19 | 基础篇总结：如何理解查询优化、通配符以及存储过程？

2019-07-24 陈旸

到这一篇的时候，意味着SQL专栏的基础部分正式更新完毕。在文章更新的时候，谢谢大家积极地评论和提问，让专栏增色不少。我总结了一些基础篇的常见问题，希望能对你有所帮助。答疑篇主要包括了DBMS、查询优化、存储过程、事务处理等一些问题。

关于各种DBMS的介绍

答疑1

文章中有句话不太理解，“列式数据库是将数据按照列存储到数据库中，这样做的好处是可以大量降低系统的 I/O”，可以解释一些“降低系统I/O”是什么意思吗？

解答

行式存储是把一行的数据都串起来进行存储，然后再存储下一行。同样，列式存储是把一列的数据都串起来进行存储，然后再存储下一列。这样做的话，相邻数据的数据类型都是一样的，更容易压缩，压缩之后就自然降低了I/O。

我们还需要从数据处理的需求出发，去理解行式存储和列式存储。数据处理可以分为OLTP（联机事务处理）和OLAP（联机分析处理）两大类。

OLTP一般用于处理客户的事务和进行查询，需要随时对数据表中的记录进行增删改查，对实时性要求高。

OLAP一般用于市场的数据分析，通常数据量大，需要进行复杂的分析操作，可以对大量历史数据进行汇总和分析，对实时性要求不高。

那么对于OLTP来说，由于随时需要对数据记录进行增删改查，更适合采用行式存储，因为一行数据的写入会同时修改多个列。传统的RDBMS都属于行式存储，比如Oracle、SQL Server和MySQL等。

对于OLAP来说，由于需要对大量历史数据进行汇总和分析，则适合采用列式存储，这样的话汇总数据会非常快，但是对于插入（INSERT）和更新（UPDATE）会比较麻烦，相比于行式存储性能会差不少。

所以说列式存储适合大批量数据查询，可以降低I/O，但如果对实时性要求高，则更适合行式存储。

关于查询优化

答疑1

在MySQL中统计数据表的行数，可以使用三种方式：SELECT COUNT(*)、SELECT COUNT(1)和SELECT COUNT(具体字段)，使用这三者之间的查询效率是怎样的？之前看到说是：SELECT COUNT(*)＞ SELECT COUNT(1)＞ SELECT COUNT(具体字段)。

解答

在MySQL InnoDB存储引擎中，COUNT(*)和COUNT(1)都是对所有结果进行COUNT。如果有WHERE子句，则是对所有符合筛选条件的数据行进行统计；如果没有WHERE子句，则是对数据表的数据行数进行统计。

因此COUNT(*)和COUNT(1)本质上并没有区别，执行的复杂度都是O(N)，也就是采用全表扫描，进行循环+计数的方式进行统计。

如果是MySQL MyISAM存储引擎，统计数据表的行数只需要O(1)的复杂度，这是因为每张MyISAM的数据表都有一个meta信息存储了row_count值，而一致性则由表级锁来保证。因为InnoDB支持事务，采用行级锁和MVCC机制，所以无法像MyISAM一样，只维护一个row_count变量，因此需要采用扫描全表，进行循环+计数的方式来完成统计。

需要注意的是，在实际执行中，COUNT(*)和COUNT(1)的执行时间可能略有差别，不过你还是可以把它俩的执行效率看成是相等的。

另外在InnoDB引擎中，如果采用COUNT(*)和COUNT(1)来统计数据行数，要尽量采用二级索引。因为主键采用的索引是聚簇索引，聚簇索引包含的信息多，明显会大于二级索引（非聚簇索引）。对于COUNT(*)和COUNT(1)来说，它们不需要查找具体的行，只是统计行数，系统会自动采用占用空间更小的二级索引来进行统计。

然而如果想要查找具体的行，那么采用主键索引的效率更高。如果有多个二级索引，会使用key_len小的二级索引进行扫描。当没有二级索引的时候，才会采用主键索引来进行统计。

这里我总结一下：

1. 一般情况下，三者执行的效率为 COUNT(*)= COUNT(1)> COUNT(字段)。我们尽量使用COUNT(*)，当然如果你要统计的是某个字段的非空数据行数，则另当别论，毕竟比较执行效率的前提是结果一样才可以。
2. 如果要统计COUNT(*)，尽量在数据表上建立二级索引，系统会自动采用key_len小的二级索引进行扫描，这样当我们使用SELECT COUNT(*)的时候效率就会提升，有时候可以提升几倍甚至更高。

答疑2

在MySQL中，LIMIT关键词是最后执行的，如果可以确定只有一条结果，那么就起不到查询优化的效果了吧，因为LIMIT是对最后的结果集过滤，如果结果集本来就只有一条，那就没有什么用了。

解答

如果你可以确定结果集只有一条，那么加上LIMIT 1的时候，当找到一条结果的时候就不会继续扫描了，这样会加快查询速度。这里指的查询优化针对的是会扫描全表的SQL语句，如果数据表已经对字段建立了唯一索引，那么可以通过索引进行查询，不会全表扫描的话，就不需要加上LIMIT 1了。

关于通配符的解释

关于查询语句中通配符的使用理解，我举了一个查询英雄名除了第一个字以外，包含“太”字的英雄都有谁的例子，使用的SQL语句是：

```
SQL> SELECT name FROM heros WHERE name LIKE '_%太%'
```

（_）匹配任意一个字符，（%）匹配大于等于0个任意字符。

所以通配符'_%太%'说明在第一个字符之后需要有“太”字，这里就不能匹配上“太乙真人”，但是可以匹配上“东皇太一”。如果数据表中有“太乙真人太太”，那么结果集中也可以匹配到。

另外，单独的LIKE '%'无法查出NULL值，比如：SELECT * FROM heros WHERE role_assist LIKE '%'。

答疑4

可以理解在WHERE条件字段上加索引，但是为什么在ORDER BY字段上还要加索引呢？这个时候已经通过WHERE条件过滤得到了数据，已经不需要再筛选过滤数据了，只需要根据字段排

序就好了。

解答

在MySQL中，支持两种排序方式，分别是FileSort和Index排序。在Index排序中，索引可以保证数据的有序性，不需要再进行排序，效率更高。而FileSort排序则一般在内存中进行排序，占用CPU较多。如果待排结果较大，会产生临时文件I/O到磁盘进行排序的情况，效率较低。

所以使用ORDER BY子句时，应该尽量使用Index排序，避免使用FileSort排序。当然你可以使用explain来查看执行计划，看下优化器是否采用索引进行排序。

优化建议：

1. SQL中，可以在WHERE子句和ORDER BY子句中使用索引，目的是在WHERE子句中避免全表扫描，在ORDER BY子句避免使用FileSort排序。当然，某些情况下全表扫描，或者FileSort排序不一定比索引慢。但总的来说，我们还是要避免，以提高查询效率。一般情况下，优化器会帮我们进行更好的选择，当然我们也需要建立合理的索引。
2. 尽量使用Index完成ORDER BY排序。如果WHERE和ORDER BY后面是相同的列就使用单索引列；如果不同就使用联合索引。
3. 无法使用Index时，需要对FileSort方式进行调优。

答疑5

ORDER BY是对分的组排序还是对分组中的记录排序呢？

解答

ORDER BY就是对记录进行排序。如果你在ORDER BY前面用到了GROUP BY，实际上这是一种分组的聚合方式，已经把一组的数据聚合成为了一条记录，再进行排序的时候，相当于对分的组进行了排序。

答疑6

请问下关于SELECT语句内部的执行步骤。

解答

一条完整的SELECT语句内部的执行顺序是这样的：

1. FROM子句组装数据（包括通过ON进行连接）；
2. WHERE子句进行条件筛选；
3. GROUP BY分组；
4. 使用聚集函数进行计算；
5. HAVING筛选分组；
6. 计算所有的表达式；

7. SELECT 的字段；
8. ORDER BY排序；
9. LIMIT筛选。

答疑7

不太理解哪种情况下应该使用EXISTS，哪种情况应该用IN。选择的标准是看能否使用表的索引吗？

解答

索引是个前提，其实选择与否还是要看表的大小。你可以将选择的标准理解为小表驱动大表。在这种方式下效率是最高的。

比如下面这样：

```
SELECT * FROM A WHERE cc IN (SELECT cc FROM B)
SELECT * FROM A WHERE EXISTS (SELECT cc FROM B WHERE B.cc=A.cc)
```

当A小于B时，用EXISTS。因为EXISTS的实现，相当于外表循环，实现的逻辑类似于：

```
for i in A
    for j in B
        if j.cc == i.cc then ...
```

当B小于A时用IN，因为实现的逻辑类似于：

```
for i in B
    for j in A
        if j.cc == i.cc then ...
```

哪个表小就用哪个表来驱动，A表小就用EXISTS，B表小就用IN。

关于存储过程

答疑1

在使用存储过程声明变量时，都支持哪些数据类型呢？

解答

不同的DBMS对数据类型的定义不同，你需要查询相关的DBMS文档。以MySQL为例，常见的

数据类型可以分成三类，分别是数值类型、字符串类型和日期 / 时间类型。

答疑2

“IN参数必须在调用存储过程时指定”的含义是什么？我查询了MySQL的存储过程定义，可以不包含 IN 参数。当存储过程的定义语句里有 IN 参数时，存储过程的语句中必须用到这个参数吗?

解答

如果存储过程定义了IN参数，就需要在调用的时候传入。当然在定义存储过程的时候，如果不指定参数类型，就默认是IN类型的参数。因为IN参数在存储过程中是默认值，可以省略不写。比如下面两种定义方式都是一样的：

```
CREATE PROCEDURE `add_num`(IN n INT)
```

```
CREATE PROCEDURE `add_num`(n INT)
```

在存储过程中的语句里，不一定要用到IN参数，只是在调用的时候需要传入这个。另外IN参数在存储过程中进行了修改，也不会进行返回的。如果想要返回参数，需要使用OUT，或者INOUT参数类型。

关于事务处理

答疑1

如果INSERT INTO test SELECT '关羽';之后没有执行COMMIT，结果应该是空。但是我执行出来的结果是'关羽'，为什么ROLLBACK没有全部回滚？

代码如下：

```
CREATE TABLE test(name varchar(255), PRIMARY KEY (name)) ENGINE=InnoDB;
BEGIN;
INSERT INTO test SELECT '关羽';
BEGIN;
INSERT INTO test SELECT '张飞';
INSERT INTO test SELECT '张飞';
ROLLBACK;
SELECT * FROM test;
```

解答

先解释下连续BEGIN的情况。

在MySQL中BEGIN用于开启事务，如果是连续BEGIN，当开启了第一个事务，还没有进行COMMIT提交时，会直接进行第二个事务的BEGIN，这时数据库会隐式地COMMIT第一个事务，然后再进入到第二个事务。

为什么ROLLBACK没有全部回滚呢？

因为ROLLBACK是针对当前事务的，在BEGIN之后已经开启了第二个事务，当遇到ROLLBACK的时候，第二个事务都进行了回滚，也就得到了第一个事务执行之后的结果即“关羽”。

关于事务的ACID，以及我们使用COMMIT和ROLLBACK来控制事务的时候，有一个容易出错的地方。

在一个事务的执行过程中可能会失败。遇到失败的时候是进行回滚，还是将事务执行过程中已经成功操作的来进行提交，这个逻辑是需要开发者自己来控制的。

这里开发者可以决定，如果遇到了小错误是直接忽略，提交事务，还是遇到任何错误都进行回滚。如果我们强行进行COMMIT，数据库会将这个事务中成功的操作进行提交，它会认为你觉得已经是ACID了（就是你认为可以做COMMIT了，即使遇到了一些小问题也是可以忽略的）。

我在今天的文章里重点解答了一些问题，还有一些未解答的会留在评论里进行回复。最后出一道思考题吧。

请你自己写出下面操作的运行结果（你可以把它作为一道笔试题，自己写出结果，再与实际的运行结果进行比对）：

```
DROP TABLE IF EXISTS test;
CREATE TABLE test(name varchar(255), PRIMARY KEY (name)) ENGINE=InnoDB;
BEGIN;
INSERT INTO test SELECT '关羽';
BEGIN;
INSERT INTO test SELECT '张飞';
INSERT INTO test SELECT '张飞';
COMMIT;
SELECT * FROM test;
```

欢迎你在评论区写下你的思考，我会与你一起交流，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

精选留言

Destroy、 3

结果是：

```
+------+
| name |
+------+
| 关羽 |
| 张飞 |
+------+
```

因为插入关羽这个是第一个事务，虽然没有commit，但是第二个begin数据库会隐式地 COMMIT 第一个事务，第二事务，插入张飞两次，第一次插入成功，第二次插入失败。强制commit，第一次插入的张飞会进行提交。所以结果是关羽和张飞。

2019-07-24

作者回复

对的 这个解释也正确

2019-07-26

ack 2

思考题：

自己想出来是只有关羽一条，因为name是主键，插入两条导致第二个事务回滚。但实际运行后结果是关羽、张飞。不知道是为什么，望老师解答。（mysql的autocommit=1，隔离级别是可

重复读）

2019-07-24

许童童 1

老师你好，能否说一下varchar和nvarchar有什么区别，分别用在什么场景？

2019-07-24

作者回复

相同点：可变长度，字符类型数据

不同点：varchar(n)是n个字节，非Unicode字符。（英文字母占1个字节，中文占2个字节）

而nvarchar(n)是n个字符，Unicode字符。（英文字母或者中文都是占用2个字节）

举个例子，varchar(10)代表10个字节，所以可以是10个英文字母，也可以是5个汉字。

而nvarchar(10)代表10个字符，这10个字符可以是10个字母，也可以是10个汉字（英文字母或者中文 都是占用2个字节）

2019-07-26

Cue 1

很详细的答疑，赞

2019-07-24

hxd 0

MySQL 5.7 测试是只有一条关羽的数据插入成功，第二个begin隐式提交了第一条，然而第二个事务中途失败回滚了，相当于张飞的数据没有插入成功。

2019-07-29

Geek_c76f38 0

请教一个sql如何在一个表里间隔固定刚修改数据，如每间隔3行将一个字段设置成0?

2019-07-25

另至 0

答案是：“关羽”

根据第十四篇-事务中原子性的描述：要不全部成功要不全部失败。

第一个事务成功插入“关羽”

第二个事务，第一条插入“张飞”成功，第二条插入“张飞”失败。

所以第二个事务整体回滚，一条“张飞”都没插入。

所以结果只有“关羽”

2019-07-25

作者回复

对事务ACID的理解是这样的，不过在程序中是需要自己来控制的，如果遇到了错误，还继续执行COMMIT的话，也会让事务中正确的部分进行提交。所以你可以跑一遍代码，运行结果应该是 关羽，张飞。

2019-07-26

另至 0

根据第十四篇-事务，原子性：要不全部成功，要不全部失败。

2019-07-25

作者回复

原子是这个特点，不过在代码操作的时候，如果在一个事务中遇到的错误，还是可以强制进行COMMIT的，这时会把这个事务中成功执行的部分进行提交。你可以运行下 文章中给到的代码

2019-07-26

一叶知秋 0

老师 我也想问个问题。。。

之前提到过SQL执行的顺序是：FROM > WHERE > GROUP BY > HAVING > SELECT 的字段 > DISTINCT > ORDER BY > LIMIT

1）既然limit是最后执行的那么为何limit可以避免全表扫描。

2) 假如select的字段不包含order by字段那么是否在distinct产生的虚拟表上还要添加列？

2019-07-25

苏极 0

对于COUNT 这个有些困惑

1、老师你说，聚簇索引比二级索引信息多？是指的什么信息多，它是怎么导致使用主键索引慢了呢

2019-07-25

庞鑫华 0

老师，请问join查询，on后面的条件、连接条件，where后面的条件，数据过滤顺序是怎样的呢？

2019-07-25

庞鑫华 0

老师，能否举例说明一条复杂sql的查询解析过程，包括join操作的

2019-07-25

悟空 0

太乙真人太太，真开心，哈哈哈。

2019-07-25

作者回复

哈哈 这时专栏里的一个同学举的例子，我觉得不错，挺好的说明了通配符的使用

2019-07-26

Ronnyz 0

作业：关羽 张飞

2019-07-24

作者回复

对的 答案正确，是关羽，张飞

2019-07-26

NO.9 0

在sql语句里 怎么指定使用哪个索引呢？

是像oracle里那样用hint么？

2019-07-24

作者回复

对的，可以使用hint，不同DBMS方法略有不同：

Oracle：/*+ index(索引名称) */

SQL Serve：WITH (INDEX(索引名称))

MySQL：FORCE INDEX(索引名称)

比如我们查询player表的时候想要强制使用player_name进行索引，可以写成：

Oracle: SELECT /*+ INDEX(player_name) */ FROM player

SQL Server: SELECT player_id, team_id, player_name FROM player WITH(INDEX(player_name))

MySQL：SELECT player_id, team_id, player_name FROM player FORCE INDEX(player_name)

2019-07-26

mickey 0

答案是

name

关羽

张飞

2019-07-24

作者回复

正确

2019-07-26

liuyyy 0

老师我想问下索引，你文中多次提到了这个，还包括二级索引，添加索引等，我理解的索引就只有像python中的index等，为什么还有多个索引这个说法呢

2019-07-24

ttttt 0

思考题：

事务隔离级别为可重复度。

当前连接（操作人小王）：会产生幻读，读到两条信息。

别的连接（操作人小李）：不会产生幻读，会只读到一条信息。

2019-07-24

一叶知秋

0

关羽、张飞。

1）MySQL默认开始自动提交事务。那么第一条insert会自动提交。

2）set autocommit=0; 结果一致, 因为没有写rollback 只有commit 因此仍会执行可执行的SQL 不会导致整个事务的回滚。

2019-07-24

pain

0

老师，我问下，union 左右的两个子句是按照顺序执行的，还是并发执行的啊

2019-07-24