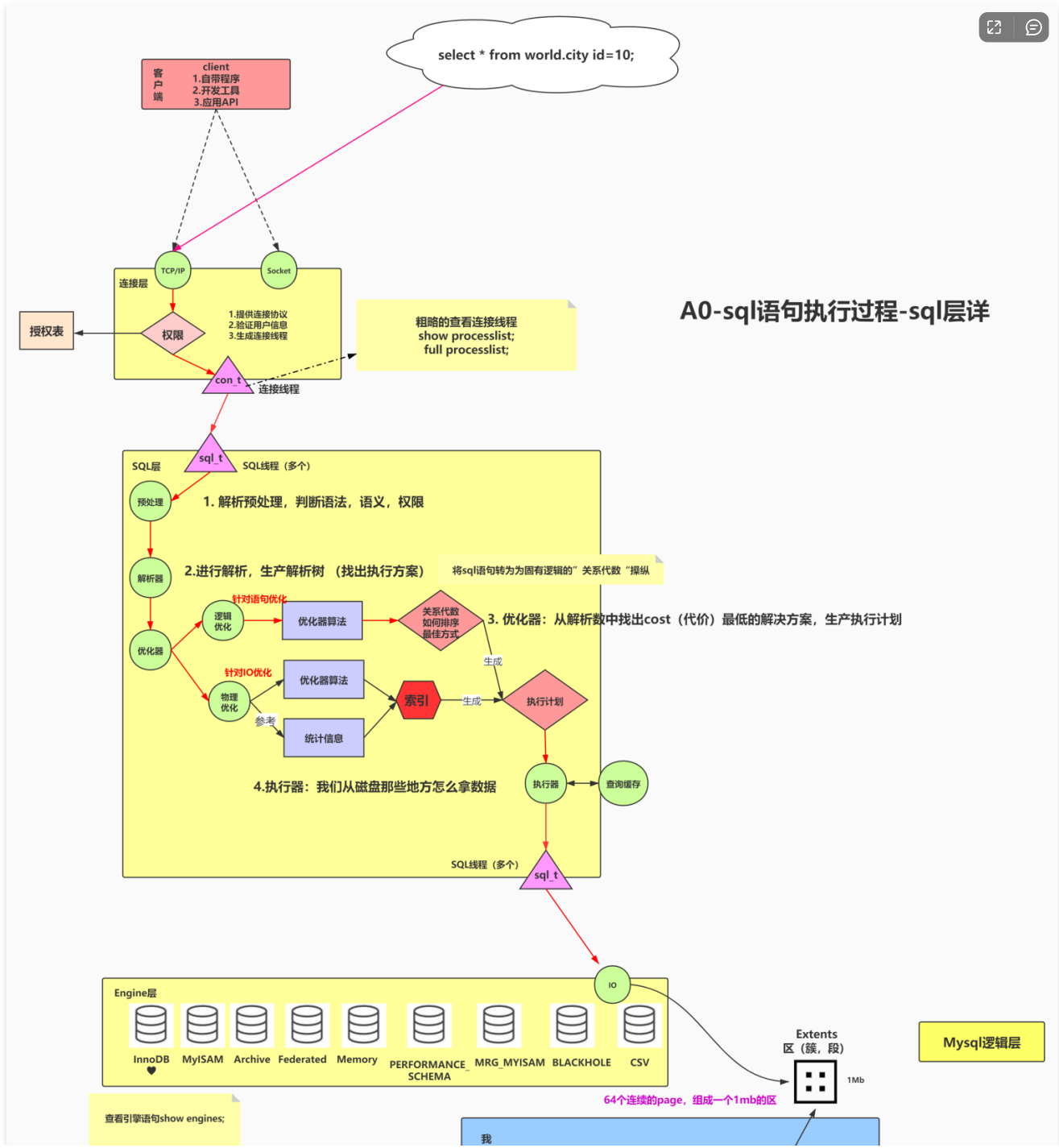
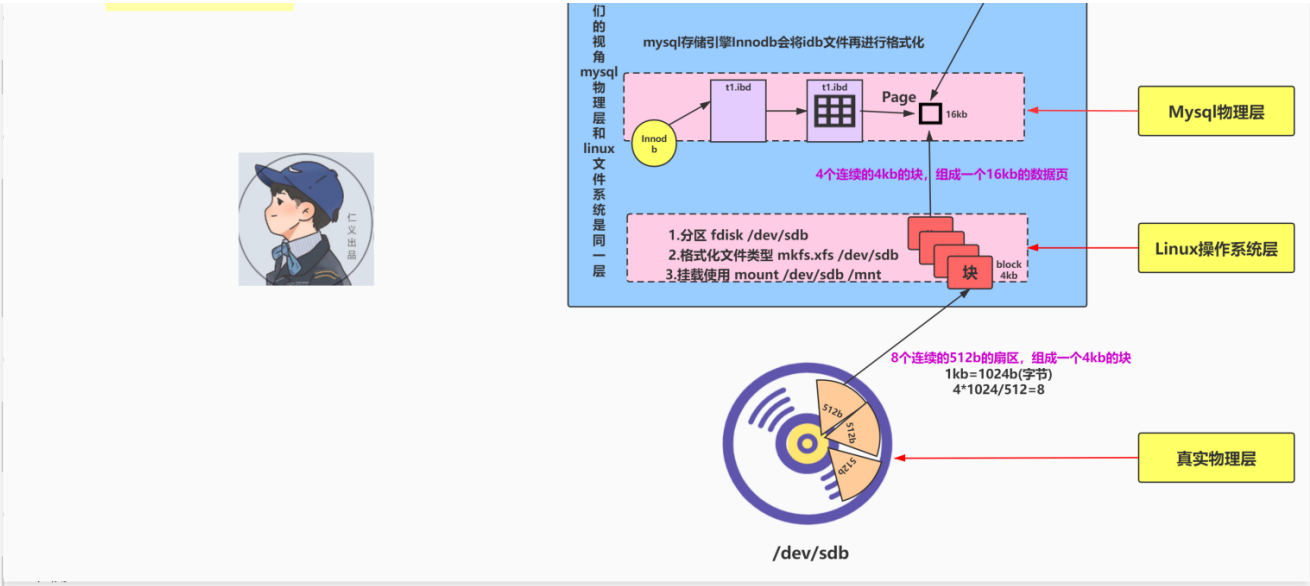
15.MySQL-索引√

1.索引BTREE算法介绍

1.0 SQL语句执行过程（sql层详）


select * from world.city id=10;
客户端
client
1.自带程序
2.开发工具
3.应用API
TCP/IP
Socket
连接层
1.提供连接协议
2.验证用户信息
3.生成连接线程
授权表
权限
con_t
连接线程
粗略的查看连接线程
show processlist;
full processlist;
A0-sql语句执行过程-sql层详
SQL层
sql_t
SQL线程（多个）
预处理
1. 解析预处理，判断语法，语义，权限
解析器
2.进行解析，生产解析树 （找出执行方案）
将sql语句转为为固有逻辑的”关系代数“操纵
针对语句优化
逻辑优化
优化器算法
关系代数
如何排序
最佳方式
3. 优化器：从解析数中找出cost（代价）最低的解决方案，生产执行计划
优化器
生成
针对IO优化
物理优化
优化器算法
索引
生成
执行计划
参考
统计信息
执行器
查询缓存
4.执行器：我们从磁盘那些地方怎么拿数据
SQL线程（多个）
sql_t
IO
Engine层
InnoDB
MyISAM
Archive
Federated
Memory
PERFORMANCE_SCHEMA
MRG_MYISAM
BLACKHOLE
CSV
Extents
区（簇，段）
1Mb
64个连续的page，组成一个1mb的区
Mysql逻辑层
查看引擎语句show engines;
我



不管查找哪个数,需要查找次数理论上是相同的.对于一个三层b树来讲,理论上查找每个值都是三次IO.

2.擅长范围查找

讲究快速锁定范围. B+tree,加入了双向指针(头尾相接),进一步增强范围查找.减少对于ROOT和NON-LEAF的访问次数.

1.5 BTREE原理图

2.MySQL索引BTREE算法使用

2.1 索引btree算法分类

1.聚簇索引
2.辅助索引
- 单列索引
- 联合索引
- 前缀索引

3.唯一索引

2.2 聚簇索引

2.2.1 前提

1、 如果表中设置了主键（例如ID列），自动根据ID列生成聚簇索引
2、 如果没有设置主键，自动选择第一个唯一键的列作为聚簇索引
3、 自动生成隐藏（6字节row_id）的聚簇索引。
建议： 在建表时，显示的创建主键，最好是数字自增列。

2.2.2 功能

聚簇索引组织表。
将逻辑上连续的数据，在磁盘存储时也是物理（同一个区内）上连续的。
1. 录入数据时，按照聚簇索引组织存储数据，在磁盘上有序存储数据行。
2. 加速查询。基于ID作为条件的判断查询。

2.2.3 btree构建过程

a. 叶子节点： 存储数据行时就是有序的，直接将数据行的page作为叶子节点（相邻的叶子结点，有双向指针）
b. 枝节点 ： 提取叶子节点ID的范围+指针，构建枝节点（相邻枝节点，有双向指针）
c. 根节点 ： 提取枝节点的ID的范围+指针，构建根节点

2.2.4 原理图

2.3 辅助索引

2.3.0 辅助索引介绍

1.前提

> 需要人为创建辅助索引，将经常作为查询条件的列创建辅助索引，起到加速查询的效果。

2.功能

> 按照辅助索引列，作为查询条件时。
> 1. 查找辅助索引树，得到ID值
> 2. 拿着ID值回表（聚簇索引）查询

3.btree构建过程

> btree 构建过程
> a. 叶子节点：提取主键（ID）+辅助索引列，按照辅助索引列进行从小到大排序后，生成叶子节点。（相邻的叶子结点，有双向指针）
> b. 枝节点 ： 提取叶子节点辅助索引列的范围+指针，构建枝节点（相邻枝节点，有双向指针）
> c. 根节点 ： 提取枝节点的辅助索引列的范围+指针，构建根节点

4.原理图

2.3.1 单列索引

构建过程

> `alter table t1 add index idx(name)`
> 1. 从原表中获取:索引列(name)+ID值
> 2. 安装索引列值(name)从小到大排序,生成叶子节点中.
> 3. 枝节点:叶子节点的name范围+指针
> 4. 根节点:枝节点name的范围+指针

如何提供查询优化

> 基于name 列进行条件时.
> 1. 根据name列的条件值,在辅助索引扫描,获取到ID
> 2. 拿着ID回表查询,最中获得想要的数据页

2.3.2 联合索引

构建过程

> 叶子节点: 获取ID+name+age ,按照name和age组合排序. 将有序的值存储到连续的数据页中.
> 枝节点 ：获取叶子节点name列值范围+指针.
> 根节点 ：获取枝节点 name值的范围+指针.

如何提供查询优化

> 例如:
> where name = and age=
> 1. 按照name条件值,扫描根节点和枝节点,找到叶子结点.
> 2. 根据叶子节点内容在做age 条件过滤,最终获得ID
> 3. 回表查询,根据ID扫描聚簇索引,最终得到数据页.

3.MySQL索引管理命令

3.0 压测

> source /root/t100w.sql
> mysqlslap --defaults-file=/etc/my.cnf --concurrency=100 --iterations=1 --
> create-schema='test' --query="select * from test.t100w where k2='780P'"
> engine=innodb --number-of-queries=2000 -uroot -p123 -verbose
>
> --concurrency=100： 模拟同时100会话连接
> --create-schema='test'： 操作的库是谁

> --query="select * from test.t100w where k2='780P'"： 做了什么操作

> --number-of-queries=2000： 一共做了多少次查询

> Average number of seconds to run all queries: 719.431 seconds

> Minimum number of seconds to run all queries: 719.431 seconds

> Maximum number of seconds to run all queries: 719.431 seconds

3.1 查询索引

Bash | Copy

```
1.查看列信息
desc world.city;
Key
----
PRI     ---> 主键索引(聚簇索引)
MUL     ---> 辅助索引
UNI     ---> 唯一索引
2.查看索引常用命令
mysql> show index from world.city;
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+
| Table | Non_unique | Key_name    | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+
| city  |          0 | PRIMARY     |            1 | ID          | A         |        4188 |     NULL |   NULL |
| city  |          1 | CountryCode |            1 | CountryCode | A         |         232 |     NULL |   NULL |
+-------+------------+-------------+--------------+-------------+-----------+-------------+----------+--------+
Table      : 表名
Key_name   : 索引名
Column_name: 列名
Cardinality :基数 ,不重复的值的个数(根据统计信息,采样获得的错略值).越大越好.
量化方法: Cardinality/总行数  建议 80% 以上
```

3.2 创建索引

Bash | Copy

```
a. 单列索引
mysql> alter table world.city add index i_pop(population);

b. 联合索引
mysql> alter table world.city add index i_c_p(countrycode,population);

c. 前缀索引
mysql> alter table world.city add index i_name(name(10));

d. 主键索引
mysql> create table aa (id int);
mysql> alter table aa modify  id int not null primary key auto_increment;

e. 唯一索引
mysql> alter table aa add telnum  char(11);
mysql> alter table aa add unique index i_tel(telnum);
```

3.3 删除索引

```bash
mysql> alter table aa drop index i_tel;
```

4. MySQL索引考虑的事项

4.1 回表问题

1. 回表是什么?

> 按照辅助索引列，作为查询条件时，先查找辅助索引树得到ID，再到聚簇索引树查找数据行的过程

2.回表产生的问题?

> IO量多、IO次数多、随机IO会增多

3.怎么减少回表?

> 减少回表建议：
>
> 1. 索引覆盖
>
> 辅助索引能够完全覆盖查询结果，可以使用联合索引。
>
> 2. 精细化查询条件+合理的联合索引
>
> 尽量让查询条件精细化，尽量使用唯一值多的列作为查询条件
>
> 3.调整优化器算法
> 优化器：MRR（Multi-Range-Read）， 锦上添花的功能。
>
> mysql> select @@optimizer_switch;
>
> mysql> set global optimizer_switch='mrr=on';
>
> 功能：
>
> 1. 辅助索引查找后得到ID值，进行自动排序
>
> 2. 一次性回表，很有可能受到B+TREE中的双向指针的优化查找。

4.2 索引树高度问题

1.索引高度影响因素及解决

> a. 高度越低越好
>
> b. 数据行越多，高度越高。
>
> 1. 分区表。一个实例里管理。

> 2. 按照数据特点，进行归档表。

> 3. 分布式架构。针对海量数据、高并发业务主流方案。

> 4. 在设计方面，满足三大范式。

> c. 主键规划：长度过长。

> 1. 主键，尽量使用自增数字列。

> d. 列值长度越长，数据量大的话，会影响到高度。

> 1. 使用前缀索引

> 100字符 只取前10个字符，构建索引树。

> e. 数据类型的选择。

> 选择合适的、简短的数据类性。

2.索引树高度计算

第一步：确定那个数据页

通过information_schema.information_schema的PAGE_NO列确定那个页

第二步：跳过数据页中前64个字节 找到PAGE_LEVEL占用字节显示的16进制转化为10进制

5.索引应用规范

5.1 建立索引的原则（DBA运维规范）

> 说明

> 为了使索引的使用效率更高，在创建索引时，必须考虑在哪些字段上创建索引和创建什么类型的索引。

> (1) 必须要有主键,最好数字自增列。

> (2) 经常做为where条件列 order by group by join on, distinct 的条件(业务:产品功能 +用户行为)

> (3) 联合索引最左原则。

(4) 列值长度较长的索引列,我们建议使用前缀索引.

(5) 降低索引条目,一方面不要创建没用索引,不常使用的索引清理,percona toolkit(xxxxx)

(6) 索引维护要避开业务繁忙期，建议用pt-osc

5.1 建立索引的原则（DBA开发规范）

(1)没有查询条件，或者查询条件没有建立索引

(2)查询的结果集，超过了总数行数25%，优化器觉得就没有必要走索引了。

(3)索引本身失效，统计信息不真实（过旧）

(4)查询条件使用函数在索引列上，或者对索引列进行运算，运算包括(+，-，*，/，！等)

(5)隐式转换导致索引失效.这一点应当引起重视.也是开发中经常会犯的错误.

(6)<> ，not in 不走索引（辅助索引）

(7)单独的>,<,in 有可能走，也有可能不走，和结果集有关，尽量结合业务添加limit

(8)like "%_" 百分号在最前面不走

%E7%B4%A2%E5%BC%95%E2%88%9A%20%7C%201.%E7%B4%A2%E5%BC%95BTREE%E7%AE%97%E6%B3%95%E4%BB%8B%E7%BB%8D1.0%20SQL%E8%AF%AD%E5%8F%A5%E6%89%A7%E8%A1%