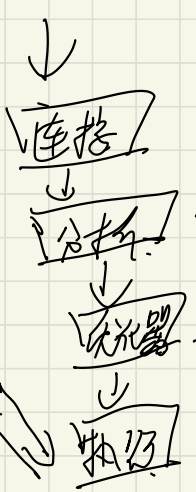
MySql 相关

为报错还是20的问题



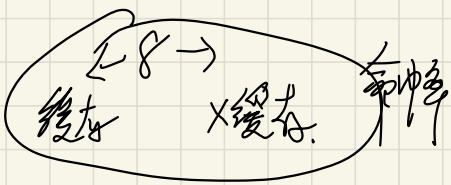
AST：抽象语法树

RBO：基于规则

CBO 基于成本

IO

聚集



---

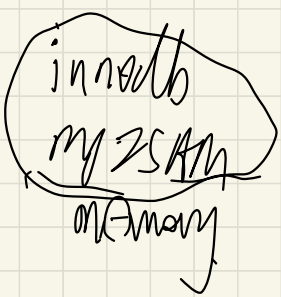
show profile：具体时间 all

performance_schema

在执行化.

instruments：生产者

consumers：消费者



---

show processlist. 查看连接.

数据库连接池：C3P0，druid（德鲁伊）

更小的通常更好.

占用空间不一样.

frm 表结构.
ibd. 数据文件

myisam { MYD: 数据文件.
MYI: 索引.

ORM?

存IP. INET_ATON 把IP地址整数存储.
NTOA

Schame

null ≠ null 特殊字符代替null.

Varchar 根据实际长度保存 { <255 额外一个字节保存长度
>255 两个字节

读取4K. innodb 16K 20数据.

使用场景 { 1. 存储长度波动较大.
2. 字符串很少更新的场景, 每次更新都重新计算
3. 适合保存多字节字符. 如: 汉字, 特殊字符

char { 类型自动把以尾空格
检索效率.

存储数据长度不大 (md5加密)
存储短字符串.
经常更新

BLOB和TEXT 代替者 FTP啥的, 加和使用问题

dateTime
- 8个字节
- 日期类型
- 不要用字符串存储
- 范围大

timestamp
- 4个
- 1970-01-01 - 2038-01-19
- 精确到秒
- 整形存储
- 设置日期

date
- 3个
- 日期之间比较
- date 1000-01-01 到 9999-12-31

枚举
- 男女 直接数据库进行转化操作
- 0 1

ifm

特殊类型: INET_ATON 进行IP的转换

空间换时间
反范式

缓存衍生值 { 自己维护 / 发消息 / 进行更新

主键选择： {
代理主键 与业务无关
自然主键 如身份证号 与业务相关
推荐代理主键 { 与业务不耦合 / 键策略，可继续扩展

流水号的概念

字符集选择： {
UTF-8 存储2字节
UTF8Mb4

存储引擎选择！

innodb.
myISAM
memery：不能持久化！

	MyISAM	Innodb
索引类型	非聚簇索引（索引—数据）	聚簇索引（数据+索引）
支持事务	否	是
支持表锁	是	是
支持行锁	否	是
支持外键	否	是
支持全文索引	是	是（5.6+）
适合操作类型	大量select	大量insert, update, delete

适当冗余 → mysql：用一个单独的表。
√ 视图 oracle { mysql没有。
物化视图

适当拆分：{ 垂直切分：按业务切分 mycat？
水平切分：1-1000
1000-2000

处投文，单独存放！！！四张表。

执行计划！explain：

{ 以相同 从上到下
  以不同，从下向上

select_type { simple 类型的意思
primary
union
subquery

type { 访问方式. 从最底

range 范围查询

const 常量最优，二级索引指定一行

ref 相等于二.

---

possible_keys 可能有的索引

key 实际使用的索引

key_len. 索引长度.

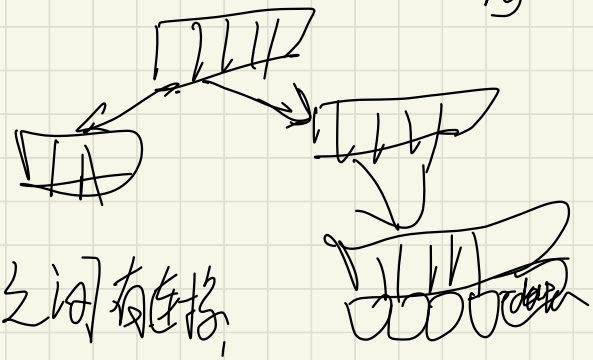
索引优化：B+树 索引是数据

Hash. 是memory 内存不能持久化.

innodb
MyISAM.

16K.
4K的整数倍

B+树 只有叶子节点才存数据.



B+树：非叶子节点之间有连接.

---

innodb：聚簇索引 叶子节点直接放数据

MyISAM：非聚簇索引. 放的是数据的地址.

组合索引：如 idx_name_age_no idx_cols

主键索引是唯一索引的体现.

回表：name中存的主键，再回到表中查整行数据
子查询中可用

索引覆盖：查的就是id，就不回表了，就是覆盖了

最左匹配：

select * from emp name = ? and age = ? ✓

select * from emp where age = ? ✗

组合索引 name, age.

解决 { (name)的(age) 两个 ✗
(age, name) 组合索引 ✓ }

先左再右
无左无右

select * from emp where name = ?   (name, age), age ✓
存储量小

谓词下推：取的量小

索引下推 before：→ name 取出 → 过滤 age   server

存储引擎中做 → 直接取值的时候就过滤 age 了

页分裂问题导致的一些问题，以及放到内存中会

更加大. 每次插入. 删除. 都可能会导致页分裂和页合并. 移动.

先关闭, 再传数据再开索引快.

覆盖索引 只查索引列就覆盖了. 加一个就不对了.

优化级 type system - const - ref - range - index - All

explain 强制类型转换会全表

> < >= <= 会用到索引.

更新频繁不建议索引.

创建索引的列不允许为null. 可能会得到不一样的结果.

join_Buffer.

能用limit就用limit. 但用oracle时row时 1-5可以 6时得查一层.

问优化经历. 有过一个查询很慢, 用执行计划看, 发现type是All, 最后优化到const.

Handler_read_key
Handler_read_rnd_next
多或者大索引失效

查询缓存在8后已被干掉. LRU: 最少使用.

连接器

ASTree. Calcite 抽象语法树 语法解析

查询优化器：选择最有效的执行计划 CBU 成本 RBU 规则

成本最小

重新关联表的顺序 优化器决定比较高

将外连接转化为内连接

等价变化规则

用等价替换规则

count() min(), max() 加上分组条件，可以用上索引

覆盖索引

子查询优化 放入缓存中

等值传播 如果两个列通过等值关联，能够把where对一个列的条件性列另一个上

Straight join 强制使用自己的定义表顺序

排序优化：{ 两次 / 一次

先把数据拿出来排序 再去表中读

到

取消 随机IO

占用大量的存储空间，可能不自多

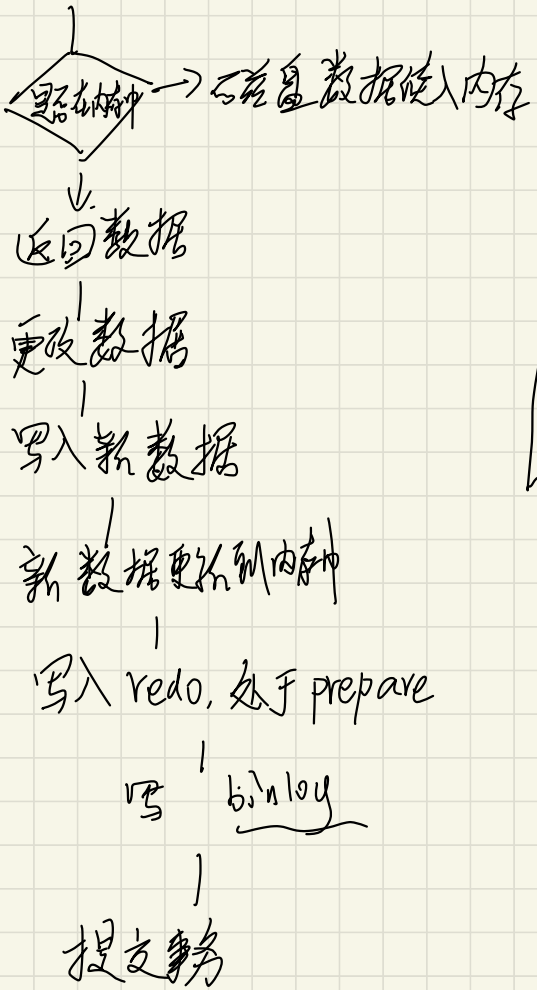
---

count(*) count(?) count(1) 都是一样的

带 where count的都不快，单独 count的快

标识列分组会更高的查询效率

数据更新流程



先写 redo log 再写 binlog

Undo log 事务原子性

insert — delete

update — up

delete — insert

可以随时返回当时的状态

返回事务开始之前的状态

查找 count 可以用

select 1 from table a where a.a>1 and a.b<2 limit 1

读锁：被锁后无法更新 无法读其他表

写锁 锁一张 其他对这张的查询不能查询出来。

劣点 MyISAM

Innodb：共享锁 X 排它锁，可以理解为行锁读锁

排他锁：可以理解为写锁