主要针对关系型数据库MySQL。键值类数据库参考

<https://www.jianshu.com/p/098a870d83e4>

创建时和查询时两个阶段的优化。

### 1、逻辑架构



* 第一层：客户端通过连接服务，将要执行的sql指令传输过来
* 第二层：服务器解析并优化sql，生成最终的执行计划并执行
* 第三层：存储引擎，负责数据的储存和提取

### 2、锁

数据库通过锁机制来解决并发场景-共享锁（读锁）和排他锁（写锁）。读锁是不阻塞的，多个客户端可以在同一时刻读取同一个资源。写锁是排他的，并且会阻塞其他的读锁和写锁。简单提下乐观锁和悲观锁。

* 乐观锁，通常用于数据竞争不激烈的场景，多读少写，通过版本号和时间戳实现。
* 悲观锁，通常用于数据竞争激烈的场景，每次操作都会锁定数据。

要锁定数据需要一定的锁策略来配合。

* 表锁，锁定整张表，开销最小，但是会加剧锁竞争。
* 行锁，锁定行级别，开销最大，但是可以最大程度的支持并发。

但是MySql的存储引擎的真实实现不是简单的行级锁，一般都是实现了多版本并发控制（MVCC）。MVCC是行级锁的变种，多数情况下避免了加锁操作，开销更低。MVCC是通过保存数据的某个时间点快照实现的。

### 3、事务

事务保证一组原子性的操作，要么全部成功，要么全部失败。一旦失败，回滚之前的所有操作。MySql采用自动提交，如果不是显式的开启一个事务，则每个查询都作为一个事务。

隔离级别控制了一个事务中的修改，哪些在事务内和事务间是可见的。四种常见的隔离级别：

* 未提交读（Read UnCommitted），事务中的修改，即使没提交对其他事务也是可见的。事务可能读取未提交的数据，造成脏读。
* 提交读（Read Committed），一个事务开始时，只能看见已提交的事务所做的修改。事务未提交之前，所做的修改对其他事务是不可见的。也叫不可重复读，同一个事务多次读取同样记录可能不同。
* 可重复读（RepeatTable Read），同一个事务中多次读取同样的记录结果时结果相同。
* 可串行化（Serializable），最高隔离级别，强制事务串行执行。

### 4、存储引擎

InnoDB引擎，最重要，使用最广泛的存储引擎。被用来设计处理大量短期事务，具有高性能和自动崩溃恢复的特性。

MyISAM引擎，不支持事务和行级锁，崩溃后无法安全恢复。

### 5、创建时优化

#### 5.1 Schema和数据类型优化

整数

TinyInt,SmallInt,MediumInt,Int,BigInt 使用的存储8,16,24,32,64位存储空间。使用Unsigned表示不允许负数，可以使正数的上线提高一倍。

实数

* Float,Double , 支持近似的浮点运算。
* Decimal，用于存储精确的小数。

字符串

* VarChar，存储变长的字符串。需要1或2个额外的字节记录字符串的长度。
* Char，定长，适合存储固定长度的字符串，如MD5值。
* Blob，Text 为了存储很大的数据而设计的。分别采用二进制和字符的方式。

时间类型

* DateTime，保存大范围的值，占8个字节。
* TimeStamp，推荐，与UNIX时间戳相同，占4个字节。

优化建议点

* 尽量使用对应的数据类型。比如，不要用字符串类型保存时间，用整型保存IP。
* 选择更小的数据类型。能用TinyInt不用Int。
* 标识列（identifier column），建议使用整型，不推荐字符串类型，占用更多空间，而且计算速度比整型慢。
* 不推荐ORM系统自动生成的Schema，通常具有不注重数据类型，使用很大的VarChar类型，索引利用不合理等问题。
* 真实场景混用范式和反范式。冗余高查询效率高，插入更新效率低；冗余低插入更新效率高，查询效率低。
* 创建完全的独立的汇总表\缓存表，定时生成数据，用于用户耗时时间长的操作。对于精确度要求高的汇总操作，可以采用 历史结果+最新记录的结果 来达到快速查询的目的。
* 数据迁移，表升级的过程中可以使用影子表的方式，通过修改原表的表名，达到保存历史数据，同时不影响新表使用的目的。

#### 5.2 索引

索引包含一个或多个列的值。MySql只能高效的利用索引的最左前缀列。索引的优势：

* 减少查询扫描的数据量
* 避免排序和零时表
* 将随机IO变为顺序IO （顺序IO的效率高于随机IO）

B-Tree

使用最多的索引类型。采用B-Tree数据结构来存储数据（每个叶子节点都包含指向下一个叶子节点的指针，从而方便叶子节点的遍历）。B-Tree索引适用于全键值，键值范围，键前缀查找，支持排序。

B-Tree索引限制：

* 如果不是按照索引的最左列开始查询，则无法使用索引。
* 不能跳过索引中的列。如果使用第一列和第三列索引，则只能使用第一列索引。
* 如果查询中有个范围查询，则其右边的所有列都无法使用索引优化查询。

哈希索引

只有精确匹配索引的所有列，查询才有效。存储引擎会对所有的索引列计算一个哈希码，哈希索引将所有的哈希码存储在索引中，并保存指向每个数据行的指针。更多面试题，欢迎关注公众号 Java面试题精选

哈希索引限制：

* 无法用于排序
* 不支持部分匹配
* 只支持等值查询如=，IN（），不支持 < >

优化建议点

* 注意每种索引的适用范围和适用限制。
* 索引的列如果是表达式的一部分或者是函数的参数，则失效。
* 针对特别长的字符串，可以使用前缀索引，根据索引的选择性选择合适的前缀长度。
* 使用多列索引的时候，可以通过 AND 和 OR 语法连接。
* 重复索引没必要，如（A，B）和（A）重复。
* 索引在where条件查询和group by语法查询的时候特别有效。
* 将范围查询放在条件查询的最后，防止范围查询导致的右边索引失效的问题。
* 索引最好不要选择过长的字符串，而且索引列也不宜为null。

### 6、查询时优化

#### 6.1 查询质量的三个重要指标

* 响应时间 （服务时间，排队时间）
* 扫描的行
* 返回的行

#### 6.2 查询优化点

* 避免查询无关的列，如使用Select \* 返回所有的列。
* 避免查询无关的行
* 切分查询。将一个对服务器压力较大的任务，分解到一个较长的时间中，并分多次执行。如要删除一万条数据，可以分10次执行，每次执行完成后暂停一段时间，再继续执行。过程中可以释放服务器资源给其他任务。
* 分解关联查询。将多表关联查询的一次查询，分解成对单表的多次查询。可以减少锁竞争，查询本身的查询效率也比较高。因为MySql的连接和断开都是轻量级的操作，不会由于查询拆分为多次，造成效率问题。
* 注意count的操作只能统计不为null的列，所以统计总的行数使用count（\*）。
* group by 按照标识列分组效率高，分组结果不宜出行分组列之外的列。
* 关联查询延迟关联，可以根据查询条件先缩小各自要查询的范围，再关联。
* Limit分页优化。可以根据索引覆盖扫描，再根据索引列关联自身查询其他列。如

SELECT

id,

NAME,

age

WHERE

student s1

INNER JOIN (

SELECT

id

FROM

student

ORDER BY

age

LIMIT 50,5

) AS s2 ON s1.id = s2.id

* Union查询默认去重，如果不是业务必须，建议使用效率更高的Union All

### 7、补充

1. 条件中的字段类型和表结构类型不一致，mysql会自动加转换函数，导致索引作为函数中的参数失效。
2. like查询前面部分未输入，以%开头无法命中索引。
3. 补充2个5.7版本的新特性：

generated column，就是数据库中这一列由其他列计算而得

CREATE TABLE triangle (sidea DOUBLE, sideb DOUBLE, area DOUBLE AS (sidea \* sideb / 2));

insert into triangle(sidea, sideb) values(3, 4);

select \* from triangle;

+-------+-------+------+

| sidea | sideb | area |

+-------+-------+------+

| 3 | 4 | 6 |

+-------+-------+------+

支持JSON格式数据，并提供相关内置函数

CREATE TABLE json\_test (name JSON);

INSERT INTO json\_test VALUES('{"name1": "value1", "name2": "value2"}');

SELECT \* FROM json\_test WHERE JSON\_CONTAINS(name, '$.name1');

关注explain在性能分析中的使用：

EXPLAIN SELECT settleId FROM Settle WHERE settleId = "3679"



* select\_type，有几种值：simple（表示简单的select，没有union和子查询），primary（有子查询，最外面的select查询就是primary），union（union中的第二个或随后的select查询，不依赖外部查询结果），dependent union（union中的第二个或随后的select查询，依赖外部查询结果）
* type，有几种值：system（表仅有一行（=系统表），这是const连接类型的一个特例），const（常量查询）, ref(非唯一索引访问，只有普通索引)，eq\_ref（使用唯一索引或组件查询），all（全表查询），index（根据索引查询全表），range（范围查询）
* possible\_keys: 表中可能帮助查询的索引
* key，选择使用的索引
* key\_len，使用的索引长度
* rows，扫描的行数，越大越不好
* extra，有几种值：Only index（信息从索引中检索出，比扫描表快），where used（使用where限制），Using filesort （可能在内存或磁盘排序），Using temporary（对查询结果排序时使用临时表）