MySQL小知识彩蛋大集合

基础篇

1.MySQL有哪些安装方式?

```
1. yum安装
2. rpm安装
3. 二进制安装
4. 如果公司有定制性的需求，可以使用源码安装
```

2.MySQL授权表有哪些?

1. user ：保存全库的授权信息，用户、密码、密码插件、地址来源
2. db ：记录各个账号在各个数据库上的操作权限
3. tables_priv ：单表级别的授权信息
4. columns_priv ：记录数据列级别的操作权限
5. proxies_priv ：代理授权信息
6. procs_priv ：存储过程授权信息

体系结构篇

1.如何查询数据库连接会话线程id对应的系统id?

我们以数据库中id=9为例

第一步：在数据库中 show processlist 查询线程id

```
mysql> show processlist;
+----+-----------------+-----------+-------+---------+------+-----------------------+-----------------+
| Id | User            | Host      | db    | Command | Time | State                 | Info            |
+----+-----------------+-----------+-------+---------+------+-----------------------+-----------------+
|  5 | event_scheduler | localhost | NULL  | Daemon  | 4903 | Waiting on empty queue | NULL            |
|  9 | root            | localhost | mysql | Query   |    0 | starting              | show processlist |
+----+-----------------+-----------+-------+---------+------+-----------------------+-----------------+
2 rows in set (0.00 sec)
```

第二步：提取系统库performance_schema下面threads表中 processlist_id和thread_os_id两列的信息。**数据库id=9对应操作系统中的1609**

```
1  mysql> select PROCESSLIST_ID , THREAD_OS_ID  from performance_schema.thr
```

```
mysql> select PROCESSLIST_ID , THREAD_OS_ID  from performance_schema.threads where processlist_id=9;
+----------------+--------------+
| PROCESSLIST_ID | THREAD_OS_ID |
+----------------+--------------+
|              9 |         1609 |
+----------------+--------------+
1 row in set (0.00 sec)
```

第三步：top命令查询数据库进程id

```
top - 23:31:43 up  2:19,  3 users,  load average: 0.00, 0.01, 0.05
Tasks: 100 total,   2 running,  98 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.6 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2027876 total,  1310032 free,   484480 used,   233364 buff/cache
KiB Swap:  1048572 total,  1048572 free,        0 used.  1392960 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1487 mysql     20   0 1296844 366004  15724 S  0.6 18.0   0:34.16 mysqld
    1 root      20   0  125720   4268   2572 S  0.0  0.2   0:02.09 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    4 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    5 root      20   0       0      0      0 S  0.0  0.0   0:00.24 kworker/u256:0
    6 root      20   0       0      0      0 S  0.0  0.0   0:00.20 ksoftirqd/0
    7 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S  0.0  0.0   0:00.56 rcu_sched
   10 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 lru-add-drain
```

第四步：通过top –Hp 1487(数据库进程id) **找到对应连接会话线程对应的操作系统id1609**

```
top - 23:33:49 up  2:21,  3 users,  load average: 0.00, 0.01, 0.05
Threads:  40 total,   0 running,  40 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2027876 total,  1310160 free,   484312 used,   233404 buff/cache
KiB Swap:  1048572 total,  1048572 free,        0 used.  1393108 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1504 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.84 mysqld
 1505 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.83 mysqld
 1506 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.71 mysqld
 1507 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.89 mysqld
 1508 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.04 mysqld
 1509 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.10 mysqld
 1510 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.04 mysqld
 1512 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1513 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.22 mysqld
 1514 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.02 mysqld
 1515 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.04 mysqld
 1516 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 xpl_worker0
 1517 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 xpl_worker1
 1518 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.18 mysqld
 1522 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1523 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:01.02 mysqld
 1524 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1525 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.01 mysqld
 1526 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.01 mysqld
 1527 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.03 mysqld
 1528 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1529 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1530 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1532 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.00 mysqld
 1609 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.03 mysqld
 1669 mysql     20   0 1296844 366004  15724 S  0.0 18.0   0:00.09 mysqld
```

用户管理篇

1.mysql8.0默认加密插件发生了变化，不同版本兼容性问题如何解决?

我们在做mysql升级或者降级的情况下，默认加密插件可能出现不兼容，导致升级后用户无法登陆

mysql5.7版本（mysql8.0版本之前）默认加密插件是mysql_native_password （可以进行暴力破解）

mysql8.0版本默认加密插件是 caching_sha2_password

caching_sha2_password比mysql_native_password 加密程度更高更加的安全

```
mysql> mysql> select user,host,authentication_string,plugin from mysql.user;
+------------------+-----------+-------------------------------------------------------------------------+-----------------------+
| user             | host      | authentication_string                                                   | plugin                |
+------------------+-----------+-------------------------------------------------------------------------+-----------------------+
| oldguo           | 10.0.0.%  | *23AE809DDACAF96AF0FD78ED04B6A265E05AA257                               | mysql_native_password |
| mysql.infoschema | localhost | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED  | caching_sha2_password |
| mysql.session    | localhost | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED  | caching_sha2_password |
| mysql.sys        | localhost | $A$005$THISISACOMBINATIONOFINVALIDSALTANDPASSWORDTHATMUSTNEVERBRBEUSED  | caching_sha2_password |
| root             | localhost |                                                                         | caching_sha2_password |
+------------------+-----------+-------------------------------------------------------------------------+-----------------------+
5 rows in set (0.00 sec)
```

加密程度不同

2. mysql8.0创建用户和授权方式的改变

> mysql8.0版本之前 授权和创建用户操作可同时（grant）
> mysql> grant all on *.* to cry@'10.0.0.%' identified by '123';

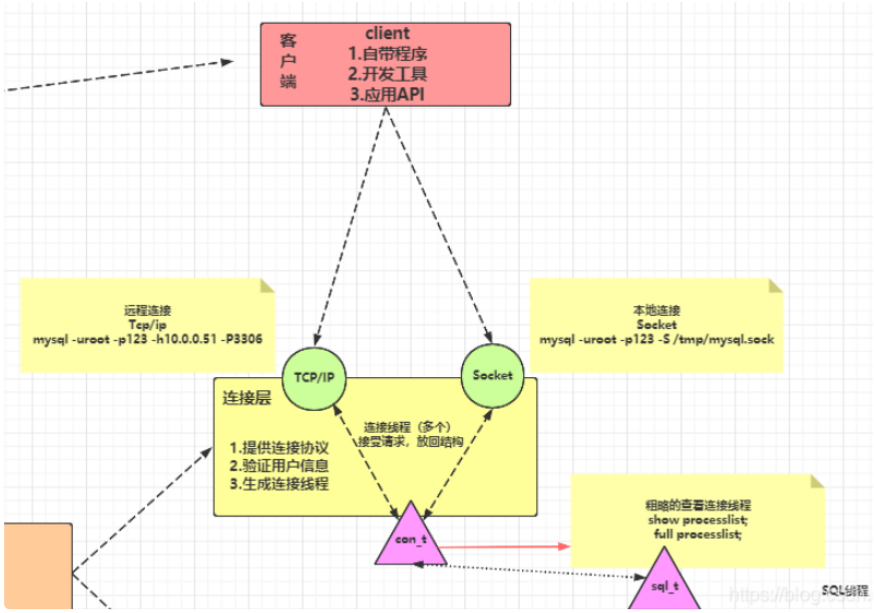
8.0版本不支持grant 命令后使用 identified by语句

3.mysql配置文件优先级

> my.cnf可以在四个路径下数据库进行读取
> [root@db01 ~]# mysqld --help --verbose | grep my.cnf
> /etc/my.cnf /etc/mysql/my.cnf /usr/local/mysql/etc/my.cnf ~/.my.cnf
> 优先级从左至右依次读取，当发生冲突会发生覆盖，以最后读取的文件为准

4.管理员root@'locahost' 丢失,怎么处理?

1.基础知识



知识点涉及到我们的客户端连接mysql服务端通过mysql服务端的连接层。

mysql服务端的连接层，经历了三个阶段

1.提供连接协议

2.验证用户信息

3.生成连接线程

我们在第二步验证用户信息的时候会加载系统授权表中的mysql.user中的用户密码信息

2.我们使用的是两个登陆连接选项，跳过验证用户信息步骤

> --skip-grant-tables 启动mysql时，不加载授权表，无密码登陆。
> 痛点：在于所有能够连接mysql的用户都是无密码登陆，包括远程连接用户。
> 我们为了减低不安全性，所以引入下面这个选项拒绝远程连接用户，使用本地登陆的方式。
> （连接mysql有两种方式，一种是基于tcp/ip协议的远程连接方式，一种是套接字文件的本地登陆方式）
> --skip-networking 拒绝远程连接用户连接

3.实际操作流程

```
a. 关闭数据库实例
/etc/init.d/mysqld stop
b. 特殊模式启动
mysqld_safe --skip-grant-tables --skip-networking &
c. 登录刷新授权表
[root@db01 data]# mysql
mysql> flush privilges;
d. 改密码
mysql> alter user root@'localhost' identified by '123';
e. 重启到正常模式
[root@db01 data]# /etc/init.d/mysqld restart
```

Mysql多实例篇

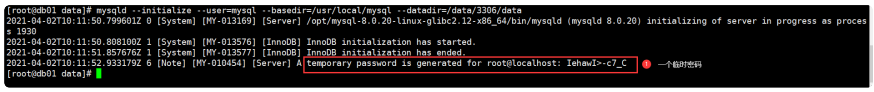
1.不同版本初始化的区别?

5.6（5.5）版本初始化调用的是一个脚本，脚本路径是/usr/local/mysql/scripts/mysql_install_db

5.7版本初始化调用的mysqld --initialize或--initialize-insecure

--initialize(官方建议但不常用)：初始化自动为root@'localhost'(管理员用户)创建临时密码，四种密码复杂度，一共12位

我们使用mysqld --initialize初始化看一下效果



mysqld --initialize使用临时免密登陆后，只有连接权限。所以需要重新给root@'localhost'设置密码

mysql> alter user root@'localhost' indetified by '123';

SQL语句篇

1.sql_mode参数

sql_mode的作用

在MYSQL存储和应用数据时,能够保证数据时准确有效的.防止录入不规矩的数据

查看sql_mode

```
mysql> select @@sql_mode;
```

sql_mode8.0版本规范

ONLY_FULL_GROUP_BY：对于GROUP BY聚合操作，如果在SELECT中的列、HAVING或者ORDER BY子句的列，没有在GROUP BY中出现，或者不在函数中聚合，那么这个SQL是不合法的。

STRICT_TRANS_TABLES：进行数据的严格校验，错误数据不能插入，报error错误。如果不能将给定的值插入到事务表中，则放弃该语句。对于非事务表，如果值出现在单行语句或多行语句的第1行，则放弃该语句。

NO_ZERO_IN_DATE：在严格模式，不接受月或日部分为0的日期。

NO_ZERO_DATE：在严格模式，不要将 '0000-00-00'做为合法日期。

ERROR_FOR_DIVISION_BY_ZERO：在严格模式，在INSERT或UPDATE过程中，如果被零除(或MOD(X，0))，则产生错误(否则为警告)。

NO_ENGINE_SUBSTITUTION：如果需要的存储引擎被禁用或未编译，那么抛出错误。

更多详细信息，看官方文档 <https://dev.mysql.com/doc/refman/8.0/en/sql-mode.html>

2.tinyint int bigint 区别?

tinyint：占用字节1(8位)， 数据长度3

int ： 占用字节4（32位），数据长度10

bigint：占用字节8， 数据长度20

3.char varchar 的区别?

1、 char使用指定的空间；varchar是根据数据量来使用空间

2、 char的插入数据效率理论上比varchar高，varchar是需要通过后面的记录数来计算占用多少字节。

3、两种字符串类型应该如何选择?

（1）如果确定数据一定是占指定长度，那么使用char类型；

（2）如果不确定数据到底有多少，那么使用varchar类型；

（3）如果数据存储长度超过255个字符，不论是否固定长度，都会使用text，不再使用char和varchar

varchar(250) utf8 751字节 不超过 765字节

varchar(190) utf8mb4 761字节 不超过 765字节

在实际生产环境当中不会真的将varchar的值设置到65535，因为索引默认支持的字符量为765，一旦超这个字符限制，是不会走索引的，不走索引必定会拉低整体的数据库性能

4..浮点数存储操作?

我们在存储小数时，会先扩大多少倍，转化为整数，减少存储空间。再使用的时候再把整数缩小多少倍为小数进行使用。
例子
123.456 ----> 扩大1000倍化为整数存储=123456------>缩小1000倍使用=123.456

5. drop truncate delete 区别?

drop table t1
truncate table t1
delet table t1
删除层面 表结构 数据
高水位线 空间释放
drop(DDL语句)： 物理删除 删除表结构(删除元数据) 删除数据
降低高水位线 立即释放空间

truncate(DDL语句)：物理删除 保留表结构 删除数据
降低高水位线 立即释放空间
delete(DML语句)： 逻辑删除 保留表结构 删除数据（底层打上删除标签） 不降低高水位线 不会立即释放空间（会产生碎片）

6.伪删除: 用update替代delete

a. 添加状态列

mysql> alter table stu add state tinyint not null default 1 comment '状态列,1存在,0不存在';

b. 原删除语句替换

原来: delete from stu where sid=7;

替换为: update stu set state=0 where sid=7;

c. 原业务语句查询替换

原来: select * from stu;

替换为: select * from stu where state=1;

7.select查询系统变量与设置用户变量

系统变量是两个@符合，用户变量是一个@符号
1.查看所有数据库中的系统变量
mysql> show variables;
2.查看单个系统变量信息
mysql> select @@变量名称;
3.设置用户变量两种方式 := 赋值符号
方式一：
mysql> select @oldguo:=1;
方式二：
mysql> set @oldguo:=1;
4.调用设置的用户变量
mysql> select @oldguo;

8.union与 union all的区别

union自动去除重复行，会有更多性能消耗（因为要先排序，再去重）

union all 不排序去重，只是将结果集合并在一起

9.mysql5.7版本及之后 sql_mode=only_full_group _by的限制

1.如果select 后的查询列表,不是group by 的条件,又不在聚合函数中存在.就会和SQL_mode不兼容

2.如果group by后的列是主键或唯一键时,可以不使用group by，因为本来就没有重复

所以其他的列我们用到函数 group_concat()

10.group by 执行原理

高级开发上篇

1.生成随机密码的方法?

两种方法：一种纯使用函数，一种是函数加用户变量的方式

Bash | Copy

```
案例一：生成随机密码，随机密码格式要求总共12个，开头是大写字母后面跟上11个数字字母组
1.随机生成开头大写字母
select substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',1+floor(rand()*26),1) as tes
       截取函数 截取字符集                    截取位置          截取长度
mysql> select substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',1+floor(rand()*26),1)
+------+
| test |
+------+
| J    |
+------+
2.随机生成11数字组合，我们可以借用select uuid()随机生成
2.1先进行uuid随机生成的替换
mysql> select replace(uuid(),'-','');
+----------------------------------+
| replace(uuid(),'-','')           |
+----------------------------------+
| d6f245f2986411eb8beb000c29ef43a9 |
+----------------------------------+
2.2 然后截取替换后的uuid结果11个字符。
mysql> select substr(replace(uuid(),'-',''),1+floor(rand()*21),11);
+-----------------------------------------------------+
| substr(replace(uuid(),'-',''),1+floor(rand()*21),11) |
+-----------------------------------------------------+
| 0986511eb8b                                         |
+-----------------------------------------------------+
3.最后将两部分拼接在一起生成我们需要的随机密码
mysql> select concat(substr('ABCDEFGHIJKLMNOPQRSTUVWXYZ',1+floor(rand()
,substr(replace(uuid(),'-',''),1+floor(rand()*21),11)) as '随机密码';
+--------------+
| 随机密码     |
+--------------+
| B1eb8beb000c |

案例二：生成随机密码
0. 将uuid中的横杠替换成空，再赋值给str
mysql> select replace(uuid(),'-','') into @str;
```

2.碎片怎么产生的?

> delete和update操作对于底层而言都没有真正删除数据，只是打上了删除标签，占用数据页空间，可以被重复用。当大批量的delete和update操作就会产生大量的可重复用数据，不被复用就是碎片。

3.产生碎片对查询或更新有什么影响?

> 对查询有影响，对更新无影响
> 对全表查询和范围查询会花费代价，因为碎片也是占用数据页空间空间

4.为什么 alter table table_name engine=innodb.第一次碎片处理效果明显，第二次没有第一次明显?

> 产生大量碎片 第一次会进行碎片整理，降低高水位线，将数据页空间进行合并，减少碎片空间占用量。
>
> 第二次：短时间内多次碎片整理没有明显效果。

高级开发下篇

. 简述数据库三范式?

> 第一范式：保证每列的原子性和不和在分性

> 第二范式：保证每个表只描述一个实体

> 第三范式：保证每个列都和主键相关，字段和主键直接对应，不依赖中间字段

. 简述pt-osc、gh-ost第三方工具处理DDL时的工作原理?

> 1. 查看是否有从节点
> 2. 查看是否有外键
> 3. 创建新表，修改表结构
> 4. 创建触发器，保证拷贝过程中的数据同步
> 5. 拷贝数据
> 6. 给新表重名名
> 7. 删除原表
> 8. 删除触发器

.对SQL语句优化有哪些方法?

> 1. where子句中where表之间的连接必须放在其他where条件之前，那些可以过滤到最大数量的记录条件必须放在where语句末尾，having语句后。
> 2，where子句中使用EXISTS代替IN，使用NOT EXISTS代替NOT IN。
> 3. 避免在索引列上使用计算
> 4. 避免全表扫描，首先应该考虑在where和order by所涉及的列上建立索引
> 5. 尽量避免where子句中对字段进行noll值判断，否则存储引擎将放弃使用索引而进行全表扫描
> 6. 尽量避免where子句中对字段进行表达式操作，否则存储引擎放弃使用索引而进行全表扫描
> 7. 避免在索引列上使用IS NULL和IS NOT NULL。

索引篇

1..如何计算索引树高度?

807f3c1c858f.png&title=MySQL%E5%B0%8F%E7%9F%A5%E8%AF%86%E5%BD%A9%E8%9B%8B%E5%A4%A7%E9%9B%86%E5%90%88%