Leverage SI till segment level

Author: Nihal kumar ojha
Version: v1

Background:

Secondary index tables are created as indexes and managed as child tables internally by Carbondata. Users can create a secondary index based on the column position in the main table(Recommended for right columns) and the queries should have filter on that column to improve the filter query performance. In the current design of SI, performance may degrade for some specific scenarios. This document will explain that scenario and the way of handling it.

Currently, query can be pruned with SI in two ways:

1. With spark plan rewrite.
2. SI as coarse grain index(without plan rewrite).

Bottleneck:

In the existing architecture, if the parent(main) table and SI table don't have the same valid segments then we disable the SI table. And then from the next query onwards, we scan and prune only the parent table until we trigger the next load or REINDEX command (as these commands will make the parent and SI table segments in sync). Because of this, queries take more time to give the result when SI is disabled.

Proposed Solution:

We are planning to leverage SI till the segment level. It means at place of disabling the SI table(when parent and child table segments are not in sync) we will do pruning on SI tables for all the valid segments(segments with status success, marked for update and load partial success) and the rest of the segments will be pruned by the parent table.

Example:

Let's assume segments available in parent table are 0,1,2,3,4

And segments available in the index table are 0,2,3

Current behavior: Index table will be disabled and all the filter queries will be pruned with help of the parent table.

Expected behavior: Don't disable index table while prune segments 0,2,3 with index table and segment 1,4 with parent table.

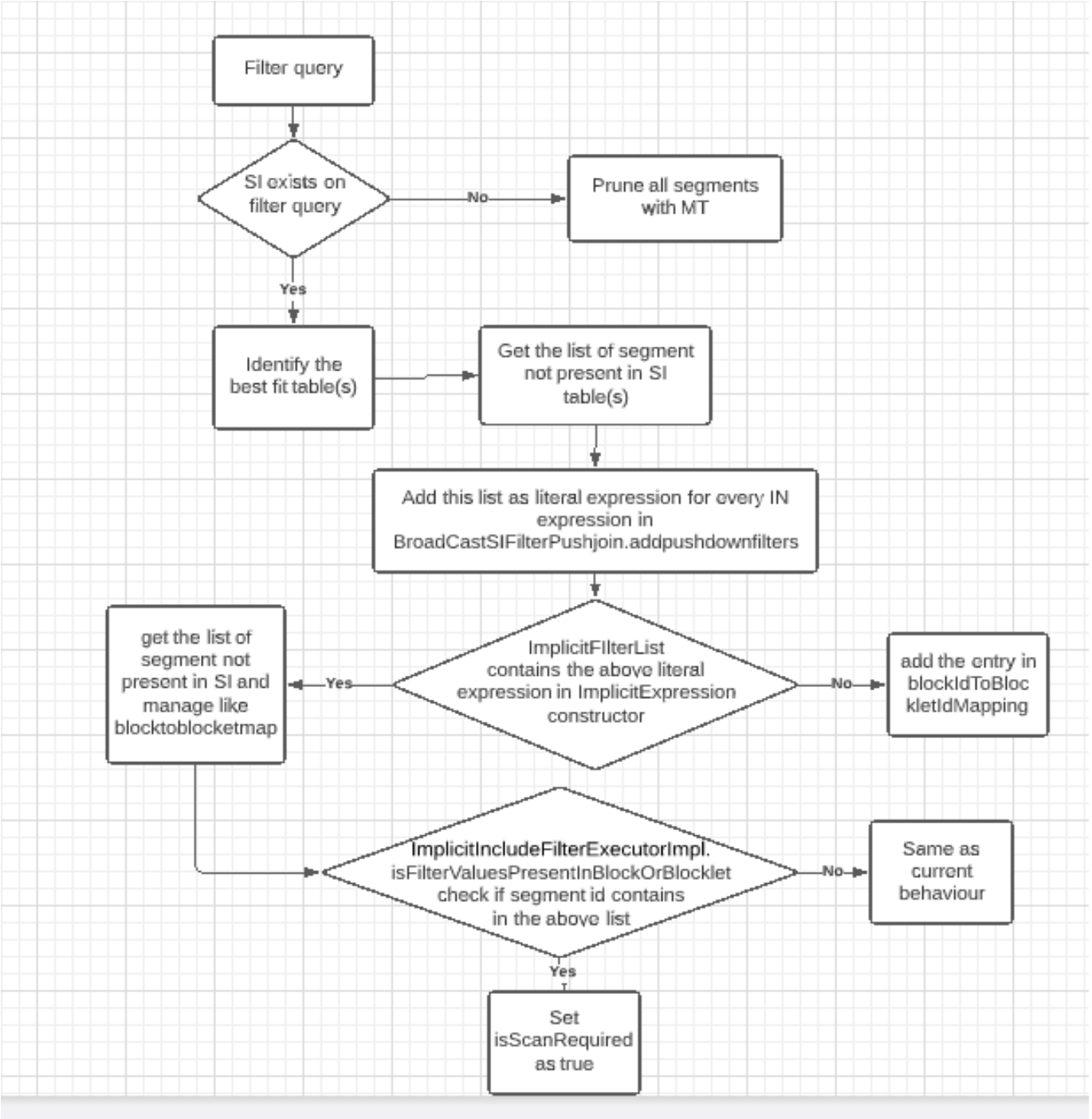
Leverage SI till segment level with spark plan rewrite

1. Property isSITableEnabled will be removed completely and will not be referred anywhere. For the old table also will check And remove if this property exists.
2. For every filter query, check if SI exists on the filtered column.
    a. If yes then identify the best fit SI table.
    b. If not then perform pruning with the main table and exit.
3. At the time of evaluating inputCopy in BroadCastSIFIlterPushJoin, we will identify the set of segments that are not present in the SI table(In case of more than one SI table we will keep adding the missing segments in the existing list).

    In the above example, we will have the set of missing segments as (1,4).
4. We will create a new literal which value is comma-separated missing segments(identified from step 3) with prefix MISSING_SI_SEGMENTS. And this literal will be added for every IN filter(every key) in BroadCastSIFilterPushJoin.addPushDownFilters.

    In the above example, we will have the new literal with value as MISSING_SI_SEGMENTS_1,4 and we will add this literal inside IN expression.
5. At the time of creating new ImplicitExpression from CarbonFilters, we will check if implicitFilterList contains the literal which value starts with MISSING_SI_SEGMENTS
    a. If yes then get the Set of missing SI segments from literal expression and we will handle this in the same way we have handled blockIdToBlockletIdMapping.
    b. If not then add the entry in blockIdToBlockletIdMapping like current behavior.
6. In ImplicitIncludeFilterExecutorImpl.isFilterValuesPresentInBlockOrBlocklet we will check if the above set(evaluated in step 5) contains the segment id present in uniqueBlockPath.
    a. If yes then set isScanRequired as true.
    b. If not then continue like current behavior.

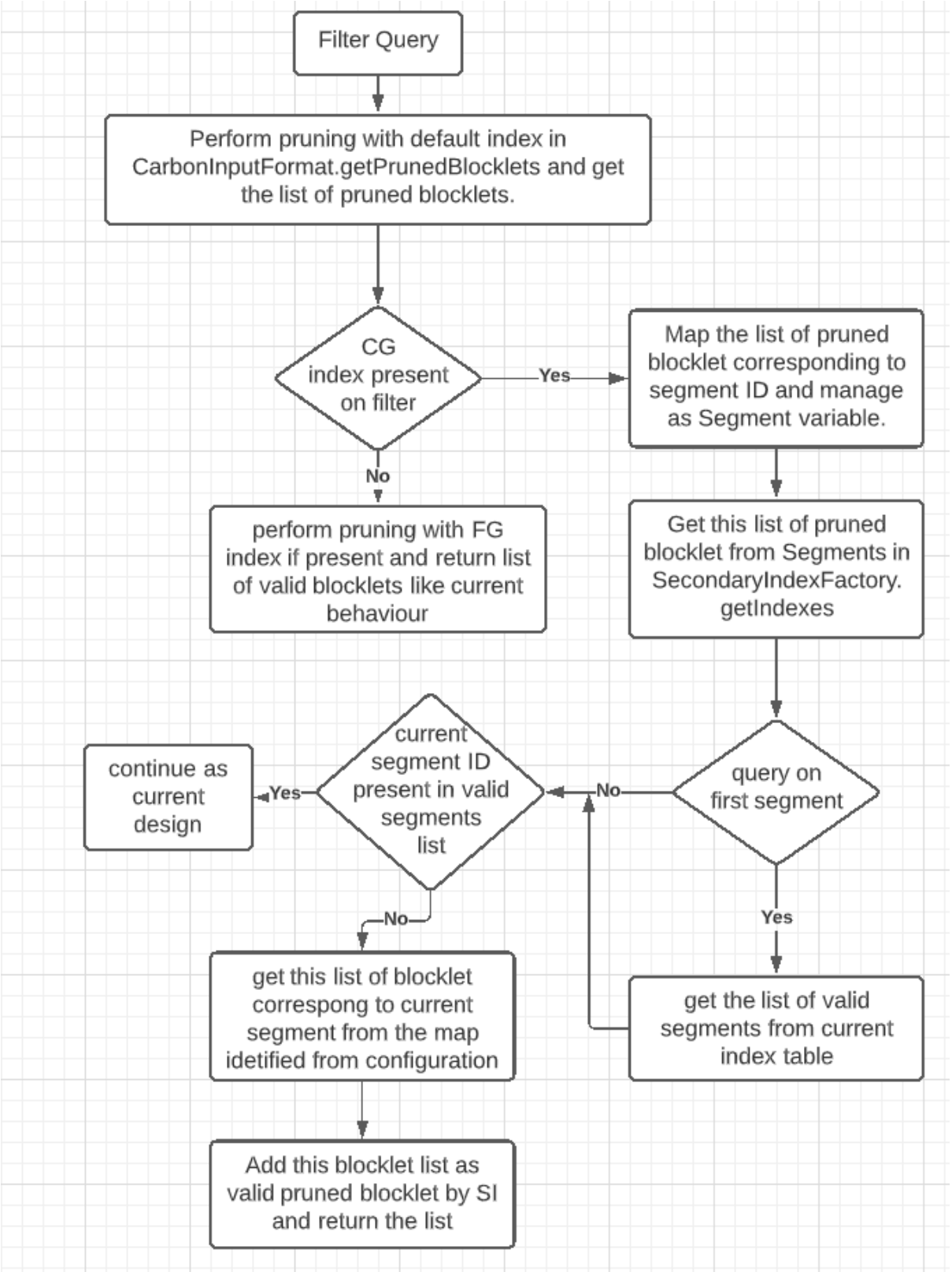
The above steps can be understand by the below diagram:



Leverage SI till segment level without spark plan rewrite

1. Don’t disable SI table when parent and child table segments are not in sync(make IndexStatus as ENABLED).
2. Perform pruning with default index in CarbonInputFormat.getPrunedBlocklets and get the pruned blocklet list (same as current behavior).
3. Iterate over the pruned blocklets and map it to the corresponding segment ID (Create a map with segment Id as key and list of blocklets as value).
4. Get the list of pruned blocklet corresponding to segment ID in SecondaryIndexFactory.getIndexes.
5. In SecondaryIndex.getPositionReferences if PositionReferenceInfo.fetched is false then get the list of all valid segments in current index table.
6. If currentSegmentId in SecondaryIndex.prune is not a valid segment (not present in the list identified in step 5) in SI table then return the list of blocklet corresponding to currentSegmentId from the identified map in step 4.
7. If currentSegmentId is valid SI segments then return the identified blocklets corresponding to currentSegmentId from SI table(same as existing design).

The above steps can be understood by the below diagram:


Filter Query
Perform pruning with default index in CarbonInputFormat.getPrunedBlocklets and get the list of pruned blocklets.
CG index present on filter
Yes
No
Map the list of pruned blocklet corresponding to segment ID and manage as Segment variable.
perform pruning with FG index if present and return list of valid blocklets like current behaviour
Get this list of pruned blocklet from Segments in SecondaryIndexFactory. getIndexes
query on first segment
No
Yes
current segment ID present in valid segments list
Yes
No
continue as current design
get the list of valid segments from current index table
get this list of blocklet correspong to current segment from the map idetified from configuration
Add this blocklet list as valid pruned blocklet by SI and return the list