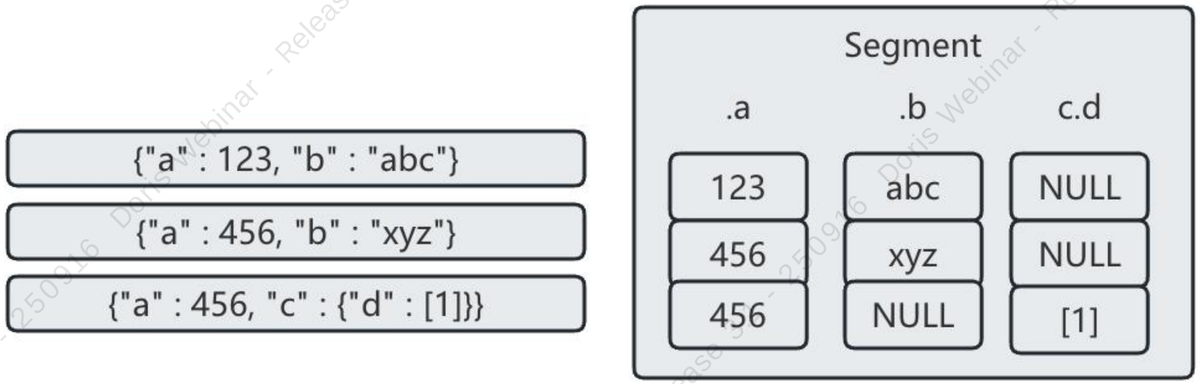
Doris 3.1 新版本解读（二）
半结构化能力全面升级之 Variant 数据类型

李航宇 Apache Doris Committer、飞轮科技资深技术专家

目录

Variant 数据类型简介



列式存储(空间节省N倍)
1. 字典编码
3. 强大的压缩能力
4. 动态适应数据类型

索引(检索、高并发能力)
1. min、max、bloom filter索引
2. 倒排索引

查询高性能
1. 向量化引擎
2. 高效数据裁剪（文件、page、动态剪枝）
3. 支持高并发查询

跟 JSON 类型对比收益

存储空间

类型	存储空间
预定义静态列	12.618 GB
VARIANT 类型	12.718 GB
JSON 类型	35.711 GB

节省约 65% 存储容量

查询次数	预定义静态列	VARIANT 类型	JSON 类型
第一次查询 (cold)	233.79s	248.66s	大部分查询超时
第二次查询 (hot)	86.02s	94.82s	789.24s
第三次查询 (hot)	83.03s	92.29s	743.69s

3.1 Doris 稀疏列开启后效果

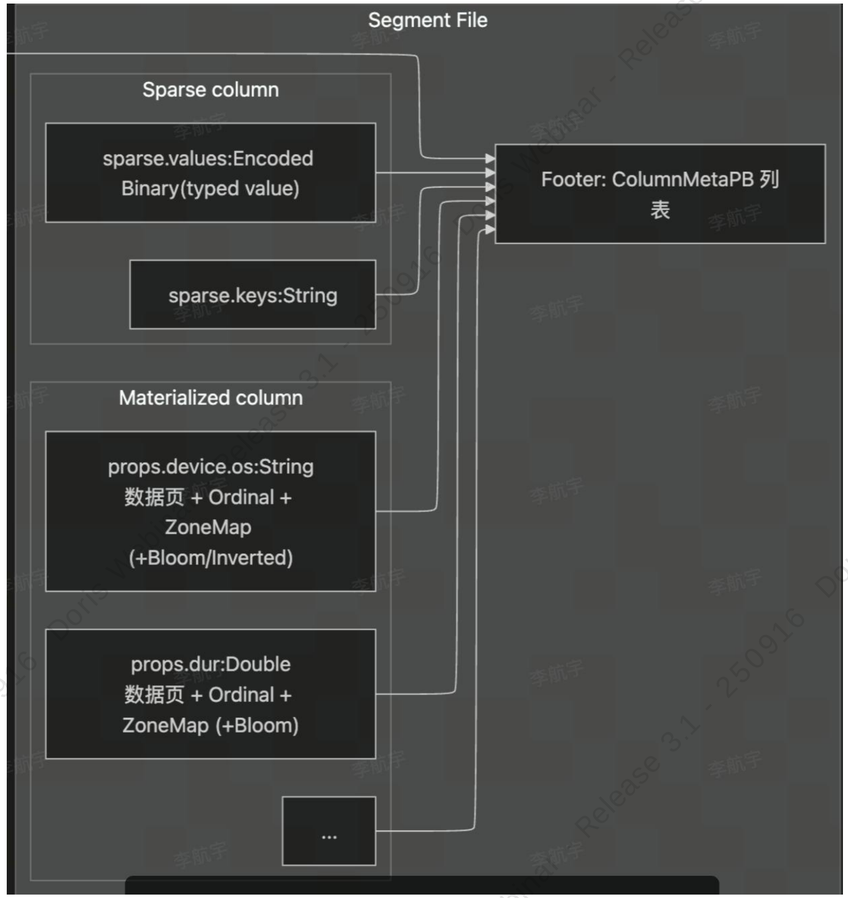
- 列数不再有硬限制， 2.1、3.0默认是单个tablet 1024个子列提取上限
- 更高的存储效率，Variant扩展性提升，极大减少schema内存占用
- 部分稀疏场景下空间再节省1/3
- 查询自适应稀疏和稠密列， 稠密列会自动根据稀疏度变成稀疏列，反之亦可

Doris 稀疏列使用方式

```
CREATE TABLE example_table (
  id INT,
  data_variant VARIANT<
      properties(
          'variant_max_subcolumns_count' = '2048',
       )
  >
);
```

variant_max_subcolumns_count：默认 0（不限制 Path 物化列数）。建议在生产设置为 2048（Tablet 级别）以控制列数。超过阈值后，低频/稀疏路径会被收敛到共享数据结构，从该结构查询可能带来性能下降。

Doris Variant稀疏列存储



原理

稀疏子列（Sparse Subcolumns）：按 JSON key 频次排序，只提取 Top-N 高频子列入“真列式”；长尾保持在Sparse列存储，避免无序扩张。

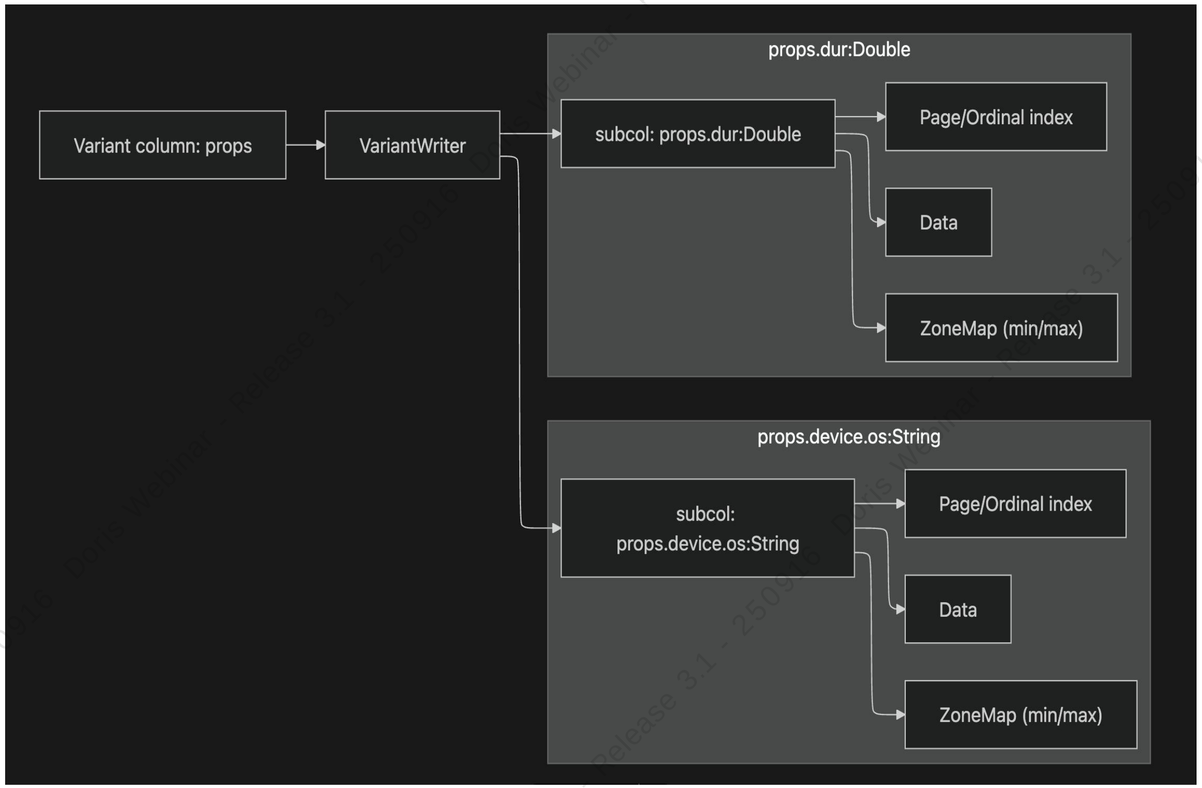
- 物化子列：每条“JSON path：最终类型”形成独立列，具备各自数据页、Ordinal、ZoneMap，且按需要具备Bitmap/Bloom/倒排索引。

- Sparse列：键列（path）、值列（typed binary），配合VariantStatistics.sparse_column_non_null_size 提供稀疏JSON path存在感知，支持 Extract/Merge 访问模式。

- Footer：集中维护 ColumnMetaPB、ColumnPathInfo、子列 none_null_size 与 VariantStatistics，供 Segment 构建reader 与执行阶段做类型/JSON path判定。

写入流程

写入流程（列式展开与稀疏列Flush逻辑）



- JSON key按照叶子JSON path（Leaf Path）切分为多条子列（如 props.device.os:String、props.dur:Double）

- 每条子列拥有：
  - 单独的列式存储、列级别的元数据、索引（ZoneMap、BloomFilter、InvertedIndex）
  - 位图/稀疏索引（映射原始 row ordinal 到子列 ordinal）；
  - Page/Ordinal Index 与 ZoneMap（Page/Column/Segment/级 min-max）

- 稀疏列存储，将JSON key按照出现频次进行排序，取前N个左右子列，余下的单独拼接到稀疏列中（减少额外的存储开销）

目录

超多列的适用场景

车联网/IoT 遥测

- 设备型号多、传感器维度动态增减。
- 营销自动化/CRM：事件/用户属性持续扩展（如自定义 event/property）

广告/埋点事件

- 海量可选 properties，字段稀疏且不断演进

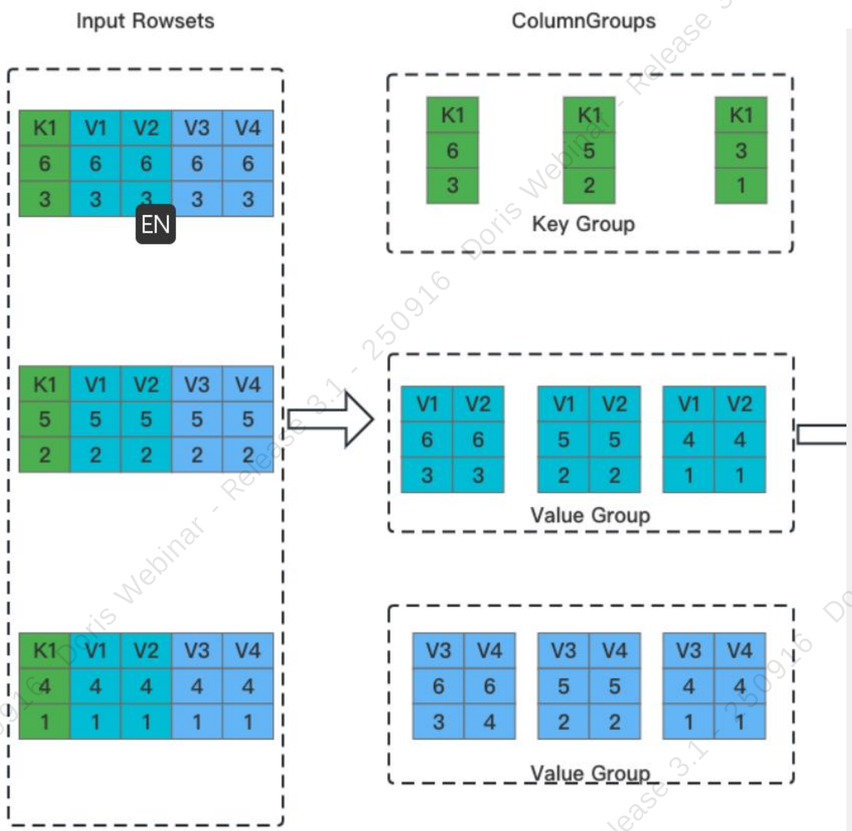
安全审计/日志

不同源日志字段各异，需按模式聚合检索

电商商品属性

类目跨度大，商品属性高度可变。

内存优化：Vertical compaction机制



– Variant schema拆分，根据叶子节点生成compaction schema

– 切分列组。将输入 Rowset 按照列进行切分，所有的 Key 列一组、Value 列按 N 个一组，切分成多个 Column Group；

– 按照Column Group 分组写入文件

效果

500列Compaction内存只占优化前1/10，优化前最多支持1000 variant子列，优化后实测Compaction可支持10000子列（不包括稀疏列部分）

进一步优化

大量稀疏值插入效率优化

优化值填充默认值效率，按 batch 的方式进行批量填充（减少虚函数开销）

更加高效元数据管理

通过 LRU 机制减少内存中列存储相关元数据缓存内存开销。并按需加载查询的元数据

目录

为什么要Schema Template

提升子路径类型可维护性

- 关键子路径类型可在 DDL 中固定，避免类型漂移引发的查询报错、索引失效与隐式转换开销
- 为不同子路径定制倒排策略（分词/非分词、解析器、短语搜索等），常用查询延迟更低、命中更稳定。
- 不再“整列统一继承索引”（2.1 的做法易膨胀），而是“按子路径精细化配置”，显著降低索引数量、写放大与存储成本

Schema Template使用举例

```
CREATE TABLE IF NOT EXISTS tbl (
    k BIGINT,
    v VARIANT<'content' : STRING>,
INDEX idx_tokenized(v) USING INVERTED
PROPERTIES("parser" = "english",
"field_pattern" = "content"),
INDEX idx_v(v)  USING INVERTED
PROPERTIES("field_pattern" = "content")
);
-- Use tokenized index
SELECT * FROM tbl WHERE v['content']
MATCH 'Doris';

-- Use non-tokenized index
SELECT * FROM tbl WHERE v['content'] =
'Doris';
```

- 常用属性：field_pattern（目标子路径）、analyzer、parser、support_phrase 等
- v.content 将同时拥有一个英文分词倒排索引和一个非分词倒排索引
- 查询自适应索引类型

Variant Schema 使用举例

```
CREATE TABLE IF NOT EXISTS tbl2 (
  k BIGINT,
  v VARIANT<
      'pattern1_*' : STRING, -- batch-typing: all
subpaths matching pattern1_* are STRING
      'pattern2_*' : BIGINT, -- batch-typing: all
subpaths matching pattern2_* are BIGINT
      properties("variant_max_subcolumns_count" =
"2048")   >,
  INDEX idx_p1 (v) USING INVERTED
    PROPERTIES("field_pattern"="pattern1_*",
"parser" = "english"),
INDEX idx_p2 (v) USING INVERTED
    PROPERTIES("field_pattern"="pattern2_*") --
non-tokenized inverted index for pattern2_*
) DUPLICATE KEY(k);
```

- 使用通配符匹配类型
  - 'pattern1
  - 'pattern2
- 配合稀疏列机制控制子列 properties("variant_max_subcolumns_count" = "2048")
- 分别对不同的pattern配置不同的索引

Thanks !