查询优化 索引优化 表结构优化

1.索引的顺序 b-tree

2.使用b-tree数据结构来存储数据

b-tree所有的值都是按顺序存储的

从索引的根节点开始进行索引 根节点的槽中存放了指向子节点的指针，存储引擎根据这些指针向下层查找。

全值匹配

匹配最左前缀

匹配列前缀

匹配范围值

1.如果不是按照索引的最左列开始查找，则无法使用索引

2.不能跳过索引中的列

3.如果查询中某个列的范围查询，则其右边所有列都无法使用索引优化查询

分析是否是需要的行和需要的列。

1.sql查询优化:分析是否是需要的行和需要的列。

2.确认MySQL服务器层是否在分析大量超过需要的数据行

少用select \*

看查询结果是否扫描了过多的数据。响应时间 扫描的行 返回的行。

响应时间:服务时间+排队时间；服务时间（数据库处理这个查询真正花了多长时间） 排队时间:等I/O,锁等

explain type列反应了访问类型，全表扫描，范围扫描，唯一索引查询、常数引用 ALL ref

MySQL能使用如下三种方式应用WHERE条件，从好到坏依次为：

1）在索引中使用WHERE条件来过滤不匹配的记录

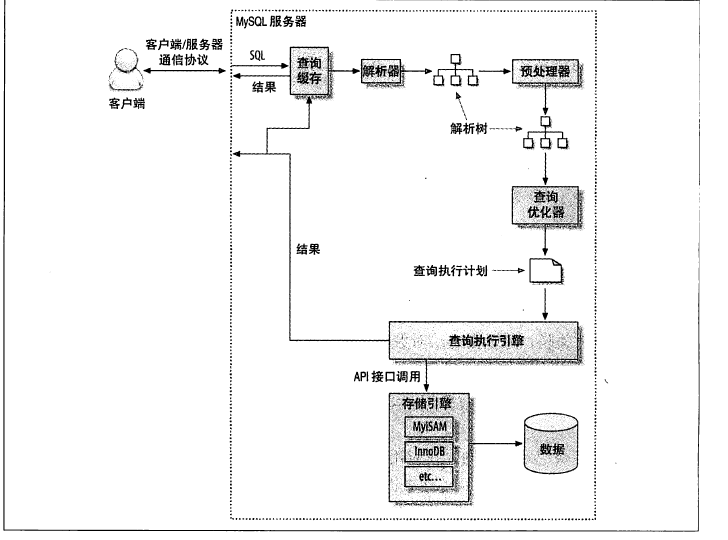
2）使用索引覆盖扫描来返回记录，直接从索引中过滤不需要的继续并返回命中的结果。

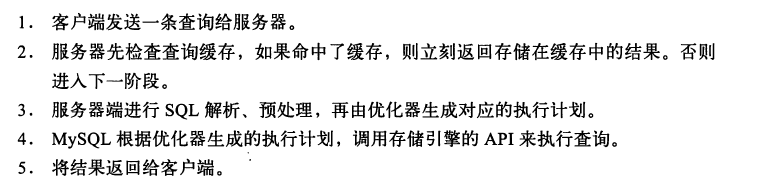
3）从数据中返回数据，然后过滤不满足条件的记录出现Extra列中出现了Using index

1.复杂查询分为多个简单查询

2.切分查询

3.分解关联查询





show full processlist 返回结果中的comand列表示当前的状态

Sleep Query Locked Analyzing and Statistics

Coping to tmp table on[on desk]

Sorting Result

Sending data

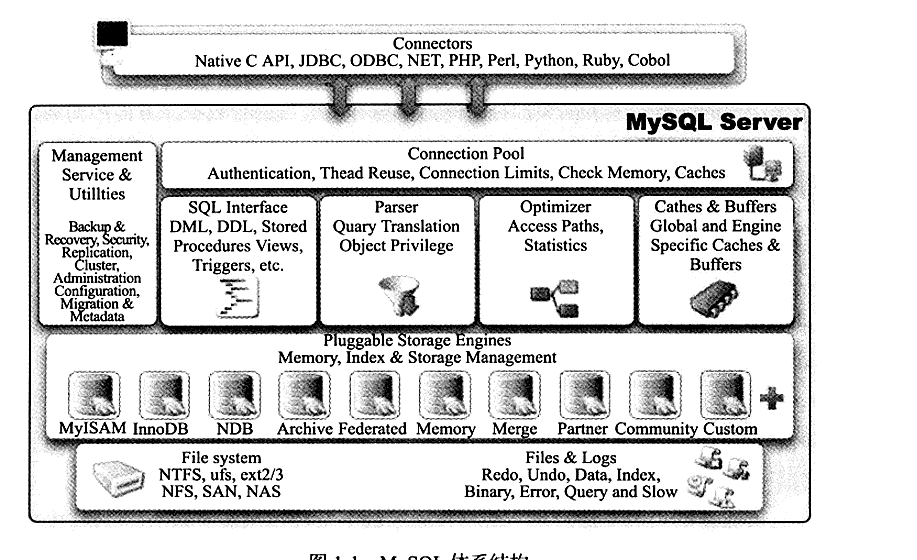
解析SQL，预处理、优化SQL执行计划。

show STATUS like 'last\_query\_cost'

重新定义关联表的顺序

将外链转化为内连接

使用等价变换规则



5.5.8行级锁设计 支持外键 非锁定读 读取操作不会产生锁