;
  select database();
4、切换库;
  use 库名;
5、查看库中已有表;
  show tables;
6、删除库;
  drop database 库名;
```

表管理

```
1、创建表并指定字符集;
  create table 表名(字段名 数据类型,xxx)charset=utf8;
2、查看创建表的语句 (字符集、存储引擎);
  show create table 表名;
3、查看表结构;
  desc 表名;
4、删除表;
  drop table 表1,表2,表3;
```

表记录管理

```
1、增 ： insert into 表名(字段名) values(),();
2、删 ： delete from 表名 where 条件;
3、改 ： update 表名 set 字段名=值,字段名=值 where 条件;
4、查 ： select 字段名,xxx from 表名 where 条件;
```

表字段管理（alter table 表名）

```
1、增 ： alter table 表名 add 字段名 类型 first | after 字段名;
2、删 ： alter table 表名 drop 字段名;
3、改 ： alter table 表名 modify 字段名 新类型;
4、表重命名: alter table 表名 rename 新表名;
```

5、数据类型

四大数据类型

- 数值类型

```
int smallint bigint tinyint
float(m,n) double decimal
```

- 字符类型

```
char() varchar() text longtext blob longblob
```

- 枚举类型

```
enum() set()
```

- 日期时间类型

```
date time year datetime timestamp
```

日期时间函数

```
NOW() CURDATE() YEAR(字段名) DATE(字段名) TIME(字段名)
```

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
  select * from tab where time<(NOW()-interval 1 year);
```

6、MySQL运算符

- **数值比较**

```
> >= < <= = !=
eg1 : 查询成绩不及格的学生
      select * from students where score<60;
eg2 : 删除成绩不及格的学生
      delete from students where score<60;
eg3 : 把id为3的学生的姓名改为 周芷若
      update students set name='周芷若' where id=3;
```

- **逻辑比较**

```
and  or
eg1 : 查询成绩不及格的男生
  select * from students where score<60 and gender='M';
eg2 : 查询成绩在60-70之间的学生
  select * from students where score>=60 and score<=70;
```

- **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
  select name,score from students where score between 0 and 59;
eg2 : 查询AID1903和AID1902班的学生姓名及成绩
  select name,score from students where class in('AID1903','AID1902');
```

- **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
  select * from students where address='北京' and name like '赵%';
```

- **NULL判断**

```
is NULL 、is not NULL
eg1 : 查询姓名字段值为NULL的学生信息
  select * from students where name is NULL;
```

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
  select * from students order by score DESC;
```

- **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
     limit (6-1)*10,10
```

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**

  ```
  select * from sanguo where country='蜀国';
  ```
- **将赵云的攻击力设置为360,防御力设置为68**

  ```
  update sanguo set attack=360,defense=68 where name='赵云';
  ```
- **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**

  ```
  update sanguo set attack=100,defense=60 where country='吴国';
  ```
- **找出攻击值高于200的蜀国英雄的名字、攻击力**

  ```
  select name,attack from sanguo where attack>200 and country='蜀国';
  ```
- **将蜀国英雄按攻击值从高到低排序**

  ```
  select * from sanguo where country='蜀国' order by attack DESC;
  ```
- **魏蜀两国英雄中名字为三个字的按防御值升序排列**

- ```
  select * from sanguo where country in('魏国','蜀国') and name like '___' order by defense ASC;
  ```

- **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**

  ```
  select name,attack,country from sanguo
  where country='蜀国' and name is not NULL
  order by attack DESC
  limit 3;
  ```

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?

```
select max(attack) from sanguo;
```

eg2 : 表中共有多少个英雄?

```
select count(name) as number from sanguo;
```

eg3 : 蜀国英雄中攻击值大于200的英雄的数量
```

```
select count(id) from sanguo where country='蜀国' and attack>200;
聚合函数在默认情况下是不能与其他列一起做查询的。
```

- **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

```
select country,avg(attack) from sanguo
group by country;
魏国
魏国
魏国          160.0000       魏国
蜀国
蜀国          170.0000       蜀国
吴国          150.0000       吴国
```

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

```
select country,count(id) as number from sanguo
where gender='M' group by country
order by number DESC
limit 2;
```

==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
select country,avg(attack) from sanguo
group by country
having avg(attack)>105
order by avg(attack) DESC
limit 2;
```

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
  select distinct name,country from sanguo;
eg2 : 计算一共有多少个国家
  select count(distinct country) from sanguo;
```

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
  select name,attack*2 from sanguo;
eg2: 更新蜀国所有英雄攻击力 * 2
  update sanguo set attack=attack*2 where country='蜀国';
```

嵌套查询(子查询)

- **定义**

把内层的查询结果作为外层的查询条件

- **语法格式**

```
select ... from 表名 where 条件(select ....);
```

- **示例**

```
1、把攻击值小于平均攻击值的英雄名字和攻击值显示出来
  select name,attack from sanguo where attack<(select avg(attack) from sanguo);
2、找出每个国家攻击力最高的英雄的名字和攻击值(子查询)
  select name,attack from sanguo where (country,attack) in(select country,max(attack) from
sanguo group by country);
```

多表查询

sql脚本资料：join_query.sql

```
mysql -uroot -p123456
mysql>source /home/tarena/join_query.sql
```

```
create database if not exists db1 character set utf8;
use db1;

create table if not exists province(
id int primary key auto_increment,
pid int,
pname varchar(15)
)default charset=utf8;

insert into province values
(1, 130000, '河北省'),
(2, 140000, '陕西省'),
(3, 150000, '四川省'),
(4, 160000, '广东省'),
(5, 170000, '山东省'),
(6, 180000, '湖北省'),
(7, 190000, '河南省'),
(8, 200000, '海南省'),
(9, 200001, '云南省'),
(10,200002,'山西省');

create table if not exists city(
id int primary key auto_increment,
cid int,
cname varchar(15),
cp_id int
)default charset=utf8;

insert into city values
(1, 131100, '石家庄市', 130000),
(2, 131101, '沧州市', 130000),
(3, 131102, '廊坊市', 130000),
(4, 131103, '西安市', 140000),
(5, 131104, '成都市', 150000),
(6, 131105, '重庆市', 150000),
(7, 131106, '广州市', 160000),
(8, 131107, '济南市', 170000),
(9, 131108, '武汉市', 180000),
(10,131109, '郑州市', 190000),
(11,131110, '北京市', 320000),
(12,131111, '天津市', 320000),
(13,131112, '上海市', 320000),
(14,131113, '哈尔滨', 320001),
(15,131114, '雄安新区', 320002);

create table if not exists county(
id int primary key auto_increment,
coid int,
coname varchar(15),
copid int
)default charset=utf8;

insert into county values
```

```
(1, 132100, '正定县', 131100),
(2, 132102, '浦东新区', 131112),
(3, 132103, '武昌区', 131108),
(4, 132104, '哈哈', 131115),
(5, 132105, '安新县', 131114),
(6, 132106, '容城县', 131114),
(7, 132107, '雄县', 131114),
(8, 132108, '嘎嘎', 131115);
```

- **笛卡尔积** -- 交叉连接

```
select 字段名列表 from 表名列表;
```

- **多表查询**

```
select 字段名列表 from 表名列表 where 条件;
```

- **示例**

```
1、显示省和市的详细信息
   河北省   石家庄市
   河北省   廊坊市
   湖北省   武汉市
   select province.pname,city.cname from province,city
   where province.pid=city.cp_id;
2、显示 省 市 县 详细信息
   select province.pname,city.cname,county.coname from province,city,county
   where province.pid=city.cp_id and city.cid=county.copid;
```

连接查询

- **内连接（结果同多表查询，显示匹配到的记录）**

```
在关联的两张表中，把满足条件的数据筛选出来。
select 字段名 from  表1 inner join 表2 on 条件 inner join 表3 on 条件;
eg1 : 显示省市详细信息
  select province.pname,city.cname from province
  inner join city on province.pid=city.cp_id;
eg2 : 显示 省 市 县 详细信息
  select province.pname,city.cname,county.coname from province inner join city on
province.pid=city.cp_id
  inner join county on city.cid=county.copid;
```

- **左外连接**

以 左表 为主显示查询结果

```
select 字段名 from 表1 left join 表2 on 条件 left join 表3 on 条件;
eg1 : 显示 省 市 详细信息（要求省全部显示）
  select province.pname,city.cname from province
  left join city on province.pid=city.cp_id;
```

- **右外连接**

用法同左连接,以右表为主显示查询结果

```
select 字段名 from 表1 right join 表2 on 条件 right join 表3 on 条件;
```

```
完整外连接
1.作用
将两张表的数据做关联查询，关联的上的则正常显示，关联不上的，则以null值填充
2.语法
select 字段名 from 表1 full join 表2 on 条件 full join 表3 on 条件;
```

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
索引的比对手段
1.没有创建索引
    1.查询系统事件
    2.执行查询
    3.查看执行事件
    4.对比时间差

2.在某列创建索引
    1.查询系统事件
    2.执行查询
    3.查看执行事件
    4.对比时间差
一般经常查询的列,
```

- **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)

```
索引分列
1.主键索引
    特点：增加逐渐之后，主键列自动会被增加索引
    2.增加主键【索引】
     1.已有表增加主键
     alter table 表名 add primary key(id);
2.唯一索引
    1.特点
        1.可以有多个
        2.唯一索引所在的列的值必须唯一
    2.实施手段
        1.创建表的时候指定唯一性
        2.对已有表创建索引
            create unique index 索引名 on 表名（字段名）;
```

- **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;
```

外键(foreign key) - 记住3点

- **定义**

让当前表字段的值在另一个表的范围内选择

- **语法**

```
foreign key(参考字段名)
references 主表(被参考字段名)
on delete 级联动作
on update 级联动作
```

- **使用规则**

```
1、主表、从表字段数据类型要一致
2、主表被参考字段 : KEY的一种, 一般为主键
1.作用: 约束当前表的某列值必须取自于另外一张表的主键列值
外键所在的列称之为外键列
外键所在的表称之为外键表或者字表
被外键列所引用的表称之为主表或主键表;
语法: constraint fk_course_teacher foreign key(couse) referenses Course(id);
1.创建表的同时指定外键
语法:
create table 表名(字段(类型),... constraint 外键名 foreign key(字段) references 主键表(主键列))

2.对已有表增加外键
alter table 表名 add constraint 外键名 foreign key(字段) references 主键表(主键列))
```

- **示例**

表1、缴费信息表(财务)

```
id   姓名      班级      缴费金额
1   唐伯虎    AID1903     300
2   点秋香    AID1903     300
3   祝枝山    AID1903     300
create database db2 charset utf8;
use db2;
create table master(
id int primary key,
name varchar(20),
class char(7),
money decimal(10,2)
)charset=utf8;
insert into master values(1,'唐伯虎','AID1903',300),(2,'点秋香','AID1903',300),(3,'祝枝
山','AID1903',300);
```

表2、学生信息表(班主任) -- 做外键关联

```
stu_id   姓名    缴费金额
  1     唐伯虎     300
  2     点秋香     300
create table slave(
stu_id int,
name varchar(20),
money decimal(10,2),
foreign key(stu_id) references master(id)
on delete cascade on update cascade
)charset=utf8;
insert into slave values(1,'唐伯虎',300),(2,'点秋香',300),(3,'祝枝山',300);
```

- **删除外键**

```
alter table 表名 drop foreign key 外键名;
外键名 : show create table 表名;
```

- **级联动作**

```
cascade
数据级联删除、更新(参考字段)
restrict(默认)
从表有相关联记录,不允许主表操作
set null
主表删除、更新,从表相关联记录字段值为NULL
```

- **已有表添加外键**

```
alter table 表名 add constraint 外键名 foreign key(参考字段) references 主表(被参考字段) on delete
级联动作 on update 级联动作
```

```
alter table 表名 drop foreign key 外键名;
外键名 : show create table 表名;
```

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

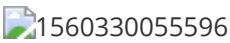
comment_id	article_id	user_id	date
1	10000	10000	2018-01-30 09:00:00
2	10001	10001	… …
3	10002	10000	… …
4	10003	10015	… …
5	10004	10006	… …
6	10025	10006	… …
7	10009	10000	… …

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

 1560330055596

- **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
```

```
numpy模块
reshape()
shape()
zeros()
ones()
dot()矩阵
```

```
(单选题)____ 命令将删除一个或多个表定义以及这些表的所有数据、索引、触发器、约束和权限规定。
A.DROP TABLE
B.TRUNCATE TABLE
C.ALTER TABLE
D. CREATE TABLE
【正确答案】A
【答案解析】truncate和delete的区别
         1、事务：truncate是不可以rollback的，但是delete是可以rollback的；
               原因：truncate删除整表数据(ddl语句,隐式提交)，delete是一行一行的删除，可以rollback
 2、效果：truncate删除后将重新水平线和索引(id从零开始) ,delete不会删除索引
 3、 truncate 不能触发任何Delete触发器。
 4、delete 删除可以返回行数


(多选题)在MySQL数据库中，主键约束与唯一约束的区别有（）
A.主键列的数据类型不限，但此列必须是唯一并且非空
B.一张数据表只能有一个主键
C.唯一性约束所在的列允许空值
D.数据表可以包含有多个唯一约束
【正确答案】A,B,C,D
【答案解析】此题目考查MySQL数据库中主键约束和唯一约束的区别。
主键约束（PRIMARY KEY）：
1.主键用于唯一的标识表中的每一条记录，可以定义一类或多列为主键。
2.表里面只能有一个主键约束，但可以有多个唯一约束。
```

3.主键列上没有任何两行具有相同值（即重复值），不允许空（null）。
4.主键可作外键，唯一索引不可。
唯一约束（UNIQUE）：
1.唯一约束用来限制不受主键约束的列上的数据的唯一性，用于作为访问某行的可选手段，一个表上可以防止多个唯一性约束。
2.只要唯一就可以更新。
3.表中任意两行在指定列上都不允许有相同的值，允许空（NULL）。
4.一个表上可以放置多个唯一约束。