

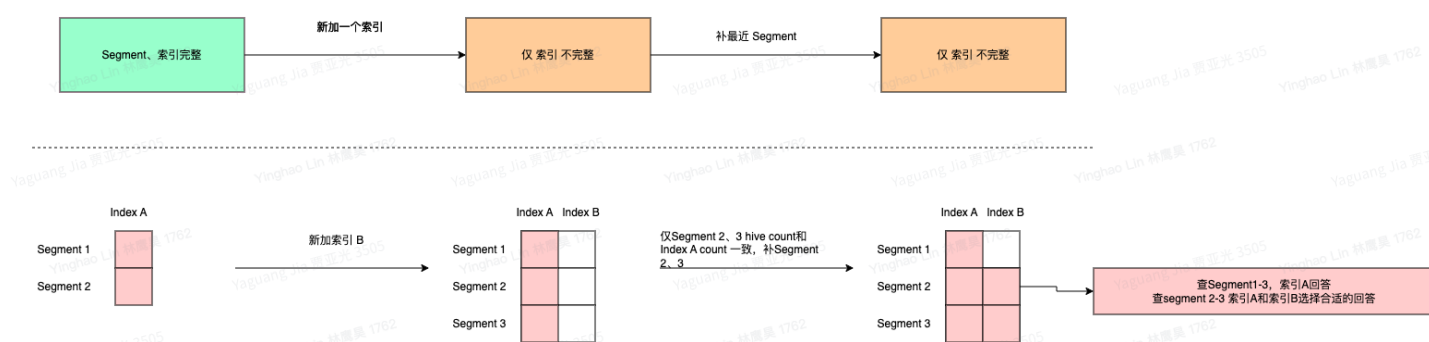
KYLIN-5527 Kylin job engine adds the ability to check entries with data source(e.g. hive)

背景

如果用户定期删hive表中的旧数据，同时一些新增加的索引又需要在历史segment上要进行补数，那么就要做到

- 数据要准确，对于hive表数据历史存在而现清空的情况，不能补出空数据的新索引，而导致新老索引数据不一致

解决方案：构建出异构segment



异构segment的构建和查询之前 KE 已经支持，但构建需要人工判别空segment并逐个构建，没有做到针对空segment的自动判断

此外，针对 hive表历史数据条目发生变化而非删除的情况，当需要从hive重新获取数据构建索引时，目前也没有相应逻辑去保证此种场景下模型内新旧索引数据的一致性

综上，需要增加索引补数流程中的 数据一致性 检查，解决上述招行问题的同时，更进一步增强 KE 产品在面对数据不一致场景下的识别和处理能力

Dev Design

主要有2处改动点：索引补数构建增加 Data Count Check 和 构建任务支持 segment 级别的部分成功

1. 索引补数构建增加 Data Count Check

-

上述任何一个check失败，则该segment的补数构建任务直接跳过

- 该count check逻辑由一个模型级开关控制，默认关闭

`kylin.build.data-count-check-enabled`

- 仅对任务类型为 INDEX_BUILD 的任务进行 count check
 - 该类型任务界面展示类型为：构建索引
 - 数据范围2种：全量加载 和 segment时间范围

构建索引	model1	全量加载
构建索引	test1	2022-10-02 00:00:00...

- 当索引补数需从hive进行
 - 先执行check1
 - 再执行check2（因为补数时一定会先产生平表数据，因此实际是检查已有索引与平表count值的一致性）
- 当索引补数均可从父索引来
 - 只执行check1
- Count check位置，做在生成平表里面

- 等待资源

持续时间: 1.6m

- 构建或刷新快照

持续时间: 0.04m

- 物化事实表视图

持续时间: < 0.01m

- 生成全局字典

持续时间: < 0.01m

- 生成平表

持续时间: < 0.01m

- 获取平表统计信息

持续时间: 0.02m

- 分层构建索引 3/3

持续时间: 0.05m

- 更新平表统计信息

持续时间: 0.65m

count check

- 如何判断本次索引补数是否涉及平表?

AdaptiveSpanningTree#fromFlatTable

```
public boolean fromFlatTable() {  
    return level0thNodes.stream().anyMatch(TreeNode::parentIsNull);  
}
```

- Count check具体实现?

- 索引的count check已有现成方法

```
64 object SanityChecker extends LogEx {  
65  
66     val SKIP_FLAG: Long = -1L  
67  
68     def getCount(df: DataFrame, layout: LayoutEntity): Long = {  
69         if (IndexEntity.isTableIndex(layout.getId)) {  
70             df.count()  
71         } else {  
72             layout.getOrderedMeasures.values().asScala.find(m => m.getFunction.isCountConstant && !m.isTomb) match {  
73                 case Some(countMeasure) =>  
74                     val countMeasureCol = countMeasure.getId.toString  
75                     val row = df.select(countMeasureCol).agg(sum(col(countMeasureCol))).collect().head  
76                     if (row.isNullAt(0)) 0L else row.getLong(0)  
77                 case _ =>  
78                     SKIP_FLAG  
79             }  
80         }  
81     }  
82 }
```

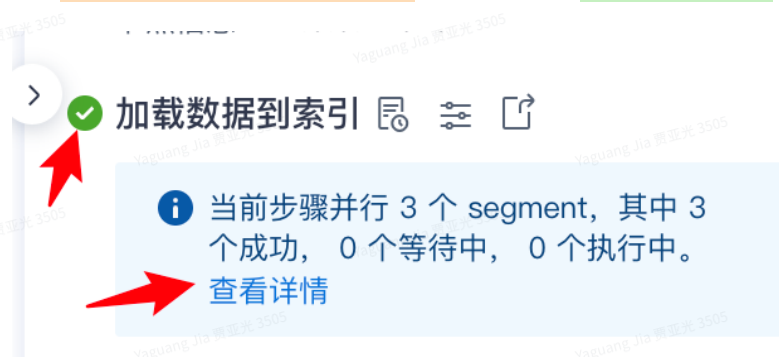
- 平表的count check: 直接count即可

- count计算需通过slowStartExec进行调用, 防止spark任务不受控提交导致的堵塞

- 变更为

- 新增子步骤状态：WARNING
- 在count check机制下，允许segment级别部分成功
 - 单 segment 任务，子步骤 WARNING，整体任务仍为 FINISHED
 - 同一任务多 segment 并行构建，存在 WARNING 的 segment 任务，整体任务仍为 FINISHED
- 其他原因的失败（ERROR）依旧会导致整体任务失败，这点和原产品行为一致
- 成功构建的segment，索引可以使用
- 提示信息变更如下：

- “当前步骤并行3个 segment，其中 2 个成功，1 个因数据不一致未构建，0 个等待中，0 个执行中”
- 若所有segment都未构建，则左上角 绿色打勾图标 需变更为 黄色三角图标





- 点开“查看详情”后



- 展开某个未构建的segment后

Segment 详情

>  2022-10-01 00:00:00 GMT+8 – 2022-11-01 00:00:00 GMT+8 100.0%

✓  2022-11-01 00:00:00 GMT+8 – 2022-12-01 00:00:00 GMT+8 100.0%

等待时间: < 0.01m

持续时间: 2.5m [详情](#)

节点信息:

- 等待资源
持续时间: 1.47m
- 构建或刷新快照
持续时间: 0.04m
- 物化事实表视图
持续时间: < 0.01m
- 生成全局字典
持续时间: < 0.01m
- 生成平表
持续时间: < 0.01m
- 获取平表统计信息
持续时间: 0.02m
- 分层构建索引 1/1
持续时间: 0.06m

当前 segment 因数据不一致未构建

此处后续的子步骤状态置为跳过

- 如果一个任务仅包含一个跳过构建的segment，整体任务状态如何？

- 整体任务状态仍为 FINISHED

> 任务 ID: 6ed77c50-b209-f529-c021-f21146d42070
-fac6a04e-d007-e3d1-8bdf-ee24f8e9b98c

对象: test1

状态: **FINISHED**

等待时间: 0.03m

持续时间: 3.87m

- 但子步骤状态如下:

✓ 加载数据到索引

等待时间: < 0.01m
持续时间: 2.08m 详情
节点信息: 127.0.0.1:7070

黄色三角图标

当前 segment 因数据不一致未构建

- 等待资源
持续时间: 0.89m
- 构建或刷新快照
持续时间: 0.2m
- 物化事实表视图
持续时间: < 0.01m
- 生成全局字典
持续时间: 0.04m
- 生成平表
持续时间: < 0.01m

- 获取平表统计信息
持续时间: 0.01m
- 分层构建索引 1/1
持续时间: 0.07m
- 更新平表统计信息
持续时间: 0.85m

这3个步骤
状态置为跳过

子步骤效果

➤ 获取平表统计信息

持续时间: -

➤ 分层构建索引 0/1

持续时间: -

➤ 更新平表统计信息

持续时间: -

其他任务包含某个跳过的segment，是否可以提交？

- 只要segment处在构建过程中，其他任务若包含该segment就无法提交

仅剩一个异常索引的segment（其中仅包含一个异常的索引），该segment状态如何？

- 现状如图

基本信息
数据特征
Segment
索引
开发者

Segment 列表 ⓘ

刷新 | v

<input type="checkbox"/>	开始时间 ▾	结束时间 ▾	索引数 ⓘ ▾	状态 ▾	量
<input type="checkbox"/>	2022-10-02 00:00:00 GMT+8	2022-10-05 00:00:00 GMT+8	2/4	ONLINE	21
<input type="checkbox"/>	2022-10-01 00:00:00 GMT+8	2022-10-02 00:00:00 GMT+8	0/4	ONLINE	21

共 2 条 1 条/页 1/2 页

- 本次维持现状

- 关于segment覆盖机制 [KE-39001 CLONE - KE4构建支持segment覆盖 Acceptance](#)
 - segment覆盖会构建所有索引（包括新的之前尚未补数过的索引）
 - 可保证该segment中所有索引数据的一致性，因此不影响本次改动

3. 元数据改动

- 现状

```
data row : f8c43c52-d306-62eb-300d-f8c43c52 ,
"layout_instances": [
{
  "layout_id": 20000000001,
  "build_job_id": "d76eab2c-8345-505f-cdbe-ebfe5e377c0b-fac43c52-d306-62",
  "rows": 11,
  "byte_size": 6346,
  "file_count": 1,
  "source_rows": 11,
  "source_byte_size": 0,
  "partition_num": 1,
  "partition_values": [],
  "is_ready": false,
  "create_time": 1670818040131,
  "multi_partition": []
},
{
  "layout_id": 10001,
  "build_job_id": "35262a3c-2388-6bf2-61b6-b02885868cd0-fac43c52-d306-62",
  "rows": 8,
  "byte_size": 2124,
  "file_count": 1,
  "source_rows": 11,
  "source_byte_size": 0,
  "partition_num": 1,
  "partition_values": [],
  "is_ready": false,
  "create_time": 1670818946166,
  "multi_partition": []
},
{
  "layout_id": 20000010001,
  "build_job_id": "f277d7e1-7e44-213c-5ae5-27d7d38dcd74-fac43c52-d306-62",
  "rows": 11
}
```

- 新设计，增加 abnormal_type 字段表示 layout 异常


```

{
  "layout_id": 20001,
  "build_job_id": "4c20f2c7-7b2f-0504-6db7-e5d921fd993f",
  "rows": 0,
  "byte_size": 0,
  "file_count": 0,
  "source_rows": 0,
  "source_byte_size": 0,
  "partition_num": 0,
  "partition_values": [],
  "is_ready": false,
  "create_time": 1675757603265,
  "multi_partition": [],
  "abnormal_type": "DATA_INCONSISTENT"
},
}

```

由于目前 layout_instances 中存放的都是有效的 layouts，而本次新设计需加入 layouts 状态，同时也会加入非有效状态的 layouts，故以下代码中使用到 layout_instances 的地方根据上下文语义的不同需要分别处理：

- 语义1：仅需要获取有效的 layouts
- 语义2：需要获取全部 layouts

Method <code>getLayouts()</code> of <code>org.apache.kylin.metadata.cube.model.NDataSegDetails</code>			21+ usages
Project production files			Usages or base methods
NDataLayout.java	263	for (NDataLayout cached : segDetails.getLayouts()) {	
NDataSegDetails.java	117	for (NDataLayout cuboid : getLayouts()) {	
NDataSegDetails.java	128	for (NDataLayout cuboid : getLayouts()) {	
NDataSegDetails.java	159	List<NDataLayout> currentSortedLayouts = getSortedLayouts(getLayouts());	
NDataSegDetails.java	160	List<NDataLayout> anotherSortedLayouts = getSortedLayouts(another.getLayouts());	
NDataSegment.java	347	List<NDataLayout> filteredCuboids = segDetails.getLayouts().stream()	
NDataSegment.java	353	List<NDataLayout> cuboids = segDetails.getLayouts();	
NDataflowManager.java	733	if (seg.getStatus() == SegmentStatusEnum.WARNING && segDetails != null && segDetails	
NDataflowManager.java	833	val layouts = segment.getSegDetails().getLayouts();	
SegmentPartition.java	145	.getLayouts().stream() //	
NDataSegmentResponse.java	130	long segmentFileCount = segment.getSegDetails().getLayouts().stream()	
IndexPlanService.java	730	if ((seg.getSegDetails().getLayouts().size() - lockedIndexCountInSeg) != allIndexCountWi	
ModelService.java	1183	val segLayoutIds = segment.getSegDetails().getLayouts().stream().map(NDataLayout::ge	
ModelService.java	2393	List<NDataLayout> layouts = new LinkedList<>(segDetails.getLayouts());	
PartitionDictionaryBuilderHelper.java	52	for (NDataLayout cuboid : seg.getSegDetails().getLayouts()) {	
AfterBuildResourceMerger.java	112	dfUpdate.setToAddOrUpdateLayouts(theSeg.getSegDetails().getLayouts().toArray(new N	
AfterMergeOrRefreshResourceMerger.java	83	toUpdateCuboids.addAll(new ArrayList<>(mergedSegment.getSegDetails().getLayouts())	
AfterMergeOrRefreshResourceMerger.java	158	toUpdateCuboids.addAll(new ArrayList<>(mergedSegment.getSegDetails().getLayouts())	
DictionaryBuilderHelper.java	141	for (NDataLayout cuboid : seg.getSegDetails().getLayouts()) {	
DFMergeJob.java	76	for (NDataLayout cuboid : seg.getSegDetails().getLayouts()) {	
DataflowCleanerCLI.java	81	toBeRemoved.addAll(segment.getSegDetails().getLayouts().stream().map(NDataLayout::	
...<15 usages are out of scope 'Project production files'>...			
Press <code>⌘F7</code> again to search in 'Project Files'			

文件	代码行数	使用 getLayouts() 方法的目的
NDataLayout.java	263	

		用在 isCachedAndShared() 方法中，更新属性前判断该 ND 是否已被缓存
NDataSegDetails.java	117	getTotalRowCount方法，统计row count
NDataSegDetails.java	128	getLayoutByld方法，根据id号拿layout
NDataSegDetails.java	159	Segment merge前判断layout是否一致
NDataSegDetails.java	160	Segment merge前判断layout是否一致
NDataSegment.java	347	过滤为null的 NDataLayout，并重新对 segment 中 layout !
NDataSegment.java	353	将segment layouts放到缓存的 layoutsMap 中
NDataflowManager.java	733	清空warning状态segment里全部layout后，状态变更为rea
NDataflowManager.java	833	删除layout里多级分区信息
SegmentPartition.java	145	计算多级分区数据容量
NDataSegmentResponse.java	130	判断是否存在有效的基础明细索引
IndexPlanService.java	730	计算需要补全索引的segment数量
ModelService.java	1183	过滤出索引构建不全的segment
ModelService.java	2393	删除索引逻辑中，过滤并判断删除每个segment中对应的lay
PartitionDictionaryBuilderHelper. java	52	多级分区模型索引补充全局字典
AfterBuildResourceMerger.java	112	增量构建完成后，合并元数据，获取到layout信息统计layo
AfterMergeOrRefreshResourceMer ger.java	83	合并或刷新后，合并元数据，获取到多级分区模型segment 统计layout大小
AfterMergeOrRefreshResourceMer ger.java	158	合并或刷新后，合并元数据，获取到常规模型segment layc layout大小
DictionaryBuilderHelper.java	141	常规模型索引补充全局字典
DFMergeJob.java	76	segment合并前资源探测步骤，获取layouts，供后续读取p 使用
MergeStage.scala	81	segment合并，获取layouts，提交到spark进行merge操作
PartitionMergeStage.scala	53	多级分区模型segment合并，获取layouts，提交到spark进 作
DataflowCleanerCLI.java	81	命令行工具，清理segment中，索引计划里不存在的layout:

4. Rest API 改动

- 新增 获取segment下索引列表 接口

从现有接口中进行扩展：GET /api/index_plans/index?project=&segment_id=

其中 segment_id 为本次新增字段

原接口在手册中未开放，故不会对现有客户使用有影响

```
@ApiOperation(value = "getIndex", tags = { "AI" }, notes = "Update response: total_size")
@GetMapping(value = "/index")
public EnvelopeResponse<FusionRuleDataResult<List<IndexResponse>>> getIndex(
    @RequestParam(value = "project") String project, @RequestParam(value = "model") String modelId, //
    @RequestParam(value = "sort_by", required = false, defaultValue = "") String order,
    @RequestParam(value = "reverse", required = false, defaultValue = "false") Boolean desc,
    @RequestParam(value = "sources", required = false, defaultValue = "") List<IndexEntity.Source> sources,
    @RequestParam(value = "key", required = false, defaultValue = "") String key,
    @RequestParam(value = "status", required = false, defaultValue = "") List<IndexEntity.Status> status,
    @RequestParam(value = "ids", required = false, defaultValue = "") List<Long> ids,
    @RequestParam(value = "page_offset", required = false, defaultValue = "0") Integer offset,
    @RequestParam(value = "page_size", required = false, defaultValue = "10") Integer limit,
    @RequestParam(value = "range", required = false, defaultValue = "") List<IndexEntity.Range> range) {
    checkProjectName(project);
```

5. 对其他场景的影响

- 全量构建任务：索引补数时需进行 data count check
- 拆分为多个构建任务：不影响，仅 count check 通过的索引才会被构建出来，和 segment 构建方式无关

构建索引

该索引未构建至以下数据范围。为了提高查询效率，建议您将该索引构建至：

<input checked="" type="checkbox"/>	开始时间 ▾	结束时间 ▾
<input checked="" type="checkbox"/>	2022-10-01 00:00:00 GMT+8	2022-10-02 00:00:00 GMT+8
<input checked="" type="checkbox"/>	2022-10-02 00:00:00 GMT+8	2022-10-05 00:00:00 GMT+8

共 2 条

☒ 拆分多个任务并发构建 ⓘ

- 新模型的构建：不影响，无已有索引不受count check影响
- Segment合并：不影响，索引不一致无法合并
- Segment刷新：不影响，刷新即所有索引数据从hive构建，count check也就没有意义了

6. Special Case说明

- 招行3升4场景中，将base cuboid同时作为基础聚合和基础明细

解决方案：新增开关，允许明细和聚合索引之间count值不一致

开关可以是全局、项目、模型级别的：

- 默认关闭，意为所有索引count必须一致
- 若开启，则允许明细与聚合之间count不一致，但明细和明细、聚合和聚合之间count仍需要一致
- `kylin.build.allow-non-strict-count-check`

7. 遗留问题

- 开关默认开启与否需进行性能验证
 - 本地对小数据量索引进行count操作，一个约300ms
 - 目前规格测试仅覆盖全量构建场景 [国基准四](#)，[tpch100](#)，[22sql自动建模+大view](#) 提需求