06丨数据库原理：为什么PrepareStatement性能更好更安全？

2019-11-29 李智慧

后端技术面试38讲 进入课程 >

讲述：李智慧
时长 11:18　大小 10.37M

做应用开发的同学常常觉得数据库由 DBA 运维，自己会写 SQL 就可以了，数据库原理不需要学习。其实即使是写 SQL 也需要了解数据库原理，比如我们都知道，SQL 的查询条件尽量包含索引字段，但是为什么呢？这样做有什么好处呢？你也许会说，使用索引进行查询速度快，但是为什么速度快呢？

此外，我们在 Java 程序中访问数据库的时候，有两种提交 SQL 语句的方式，一种是通过 Statement 直接提交 SQL；另一种是先通过 PrepareStatement 预编译 SQL，然后设置可变参数再提交执行。

Statement 直接提交的方式如下：

```
statement.executeUpdate("UPDATE Users SET stateus = 2 WHERE userID=233");
```

PrepareStatement 预编译的方式如下：

```
PreparedStatement updateUser = con.prepareStatement("UPDATE Users SET stateus
updateUser.setInt(1, 2);
updateUser.setInt(2,233);
updateUser.executeUpdate();
```

看代码，似乎第一种方式更加简单，但是编程实践中，主要用第二种。使用 MyBatis 等 ORM 框架时，这些框架内部也是用第二种方式提交 SQL。那为什么要舍简单而求复杂呢?

要回答上面这些问题，都需要了解数据库的原理，包括数据库的架构原理与数据库文件的存储原理。

数据库架构与 SQL 执行过程

我们先看看数据库架构原理与 SQL 执行过程。

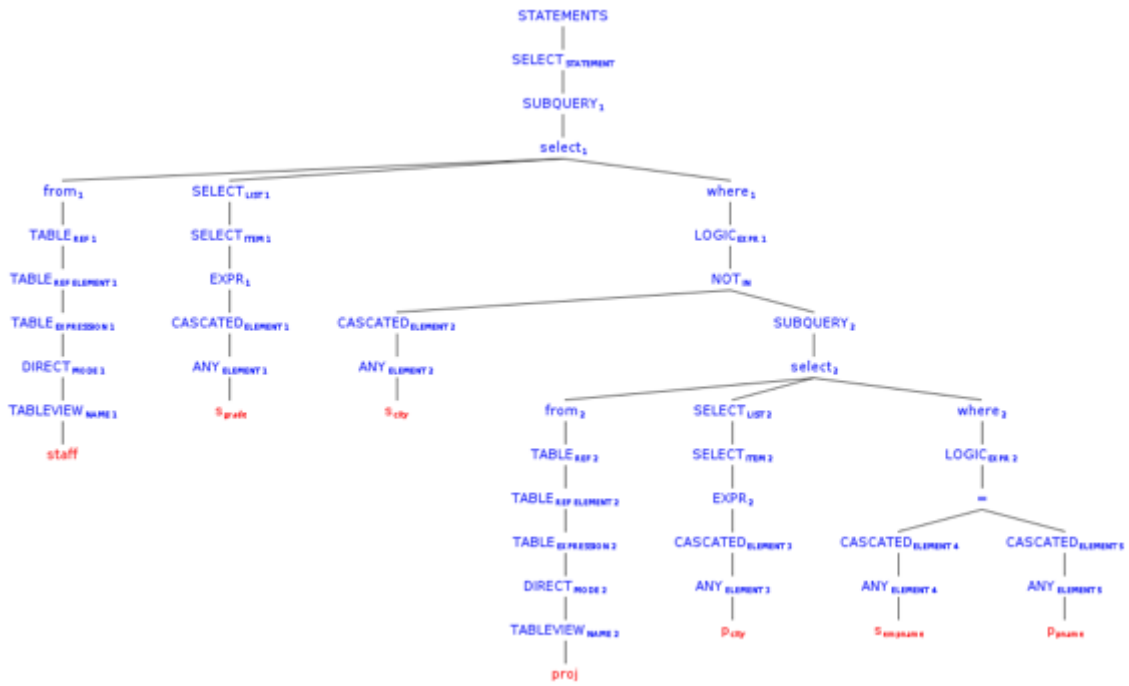
关系数据库系统 RDBMS 有很多种，但是这些关系数据库的架构基本上差不多，包括支持 SQL 语法的 Hadoop 大数据仓库，也基本上都是相似的架构。一个 SQL 提交到数据库，经过连接器将 SQL 语句交给语法分析器，生成一个抽象语法树 AST；AST 经过语义分析与优化器，进行语义优化，使计算过程和需要获取的中间数据尽可能少，然后得到数据库执行计划；执行计划提交给具体的执行引擎进行计算，将结果通过连接器再返回给应用程序。

应用程序提交 SQL 到数据库执行，首先需要建立与数据库的连接，数据库**连接器**会为每个连接请求分配一块专用的内存空间用于会话上下文管理。建立连接对数据库而言相对比较重，需要花费一定的时间，因此应用程序启动的时候，通常会初始化建立一些数据库连接放在连接池里，这样当处理外部请求执行 SQL 操作的时候，就不需要花费时间建立连接了。

这些连接一旦建立，不管是否有 SQL 执行，都会消耗一定的数据库内存资源，所以对于一个大规模互联网应用集群来说，如果启动了很多应用程序实例，这些程序每个都会和数据库建立若干个连接，即使不提交 SQL 到数据库执行，也就会对数据库产生很大的压力。

所以应用程序需要对数据库连接进行管理，一方面通过连接池对连接进行管理，空闲连接会被及时释放；另一方面微服务架构可以大大减少数据库连接，比如对于用户数据库来说，所有应用都需要连接到用户数据库，而如果划分一个用户微服务并独立部署一个比较小的集群，那么就只有这几个用户微服务实例需要连接用户数据库，需要建立的连接数量大大减少。

连接器收到 SQL 以后，会将 SQL 交给**语法分析器**进行处理，语法分析器工作比较简单机械，就是根据 SQL 语法规则生成对应的抽象语法树，下图是 Oracle 语法分析器生成的抽象语法树。



如果 SQL 语句中存在语法错误，那么在生成语法树的时候就会报错，比如，下面这个例子中 SQL 语句里的 where 拼写错误，MySQL 就会报错。

复制代码

```
mysql> explain select * from users whee id = 1;

ERROR 1064 (42000): You have an error in your SQL syntax; check the manual tha
```

因为语法错误是在构建抽象语法树的时候发现的，所以能够知道，错误是发生在哪里。上面例子中，虽然语法分析器不能知道 whee 是一个语法拼写错误，因为这个 whee 可能是表名 users 的别名，但是语法分析器在构建语法树到了`id=1`这里的时候就出错了，所以返回的报错信息可以提示，在`'id = 1'`附近有语法错误。

语法分析器生成的抽象语法树并不仅仅可以用来做语法校验，它也是下一步处理的基础。语义分析与优化器会对抽象语法树进一步做语义优化，也就是在保证 SQL 语义不变的前提下，进行语义等价转换，使最后的计算量和中间过程数据量尽可能小。

比如对于这样一个 SQL 语句，其语义是表示从 users 表中取出每一个 id 和 order 表当前记录比较，是否相等。

复制代码

```
select f.id from orders f where f.user_id = (select id from users);
```

事实上，这个 SQL 语句在语义上等价于下面这条 SQL 语句，表间计算关系更加清晰。

复制代码

```
select f.id from orders f join users u on f.user_id = u.id;
```

SQL 语义分析与优化器就是要将各种复杂嵌套的 SQL 进行语义等价转化，得到有限几种关系代数计算结构，并利用索引等信息进一步进行优化。可以说，各个数据库最黑科技的部分就是在优化这里了。

语义分析与优化器最后会输出一个执行计划，由执行引擎完成数据查询或者更新。MySQL 执行计划的例子如下：

```
mysql> explain select * from users where id = 1;
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
| id | select_type | table | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
|  1 | SIMPLE      | users | NULL       | const | PRIMARY       | PRIMARY | 4       | const |    1 |   100.00 | NULL  |
+----+-------------+-------+------------+-------+---------------+---------+---------+-------+------+----------+-------+
```

执行引擎是可替换的，只要能够执行这个执行计划就可以了。所以 MySQL 有多种执行引擎（也叫存储引擎）可以选择，缺省的是 InnoDB，此外还有 MyISAM、Memory 等，我

们可以在创建表的时候指定存储引擎。大数据仓库 Hive 也是这样的架构，Hive 输出的执行计划可以在 Hadoop 上执行。

使用 PrepareStatement 执行 SQL 的好处

好了，了解了数据库架构与 SQL 执行过程之后，让我们回到开头的问题，应用程序为什么应该使用 PrepareStatement 执行 SQL?

这样做主要有两个好处。

一个是 PrepareStatement 会预先提交带占位符的 SQL 到数据库进行预处理，提前生成执行计划，当给定占位符参数，真正执行 SQL 的时候，执行引擎可以直接执行，效率更好一点。

另一个好处则更为重要，PrepareStatement 可以防止 SQL 注入攻击。假设我们允许用户通过 App 输入一个名字到数据中心查找用户信息，如果用户输入的字符串是 Frank，那么生成的 SQL 是这样的：

```
1 select * from users where username = 'Frank';
```

复制代码

但是如果用户输入的是这样一个字符串：

```
1 Frank';drop table users;--
```

复制代码

那么生成的 SQL 就是这样的：

```
1 select * from users where username = 'Frank';drop table users;--';
```

复制代码

这条 SQL 提交到数据库以后，会被当做两条 SQL 执行，一条是正常的 select 查询 SQL，一条是删除 users 表的 SQL。黑客提交一个请求然后 users 表被删除了，系统崩溃了，这

就是 SQL 注入攻击。

如果用 Statement 提交 SQL 就会出现这种情况。

但如果用 PrepareStatement 则可以避免 SQL 被注入攻击。因为一开始构造 PrepareStatement 的时候就已经提交了查询 SQL，并被数据库预先生成好了执行计划，后面黑客不管提交什么样的字符串，都只能交给这个执行计划去执行，不可能再生成一个新的 SQL 了，也就不会被攻击了。

```
1 select * from users where username = ?;
```

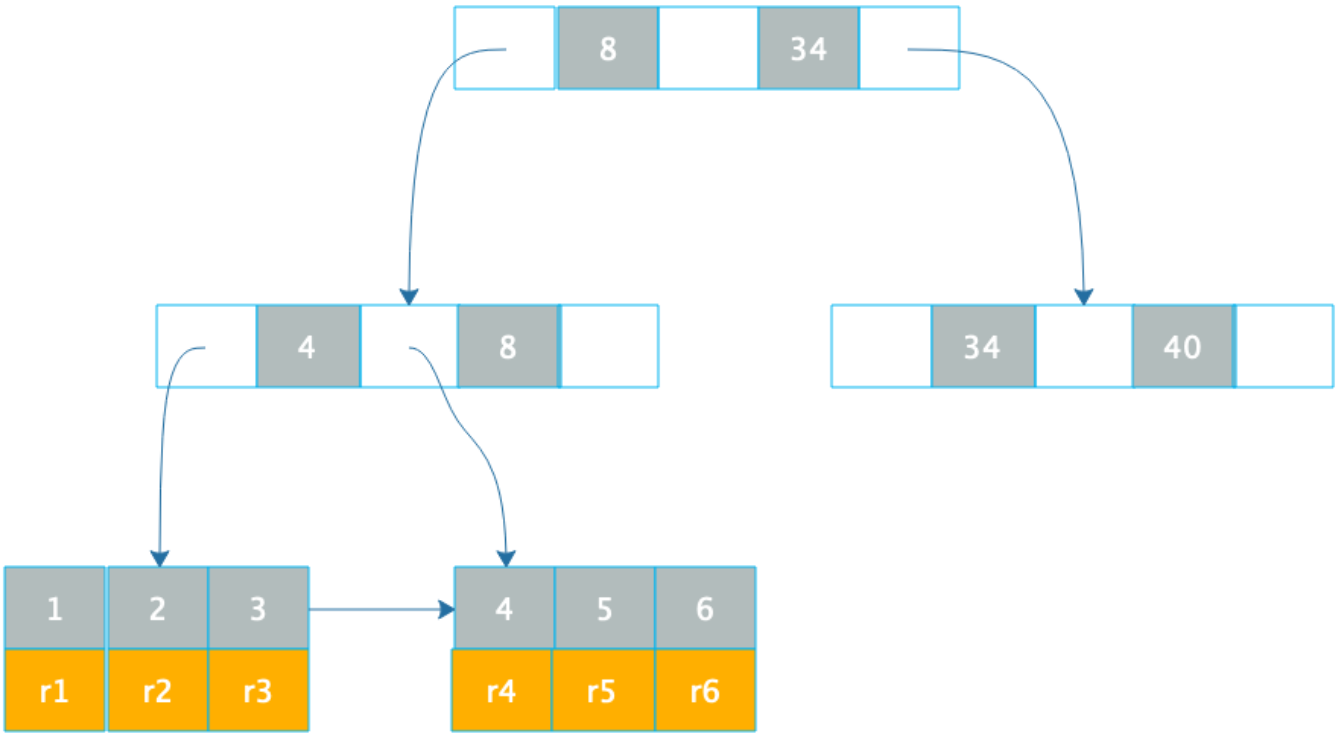
复制代码

数据库文件存储原理

回到文章开头提出的另一个问题，数据库通过索引进行查询能加快查询速度，那么，为什么索引能加快查询速度呢?

数据库索引使用 B+ 树，我们先看下 B+ 树这种数据结构。B+ 树是一种 N 叉排序树，树的每个节点包含 N 个数据，这些数据按顺序排好，两个数据之间是一个指向子节点的指针，而子节点的数据则在这两个数据大小之间。

如下图。

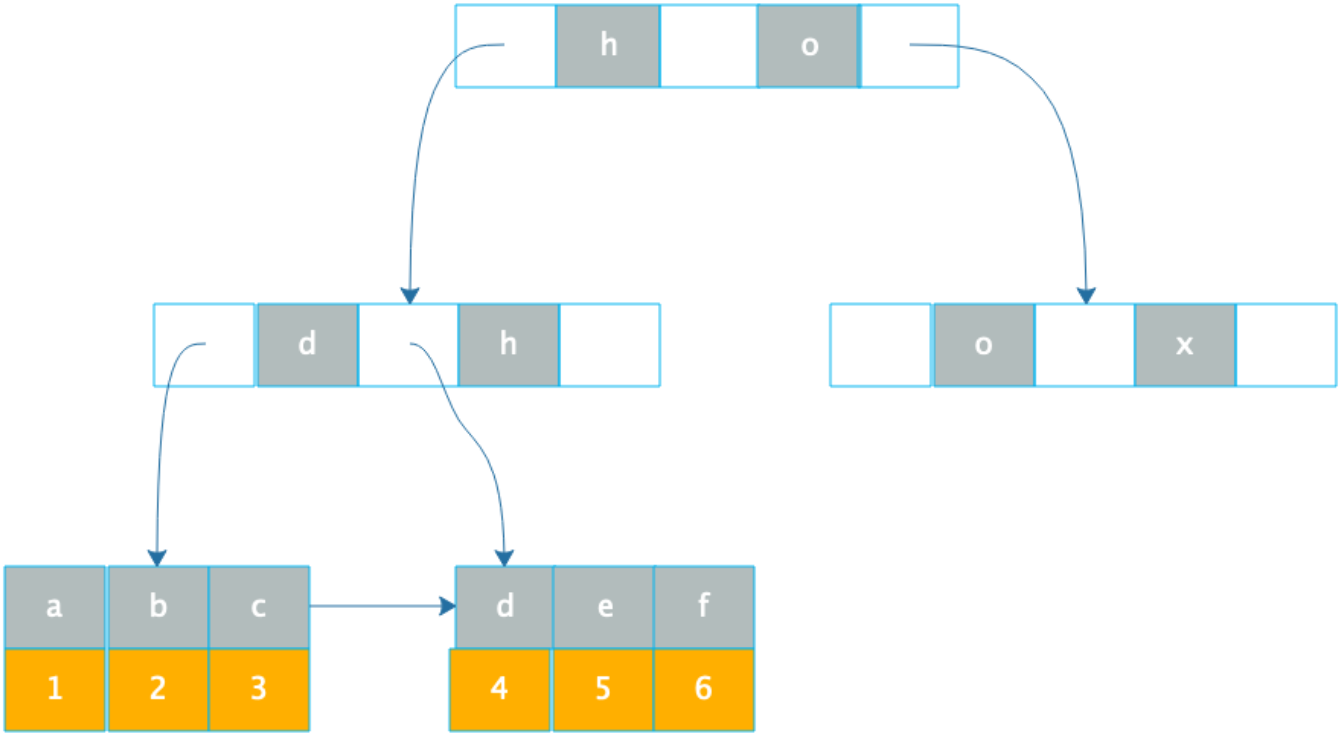


B+ 树的节点存储在磁盘上，每个节点存储 1000 多个数据，这样树的深度最多只要 4 层，就可存储数亿的数据。如果将树的根节点缓存在内存中，则最多只需要三次磁盘访问就可以检索到需要的索引数据。

B+ 树只是加快了索引的检索速度，如何通过索引加快数据库记录的查询速度呢？

数据库索引有两种，一种是聚簇索引，聚簇索引的数据库记录和索引存储在一起，上面这张图就是聚簇索引的示意图，在叶子节点，索引 1 和记录行 r1 存储在一起，查找到索引就是查找到数据库记录。像 MySQL 数据库的主键就是聚簇索引，主键 ID 和所在的记录行存储在一起。MySQL 的数据库文件实际上是以主键作为中间节点，行记录作为叶子节点的一颗 B+ 树。

另一种数据库索引是非聚簇索引，非聚簇索引在叶子节点记录的就不是数据行记录，而是聚簇索引，也就是主键，如下图。



通过 B+ 树在叶子节点找到非聚簇索引 a，和索引 a 在一起存储的是主键 1，再根据主键 1 通过主键（聚簇）索引就可以找到对应的记录 r1，这种通过非聚簇索引找到主键索引，再通过主键索引找到行记录的过程也被称作回表。

所以通过索引，可以快速查询到需要的记录，而如果要查询的字段上没有建索引，就只能扫描整张表了，查询速度就会慢很多。

数据库除了索引的 B+ 树文件，还有一些比较重要的文件，比如事务日志文件。

数据库可以支持事务，一个事务对多条记录进行更新，要么全部更新，要么全部不更新，不能部分更新，否则像转账这样的操作就会出现严重的数据不一致，可能会造成巨大的经济损失。数据库实现事务主要就是依靠事务日志文件。

在进行事务操作时，事务日志文件会记录更新前的数据记录，然后再更新数据库中的记录，如果全部记录都更新成功，那么事务正常结束，如果过程中某条记录更新失败，那么整个事务全部回滚，已经更新的记录根据事务日志中记录的数据进行恢复，这样全部数据都恢复到事务提交前的状态，仍然保持数据一致性。

此外，像 MySQL 数据库还有 binlog 日志文件，记录全部的数据更新操作记录，这样只要有了 binlog 就可以完整复现数据库的历史变更，还可以实现数据库的主从复制，构建高性能、高可用的数据库系统，我将会在架构模块进一步为你讲述。

小结

做应用开发需要了解 RDBMS 的架构原理，但是关系数据库系统非常庞大复杂，对于一般的应用开发者而言，全面掌握关系数据库的各种实现细节，代价高昂，也没有必要。我们只需要掌握数据库的架构原理与执行过程，数据库文件的存储原理与索引的实现方式，以及数据库事务与数据库复制的基本原理就可以了。然后，在开发工作中针对各种数据库问题去思考，其背后的原理是什么，应该如何处理。通过这样不断地思考学习，不但能够让使用数据库方面的能力不断提高，也能对数据库软件的设计理念也会有更深刻的认识，自己软件设计与架构的能力也会得到加强。

思考题

索引可以提高数据库的查询性能，那么是不是应该尽量多的使用索引呢？如果不是，为什么？你还了解哪些改善数据库访问性能的技巧方法？

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

上一篇　05 | 文件系统原理：如何用1分钟遍历一个100TB的文件？

精选留言 (12)

写留言

2019-11-29

您好，老师：
回答上述问题
1.创建多的索引，会占用更多磁盘空间。如果有一张很大的表，索引文件的大小可能达到操作系统允许的最大文件限制；
2.对于DML操作的时候，索引会降低他们的速度。因为MySQL不仅要把搞定的数据写入...

展开

3　15

Zend
2019-12-01

请问一下老师PreparedStatement经过语法分析生成的抽象语法树，再经过语义分析和优化器处理后生成的执行计划，会有缓存吗？

展开

作者回复: 会

1

Zend
2019-12-01

请问老师，在进行事务操作时，事务日志文件会记录更新前的数据记录，这个记录更新前的数据记录是 什么意思，是把更新之前的数据都查询出来，记录到事务日志文件嘛

作者回复: 是的，更新数据本来就会先执行查询操作。

1

尹宗昌
2019-11-29

索引可以提高查询性能，但是一定要考虑维护索引的成本。空间占用、插入修改的性能影响都要衡量

1

无形
2019-12-01

最近在广告检索中接入了用户画像标签，实现了把一大串嵌套json格式的标签数据表达式，类似dnf表达式，解析为可计算的广告匹配模型，其中就有类似sql一样，对表达式进行格式分析、数据、比较符检验、复杂的逻辑关系转换为容易处理的计算单元，最后生成一个树状匹配模型，通过对模型输入用户数据，进行匹配，并返回符合的用户。
这有点像用户画像标签就是一条SQL，对SQL进行语法分析，生成匹配模型（类比SQL的...
展开

无形
2019-12-01

数据变动的时候同时需要更新索引，索引多了，数据变动的效率就会降低，索引也是文件，会占用更多的磁盘空间
。
另外，我之前提到的要动手实现的高性能检索系统，在数据的存储部分，用文件存储，为了提高文档的检索效率，为文件创建了索引，索引是这种格式id:start:length，...
展开

一步
2019-11-30

SQL 的分析器 包括词法分析器和语法分析器，经过词法分析器生成 AST ，不是语法分析

乘坐Tornado的线程魔...
2019-11-30

印象中丁奇的《MySQL实战45讲》中，也讲到了回表。但是文中提到的MySQL是聚簇索引，不需要回表。是不是这里面有些概念需要根据具体情况再进一步拆分讨论。并不能一概而论?

7

乘坐Tornado的线程魔...
2019-11-30

请问下，文中的第一个B+树示意图的第一层中间节点，8和34中间的部分是不是也应该指向一个子节点?

作者回复: 是的，图中只是示意，没有画出全部节点。

老王的老李头

2019-11-30

李老师，堪比百科全书啊！非常接地气！给老师提个建议，文章中有些是针对MySQL说的，有些不是，希望老师能区分一下。
索引么，也是一种空间换时间的思路，细思极恐。改善数据库访问性能的手段也不应该仅限于索引这种手段。

展开

俊伟

2019-11-29

1.索引较多，插入删除的时候会有额外的开销。
2.索引字段尽量小，因为索引字段的大小会影响每一个b+树节点数据块中的数据项的个数。
3.多了解业务，有些地方可以使用索引有些则不用。
4.可以综合利用联合索引和单独索引

展开

幸福来敲门

2019-11-29

哎

展开