

2 Mysql基础⼊⻔-部署与管理体系



2.Mysql基础⼊⻔-部署与管理体系



1.5.7,8.0版本企业规范部署,启动

1.1.准备

 硬件:服务器标准化

1

与公司基础运维团队提供采购建议,根据业务场景和压力选择适合

 软件:系统版本与系统参数优化

 数据库版本标准化

1

1. 确认Supported Platforms https://[www.mysql.com/support/](http://www.mysql.com/support/)
2. 确认安装版本

推荐:20之后的小版本

1. 获取MySQL软件包 https://downloads.mysql.com/archives/community/
2. MD5验证

2

3

4

5

6

7

8

9

10

11

 数据库软件⼯具标准化





CMDB平台标准化-软件打包



1.2.Mysql部署安装

<https://dev.mysql.com/doc/refman/8.0/en/binary-installation.html>

1

1.解压

tar xf mysql-8.0.24-linux-glibc2.12-x86\_64.tar.xz

--可直接用最新版本

2

3

4

1. 2.软连接
2. shell> ln -s /opt/mysql-8.0.29-linux-glibc2.12-x86\_64 /usr/local/mysql

7

1. 3. 环境清理
2. shell> yum remove -y mariadb-libs
3. shell> yum install -y libaio-devel

11

1. 4. 创建用户
2. shell> useradd mysql

14

1. 5. 创建目录
2. shell> mkdir -p /data/mysql3306/

17

1. 6. 授权
2. shell> chown -R mysql.mysql /data/mysql3306/

20

1. 7. 初始化数据
2. shell> vim /etc/profile
3. shell> export PATH=/usr/local/mysql/bin:$PATH
4. shell> source /etc/profile
5. shell> mysqld --initialize-insecure --user=mysql --basedir=/usr/local/mysq l --datadir=/data/mysql3306/data

26

1. 8. 配置文件
2. mkdir -p /data/mysql3306/{log,etc,tmp,data}

29

1. vim /data/mysql3306/etc/my.cnf
2. [mysql]
3. # CLIENT #
4. no\_auto\_rehash
5. port = 3306
6. socket = /data/mysql3306/tmp/mysql.sock
7. default\_character\_set = utf8mb4
8. prompt = "\u:\p:\d>"

38

1. [mysqld]
2. user = mysql
3. port = 3306
4. default\_storage\_engine = InnoDB
5. character\_set\_server = utf8mb4
6. init\_connect = 'SET NAMES utf8mb4'
7. transaction\_isolation = REPEATABLE-READ
8. basedir = /usr/local/mysql
9. datadir = /data/mysql3306/data
10. socket = /data/mysql3306/tmp/mysql.sock
11. pid\_file = /data/mysql3306/data/mysql.pid
12. tmpdir = /data/mysql3306/tmp/
13. slave\_load\_tmpdir = /data/mysql3306/tmp/

52

53 ### LOGGING ###

54

#log\_error\_verbosity log\_error general\_log

general\_log\_file

= 2

= /data/mysql3306/log/mysql.err

= 0

= /data/mysql3306/log/mysql.log

1. 启动数据库

touch /data/mysql3306/log/mysql.err chown -R mysql.mysql /data/mysql3306

mysqld\_safe --defaults-file=/data/mysql3306/etc/my.cnf --user=mysql &

1. 连接数据库

mysql -uroot -p -S /data/mysql3306/tmp/mysql.sock #端口连接 mysql -uroot -p -hlocalhost -P 3306 #TCP链接

1. 关闭数据库

mysqladmin shutdown -S /data/mysql3306/tmp/mysql.sock

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71



2.MySQL架构体系讲解



2.1.客户端与服务端模式

1

客户端遵从MySQL Protocol,一套用来连接MySQL client和MySQL Server端的协议，用来支持

* Connectors (Connector/C, Connector/J, and so forth)
* MySQL Proxy
* Communication between master and slave replication servers

连接方式:

Socket (Unix Scoket) TCP/IP

API:C、PHP、JDBC、ODBC、.NET、Python、Go...

2

3

4

5

6

7

8

9

MySQL基架的⼏⼤模块组件

Connectors:MySQL向外提供的交互接⼝

对外交互组件,API,如java,python,go等语⾔可以通过该组件连接数据库管理服务组件和⼯具组件(Management Service & Utilities)

集成对MySQL的管理,⽐如备份，恢复等

Connection Pool:连接池组件

负责监听对客户端向MySQL Server端的各种请求，接收请求，转发请求到⽬标模块。

每个成功连接MySQL Server的客户请求都会被创建或分配⼀个线程，该线程负责客户端与MySQL Server端的通信，接收客户端发送的命令，传递服务端的结果信息 等。

SQL Interface:SQL接⼝组件

接收⽤户SQL命令，如DML,DDL和存储过程等，并将最终结果返回给⽤户

Parser:查询分析器组件分析SQL语法合法

SQL命令分解成数据结构失败则提示语法错误

Optimizer:优化器组件

对SQL命令按照标准流程进⾏优化分析

Caches & Buffers:缓存主件缓存和缓冲组件

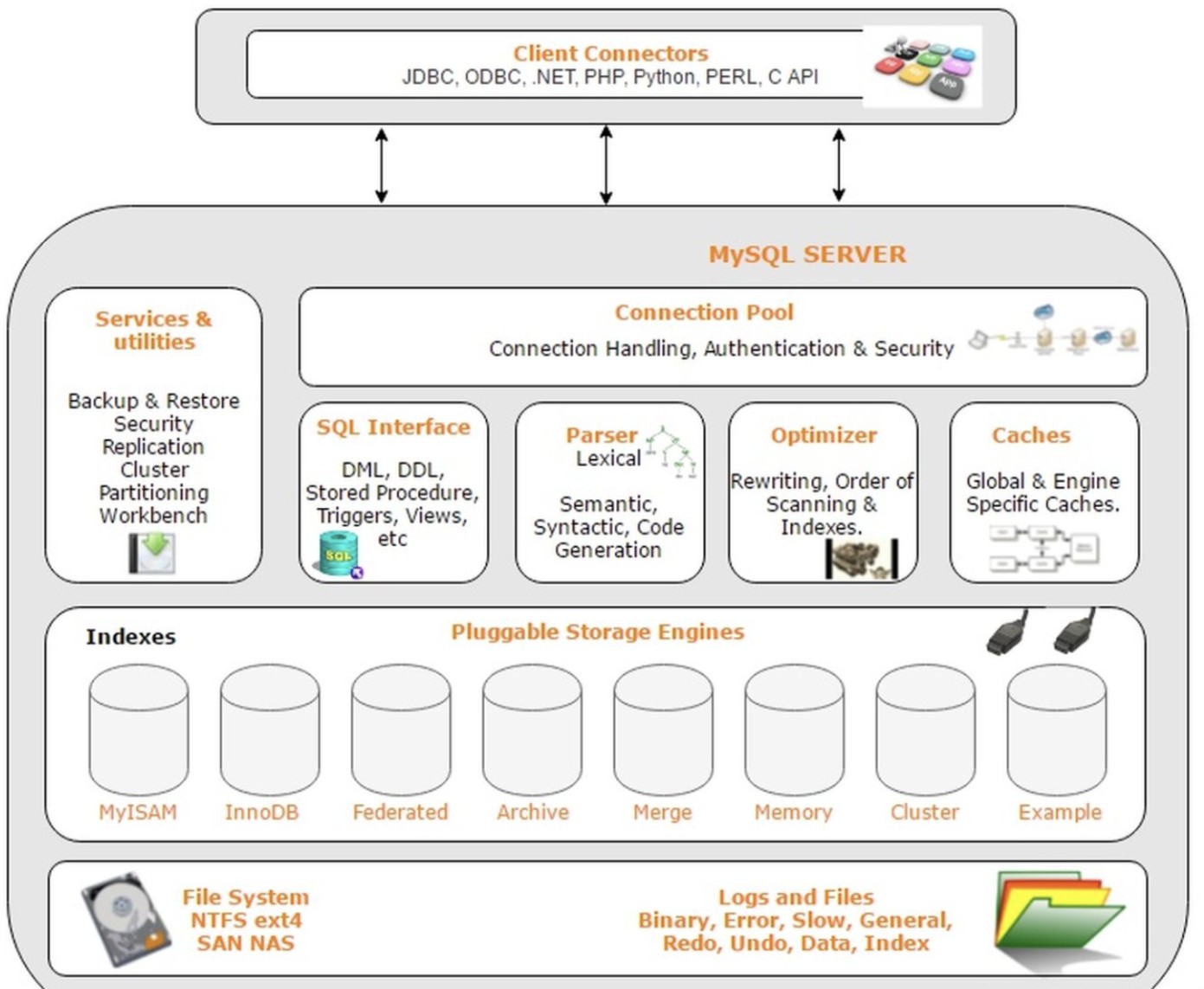
Pluggable Storage Engines:插件式存储引擎

MySQL对表的创建,数据的存储与检索靠引擎来实现

File System:物理⽂件 data

⽇志

配置⽂件等





2.2.⼀条SQL语句的执⾏

1

查询语句的执行分为以下几步：

1. 查询缓存
2. 解析器生成解析树
3. 预处理再次生成解析树
4. 查询优化器

2

3

4

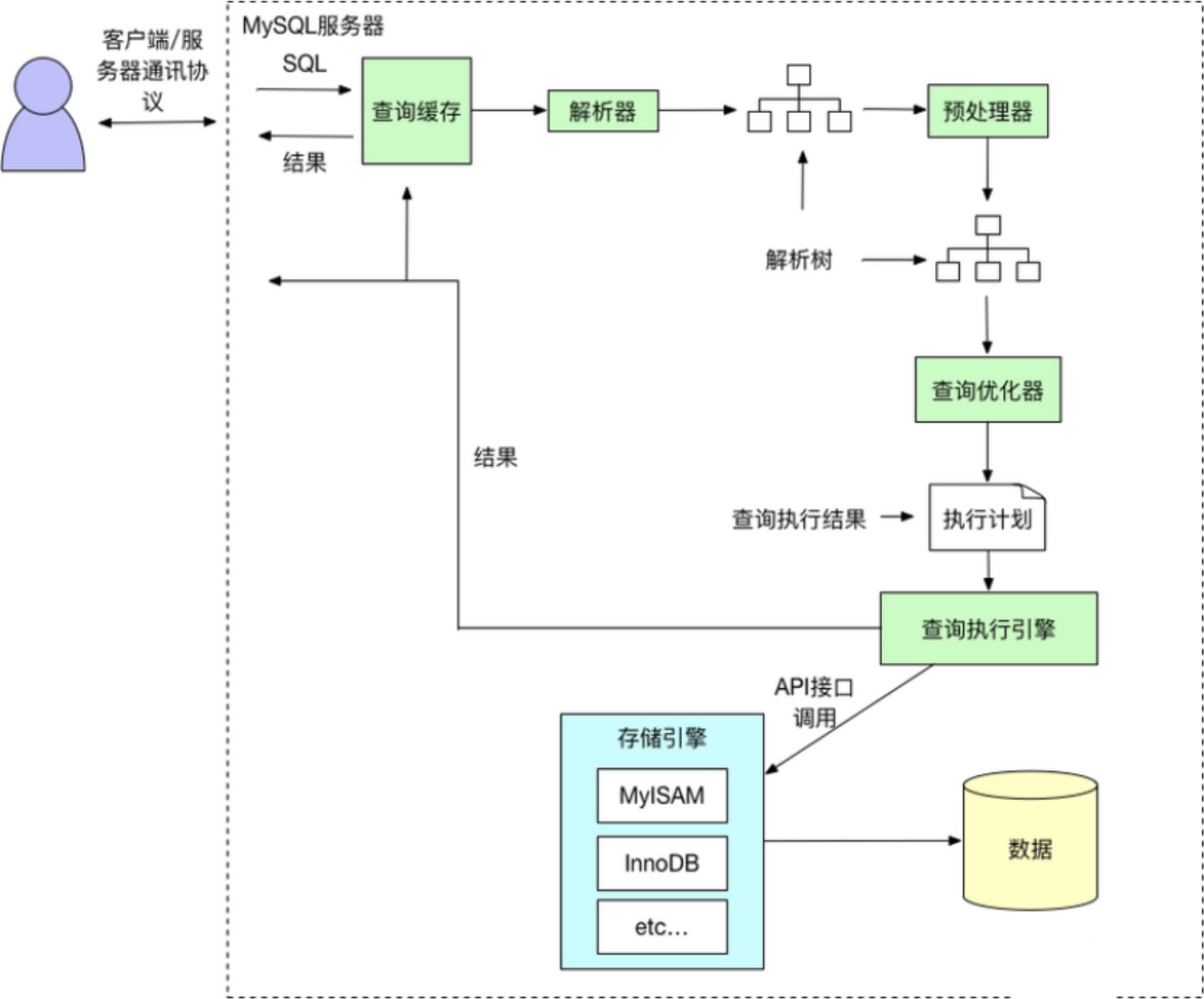
5

6

1. 查询执行计划
2. 查询执行引擎
3. 查询数据返回结果

7

8





2.2.1.查询缓存

已经删除了

1

mysql> show variables like 'query\_cache%';

1. MySQL 拿到一个查询请求后先会在查询缓存中看看是否执行过此语句，之前执行的语句会以

key-value 的形式缓存在内存中，key 是缓存的语句，value 是查询的结果

1. 如果命中缓存则直接将结果返回,如果没有命中则继续执行后面

\*\*额外补充\*\*生产环境中我一般会关闭cache，why?

1. 命中率低:请求sql与环境(连接的数据库、协议版本、字符集等)要与缓存完全一致
2. 资源消耗大:QC需要缓存最新的数据,所以在数据发生变化时(比如增删改操作)QC就会刷新,这导致缓存会频繁的写入、擦除,消耗过多资源
3. QC适用场景少:以下场景无效读取mysql的系统表

子查询或者外层查询

查询语句中用到随机函数,或者now这种函数,给查询结果带来不确定性(每次查询结果会有差异)

查询中用到了视图,临时表

存储过程,触发器中调用的sql也不适合查询语句加了SQL\_NO\_CACHE

查询语句未引用任何表

类似select into这种sql,显示加锁的sql都不会去缓存验证用户的访问权限(用户无权限则拒绝)

MySQL 8.0

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

\*\*全局锁控制\*\*每次更新QC的内存块都需要进行锁定，查询结果必须大于 MySQL 8.0 直接删除查询缓存

最为重要的是，在MySQL里QC是由一个全局锁在控制，每次更新QC的内存块都需要进行锁定。例如，一次查询结果是20KB，参数query\_cache\_min\_res\_unit 值设置为 4KB），查询结果共需要分为5次写入QC，每次都要锁定，而且容易造成block碎片

所以QC适合表数据量少,更新频率低,查询比较多,比如配置表、权限表。关闭方法很简单,有两种:

1、同时设置选项 query\_cache\_type = 0 和 query\_cache\_size = 0；

2、如果用源码编译MySQL的话，编译时增加参数 --without-query-cache 即可；

\*\*生产环境怎么办\*\*

缓存数据给第三方处理,比如redis

19

20

21

22

23

24

25

26

27

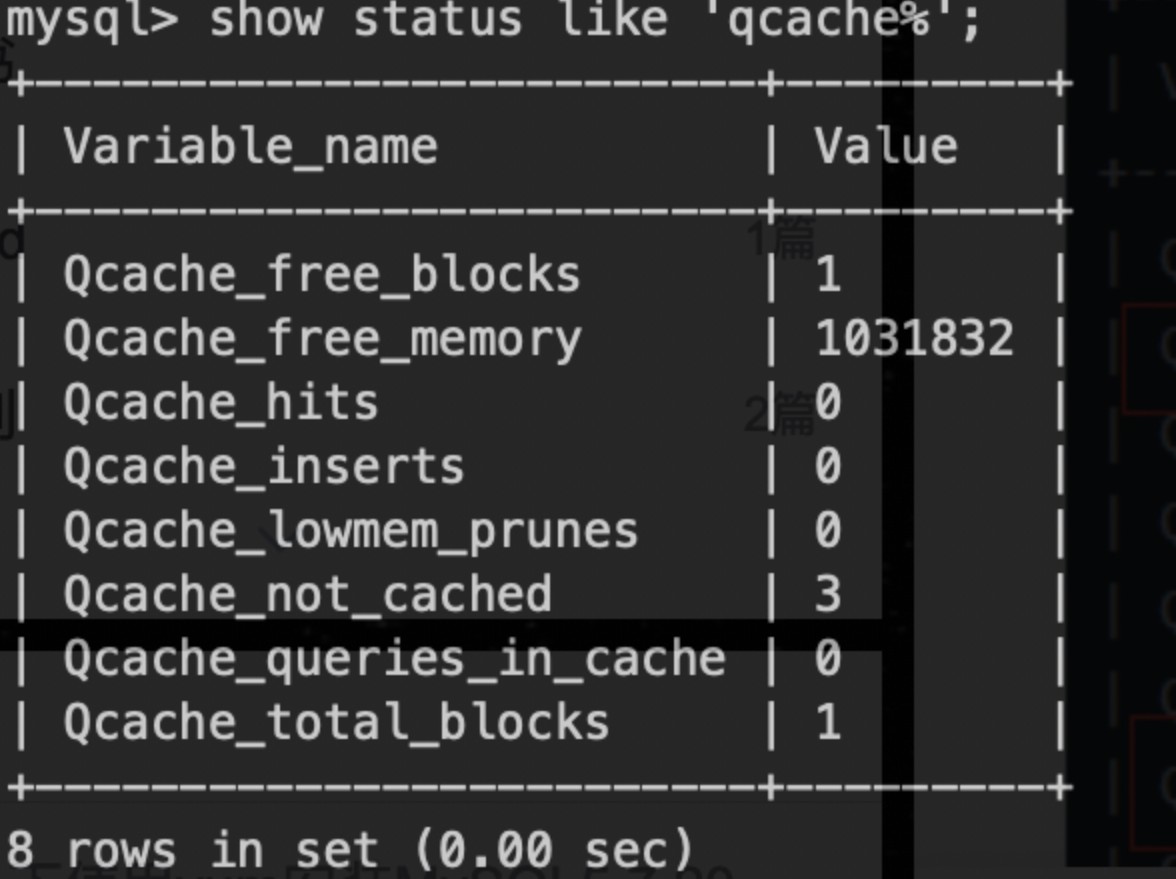
28

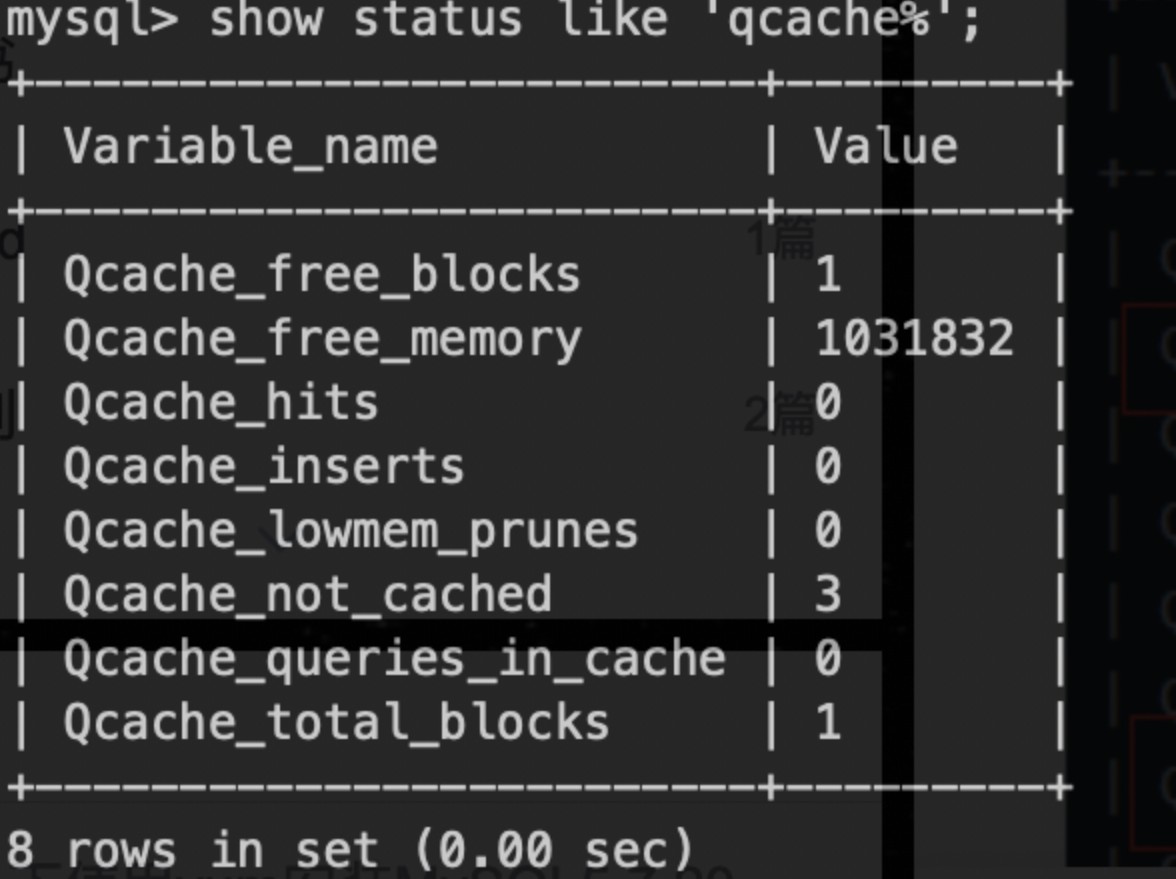
29

30

31

32





课后作业

1

show status; show variables;

是两个DBA运维MySQL最常用的两个查看命令，尝试去概览一遍大概的变量，没有基础的同学可以挑几个感兴趣的值去官方文档或者百度一下，有基础的同学建议导图进行功能分类,比如 show variables like '%innodb%';

show status like '%Com%'; Com\_select Com\_insert Com\_delete

2

3

4

5

6



2.2.2.解析器与解析树

1. 语法解析

1

语法解析是解析你的语句是不是满足 MySQL 语法标准，如果不对则会 ：

ERROR 1064 (42000): You have an error in your SQL syntax … 关于错误码在官网有说明

官方地址:

https://dev.mysql.com/doc/refman/8.0/en/error-message-elements.html https://dev.mysql.com/doc/mysql-errors/8.0/en/server-error-reference.html# error\_er\_open\_as\_readonly

2

3

4

5

6

7

8

#中文对照表

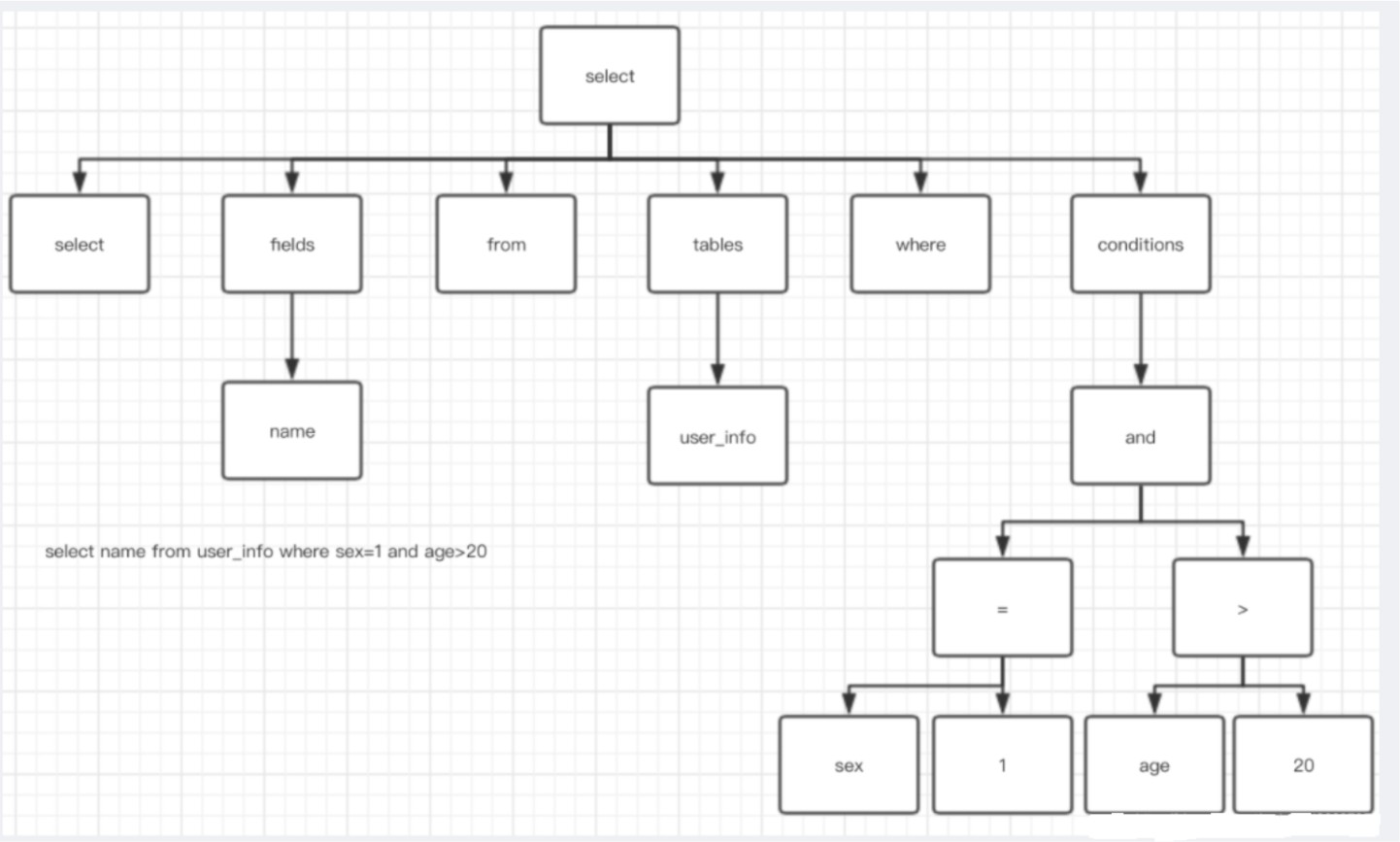
https://[www.cnblogs.com/Nedved/p/10472467.html](http://www.cnblogs.com/Nedved/p/10472467.html)

9

1. 词法解析

1

select name from user\_info where sex=1 and age>20





2.2.3.⽣成解析树

同上



2.2.4.查询优化器



1.⽣成执⾏计划

1

生成一个执行计划,要做一些事情：选择最合适的索引；

选择表扫还是走索引；选择表关联顺序；

优化 where 子句；排除管理中无用表；

决定 order by 和 group by 是否走索引；尝试使用 inner join 替换 outer join；简化子查询，决定结果缓存；

合并视图；

2

3

4

5

6

7

8

9

10

11



2.查询执⾏计划

查询最后⼀次查询的消耗,选择开销最⼩的计划进⾏

1

show status like 'Last\_query\_cost';



2.2.5.查询执⾏引擎与返回结果

这⾥执⾏器会先对权限做⼀个判断，如果有权限，才会执⾏以下步骤，否则跑出权限异常：

1

调用 Innodb 引擎接口获取这个表或索引的第一行，筛选sex=1和age>20的结果集如果不是跳过，如果是则存在结果集中；

引擎执行下一行，重复判断相同的逻辑，直到最后一行；最后将满足结果的结果集返回；

2

3

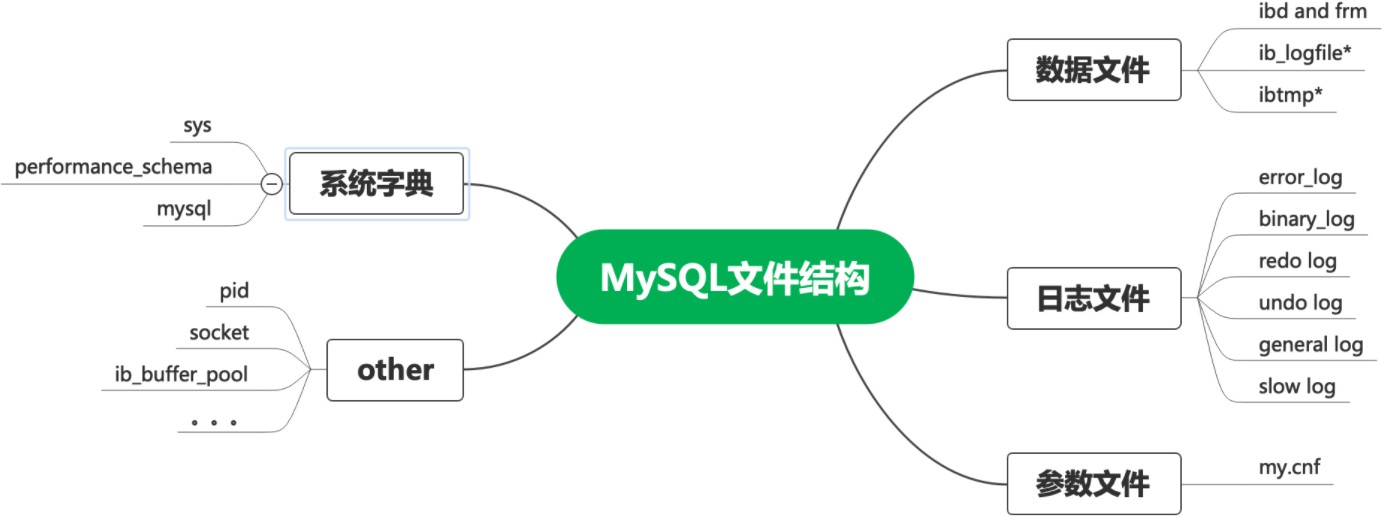
4



2.3.MySQL内部结构



2.3.1.⽂件结构





2.3.2.内存结构

共享内存:整个数据库实例共享

1

show variables where variable\_name in ('innodb\_buffer\_pool\_size','innodb\_l og\_buffer\_size','innodb\_additional\_mem\_pool\_size','key\_buffer\_size','query

\_cache\_size');

#参数说明： 1.innodb\_buffer\_pool

* Innodb 引擎最重要的缓存区域,性能决定性参数之一。包含数据页、索引页、undo 页、i nsert buffer、自适应哈希索引、锁信息以及数据字典等信息。
* 优点:提升数据库的性能、加快 SQL 运行速度
* 缺点:故障发生后值越大恢复速度较慢

2.innodb\_log\_buffer

- 存放redo log

2

3

4

5

6

7

8

9

10

11

1. innodb\_additional\_mem\_pool
   * 该部分主要存放 InnoDB 内的一些数据结构.
2. key\_buffer
   * MyISAM 表的重要缓存区域,索引缓存再此
3. query\_cache
   * 查询缓存

12

13

14

15

16

17

18

19

Session私有内存:每个线程连接是独有

1

show variables where variable\_name in ('read\_buffer\_size','read\_rnd\_bu ffer\_size','sort\_buffer\_size','join\_buffer\_size','binlog\_cache\_size','tmp\_ table\_size');

如果连接数为1000则,MySQL实例使⽤内存为

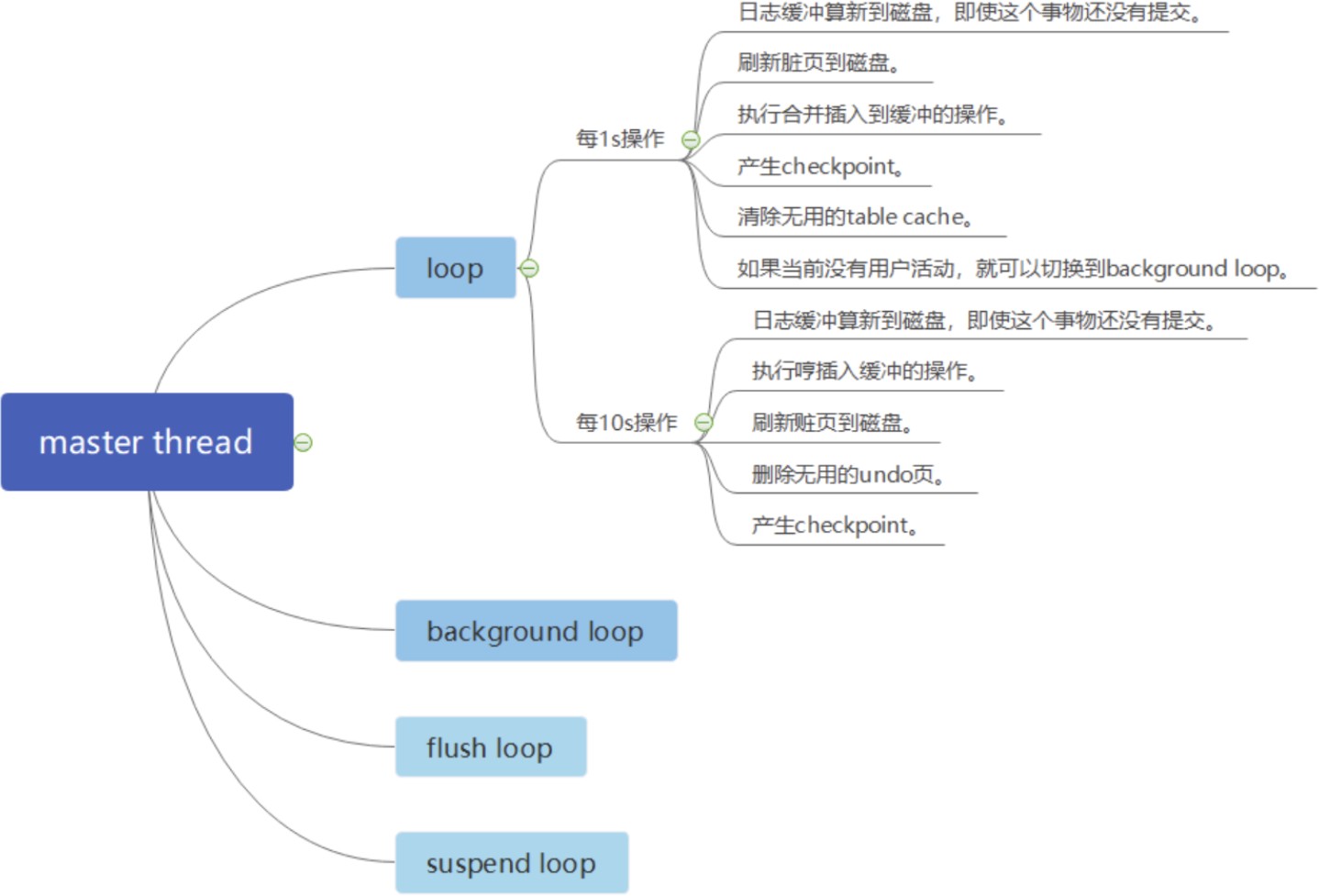
共享内存+1000Xsession内存

1



2.3.3.线程结构







2.1.3 MySQL 8.0的"新姿势"

Metadata 结构变化

5.7 版本问题

两套数据字典信息（Server层 frm，InnoDB 数据字典） DDL⽆原⼦化

frm和innodb层会出现不⼀致

并发处理需要⼩⼼处理（MDL,dict\_sys::mutex,dict\_sys::rw\_lock）崩溃⽆法恢复

8.0 变化

⽀持事务性DDL，崩溃可以回滚，保证⼀致。保留⼀份数据字典信息，取消frm数据字典。数据字典存放⾄InnoDB表中

采⽤套锁机制，管理数据字典的并发访问（MDL）全新的Plugin⽀持

8.0.17+ 加⼊Clone Plugin,更好的⽀持MGR，InnoDB Cluster的节点管理安全加密⽅式改变

改变加密⽅式为caching\_sha2\_password SSL 将⽀持到 TLSv1.3 版本。

⽤户管理及认证⽅式改变改变授权管理⽅式

加⼊role⻆⾊管理添加更多权限

原⼦性DDL

⽀持原⼦性DDL Cache && Buffer的变化

取消Query Cache