20丨当我们思考数据库调优的时候，都有哪些维度可以选择？

从这一篇开始，我们正式进入了 SQL 性能优化篇。在这一模块中，我们会关注如何提升 SQL 查询的效率。你可以思考一下，如何你是一名 DBA 或者开发人员，都有哪些维度可以进行数据库调优？

其实关于数据库调优的知识点非常分散。不同的 DBMS，不同的公司，不同的职位，不同的项目遇到的问题都不尽相同。为了能让你对数据库调优有一个整体的概览，我把这些知识点做了一个梳理，希望能对你有一些帮助。

今天的课程你需要掌握以下几个方面的内容：

1. 数据库调优的目标是什么？
2. 如果要进行调优，都有哪些维度可以选择？
3. 如何思考和分析数据库调优这件事？

数据库调优的目标

简单来说，数据库调优的目的就是要让数据库运行得更快，也就是说响应的时间更快，吞吐量更大。

不过随着用户量的不断增加，以及应用程序复杂度的提升，我们很难用“更快”去定义数据库调优的目标，因为用户在不同时间段访问服务器遇到的瓶颈不同，比如双十一促销的时候会带来大规模的并发访问；还有用户在进行不同业务操作的时候，数据库的事务处理和 SQL 查询都会有所不同。因此我们还需要更加精细的定位，去确定调优的目标。

如何确定呢？一般情况下，有两种方式可以得到反馈。

用户的反馈

用户是我们的服务对象，因此他们的反馈是最直接的。虽然他们不会直接提出技术建议，但是有些问题往往是用户第一时间发现的。我们要重视用户的反馈，找到和数据相关的问题。

日志分析

我们可以通过查看数据库日志和操作系统日志等方式找出异常情况，通过它们来定位遇到的问题。

除了这些具体的反馈以外，我们还可以通过监控运行状态来整体了解服务器和数据库的运行情况。

服务器资源使用监控

通过监控服务器的 CPU、内存、I/O 等使用情况，可以实时了解服务器的性能使用，与历史情况进行对比。

数据库内部状况监控

在数据库的监控中，活动会话（Active Session）监控是一个重要的指标。通过它，你可以清楚地了解数据库当前是否处于非常繁忙的状态，是否存在 SQL 堆积等。

除了活动会话监控以外，我们也可以对事务、锁等待等进行监控，这些都可以帮助我们对数据库的运行状态有更全面的认识。

对数据库进行调优，都有哪些维度可以进行选择？

我们需要调优的对象是整个数据库管理系统，它不仅包括 SQL 查询，还包括数据库的部署配置、架构等。从这个角度来说，我们思考的维度就不仅仅局限在 SQL 优化上了。

听起来比较复杂，但其实我们可以一步步通过下面的步骤进行梳理。

第一步，选择适合的 DBMS

我们之前讲到了 SQL 阵营和 NoSQL 阵营。在 RDBMS 中，常用的有 Oracle，SQL Server 和 MySQL 等。如果对事务性处理以及安全性要求高的话，可以选择商业的数据库产品。这些数据库在事务处理和查询性能上都比较强，比如采用 SQL Server，那么单表存储上亿条数据是没有问题的。如果数据表设计得好，即使不采用分库分表的方式，查询效率也不差。

除此以外，你也可以采用开源的 MySQL 进行存储，我们之前讲到过，它有很多存储引擎可以选择，如果进行事务处理的话可以选择 InnoDB，非事务处理可以选择 MyISAM。

NoSQL 阵营包括键值型数据库、文档型数据库、搜索引擎、列式存储和图形数据库。这些数据库的优缺点和使用场景各有不同，比如列式存储数据库可以大幅度降低系统的 I/O，适合于分布式文件系统和 OLAP，但如果数据需要频繁地增删改，那么列式存储就不太适用了。原因我在答疑篇已经讲过，这里不再赘述。

DBMS 的选择关系到了后面的整个设计过程，所以第一步就是要选择适合的 DBMS。如果已经确定好了 DBMS，那么这步可以跳过，但有时候我们要根据业务需求来进行选择。

第二步，优化表设计

选择了 DBMS 之后，我们就需要进行表设计了。RDBMS 中，每个对象都可以定义为一张表，表与表之间的关系代表了对象之间的关系。如果用的是 MySQL，我们还可以根据不同表的使用需求，选择不同的存储引擎。除此以外，还有一些优化的原则可以参考：

1. 表结构要尽量遵循第三范式的原则（关于第三范式，我在后面章节会讲）。这样可以让数据结构更加清晰规范，减少冗余字段，同时也减少了在更新，插入和删除数据时等异常情况的发生。
2. 如果分析查询应用比较多，尤其是需要进行多表联查的时候，可以采用反范式进行优化。反范式采用空间换时间的方式，通过增加冗余字段提高查询的效率。
3. 表字段的数据类型选择，关系到了查询效率的高低以及存储空间的大小。一般来说，如果字段可以采用数值类型就不要采用字符类型；字符长度要尽可能设计得短一些。针对字符类型来说，当确定字符长度固定时，就可以采用 CHAR 类型；当长度不固定时，通常采用 VARCHAR 类型。

数据表的结构设计很基础，也很关键。好的表结构可以在业务发展和用户量增加的情况下依然发挥作用，不好的表结构设计会让数据表变得非常臃肿，查询效率也会降低。

第三步，优化逻辑查询

当我们建立好数据表之后，就可以对数据表进行增删改查的操作了。这时我们首先需要考虑的是逻辑查询优化，什么是逻辑查询优化呢？

SQL 查询优化，可以分为逻辑查询优化和物理查询优化。逻辑查询优化就是通过改变 SQL 语句的内容让 SQL 执行效率更高效，采用的方式是对 SQL 语句进行等价变换，对查询进行重写。重写查询的数学基础就是关系代数。

SQL 的查询重写包括了子查询优化、等价谓词重写、视图重写、条件简化、连接消除和嵌套连接消除等。

比如我们在讲解 EXISTS 子查询和 IN 子查询的时候，会根据小表驱动大表的原则选择适合的子查询。在 WHERE 子句中会尽量避免对字段进行函数运算，它们会让字段的索引失效。

我举一个例子，假设我想对商品评论表中的评论内容进行检索，查询评论内容开头为 abc 的内容都有哪些，如果在 WHERE 子句中使用了函数，语句就会写成下面这样：

复制代码

```
SELECT comment_id, comment_text, comment_time FROM product_comment WHERE SUBSTRING(comme
```

我们可以采用查询重写的方式进行等价替换：

复制代码

```
SELECT comment_id, comment_text, comment_time FROM product_comment WHERE comment_text LI
```

你会发现在数据量大的情况下，第二条 SQL 语句的查询效率要比前面的高很多，执行时间为前者的 1/10。

第四步，优化物理查询

物理查询优化是将逻辑查询的内容变成可以被执行的物理操作符，从而为后续执行器的执行提供准备。它的核心是高效地建立索引，并通过这些索引来做各种优化。

但你要知道索引不是万能的，我们需要根据实际情况来创建索引。那么都有哪些情况需要考虑呢？

1. 如果数据重复度高，就不需要创建索引。通常在重复度超过 10% 的情况下，可以不创建这个字段的索引。比如性别这个字段（取值为男和女）。
2. 要注意索引列的位置对索引使用的影响。比如我们在 WHERE 子句中对索引字段进行了表达式的计算，会造成这个字段的索引失效。

3. 要注意联合索引对索引使用的影响。我们在创建联合索引的时候会对多个字段创建索引，这时索引的顺序就很重要了。比如我们对字段 x, y, z 创建了索引，那么顺序是 (x,y,z) 还是 (z,y,x)，在执行的时候就会存在差别。
4. 要注意多个索引对索引使用的影响。索引不是越多越好，因为每个索引都需要存储空间，索引多也就意味着需要更多的存储空间。此外，过多的索引也会导致优化器在进行评估的时候增加了筛选出索引的计算时间，影响评估的效率。

查询优化器在对 SQL 语句进行等价变换之后，还需要根据数据表的索引情况和数据情况确定访问路径，这就决定了执行 SQL 时所需要消耗的资源。SQL 查询时需要对不同的数据表进行查询，因此在物理查询优化阶段也需要确定这些查询所采用的路径，具体的情况包括：

1. 单表扫描：对于单表扫描来说，我们可以全表扫描所有的数据，也可以局部扫描。
2. 两张表的连接：常用的连接方式包括了嵌套循环连接、HASH 连接和合并连接。
3. 多张表的连接：多张数据表进行连接的时候，顺序很重要，因为不同的连接路径查询的效率不同，搜索空间也会不同。我们在进行多表连接的时候，搜索空间可能会达到很高的数据量级，巨大的搜索空间显然会占用更多的资源，因此我们需要通过调整连接顺序，将搜索空间调整在一个可接收的范围内。

物理查询优化是在确定了逻辑查询优化之后，采用物理优化技术（比如索引等），通过计算代价模型对各种可能的访问路径进行估算，从而找到执行方式中代价最小的作为执行计划。在这个部分中，我们需要掌握的重点是对索引的创建和使用。

第五步，使用 Redis 或 Memcached 作为缓存

除了可以对 SQL 本身进行优化以外，我们还可以请外援提升查询的效率。

因为数据都是存放到数据库中，我们需要从数据库层中取出数据放到内存中进行业务逻辑的操作，当用户量增大的时候，如果频繁地进行数据查询，会消耗数据库的很多资源。如果我们将常用的数据直接放到内存中，就会大幅提升查询的效率。

键值存储数据库可以帮我们解决这个问题。

常用的键值存储数据库有 Redis 和 Memcached，它们都可以将数据存放到内存中。

从可靠性来说，Redis 支持持久化，可以让我们的数据保存在硬盘上，不过这样一来性能消耗也会比较大。而 Memcached 仅仅是内存存储，不支持持久化。

从支持的数据类型来说，Redis 比 Memcached 要多，它不仅支持 key-value 类型的数据，还支持 List，Set，Hash 等数据结构。 当我们有持久化需求或者是更高级的数据处理需求的时候，就可以使用 Redis。如果是简单的 key-value 存储，则可以使用 Memcached。

通常我们对于查询响应要求高的场景（响应时间短，吞吐量大），可以考虑内存数据库，毕竟术业有专攻。传统的 RDBMS 都是将数据存储在硬盘上，而内存数据库则存放在内存中，查询起来要快得多。不过使用不同的工具，也增加了开发人员的使用成本。

第六步，库级优化

库级优化是站在数据库的维度上进行的优化策略，比如控制一个库中的数据表数量。另外我们可以采用主从架构优化我们的读写策略。

如果读和写的业务量都很大，并且它们都在同一个数据库服务器中进行操作，那么数据库的性能就会出现瓶颈，这时为了提升系统的性能，优化用户体验，我们可以采用读写分离的方式降低主数据库的负载，比如用主数据库（master）完成写操作，用从数据库（slave）完成读操作。

除此以外，我们还可以对数据库分库分表。当数据量级达到亿级以上时，有时候我们需要把一个数据库切成多份，放到不同的数据库服务器上，减少对单一数据库服务器的访问压力。如果你使用的是 MySQL，就可以使用 MySQL 自带的分区表功能，当然你也可以考虑自己做垂直切分和水平切分。

什么情况下做垂直切分，什么情况下做水平切分呢?

如果数据库中的数据表过多，可以采用垂直分库的方式，将关联的数据表部署在一个数据库上。

如果数据表中的列过多，可以采用垂直分表的方式，将数据表分拆成多张，把经常一起使用的列放到同一张表里。

如果数据表中的数据达到了亿级以上，可以考虑水平切分，将大的数据表分拆成不同的子表，每张表保持相同的表结构。比如你可以按照年份来划分，把不同年份的数据放到不同的数据表中。2017 年、2018 年和 2019 年的数据就可以分别放到三张数据表中。

采用垂直分表的形式，就是将一张数据表分拆成多张表，采用水平拆分的方式，就是将单张数据量大的表按照某个属性维度分成不同的小表。

但需要注意的是，分拆在提升数据库性能的同时，也会增加维护和使用成本。

我们该如何思考和分析数据库调优这件事

做任何事情之前，我们都需要确认目标。在数据库调优中，我们的目标就是响应时间更快，吞吐量更大。利用宏观的监控工具和微观的日志分析可以帮我们快速找到调优的思路和方式。

虽然每个人的情况都不一样，但我们同样需要对数据库调优这件事有一个整体的认知。在思考数据库调优的时候，可以从三个维度进行考虑。

首先，选择比努力更重要。

在进行 SQL 调优之前，可以先选择 DBMS 和数据表的设计方式。你能看到，不同的 DBMS 直接决定了后面的操作方式，数据表的设计方式也直接影响了后续的 SQL 查询语句。

另外，你可以把 SQL 查询优化分成两个部分，逻辑查询优化和物理查询优化。

虽然 SQL 查询优化的技术有很多，但是大方向上完全可以分成逻辑查询优化和物理查询优化两大块。逻辑查询优化就是通过 SQL 等价变换提升查询效率，直白一点就是说，换一种查询写法执行效率可能更高。物理查询优化则是通过索引和表连接方式等技术来进行优化，这里重点需要掌握索引的使用。

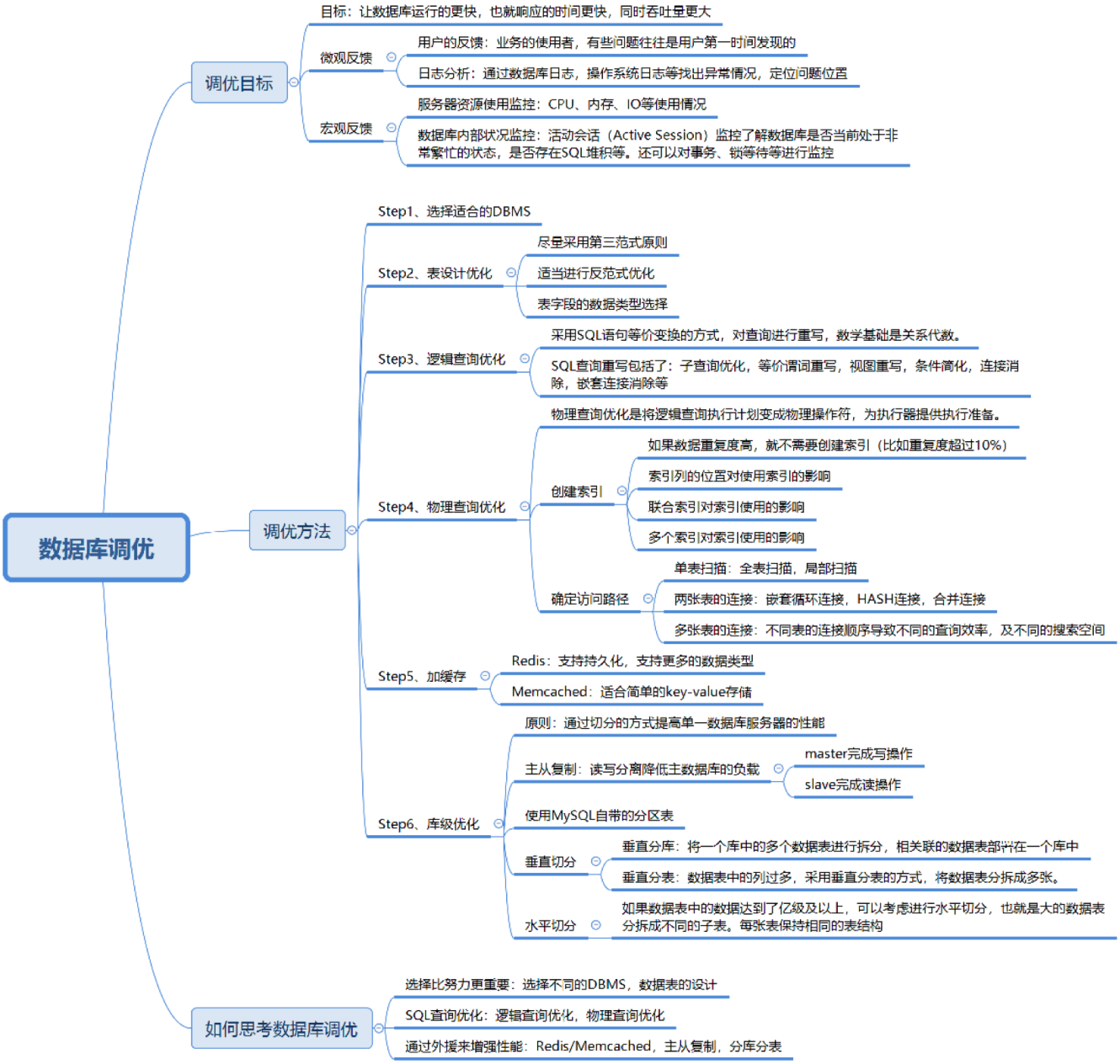
最后，我们可以通过外援来增强数据库的性能。

单一的数据库总会遇到各种限制，不如取长补短，利用外援的方式。

另外通过对数据库进行垂直或者水平切分，突破单一数据库或数据表的访问限制，提升查询的性能。

本篇文章中涉及到的概念和知识点比较多，也有可能出现纰漏，不过没有关系，我会在在后续的文章中陆续进行讲解。希望这篇文章可以让你站在一个宏观的角度对数据库的调优有系

统性的认知，对今后的工作有一些启发。



你不妨说一下，在日常的工作中你是如何发现数据库性能瓶颈的？又是怎么解决这个问题的？另外我在文章中从 6 个维度阐述了如何对数据库进行调优，前两个维度在于选择，中间两个维度在于 SQL 的查询优化，后两个维度在于外援技术。你可以说一说你对这些维度的理解吗？