TiDB 观测性解读（一） | 索引观测：快速识别无用索引与低效索引

原创 宋日杰 PingCAP 2025年03月11日 19:30 北京

导读

可观测性已经成为分布式系统成功运行的关键组成部分。如何借助多样、全面的数据，让架构师更简单、高效地定位问题、分析问题、解决问题，已经成为业内的一个技术焦点。本系列文章将深入解读 TiDB 的关键参数，帮助大家更好地观测系统的状态，实现性能的优化提升。

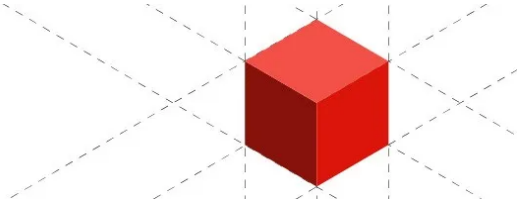
系列文章将包含以下几个章节：

- **索引观测**：快速识别无用索引与低效索引
- **SQL 执行观测**：解读 TiDB 算子的执行信息
- **数据热点观测**：分析发现造成热点的原因
- **内存观测**：分析理解 TiDB 实例的内存占用

更多内容，敬请期待。

本文为系列文章的第一篇，将探讨如何观测和管理 TiDB 中的索引，避免不必要的损失和消耗。

1 引言

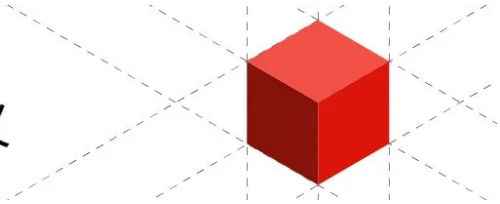


索引的设计对数据库性能优化起着至关重要的作用。索引减少了实际扫描的数据量，能够显著提升数据库系统的查询性能。然而，随着业务的复杂性不断提升，系统中的索引设计可能存在一些问题，影响数据库的整体效率：

> - **未使用的索引**：随着业务逻辑的调整、数据量变化或者新索引的创建，部分索引不再被查询优化器选中，变成了“无用索引”。
> - **低效索引**：虽然索引被查询优化器选中，但它们扫描了大量数据，导致 I/O 消耗较高，优化效果并不明显。

这些未使用或低效的索引如果没有得到及时清理，它们可能会显著影响数据库的性能和资源利用率。

2 索引优化的意义

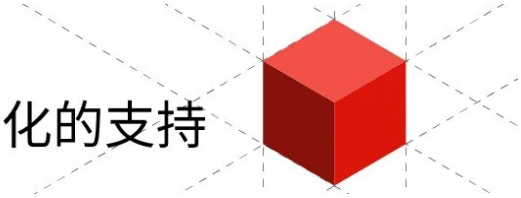


清理未使用或低效的索引对数据库性能优化工作有着重要的意义：

- **避免磁盘空间浪费**：每个索引都会消耗磁盘空间，随着数据量的增长，索引占用的空间也会随之扩大。清理未被使用的索引有助于节省存储成本。
- **降低 DML 操作的额外开销**：插入（INSERT）、更新（UPDATE）和删除（DELETE）操作需要维护所有相关的索引。未使用或低效的索引会导致这些操作的性能下降，特别是在高并发场景下，其影响尤为明显。移除这些索引可以提升 DML 操作的效率。
- **提升查询性能**：低效的索引可能导致查询时扫描的数据量过大，从而增加磁盘 I/O 和查询延迟。优化这些低效索引，能让查询操作更加高效。
- **简化数据库维护**：随着索引的增多，数据库管理的复杂性也随之提高，特别是在备份、恢复和迁移等操作中。清理不必要的索引，可以简化数据库结构，提高维护效率。

因此，定期检查并优化索引，应该成为数据库管理中的常规任务。

3 TiDB 对索引优化的支持



虽然优化索引对数据库性能至关重要，但错误地删除索引也可能带来风险，导致 SQL 性能下降，甚至导致数据库性能崩溃。因此，**删除索引的决策必须基于数据支持，通过有效手段来验证，并且在出现问题能够快速回退**。

TiDB 通过系统表 `TIDB_INDEX_USAGE`、`schema_unused_index`的引入和不可见索引（https://docs.pingcap.com/zh/tidb/stable/sql-statement-create-index#不可见索引）能力，帮助用户快速观测现有索引的状态，并实现删除索引前的验证。

系统表 TIDB_INDEX_USAGE

为了解决索引观测问题，TiDB 从 v8.0.0 版本开始引入了系统表 `INFORMATION_SCHEMA.TIDB_INDEX_USAGE`。该系统表记录了索引的关键运行指标，协助 DBA 制定优化策略。`TIDB_INDEX_USAGE` 主动观测每个 TiDB 实例中二级索引

（Secondary Index）的运行情况。自 8.4.0 版本起，聚簇表（Clustered Table）的主键也会被观测，用户可依赖这个信息优化主键设计。

要**识别无用索引**，最直观的方式是查看索引被查询优化器选择的次数。**如果某个索引查询次数为零，说明它在当前 TiDB 实例中未被使用**。通过以下几列数据，TIDB_INDEX_USAGE 提供了决策支持：

- **QUERY_TOTAL**：记录访问某个索引的查询总次数。如果某个索引的查询次数为零，则该索引未被使用。
- **LAST_ACCESS_TIME**：记录该索引的最后访问时间。如果索引长时间未被访问，可以确定该索引在此期间无用。

识别低效索引相对复杂，主要通过观察索引的选择性（selectivity）来判断。**一个优秀的索引应该能够过滤掉更多的数据，因此索引命中数据的比例越低越好**。TIDB_INDEX_USAGE 加入了以下字段来记录选择率分布情况：

- **PERCENTAGE_ACCESS_***：因为实际选择率是离散的，在视图里依据选择率范围设计了不同的“桶 (bucket)”，记录选择率落到该选择率范围的次数，以此判断索引的过滤效果。例如，PERCENTAGE_ACCESS_1_10 表示该索引选择率为 1% 到 10% 的次数，相应地，PERCENTAGE_ACCESS_0 代表没有任何行命中的次数，PERCENTAGE_ACCESS_100 表示 full index scan 的次数。每一列的数据除以 QUERY_TOTAL，能得出该选择率所占的百分比。
- **ROWS_ACCESS_TOTAL**：该索引扫描的总行数，用于衡量该索引对 I/O 的贡献。这个值除以 QUERY_TOTAL，能得出该索引的平均扫描行数。

当某个索引的选择率较高，即命中的平均行数占索引总行数的比例较高时，说明该索引的过滤效果不理想。如果能够改善这些索引，对数据库整体性能提升会有很大帮助。

作为一个分布式数据库，每个 TiDB 节点都会维护各自的索引使用情况。CLUSTER_TIDB_INDEX_USAGE **整合各个 TiDB 节点的数据，为 DBA 提供了全局的视角**。

为了降低数据观测工作对集群的性能的影响，**这个视图中的信息存在延迟，最长为 5 分钟**，用户在验证统计数据时需要注意这一点，索引的使用统计不会立即反馈到这个视图上。

另外，索引的使用统计具有一定的时效性，过旧的数据可能会对判断起反作用，所以 TiDB 并没有持久化这张系统表，**数据会随 TiDB 节点的重启而清空**。未来 TiDB 会通过 Workload Repository 对这个内存表数据做阶段性快照，方便用户观测每个时间段的精确统计。

系统表 schema_unused_indexes

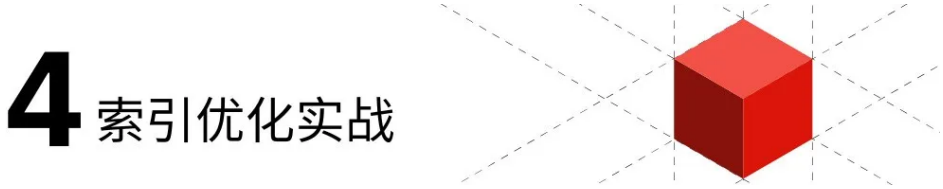
为了方便用户直接查看结果，TiDB 还提供了一个 MySQL 兼容的视图 sys.schema_unused_indexes，**该视图列出了自所有 TiDB 节点启动以来，未被使用过的索**

引。这张视图的数据来自 TIDB_INDEX_USAGE，请注意，由于 TIDB_INDEX_USAGE 在 TiDB 节点重启后会被清空，因此**在决策前需要确保节点的运行时间足够长**。

- 对于从旧版本升级到 TiDB v8.0.0 及更高版本的集群，sys schema 以及包含的视图需要手动创建，请参考官方文档（https://docs.pingcap.com/zh/tidb/stable/sys-schema-unused-indexes#手动创建-schema_unused_indexes-视图）进行操作。

不可见索引 (invisable indexes)

清理索引存在一定的风险，一旦错误地删除索引，重建索引和统计信息收集可能要花费很长时间。为降低风险，推荐先将索引设置为“不可见”状态。**设置为不可见后，优化器将不再使用该索引，但索引的统计信息仍会被更新维护，可快速恢复为“可见”**。DBA 可以先将要删除的索引设置为不可见，观察一段时间后再决定是否真正物理删除。



根据观测到的索引所处的状态，我们可以灵活选择索删除或优化相关索引。在这个部分我们将详细介绍如何安全、有效地识别优化 TiDB 问题索引。

删除未使用的索引

下面示例展示了 DBA 如何有效识别并管理未使用的索引，提升数据库性能和可维护性。

步骤 1：检查 TiDB 节点的运行时间

首先要确认至少一个 TiDB 节点已经运行了足够长的时间，覆盖了完整的业务周期，TIDB_INDEX_USAGE 中保存了足够的数据。

```
mysql> select INSTANCE,START_TIME,UPTIME
    -> from INFORMATION_SCHEMA.CLUSTER_INFO
    -> where TYPE='tidb';
+--------------------+---------------------+--------------------+
| INSTANCE           | START_TIME          | UPTIME             |
+--------------------+---------------------+--------------------+
| 192.168.0.101:4000 | 2024-12-13 17:33:52 | 1035h31m13.285074s |
| 192.168.0.102:4000 | 2024-12-13 17:34:02 | 1035h30m45.625982s |
+--------------------+---------------------+--------------------+
```

- **如果 TiDB 节点已经分组，服务不同的应用，那么要确保每个组中都有实例运行了足够长的时间。**

步骤 2：获取 schema_unused_indexes 的输出

查询 sys.schema_unused_indexes，看哪些索引自 TiDB 节点启动以来未被访问过。

```
mysql> select * from sys.schema_unused_indexes \G
*************************** 1. row ***************************
object_schema: bookshop
  object_name: users
   index_name: nickname
*************************** 2. row ***************************
object_schema: bookshop
  object_name: ratings
   index_name: uniq_book_user_idx
```

在这个例子，识别出两个索引从没有被访问过，都来自 bookshop（https://docs.pingcap.com/zh/tidb/stable/dev-guide-bookshop-schema-design）数据库。

步骤 3：将索引设置为不可见

在非高峰期，将未使用的索引设置为“不可见”，以验证其是否对查询性能产生影响。在 TiDB 中，DBA 可以使用 `ALTER TABLE` 命令将索引设置为不可见。

```
ALTER TABLE bookshop.users ALTER INDEX nickname INVISIBLE;
ALTER TABLE bookshop.ratings ALTER INDEX uniq_book_user_idx INVISIBLE;
```

步骤 4：观察数据库表现

监控业务的执行情况，确保设置为不可见的索引不影响查询性能。这一过程需要持续一段时间，具体时长应根据业务的特点来决定。如果在这段时间内，查询的性能没有显著下降，则可以确认该索引是未使用的，可以安全地移除。

步骤 5：恢复索引可见（如果出现性能回退）

在某些情况下，虽然大多数业务流程运行正常，但可能会有特定的查询场景或边缘案例受影响。因此，如果在验证过程中发现某些查询性能出现回退，可以随时将索引恢复为可见状态，并重新评估其使用情况。

```
ALTER TABLE bookshop.users ALTER INDEX nickname VISIBLE;
```

这条命令会将 nickname 这个索引恢复为可见状态，重新供查询优化器使用。

- 在诊断过程中，可以通过设置 8.0 新增的变量 `tidb_opt_use_invisible_indexes` ，在当前会话验证将索引恢复可见后的执行效果。

步骤 6：安全删除未使用的索引

经过一段时间的验证，如果确认某个索引在业务中未被使用，并且将其设置为不可见后没有引发任何性能问题，DBA 可以决定将该索引从数据库中删除。

```
ALTER TABLE bookshop.users DROP INDEX nickname;
ALTER TABLE bookshop.ratings DROP INDEX uniq_book_user_idx;
```

总结一下，通过查询视图 `schema_unused_indexes` ，DBA 可以找出未使用的索引，并通过将索引设置为“不可见”、监控验证后再删除的方式，降低删除操作的风险。在 TiDB 中，整个过程都可以通过 SQL 操作完成，灵活且高效。

识别并优化低效索引

某些低效索引不仅无法有效加速查询，反而可能导致更多的 I/O 操作，从而增加查询延迟。借助 `TIDB_INDEX_USAGE` 中收集的信息，DBA 可以根据实际需要，编写 SQL 语句筛选潜在的优化目标。**通常，访问频次较高且访问行数占比较高的索引是低效索引的典型代表**。以下查询将列出访问超过 100 万次，并且其中一半以上的选择率大于 20% 的索引 。

```
SELECT TABLE_SCHEMA,
       TABLE_NAME,
       INDEX_NAME,
       SUM(QUERY_TOTAL),
       SUM(PERCENTAGE_ACCESS_20_50),
       SUM(PERCENTAGE_ACCESS_50_100),
       SUM(PERCENTAGE_ACCESS_100)
  FROM INFORMATION_SCHEMA.CLUSTER_TIDB_INDEX_USAGE
 GROUP BY TABLE_SCHEMA, TABLE_NAME, INDEX_NAME
HAVING SUM(QUERY_TOTAL) > 1000000
   AND SUM(PERCENTAGE_ACCESS_20_50 + PERCENTAGE_ACCESS_50_100 + PERCENTAGE_ACCESS_100) / SUM
 ORDER BY SUM(QUERY_TOTAL)
;
```

下面这个例子列出发生了索引全扫描的索引。设置 100 次的访问门槛是为了排除一些手动提交的 SQL 或调试阶段的记录：

```
SELECT TABLE_SCHEMA,
       TABLE_NAME,
       INDEX_NAME,
       SUM(QUERY_TOTAL),
       SUM(PERCENTAGE_ACCESS_100)
  FROM INFORMATION_SCHEMA.CLUSTER_TIDB_INDEX_USAGE
 GROUP BY TABLE_SCHEMA, TABLE_NAME, INDEX_NAME
HAVING SUM(QUERY_TOTAL) > 100
   AND SUM(PERCENTAGE_ACCESS_100) > 0
 ORDER BY SUM(QUERY_TOTAL)
;
```

识别出低效索引后，DBA 需要评估并进行优化，这里分为几种情况。

执行计划选择问题：

索引选择是查询优化器基于统计值估算的结果，受优化器的限制，数据库无法保证在所有条件下都一定能做出正确判断，如果数据库中存在更优的索引却没法被优化器选到，可以**从数据库**

层面对优化过程进行调优。

- **执行计划管理（SPM）**：为查询引入 hints（https://docs.pingcap.com/zh/tidb/stable/optimizer-hints ）或创建执行计划绑定（ https://docs.pingcap.com/zh/tidb/stable/sql-plan-management#执行计划绑定-sql-binding ）（SQL Binding），把更优的执行计划固定下来。
- **统计信息收集**：对统计信息的收集策略进行调整，比如增加并发，提高收集频率等。确保统计信息能帮助优化器做出正确的选择。

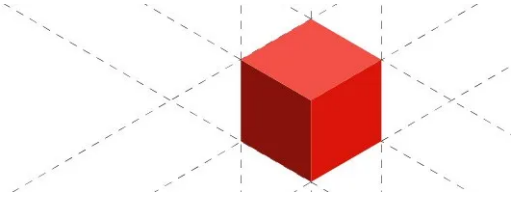
设计建模问题：

如果不是执行计划选择的问题，则需要**从设计角度进行优化**。以下是一些常见的优化思路：

- **调整索引设计**：对于选择性差的索引，可以考虑增加更多列、更换为过滤性更强的列，或者采用不同类型的索引（如前缀索引或多值索引）。
- **调整查询设计**：如果某些查询因低效索引导致性能瓶颈，DBA 可以通过调整查询语句，使其更有效地利用现有的索引。
- **调整应用建模**：在某些情况下，数据库的建模可能已经限制了某些查询的优化空间。此时需要重新审视数据结构，考虑是否需要调整表设计达到更好的索引过滤。

- **如果需要在生产环境中添加新索引，建议先将其创建为“不可见”索引，确保统计信息收集完成后，再将其设为“可见”。删除索引时，可以参考上述“步骤 3 – 步骤 6”来执行。**

5 总结



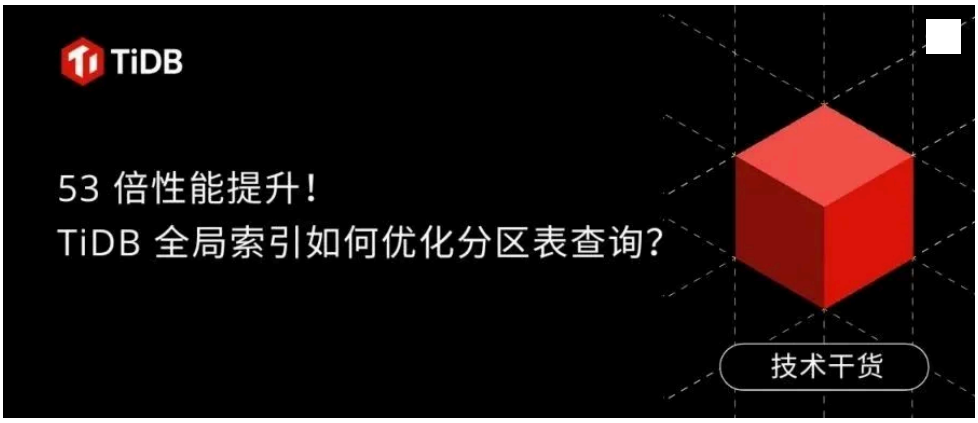
通过识别并优化未使用或低效的索引，可以减少资源浪费，并提高系统的响应速度和稳定性。在 TiDB 中，`TIDB_INDEX_USAGE` 系统表提供了相对丰富的索引使用统计数据，帮助 DBA 快速发现低效索引，并通过优化或删除它们来提升数据库效率。

尽管删除索引的操作相对简单，但在实施时仍需注意潜在的限制和风险，尤其是在大数据量和高并发环境下。执行索引优化时，DBA 应遵循以下最佳实践：

- 定期检查索引使用情况，尤其是对于大规模数据库。
- 确保用于决策的统计数据涵盖足够长的业务周期，避免误判。
- 通过设置索引为“不可见”来验证是否会影响查询性能，降低删除索引可能造成的风险。
- 创建和删除索引应选择在业务低峰时段进行。

通过遵循这些最佳实践，TiDB 用户不仅能保持系统的稳定性，还能实现高效的索引管理和优化，确保数据库的高效运行。未来，TiDB 将持续丰富索引运行数据，基于时间周期归档，为用户提供更加精准的指标和优化建议，以提升系统的性能和稳定性。

/ 相关推荐 /



53 倍性能提升！TiDB 全局索引如何优化分区表查询？

点击文末**【阅读原文】**，立即下载试用 TiDB！

阅读原文