## 数据库

1. 数据库的连接

mysql-> mysql -h 数据库服务器的地址(默认为locahost可以不输入) -u 用户名 -p 密码 -P 端口号(mysql默认为 3306也可以修改为其他)

例如：mysql -h localhost -u root -p 123456 -P 3306(默认3306可以不输入)

简写：mysql -u root -p

1. SQL语言包含四个部分
2. 数据定义语言(DDL)：用于定义和管理数据对象，包括数据库，数据表等。例如：create , drop , alter等
3. 数据操作语言(DML)：用于操作数据库对象中所包含的数据。例如：insert,update,delete语句
4. 数据查询语句(DQL)：用于查询数据库对象中所包含的数据，能够进行单表查询、连表查询、嵌套查询、以及集合查询等复杂程度不同的数据库查询，并将数据返回客户端中显示。例如：select
5. 数据控制语言(DCL)：是用来管理数据库的语言，包括管理权限及数据更改。例如：grant、revoke、commit、rollback等
6. 数据的相关操作(重点)

数据库的结构：

一个数据库服务器集群

多台数据库服务器

一台服务器可以有多个数据库

一个数据库可以有多张数据表

一张数据表可以有多个字段

字段名 字段类型 字段约束

数据库系统(DBS) -> 数据库(DB) -> 数据表(TABLE) -> 字段(field)

Mysql的sql语句不区分大小写，在windows系统下 表名和库名 不区分大小写，但是在linux系统下 库和表名严格区分大小写.

1. 数据库中的库相关操作
2. 查库

show databases;

1. 创建数据库

create database if not exists 数据库;

例如： create database if not exists laravel;

1. 打开数据库

Use 数据库名;

use laravel;

1. 查看当前所在的数据库

select database();

1. 删除数据库

drop database if exists 数据库名称;

1. 数据表相关操作(重点)
2. 创建数据表

create table 表名(

字段名1 字段1类型 字段1约束,

字段名2 字段2类型 字段2约束，

......

)engine=InnoDB charset=utf8;

1. 查看数据表的结构，有哪些字段

show create table 表名; 或者 desc 表名;(description)

show create table test; 或 desc 表名;

1. 删除表 drop table if exists 表名;

drop table if exists test;

1. 修改表名

alter table 表名 rename to 新表名;

1. 数据相关操作：增删改查（重点）
2. 增

insert into 表名(字段1名称,字段3名称,字段5名称....) values(字段1对应值,字段3对应值,字段5对应值....);(前提是字段2和字段4设置了允许为空 null并且设置了默认值)

也可以是insert into 表名 values(字段1对应值,字段2对应值,字段3对应值....);（这样要求字段与值要一一对应,不能省略）

1. 删(根据字段的值条件进行删除，否则会把整个表的内容都删除)

delete from 表 where 条件;

1. 改(alter(drop、add、modify、change))
2. 删除字段

alter table 表名 drop 字段名;

1. 添加字段

alter table 表名 add 字段名 字段类型 [first\after 字段名(表示新增加的字段在字段名名之前还是在第一个)];

1. 修改字段类型

alter table 表名 modify 字段名 新的字段类型;

1. 修改字段名称

alter table 表名 change 字段名 新字段名 字段类型

1. 设置字段默认值

alter table 表名 alter 字段名 set default 默认值

1. 删除字段的默认值

alter table 表名 alter 字段名 drop default;

1. 更新符合条件的字段值

update 表名 set 字段 = 值 where 条件

1. 查

select 字段1,字段2,字段....... from 表名 [where 条件];

select \* from 表名 [where 条件];

1. 创建用户并授权、授权、撤销权限、删除用户
2. 授权的同时创建用户

grant 权限 on 数据库.数据表 to ‘新用户名’@’登录主机’ identified by ‘密码’;

1. 创建用户

create user ‘用户名’@’主机’ identified by ‘用户密码’;

1. 授权

grant 权限 on 库.表 to ‘用户名’@’主机’;

权限: privileges 表示 select insert update等权限,如果要给与全部权限 则使用 ALL

如果要给用户在所有库中所有表的所有权限 命令如下：

grant all \*.\* to ‘用户名’@’主机’;

1. 设置与更改用户密码

set password for ‘用户名’@’主机’ = PASSWORD(‘新密码’);

如果是当前登录用户修改密码，则使用 set password = PASSWORD(‘新密码’);

1. 撤销用户权限

revoke 要撤销的权限 on 库.表 from ‘用户名’@’主机’;

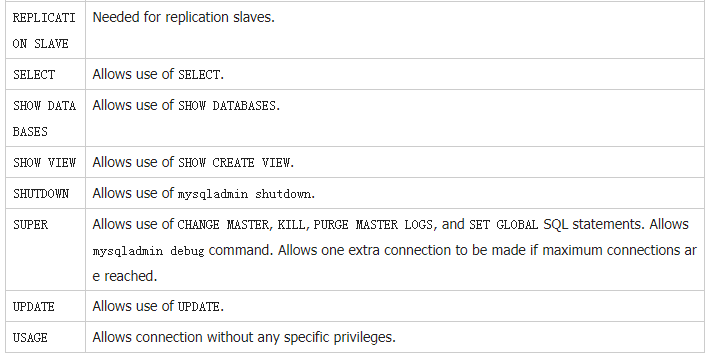
使用 show grants for ‘用户名’@’主机’;查看对应用户的权限

1. 删除用户

drop user ‘用户名’@’主机’;

1. MySQL中的操作权限表





1. 事务(begin、rollback、commit) 确保数据的原子性（完整性）

begin：开始记录对表中数据的操作

rollback：回滚,撤销之前的操作

commit：确认，保存之前的操作

讲解：

1. 只有库和表的引擎为InnoDB才支持事务 也就是engine指定为InnoDB，默认为MyISAM 可以修改为默认InnoDB
2. 事务处理可以用来维护数据库的完整性，保证成批的SQL语句要么全部执行成功，要么全部不执行
3. 事务用来管理Insert、update、delete语句

一般来说：事务是必须满足4个条件：原子性、一致性、隔离性、持久性。

1. 原子性：一个事物中的所有操作，要么全部完成、要么全部不完成，不会结束在中间某个环节。事务在执行过程中发送错误，会被回滚(rollback)到事务开始前的状态,就像这个事务从来就没有执行过一样
2. 一致性：在事务开始之前和事务结束之后，数据库的完整性没有被破坏。这表示写入的资料必须完全符合所有的预设规则，这包含资料的精确度、串联性以及后续数据库可以自发性地完成预定的工作
3. 隔离性：数据库允许多个并发事务同时对其数据进行读写和修改的能力，隔离性可以防止多个事务并发执行时，由于交叉执行而导致数据的不一致。事务隔离分为不同级别，包括读未提交(Read uncommited)、读提交(Read commited)、可重复读（repeatable read）、串行化(Seaiazable)。
4. 持久化：事务处理结束（commit）后，对数据的修改就是永久的，几遍系统故障也不会丢失。
5. 约束
6. default 给字段设置默认值
7. unsigned 表示无符号数，不能是负数,从0开始
8. unique 唯一，表示这一列数据不能有重复值，用户帐号和身份证等能辨别用户唯一性的字段
9. primary key 主键 有且唯一
10. auto\_increment 自动增长，只能跟主键配合使用，表示让主键的值从1开始依次自动+1
11. int(4) 只表示显示值的长度，不会影响存储长度（整数后面的括号中的数字，不是指数据的范围，而是数据的显示长度）
12. zerofile 零填充，数据位用0补充，跟int()配合使用

例如：int(10) 表示输入的值长度可以为 10位 ，如果输入的数值为 5位的 12345那么会在1的前面填充5个0 -> 0000012345

1. 数据类型
2. 数值类型数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 存储空间 | 说明 | 取值范围 |
| Tinyint | 1字节 | 非常小的值 | 带符号值：-128~127  无符号值：0~255 |
| Smallint | 2字节 | 较小的值 | 带符号值：-32768~32767  无符号值：0~65535 |
| Mediuint | 3字节 | 中等大小的整数 | 带符号值：-8388608~8388607  无符号值：0~16777215 |
| Int | 4字节 | 标准整型 | 带符号值：2147483648~2147483647  无符号值：0~4294967295 |
| Bigint | 8字节 | 大整形 | 带符号值：-2^63~2^63  无符号值：0~2^64-1 |
| Float | 4或8字节 | 单精度浮点数 | 最小非零值：+- 1.175494351E-38  最大非零值：+-3.402823466E+38 |
| Double | 8字节 | 双精度浮点数 | 最小非零值：+- 2.225073E-308  最大非零值：+- 1.797693E+308 |
| Decimal(m,d) | 自定义 | 义字符串形式表示的浮点数 | 取决于存储单元字节数  m是数字的最大值(1~65)  d是小数点右侧数字的数目(0~30)但不得超过m |

1. 字符串类型数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据列类型 | 存储空间 | 说明 | 取值范围 |
| Char([M]) | M字节 | 定长字符串 | M字节 |
| Varchar([M]) | L+1字节 | 可变字符串 | M字节 |
| Tinyblob,tinytext | L+1字节 | 非常小的blob（二进制大对象）和文本串 | 2^8-1字节 |
| Blob,text | L+2字节 | 小blob和文本串 | 2^16-1 |
| Mediumblob,mediumtext | L+3字节 | 中等的blob和文本串 | 2^24-1字节 |
| Longblob,longtext | L+4字节 | 大blob和文本串 | 2^32-1字节 |
| Enum(值1，值2,...) | 1或2字节 | 枚举：可赋予某个枚举成员 | 65535个成员 |
| Set(值1，值2，...) | 1,2,3,4或8字节 | 集合：可赋予多个集合成员 | 64个成员 |

Char和varchar类型的长度范围都是0~255之间

使用char()类型时，如果我们传入的值长度小于指定长度,实际长度会使用空格填充至指定长度

使用varchar类型时，如果我们传入的值的长度小于指定长度，实际长度即为传入字符串的长度，不会使用空格填充

char要比varchar效率更高

enum和set类型是特殊的字符串类型，其列值必须从固定的字符串中选择

enum只能选择其中一个值,set可以选择多个值

1. 日期和时间类型数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据列类型 | 存储空间 | 说明 | 取值范围 |
| Date | 3字节 | YYY-MM-DD格式表示的日期值 | 1000-01-01~9999-12-31 |
| Time | 3字节 | Hh:mm:ss 格式表示的时间值 | -838:59:59~838:59:59 |
| datetime | 8字节 | YYYY-MM-DD hh:mm:ss | 1000-01-01 00:00:00~9999-12-31 |
| Timestamp | 4字节 | YYYYMMDDhhmmss | 197001010000000-2037年的某个时刻 |
| Year | 1字节 | YYYY格式的年份值 | 1901~2155 |

1. 数据表的引擎
2. InnoDB（常用引擎）
3. Innodb存储引擎的mysql表提供了事务，回滚以及系统崩溃修复能力和多版本并发控制的事务的安全
4. Innodb存储引擎支持外键(foreign key)，外键所在的表称为子表而依赖的表称为父表。
5. Innodb 支持自增长列(auto\_increment)，自增长列的值不能为空，如果在使用的时候为空的话会进行存现有的值开始增值，如果有但是比现在的还大，则就保存这个值。
6. Innodb 存储引擎最重要的是支持事务，以及事务相关联功能。
7. innodb存储引擎支持mvcc的行级锁
8. Innodb 存储引擎使用的是B+Tree
9. MyISAM（少用）
10. 不支持事务，不支持行级锁，只支持并发插入的表锁，主要用于高负载的select查询
11. MyISAM 类型的表支持三种不同的存储结构：静态型、动态型、压缩型。
12. MyISAM 也是使用B + Tree 索引但是和InnoDB的在具体实现上有些不同。
13. MEMORY（基本不用）
14. memory 存储引擎相比前面的一些存储引擎，有点不一样，其使用储存在内存中的数据来创建表，而且所有的数据也都存储在内存中
15. 每个基于memory 存储引擎的表实际对应一个磁盘文件，该文件的文件名和表名是相同的，类型为.frm。该文件之存储表的结构，而其数据文件，都是存储在内存中，这样有利于对数据的快速处理，提高整个表的处理能力。
16. memory存储引擎默认使用哈希(HASH)索引，其速度比使用 B- +Tree 型要快
17. memory存储引擎文件数据都存储在内存中，如果mysqld进程发生异常，重启或关机这些数据都会消失。所以memory存储引擎中表的声明周期很短，一般只使用一次。
18. 数据导入导出

注意：在非mysql命令模式下执行

导出 mysqldump -u root -p 库名 表名>具体路径下的文件名

导入 mysql -u root -p 库名<文件名

1. 索引

主键索引(primary key)

主键索引是关系数据库中最常见的索引类型,主要作用是确定数据表里一条特定的数据记录的位置。我们可以在字段后添加primary key 来对字段设置为主键索引

注意：

1. 最好为每张表指定一个主键，但不是必要的
2. 一个表只能指定一个主键，而且主键的值不能为空
3. 主键可以有多个候选索引（例如：NOT NULL，auto\_increment）

唯一索引(unique)

唯一索引与主键索引一样，都可以防止创建重复的值。但是，不同之处在于，每个数据表中只能有一个主键索引，但是一张表中可以有多个唯一索引。

常规索引(index)

常规索引技术是关系型数据查询中最重要的技术，如果要提升数据库的性能，索引优化是首先应该考虑的，因为它能使我们的数据库得到最大性能方面的提升。常规索引也存在缺点

1. 多占用磁盘空间
2. 会减慢插入，删除和修改操作
3. 需要按照索引列上排序格式执行

创建索引我们可以使用INDEX和KEY关键字 随表一同创建。

全文索引(fulltext)

全文索引在mysql中是一个fulltext类型索引，但fulltext索引只能用于MyISAM表，并且只可以在CHAR、VARCHAR或TEXT类型的列上创建，也允许创建在一个或多个数据列上。但是fulltext是不支持中文全文索引的，所以我们将来会使用效率更高的全文索引引擎 Sphinx

1. 数据表的拆分
2. 水平拆分：（通常使用取模方式来拆分表）

水平拆分是指数据表 行 的拆分，表的行数超过了几百万几亿的时候，查询就会变慢，这是可以把一张几百万几亿行数据的表拆成多张表来存放。

优点：

1. 表关联基本能够在数据库端全部完成
2. 不会存在某些超大型数据量和高负载的表遇到瓶颈的问题
3. 应用程序端整体架构改动相对较少
4. 事务处理相对简单
5. 只要切分规则能够定义好，基本上较难遇到扩展性限制

缺点：

1. 切分规则相对更为复杂，很难抽象出一个能够满足整个数据库的切分规则
2. 后期数据的维护难度有所增加，人为手工定位数据更困难
3. 应用系统各模块耦合度较高，可能会对后面数据的迁移拆分造成一定的困难
4. 垂直拆分：

垂直拆分是指数据表 列 的拆分,把一张列比较多的表拆分为多张表。表的记录并不多，但是字段却很长，表占用空间很大，检索表的时候需要执行大量的IO，严重减低了性能。这时需要把打的字段拆分到另一个表，并且该表与原表是一对一的关系

垂直拆分原则：1、把不常用的字段单独放在一张表 2、把text,blob等大字段拆分出来放在附表中 3、经常组合查询的列放在一张表中

优点：

1. 数据库的拆分简单明了，拆分规则明确
2. 应用程序模块清晰明了，整合容易
3. 数据维护方便易行，容易定位

缺点：

1. 部分表关联无法在数据库级别完成，需要在程序中完成
2. 对于访问及其频繁且数据量超大的表仍然存在性能瓶颈，不一定能满足要求
3. 事务处理相对更为复杂
4. 切分达到一定程度之后，扩展性会遇到限制
5. 过渡切分可能会带来系统过渡复杂而难以维护

一.字段部分 select

1.查询所有的字段

select id,uname,sex,age,classid from users;

\*表示所有的字段的意思

select \* from users;

2. 部分字段

select uname,sex from users;

3.查询10年后的年龄

select uname ,age,age+10 from users;

4. 别名

age+10 as 新字段名

select uname ,age,age+10 as newage from users;

5.列合并,合并年龄与性别

select unname,age,sex from users;

select concat(列名1，列名2) from 表名;

select uname ,concat(age,sex) as agesex from users;

6. 出掉重复的行

select distinct 列名 from 表名;

select distinct classid from users;

7. 在查询结果中无中生有添加一列

select \*,'中国' as '国籍' from users;

二. 条件 where

8)沈阳校区的学生

select \* from users where classid= '沈阳校区';

9)广州校区的女生

如果需要同时满足多个条件，使用and操作符

select \* from users where classid= '广州校区' and sex='w';

10)年龄大于20的学生

select \* from users where age > 20;

年龄小于20的学生

select \* from users where age < 20;

年龄小于等于20的学生

select \* from users where age <= 20;

11)年龄 在 30 到 40 的人

select \* from users where age >=30 and age <=40;

between 在某个区间范围内

select \* from users where age between 30 and 40;

12)age 不在 20 到 35 的人

select \* from users where age < 20 or age >35;

select \* from users where age not between 20 and 35;

一定要根据题干分析出是使用or(或者的意思) 还是 and（并且的意思）

13)沈阳校区和郑州校区的女生

也就是求 沈阳校区的女生 或者 郑州校区的女生

也就是求 沈阳校区并且是女生 或者 郑州校区并且是女生

select \* from users where (classid='沈阳校区' and sex = 'w') or (classid='郑州校区' and sex = 'w');

也可以这样理解

只要是沈阳校区或者郑州校区 并且 女生

select \* from users where classid='沈阳校区' or classid='郑州校区' and sex='w';

这个地方结果不对，是因为or 和 and 的优先级不同，and的优先级更高一些，所以先执行and操作

select \* from users where (classid='沈阳校区' or classid='郑州校区') and sex='w';

14)西安校区的男生和深圳校区的女生

===>是西安校区并且是男生 或者 是深圳校区并且是女生

select \* from users where (classid='西安校区' and sex='m') or (classid='深圳校区' and sex='w');

15)找id为 3,5,8,9,12 的人

select \* from users where id=3 or id=5 or id=8 or id=9 or id=12;

in 表示是否是数组中的一个元素

select \* from users where id in(3,5,8,9,12);

找id 不是 3,5,8,9,12 的人

select \* from users where id not in(3,5,8,9,12);

找uid 不等于 7的人

select \* from users where id != 7;

select \* from users where id <> 7;

16)查询classid值为NULL的人

如果条件是null 不能用=号做判断，需要使用 is

select \* from users where classid = null;//错误的

select \* from users where classid is null;

查询classid值不为NULL的人

select \* from users where classid is not null;

模糊查询 只能使用 like关键字

17)找名字为2个字的人

\_代表任意一个字符

两个字就是 \_\_

% 表示的是任意字符任意多次（含空） 有点像正则中的.\*

select \* from users where uname like '\_\_';

18)找名字中以'李'开头的

select \* from users where uname like '李%';

19)找名字中以'龙'结尾的

select \* from users where uname like '%龙';

20)找包含'文'的人

select \* from users where uname like '%文%';

三. 分组查询 group by

聚合函数

count(\*) 统计个数 count(classid) 不会统计值为NULL的记录

统计学员总数

select count(\*) from users;

select count(classid) from users;

sum() 求和 比如符合条件的人的总年龄

求所有学员年龄的总和

select sum(列名) from users;

select sum(age) from users;

avg() 平均数

求所有学员的平均年龄

select avg(age) from users;

max() 最大值

select uname, max(age) from users;

min() 最小值

select min(age) from users;

1)石家庄校区 总共多少人

select classid, count(\*) as num from users where classid='石家庄校区';

group by 字段 是分组的意思，表示按照那个字段进行分组

2)每个班人数

校区 人数

沈阳校区 20

广州校区 30

深圳校区 10

郑州校区 9

select classid,count(\*) from users where classid is not null group by classid;

3)最大年龄,最小年龄,平均年龄

select max(age) from users;

select min(age) from users;

select avg(age) from users;

查找年龄最大的那个人

SQL语句之所以如此强大，是因为SQL语句可以相互嵌套，将一个结果集作为另一个SQL语句的条件。

分解：

1. 获取最大年龄 select max(age) from users;

2. 查询记录 满足 age = 最大年龄 select \* from users where age = ( select max(age) from users);

4)在平均年龄之上的人

分解问题

1. 获取平均年龄 select avg(age) from users;

2. 找到 用户的年龄大于 平均年龄

select \* from users where age > (select avg(age) from users);

5)每个班的男生,女生各多少人

select classid,sex, count(\*) from users where classid is not null group by classid,sex;

6)每个班的男生女生各多少人,要求在20-30岁的

select

classid,sex,count(\*)

from

users

where

age between 20 and 30

group by

classid,sex;

7)每个班多少人,只显示班级人数大于5的

如果需要对分组后查询出的结果集再进行过滤，需要使用having 子句

有group by 可以有having ,有having 就一定会有group by

select

classid,count(\*)

from

users

group by

classid

having

count(\*) > 5;

select

classid,count(\*) as num

from

users

group by

classid

having

num > 5;