<http://3brother.cn/mysql.html>

1常见的一些文件夹的作用

二进制日志log-bin ：

主重复制

错误日志log-error：

默认是关闭的,记录严重的警告和错误信息,每次启动和关闭的详细信息等.

查询日志log：

默认关闭,记录查询的sql语句，如果开启会减低mysql的整体性能，因为记录日志也是需要消耗系统资源的

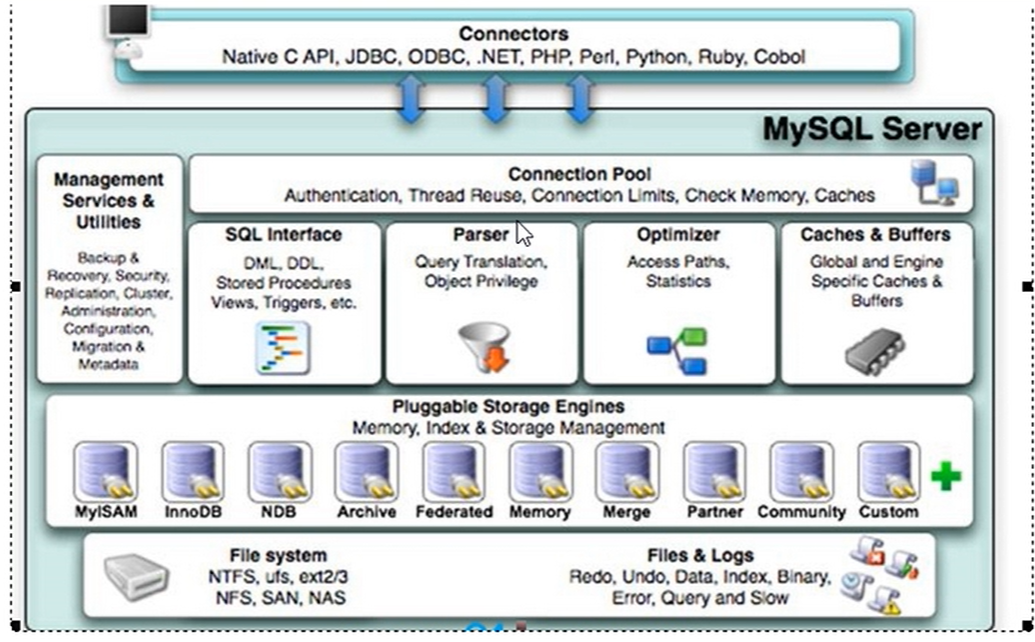
2常见的一些文件的作用

frm文件：存放表结构

myd文件：存放表数据

myi文件 ：myi文件

3、mysql的的内部分层结构



Mysql主要采取分层结构，

第一层为连接层，主要负责与各种语言的sql交互

第二层为处理层，将sql进行分类（sql interface），解析（parser），优化（optimizer），缓存

第三层为存储引擎，存储引擎有多种，是可插拨非方式进行实现

第四层为内存

4存储引擎的特点



5Sql执行满的原因

性能下降SQL慢

5.1执行时间长

查询语句写的烂

索引失效

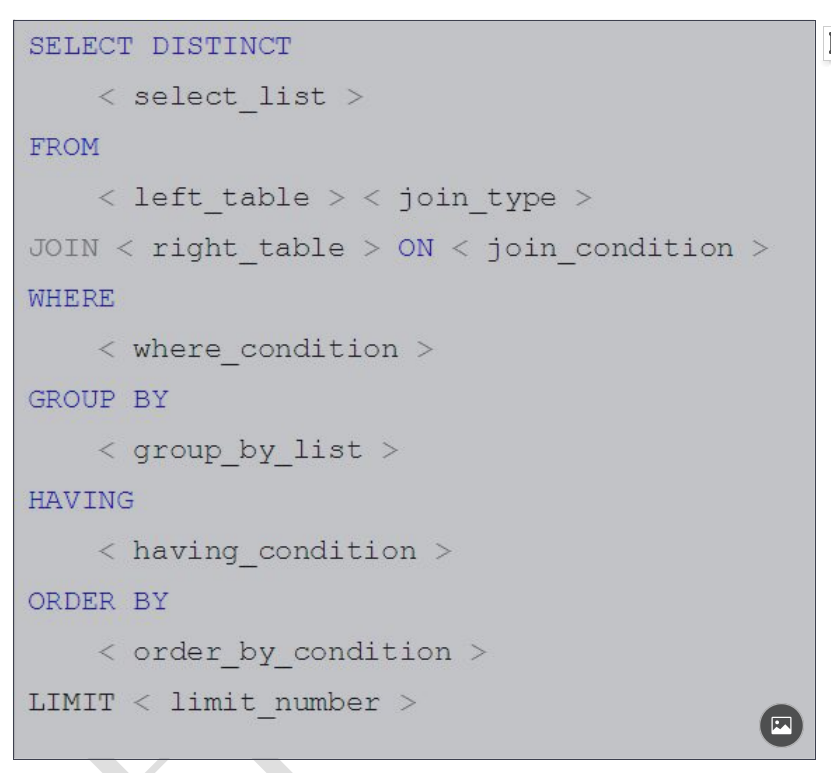
关联查询太多join

5.2等待时间长

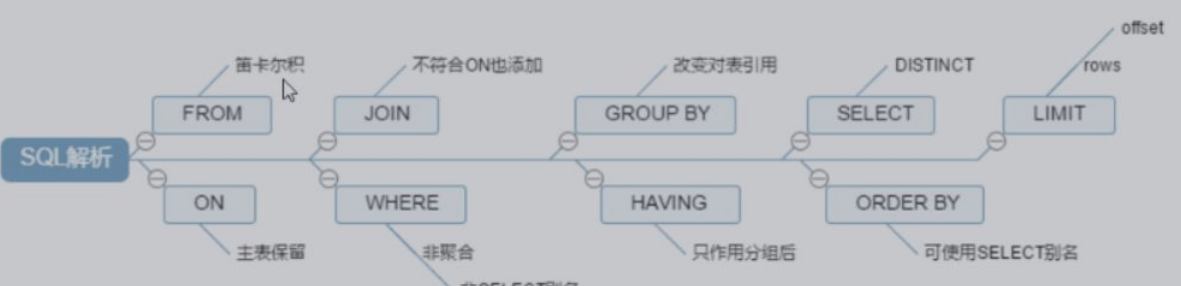
服务器调优及各个参数设置(缓冲\线程数等)

6手写sql与机器执行sql的思考方式

6.1手写



6.2机器如何执行sql



7索引的作用

索引的作用为查找和排序，以及行锁

8优缺点

优点：提高查询和排序的效率

缺点：在进行数据更新的时候也必须进行索引的维护，建议一张表的索引不能超过五个

9mysql的索引的种类

9.1b+树索引

默认的索引，相较于b树索引，查询效率更加稳定，并且在三层就可以记录上百万的数据，除此之外B+树的磁盘读写代价更低，因为相较于b树，b+树出了叶子节点，其他节点上没有指向磁盘的指针，因此每个块可以存储更多的数据，每次可以读取更多的数据

9.2hsah

Hsah索引适合做关键字查询

10索引分类

10.1单列索引

在其中一个列上建立索引

10.2主键索引

为主键创建的索引，inndb上为聚簇索引，可以快速查询，主键索引可以多个列，并且这个列不能为空，

10.3唯一索引

不能重复，可以为空

10.4复合索引

包含多个列的索引

11适合创建索引的情况

主键自动建立唯一索引；

频繁作为查询条件的字段应该创建索引

查询中与其它表关联的字段，外键关系建立索引

单键/组合索引的选择问题， 组合索引性价比更高

查询中排序的字段，排序字段若通过索引去访问将大大提高排序速度

查询中统计或者分组字段

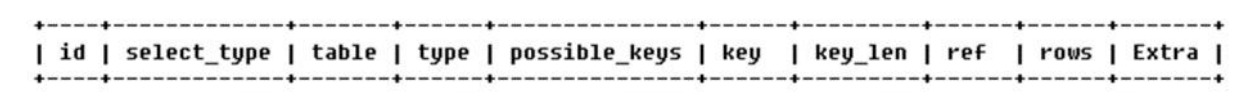
12不适合的情况

数据太少

经常删改的字段

不出现在where里

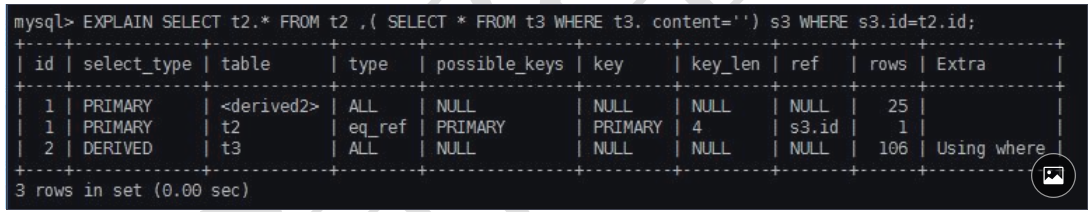
13explain的sql分析表的解读



13.1id

Id表示着对应的表的执行的顺序，id数字越高则执行的顺序在前，若id相同则按照从上到下的方式执行

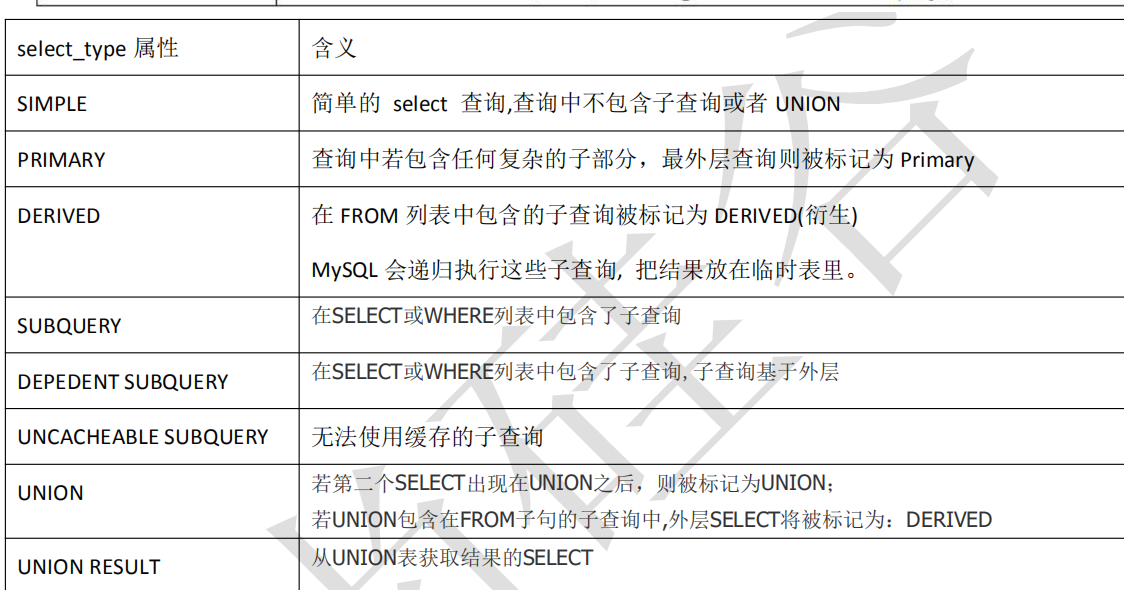
如



执行顺序我t3-》derived2（衍生表2）-》t2

13.2select\_type

select\_type 代表查询的类型，主要是用于区别普通查询、联合查询、子查询等的复杂查询



13.3table

代表哪个表

13.4type

type 是查询的访问类型。是较为重要的一个指标，结果值从最好到最坏依次是：

system > const > eq\_ref > ref > fulltext > ref\_or\_null > index\_merge > unique\_subquery > index\_subquery > range > index > ALL ，一般来说，得保证查询至少达到 range 级别，最好能达到 ref。

13.5possible\_key 肯能使用到的索引（可以忽略）

13.6key 具体使用的索引

13.7key\_len 表示索引的长度，

13.8 ref显示的是索引用到了哪一个列，做好的情况是concast，表示是一个常量

13.9rows 表示mysql执行查询的时候最少需要查询多少行，越少越好

13.10Extra 显示重要的一些其他信息

13.10.1有using fileSort 表示排序的时候无法用到索引，而是另外做一步排序的操作，可能是order by或者group by 的字段上没有索引，或者排序的字段和缩影建立的字段的顺序不同，如建立索引 x1 x2 非要排序x2 x1（不好）

13.10.2using temporary

使用了零时表存储中间结果，常见于order by group by（不好）

13.10.3using index

Using index 代表表示相应的 select 操作中使用了覆盖索引(Covering Index)，避免访问了表的数据行，效率不错！

如果同时出现 using where，表明索引被用来执行索引键值的查找;如果没有同时出现 using where，表明索引只是

用来读取数据而非利用索引执行查找。

利用索引进行了排序或分组

13.10.4 using where

表明使用了 where 过滤。

13.10.5 using join buffer

使用了连接缓存，如果不好可加大一下缓存

14优化口诀

全值匹配我最爱，最左前缀要遵守

带头大哥不能死，中间兄弟不能断

索引列上少计算，范围之后全失效

Like百分写最右，覆盖索引不写\*

不等空值还有or，索引失效要注意

字符引号不可丢，sql优化要注意

其他要注意的一些点：is null索引不会失效

is not null != <> 索引失效

一些注释：

索引列上少计算：

不在索引列上做任何操作（计算、函数、(自动 or 手动)类型转换），会导致索引失效而转向全表扫描。

范围之后全失效：

Between < > 会导致之后的索引失效，如有复合索引（a，b，c），查询where a = 1 and b>2 and c = 3

那么索引只能用到ab，c用不上

15优化的步骤

1开启慢查询日志捕获查询速度慢的sql

2explain+sql分析sql慢的原因然后优化

3show profile 查看sql的全生命周期然后优化

4调整mysql参数，如缓冲区的大小

16查询优化技巧

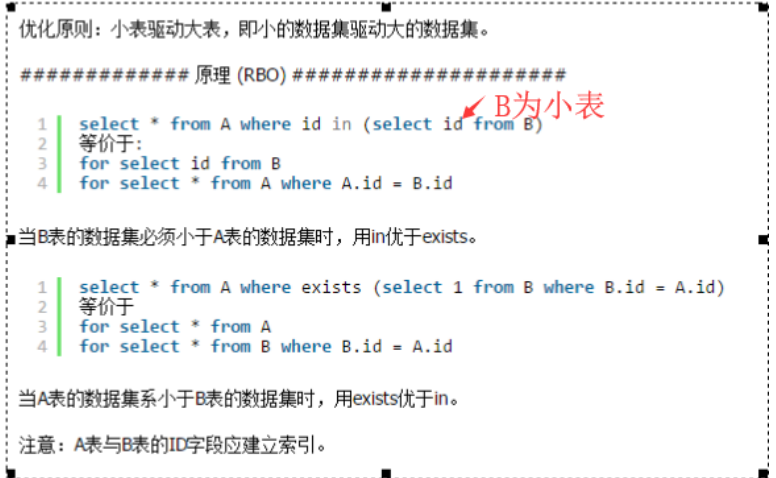
16.1小表驱动大表

Mysql会先从小表拿出一条数据然后去扫描大表进行匹配，小表驱动大表可以减少io

In和exist的优劣就和小表驱动大表有关

首先in的话是先查括号里的也就是in后面的，因此数据量越少越好，内表驱动外表

exist则是先查询外面的数据，然后将外面查出的数据放到里面去做验证，其实就是外表驱动内表



16.2order by 和group by

Orderby 和group by都需要排序

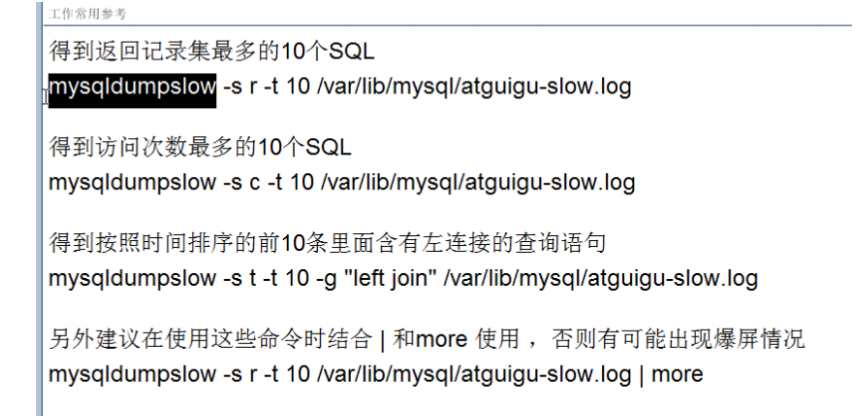
对于这两个关键字都需要排序，因此最好让排序在索引上完成，遵守最佳左前缀原则，对于不能用到索引排序的的情况，mysql会另外进行排序，也就是filesort，在mysql4.1之后，mysql采用单路排序的方式，将查出的合适数据，1读取到缓冲区，2排序 3输出，如果缓冲区太小，无法一次读取数据，那么就会创建临时文件进行合并排序，就会造成多次io反而更慢

对于这类的优化一是让排序用上索引，第二就是加大缓冲区避免多次io，第三就是对于需要排序的情况不要用select \* 因为查询过多的字段容易导致排序的时候缓存不足，导致多次io

17慢查询日志

慢查询日志一般默认是关闭的，因为打开后会消耗一定的性能，打开慢查询日志后，mysql会根据你设置的阈值记录执行时间超过这个阈值的sql，从而达到抓出慢sql的目的，mysql阈值默认为10秒

18开启日志后分析日志也是很费力的，mysql提供了mysqldumpslow进行日志的分析，如查看最近



19索引失效会导致行锁变成表锁

Mysql innodb支持行锁，这样可以更高的并发，然而行锁是建立在索引的上面，没有索引的话mysql使用全表扫描，这样的话行锁就会变成表锁，导致并发变弱。

20间隙锁

在我们使用范围更新，如我们的表里有id为1,3,4,5,6,7,8,9的记录，我们执行update aaa set x = 1 where id between（1，9）的时候，mysql会锁定1-10的id，这时候即使我们的表里没有id为2的记录，另一个会话想要插入id为2的记录也是不能插入。

21如何锁定一行（悲观锁）

Select \* from table1 where id = 10 for update

for update 在查询语句上加上for update则会锁住一行，其实这也是悲观锁的实现方式，在操作的时候先查询出要修改的记录，然后锁住，然后进行后续的操作