腾讯云数据仓库 TCHouse-X

实践教程

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

实践教程
表设计

最近更新时间：2026-05-06 16:28:12

本文为您介绍如何通过科学设计表结构（包括表类型、分桶、主键、排序键及分区）来榨干 TCHouse-X 的性能，实现查询加速与存储成本的平衡。

分桶

分桶是 TCHouse-X 实现并行计算的核心。通过 `PARTITIONED BY SPEC(BUCKET(n, column))`，数据会根据指定列的 Hash 值分布到不同的节点上。

示例代码

```
CREATE TABLE supplier (
  s_suppkey     INT NOT NULL,
  s_name        STRING NOT NULL,
  s_nationkey   INT NOT NULL,
  PRIMARY KEY (s_suppkey)
)
PARTITIONED BY SPEC (
  BUCKET(128, s_suppkey) -- 建议分桶数为 2 的幂次方
);
```

最佳实践与避坑

- **高基数列优先：** 选择值分布均匀的字段（如 `user_id`，`order_id`），避免**数据倾斜**。
- **严禁使用时间列：** 避免使用 `DATE` 或 `TIMESTAMP` 作为分桶键。否则，查询最近数据时压力会集中在少数桶，导致“一核有难，多核围观”。
- **本地 Join 优化：**将经常需要相互关联（Join）的字段设为分桶键。当两张表以相同分桶键关联时，可触发 Colocated Join，省去昂贵的数据网络传输（Shuffle）。
- **查询裁剪：** 优先选择 `WHERE` 条件中高频出现的过滤字段。

主键

主键不仅是数据的唯一标识，更是实现高效更新（Upsert/Delete）的基石。

核心规则

- **更新必备：** 只有定义了主键的表才支持 `UPDATE` 和 `DELETE` 操作。

- **字段精简：** 推荐使用数值类型（如 `BIGINT`）作为主键。主键越短，索引维护成本越低。
- **包含关系：** **主键必须包含所有的分桶键和分区键**。建议将分桶键/分区键放在组合主键的前部。

排序

底层存储会按照 `SORT BY LEXICAL` 指定的列进行物理排序，这就像为图书馆的图书编码。

收益分析

- **查询加速：** 排序键能显著收窄扫描范围。对于范围查询（`>`、`<`、`BETWEEN`），TCHouse-X 可以快速跳过不相关的数据块（Block Skipping）。
- **压缩效率：** 相同或相似的值由于物理位置相邻，压缩算法能获得更高的压缩比，降低存储开销。

分区

分区是数据管理的**第一层边界**，通常按时间或业务大类划分。

最佳实践

规避方案：分区不支持函数表达式

目前不支持 `PARTITIONED BY SPEC(HOUR(ts))` 这种写法。

- 解决方案：冗余分区列。
  - 建表时显式增加分区字段：`start_time_hour INT`。
  - 写入时手动计算：`INSERT INTO ... VALUES (..., HOUR(now()))`。

规避方案：不支持 `INSERT OVERWRITE` 目标分区

若需刷新特定分区数据：

- 标准流程：`ALTER TABLE ... DROP PARTITION (...)` → `INSERT INTO ...`。

数据更新

最近更新时间：2026-05-06 16:28:12

TCHouse-X 专为实时分析而生，旨在提供极高的数据时效。通过先进的存储模型与灵活的更新机制，TCHouse-X 将分析延迟从“天级”跨越至“秒级”，助力企业构建敏捷的决策闭环。
本文将系统阐述 TCHouse-X 的数据更新能力，涵盖核心原理、更新/删除方式及性能实践教程。

核心原理：MVCC 多版本并发控制

TCHouse-X 采用 MVCC (Multi-Version Concurrency Control) 机制来管理高并发下的数据读写。

- 原子性保障： 每个写入操作（ `INSERT` 、 `UPDATE` 、 `DELETE` ）都被封装在一个事务中，确保数据要么全部写入成功，要么完全回滚。
- 版本排序： 系统为每次提交分配递增的版本号。当主键发生冲突时，版本号最高（即最新）的数据将作为查询可见的最终结果，旧版本数据会在后台异步清理（Compaction）。

主键表数据更新方式

对于主键表，TCHouse-X 提供了两种主要的更新路径，分别对应不同的业务负载：

路径 A：高性能导入更新 (UPSERT) —— 推荐方式

这是实现高频、高并发数据同步的最佳路径。 `INSERT INTO` 语句天然具备 `UPSERT` 语义。

- **逻辑**:
  - 主键存在： 以新行覆盖旧行。
  - 主键不存在： 插入新行。
- **适用场景**： 实时流数据同步（如 Flink/Spark 写入）、CDC 数据镜像。

路径 B：标准 SQL DML (UPDATE)

支持标准的 SQL `UPDATE` 语法，提供更灵活的业务逻辑表达能力。

- 逻辑： 系统先根据 `WHERE` 子句扫描并定位满足条件的数据，计算更新值后重新写回。
- 适用场景： 修正历史数据、基于复杂关联（Join）条件的批量更新。
- 限制： 由于涉及扫描与重写过程，适合**低频、批量**任务，不建议用于高频行级更新。

事务 (Transaction) 机制

事务是 TCHouse-X 保证数据一致性的最小工作单元。

- **隐式事务**： 每个 DML 语句在执行开始时自动开启事务，执行成功则自动提交（Commit），失败则自动回滚（Rollback）。
- **隔离性**： 查询操作始终读取已提交的最新版本，不会被正在进行的写入操作阻塞。

> **注意：**
> - 目前 TCHouse-X 暂不支持跨多表的原子性事务，事务作用域仅限于单表。
> - 在进行大规模 `UPDATE` 或 `DELETE` 时，建议分批次进行，以避免产生过大的单一版本，减轻存储层压力。

实践教程

优化维度	建议方案
写入频率	避免“单条频繁提交”，建议采用小批量（Batch）写入，每秒提交 1-5 次为宜。
主键设计	主键字段应尽可能精简，推荐使用数值类型以提升版本比对效率。
谓词下推	在执行 `UPDATE` 或 `DELETE` 时，尽量带上分区键过滤条件，减小扫描范围。

查询调优

最近更新时间：2026-05-06 16:28:12

本指南旨在帮助数据工程师与分析师优化在 TCHouse-X 上的查询性能。通过理解 **Horn 自研优化器** 的工作原理、掌握诊断工具及遵循最佳实践，您可以显著降低查询延迟并提升系统并发能力。

核心机制：统计信息与 CBO

TCHouse-X 的 Horn 优化器 采用基于成本的优化模型（CBO, Cost-Based Optimizer）。它通过分析表的数据量、列基数（Cardinality）及数据分布，决定最优的执行路径（如 Join 顺序、Broadcast vs Shuffle 策略等）。

> **警告：**
> 没有统计信息，CBO 将无法生效，系统仅能回退到基础的规则优化（RBO），这通常会导致性能大幅下降。

统计信息收集 (COMPUTE STATS)

在创建新表、全量导入或增量数据变更显著后，必须手动收集统计信息，以防优化器因误判导致内存溢出 (OOM) 或全表扫描。

语法示范

```
-- 收集统计信息
COMPUTE STATS customer;

-- 查看表级摘要（行数、文件数、格式）
SHOW TABLE STATS customer;

-- 查看列级详情（基数、空值数、平均长度）
SHOW COLUMN STATS customer;
```

何时需要重新收集

- **首次加载**：数据入库后立即执行。
- **显著变更**：当数据增量超过总量的 20% 时。
- **性能回退**：原本正常的查询突然变慢，首选排查统计信息是否过期。

高级 SQL 优化最佳实践

编写准则

- **精简字段**：严禁 `SELECT *`，仅读取必要列以减少 I/O 和内存占用。
- **谓词下推**：虽然 CBO 支持自动下推，但在复杂子查询中，显式添加过滤条件更稳健。
- **类型匹配**：避免隐式转换（如 `string_col = 123`），这会导致索引失效及额外的 CPU 开销。

Join 与聚合调优

- **大表 Join 小表**：确保小表统计信息准确，优先触发 Broadcast Join。
- **倾斜处理**：对存在大量 `NULL` 或热点值的 Join Key，建议先进行过滤或加随机盐（Salt）打散。
- **去重策略**：在高基数列上，`GROUP BY` 的并行处理性能通常优于 `DISTINCT`。

存储辅助

- **分区裁剪**：查询条件必须包含分区键，避免全表扫描。
- **小文件治理**：定期合并小文件，提升 Scan 阶段的吞吐量。