Troubleshooting Secondary Indexes

Describes how to debug and troubleshoot usage of secondary indexes.

The following table lists problems you may encounter when using secondary indexes. Based on the symptoms listed in the first column, refer to the section in the third column to further troubleshoot the issue.

Symptom	Possible Cause	Troubleshooting Steps
Query performance is slow	Query is not using secondary indexes	1. Determining the Query Execution Path for OJAI Queries (DeterminingQueryExecutionPath.html#concept_sv1_dxd_pbb) 2. Determining Secondary Index Usage (DeterminingSIUsage.html#concept_zzy_yxd_pbb)
	Non-optimal OJAI query plan chosen	Examining the OJAI Query Plan (ExaminingOJAIQueryPlan.html#concept_nml_g12_pbb)
	Non-optimal query plan chosen by OJAI Distributed Query Service	Determining Index Use (../../Drill/evaluating_queries.html#evaluating_queries)
Inconsistent query results	Secondary index update lag	Identifying Secondary Index Lag (IdentifyingIndexLag.html#concept_gm3_jb2_pbb)
	Unresolved encoding errors	Troubleshooting Secondary Index Encoding Errors (IdentifyingResolvingSIEncodingErrors.html#task_dlc_1d2_pbb)
Query runs out of memory	Memory configuration in the OJAI Distributed Query Service set too low	Adjusting Memory Settings in the OJAI Distributed Query Service (AdjustMemoryOJAIQueryService.html#concept_qcf_wf2_pbb)

Secondary Index Restrictions

When troubleshooting secondary indexes, you should also keep in mind the following restrictions:

Type Restrictions

- If a composite index includes the same subfield in multiple indexed fields, the implied types of the subfields must be consistent.

  For example, you cannot create an index with the following indexed fields:

  ```
  a.b[].c, a.b.d
  ```

  Although subfield `b` appears in both indexed fields, in the first, it is an array and in the second, it is a nested document.

  See Composite Indexes and Container Field Paths (design-composite-index.html#reference_n2y_m1n_pz__section_yly_trq_jdb) for more details.

Size Restrictions

- The maximum size of all indexed fields in an index is 32 KB.

  If the collective size exceeds 32 KB, then an insert of the corresponding document results in an encoding error (`INDEX_ROW_KEY_ENCODER_ERROR_ENCODING_IS_TOO_LONG`).
- The maximum number of indexes that you can create on a JSON table is 32.

Field Definition Restrictions

- You cannot specify individual array elements as indexed fields.
- You cannot specify a table's `_id` field as an indexed field.
- If a field contains an array of nested documents and you want to index on subfields in the nested documents, then you must define the indexed field using a container field path.
- You can include a specific field only once as either an indexed or included field, with the following two exceptions:
  - The indexed field is a container field path:

    ```
    maprcli table index add -path /people \
       -index phoneNumberIdx \
       -indexedfields Phones[].Number \
       -includedfields Phones[].Number
    ```
  - The field specifies a cast to another type.

    You can create an index in which the `score` field is an indexed field cast as a `double` type, and `score` is also an included field. The included field retains the original data type of the `score` field:

    ```
    maprcli table index add -path /castTable \
       -index castIdx1 \
       -indexedfields '$CAST(score@DOUBLE)' \
       -includedFields score
    ```

    You can create an index in which the `score` field is an indexed field, cast as a `double` type, and the `score` field is also another indexed field, cast as a `long` type:

```
maprcli table index add -path /castTable \
   -index castIdx2 \
   -indexedfields '$CAST(score@DOUBLE)','$CAST(score@LONG)'
```

- You cannot use casts with included fields.
- You cannot specify a field as either an indexed or included field if the field is also specified as a column family JSON path name.

  For example, suppose you have the following JSON table:

  ```
  {
       "_id" : "ID",
       "a" :
            {
                 "b" :
                      {
                           "c" : "value",
                           "d" : "value"
                      },
                 "e" : "value"
            }
  }
  ```

  If you create a column family at field `c` in the JSON path `a.b.c`, you cannot define field `a.b.c` as either an indexed or included field. You can define the fields `a`, `a.b`, and `a.b.d` as either indexed or included fields.
- You cannot specify an included field in which the data in the field spans more than one column family.

  In the following example, the included field `sl1.sl2` spans column families, `cf2` and `cf3`:

  ```
  maprcli table cf list -path /cftab
  compressionperm  readperm  traverseperm  jsonfamilypath   writeperm  minversions  maxversions  compression  ttl         inmemory  cf
  name   memoryperm
  u:root           u:root    u:root                         u:root     0            1            lz4          2147483647  false     de
  fault  u:root
  u:root           u:root    u:root        sl1              u:root     0            1            lz4          2147483647  false     cf
  1      u:root
  u:root           u:root    u:root        sl1.sl2.sl3      u:root     0            1            lz4          2147483647  false     cf
  2      u:root
  u:root           u:root    u:root        sl1.sl2.sl3.sl4  u:root     0            1            lz4          2147483647  false     cf
  3      u:root

  maprcli table index add -path /cftab -index i1 -indexedfields sl1.sl2.sl3.sl4.l4a, sl1.l1a -includedfields sl1.sl2,sl1.sl2.sl3.sl4.s
  l5.l5b -json
  {
          "timestamp":1507419777919,
          "timeofday":"2017-10-07 04:42:57.919 GMT-0700 PM",
          "status":"ERROR",
          "errors":[
                  {
                          "id":22,
                          "desc":"Data for included field sl1.sl2 may not span more than one column family."
                  }
           ]
  }
  ```

- You cannot specify a composite index with more than one container field path as your indexed fields, unless the prefixes of the container field paths are the same.

  See Composite Indexes and Container Field Paths (design-composite-index.html#reference_n2y_m1n_pz__section_yly_trq_jdb) for more details.
- You cannot specify a composite index with an indexed field that is a subfield of another indexed field.

  For example, you cannot create an index with the following indexed fields:

  ```
  a, a.b
  ```

  The indexed field `a.b` is a subfield of the indexed field `a`.

Option Restrictions

- Because indexes are automatically split, you cannot disable splits when you create your index.

Index Use Restrictions

- Indexes do not optimize non-existence filter conditions.