AEM Query Troubleshooting

Thomas Mueller | Best Practices

Bē
Evolution Bureau

About Me

- Thomas Mueller
- Jackrabbit and Oak developer
- Query engine and indexing
- Previously, wrote Java SQL database engines Hypersonic SQL and H2

See Also

- AEM Indexing and JCR Query (2017)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-indexing-jcr-query.html

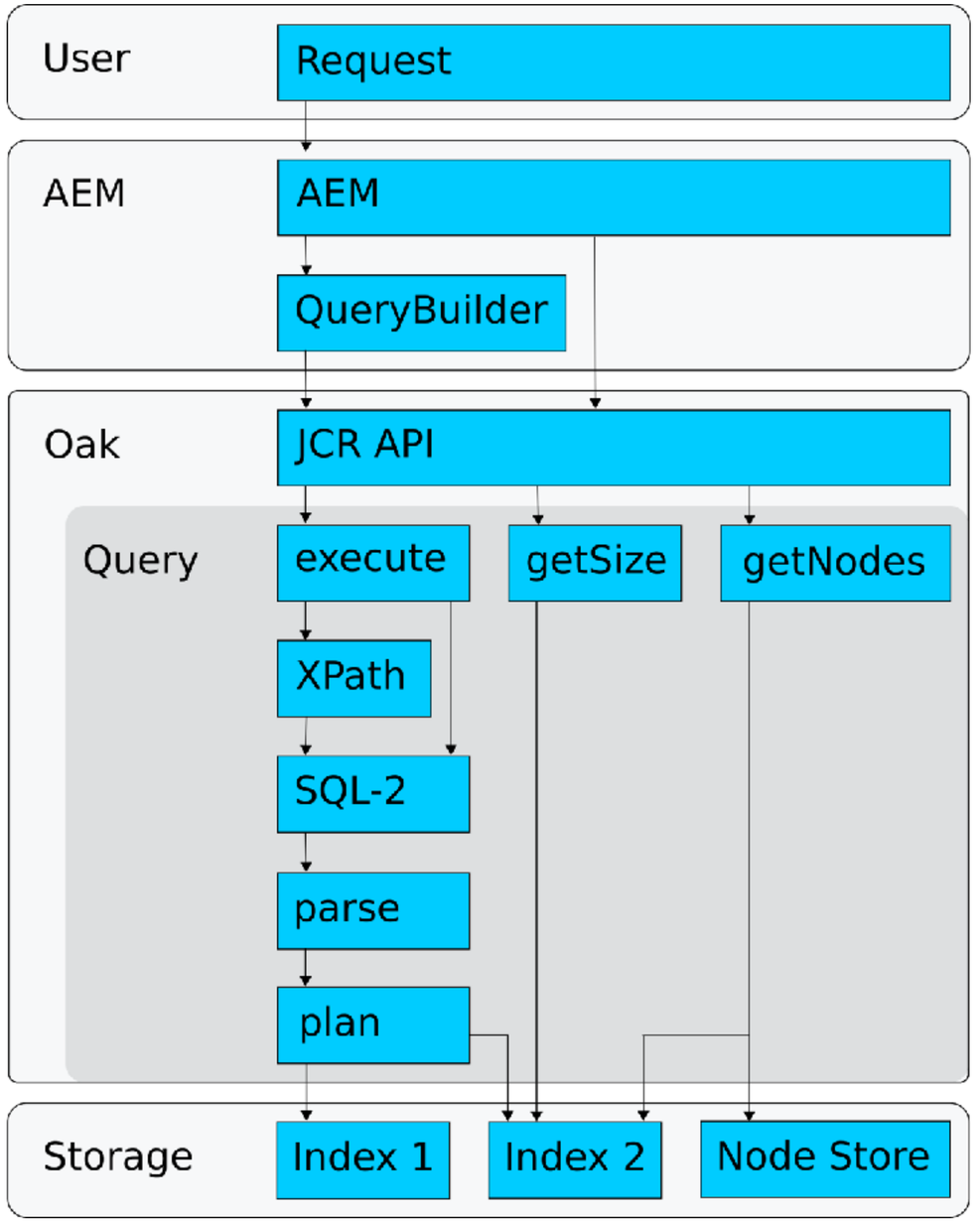
Agenda

- **Query Processing**
  - Syntax, Parsing, Preparation
  - Cost Estimation, Warnings
  - Result Size, Iteration
- **Indexing and Reindexing**
  - Analyzer, Text Extraction
- **AEM 6.4 UI Changes**
- **Error Reporting**

Adobe

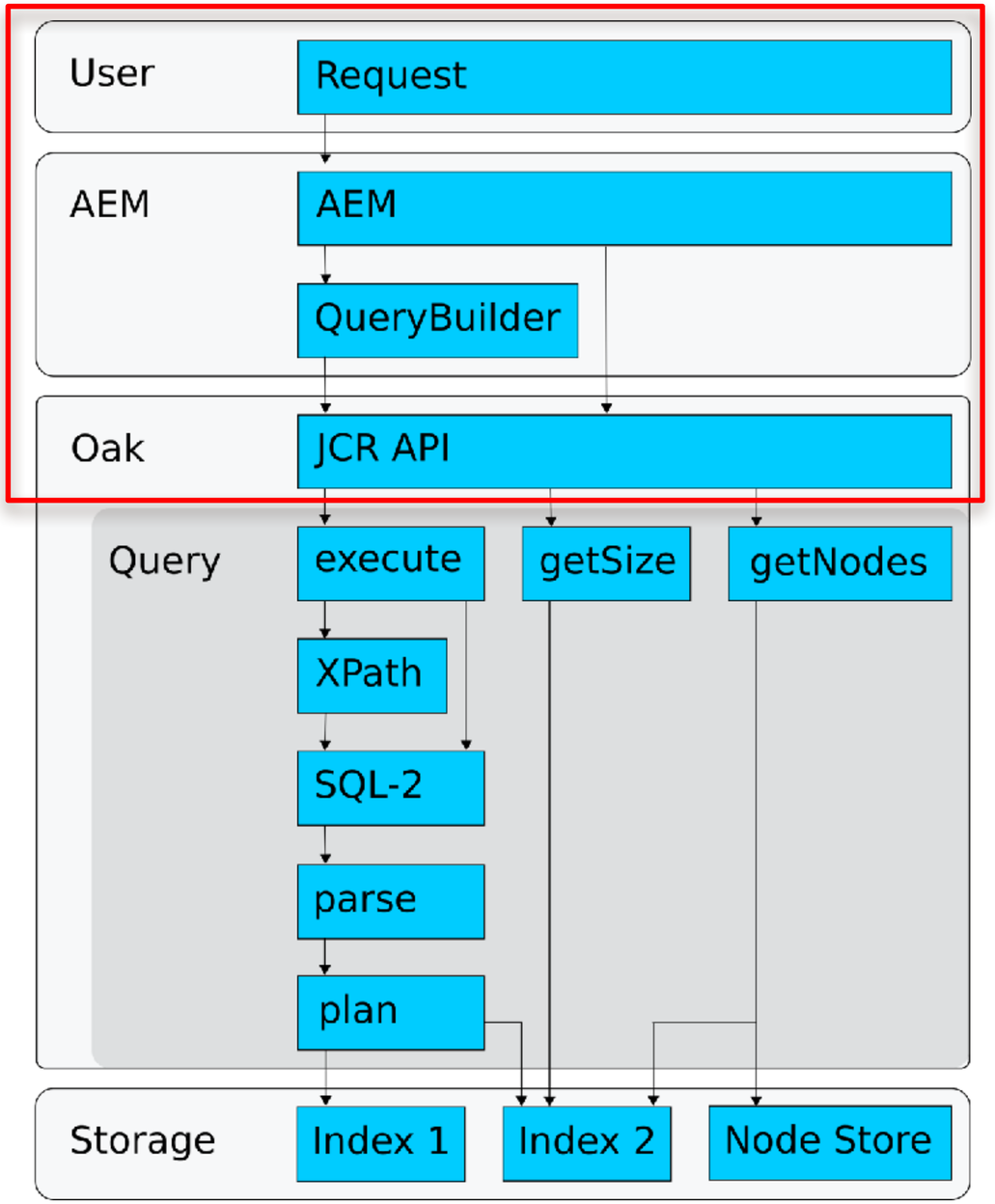
Query Processing

- We go into query processing details here
- Right: Overview
- Left: Details of this step



Query Generation

- All queries internally use the JCR API (including the QueryBuilder)

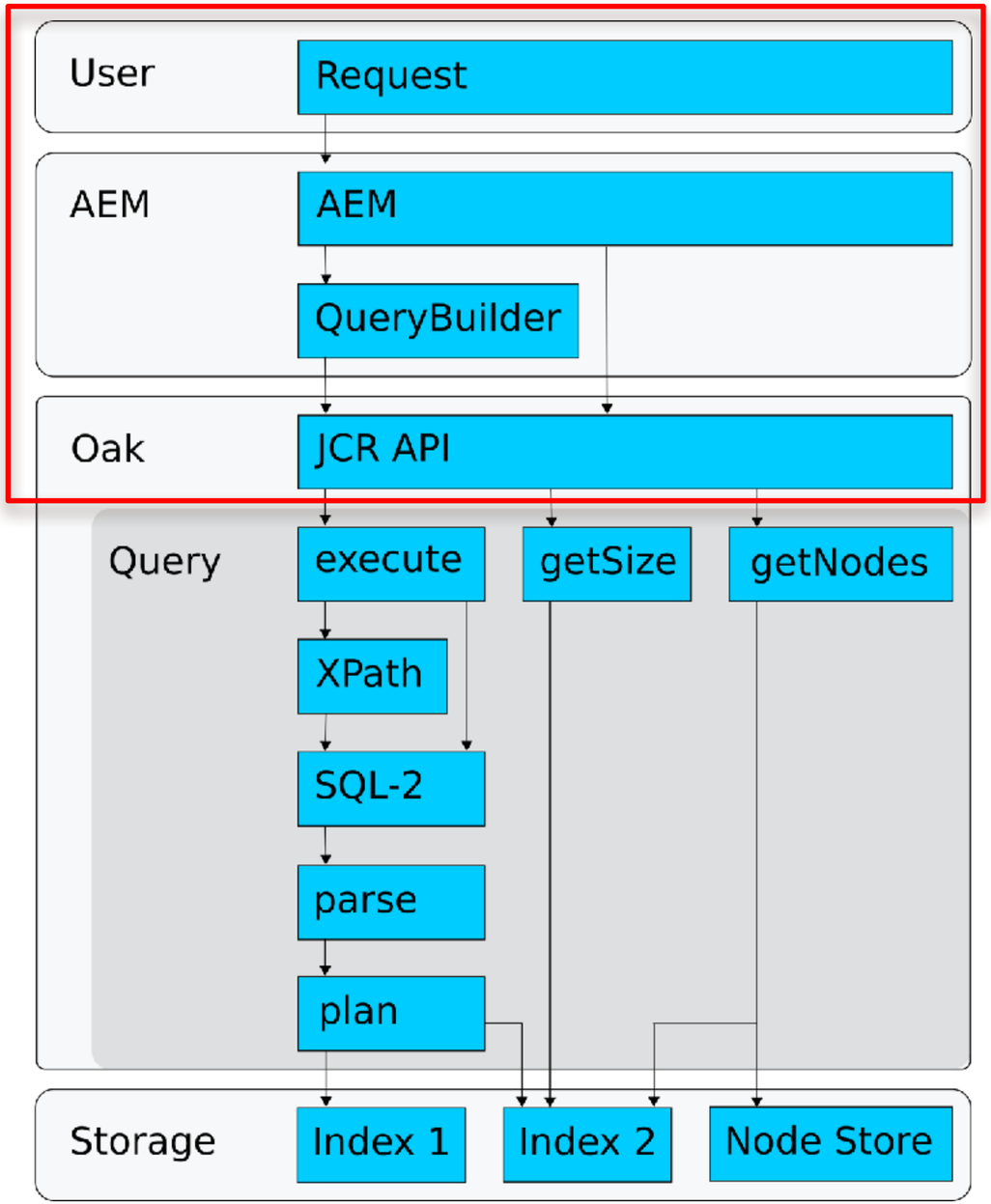


Adobe

Query Builder

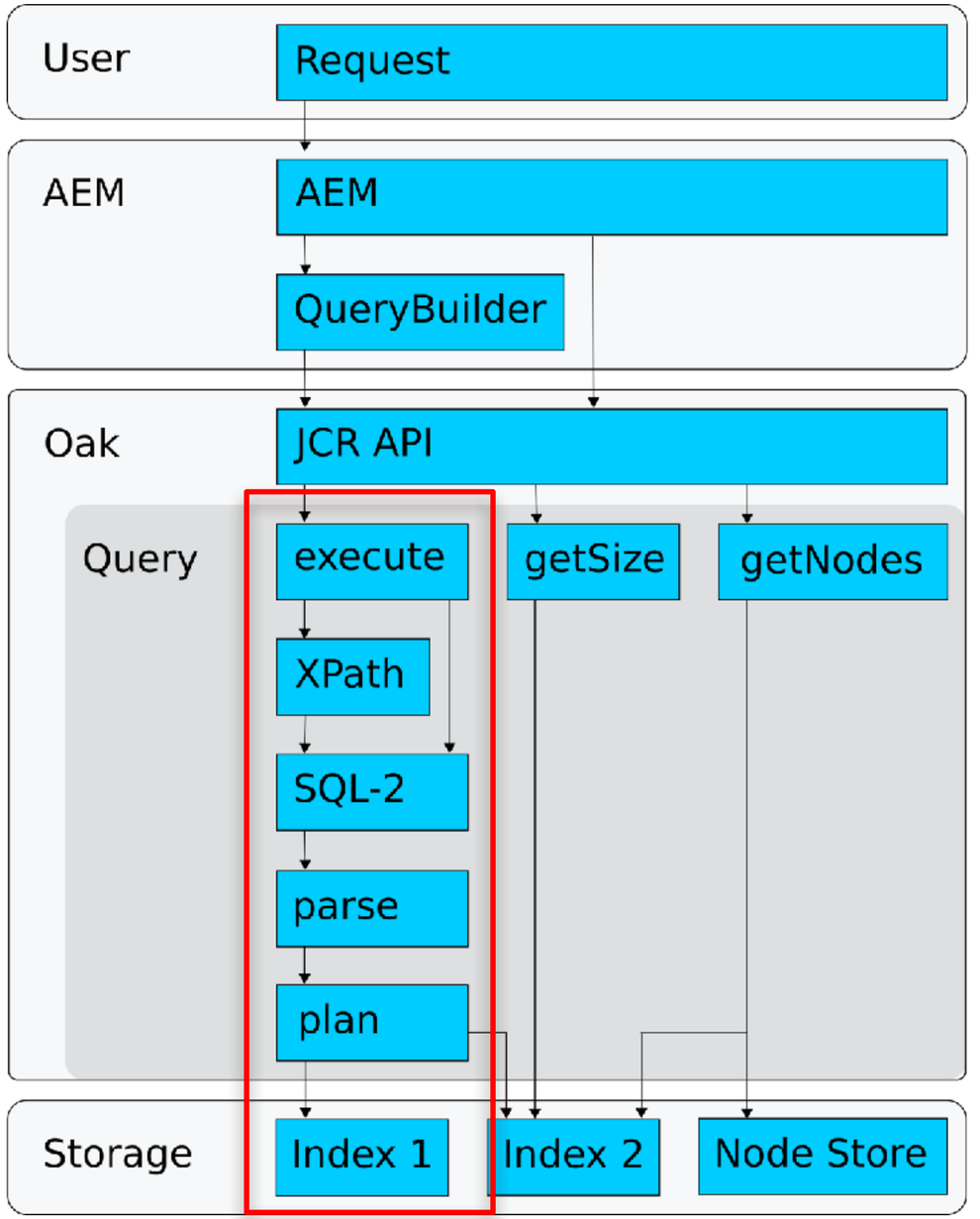
- Query Builder documentation
  https://helpx.adobe.com/experience-manager/6-4/sites/developing/using/querybuilder-api.html
- Gems session
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-search-forms-using-querybuilder.html

```
path=/content
1_property=sling:resourceType
1_property.value=foundation/components/text
1_property.operation=like
orderby=path
```



Query Parsing

- Query.execute does not read nodes (except for "order by" unindexed) - so it is usually very fast
- However, a "query plan" is built, and for this, indexes are consulted to estimate the cost



XPath is Preferred

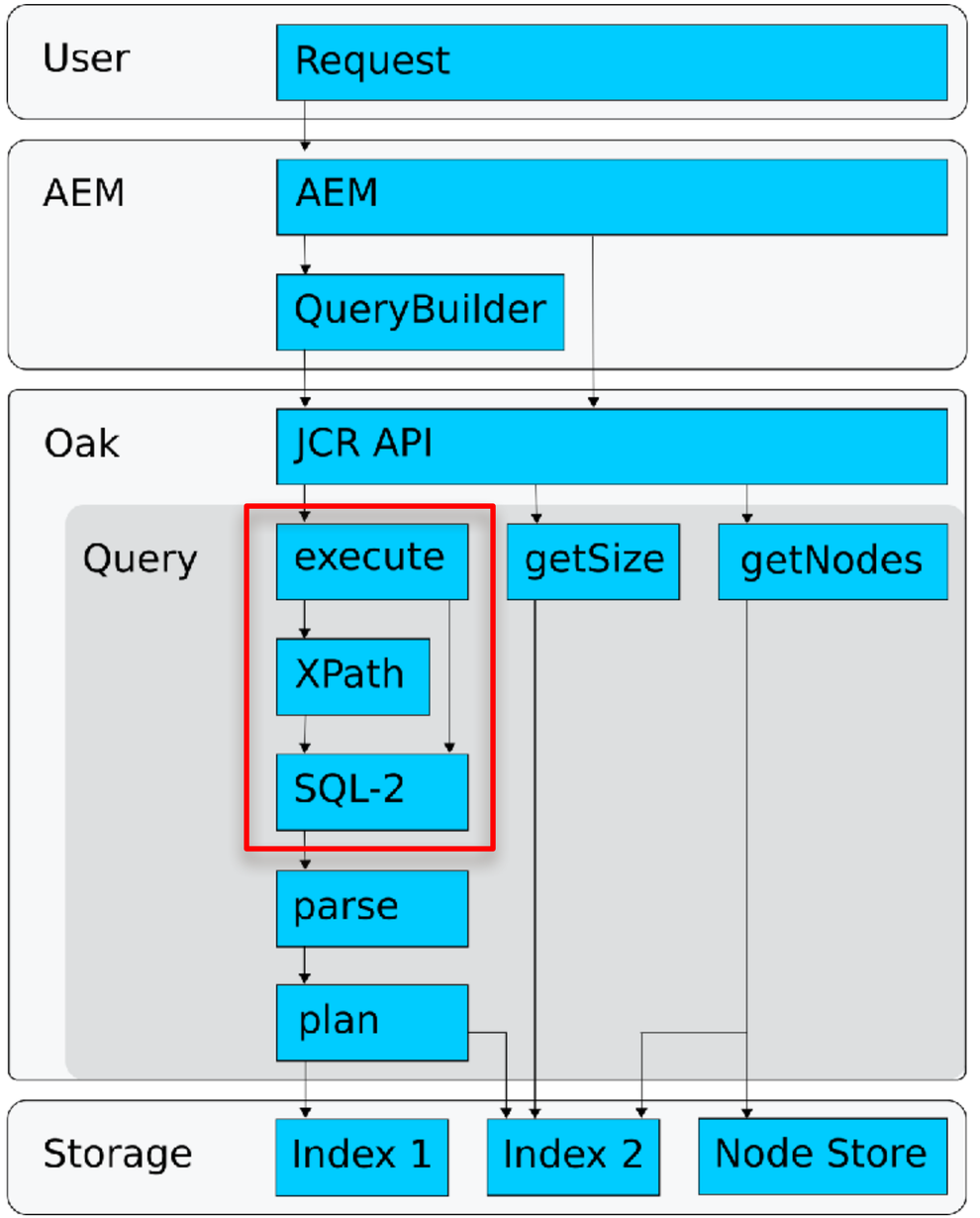
- XPath preferred over SQL-2; *not* deprecated in Oak

XPath:

```
/jcr:root/home//*[@firstName = 'x']/profile
```

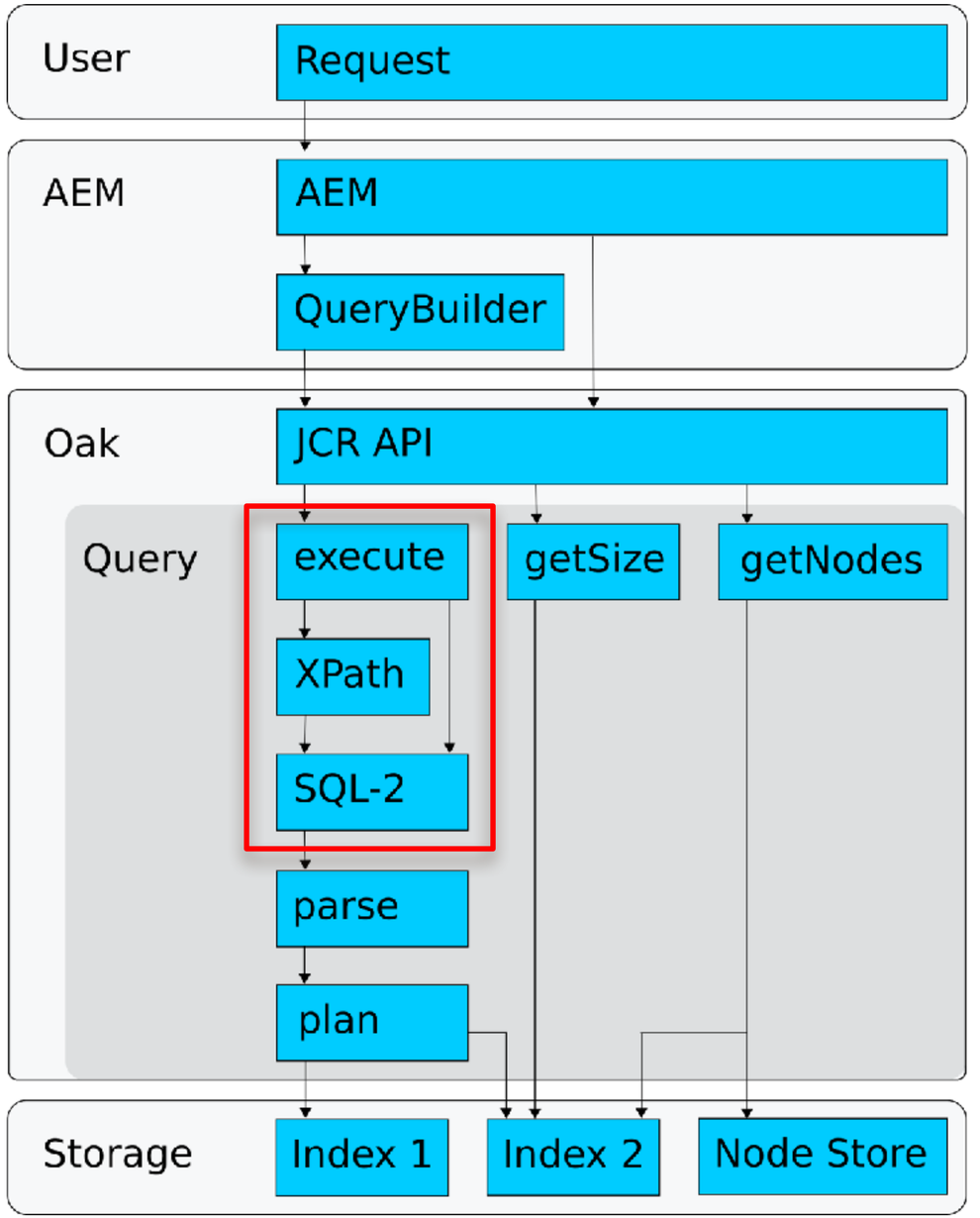
SQL-2:

```
select b.[jcr:path], b.[jcr:score], b.*
    from [nt:base] as a
    inner join [nt:base] as b on ischildnode(b, a)
    where a.[firstName] = 'x'
    and isdescendantnode(a, '/home')
    and name(b) = 'profile'
```



Query Conversion

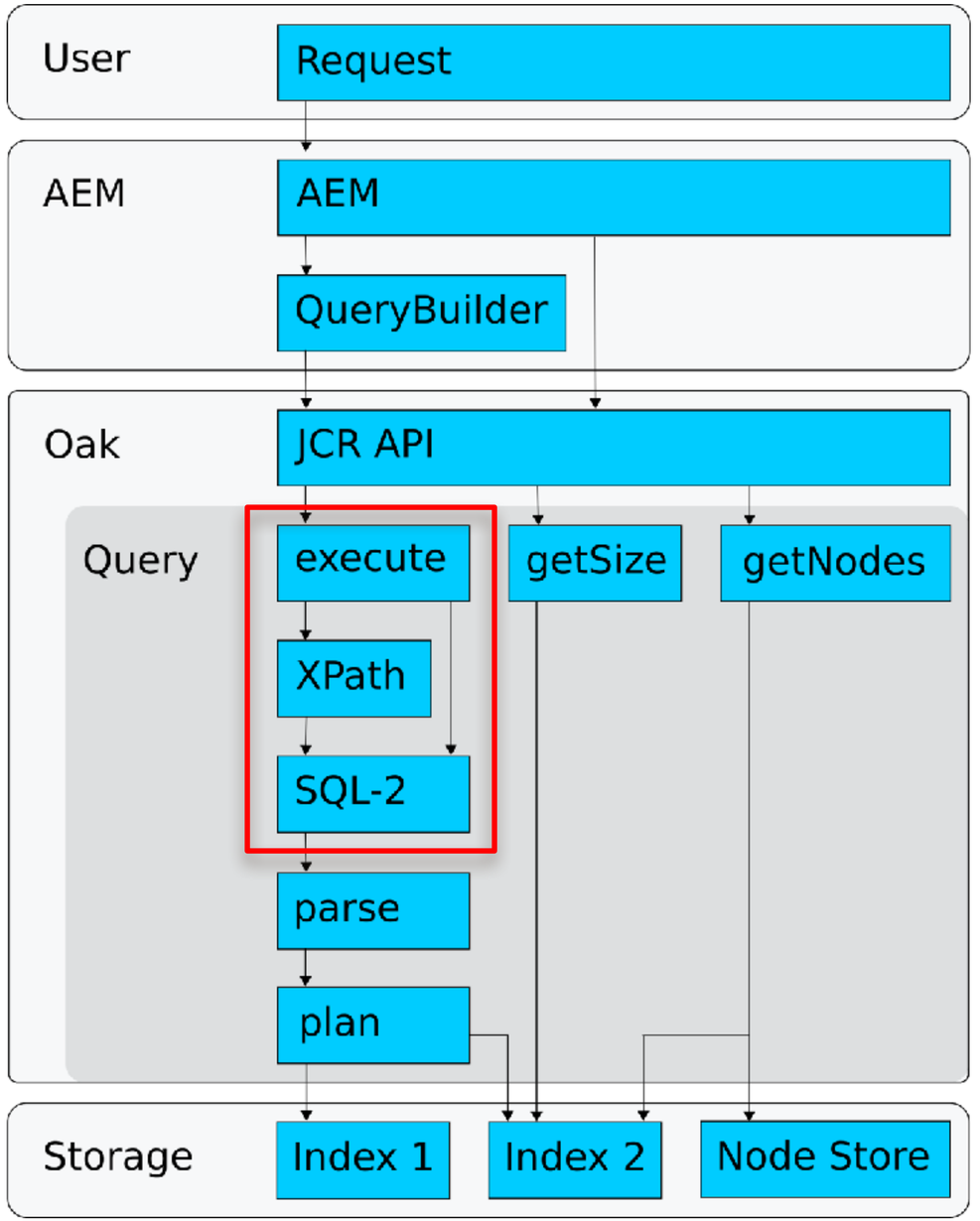
- XPath queries are converted to SQL-2
- While converting,
  - "@a=1 or @a=2" is converted to "a in(1, 2)"
  - "@a=1 or @b=1" => "a=1 ... union ... b=1"
  - "@primaryKey='x'" => "from [x]"
  - "jcr:root//*[...]/b" => "join"
  - "... | ..." => "union"
  - "order by @jcr:score desc" removed



Syntax Docs

- XPath, SQL-2 Railroad Diagrams

  http://jackrabbit.apache.org/oak/docs/query/grammar-xpath.html
  http://jackrabbit.apache.org/oak/docs/query/grammar-sql2.html

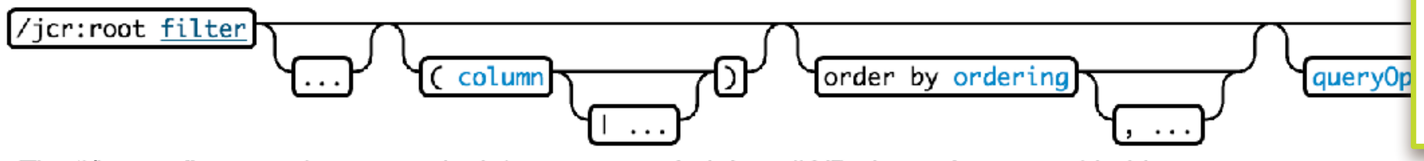


Syntax and Features Docs

Oak XPath Query Grammar

- Query
- Filter
- Column
- Constraint
- And Condition
- Condition
- Comparison
- Static Operand
- Ordering
- Dynamic Operand
- Options
- Explain
- Measure

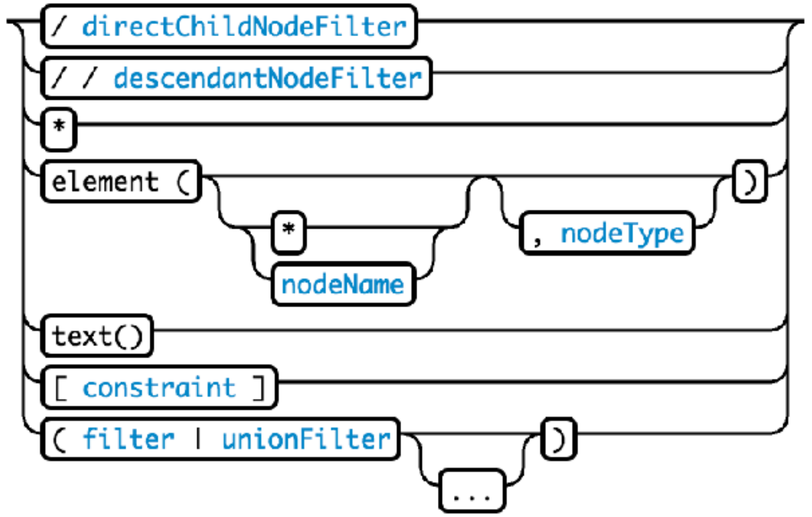
Query



The “/jcr:root” means the root node. It is recommended that all XPath queries start with this term.

All queries should have a path restriction (even if it’s just, for example, “/content”), as this allows to shrink indexes.

Filter



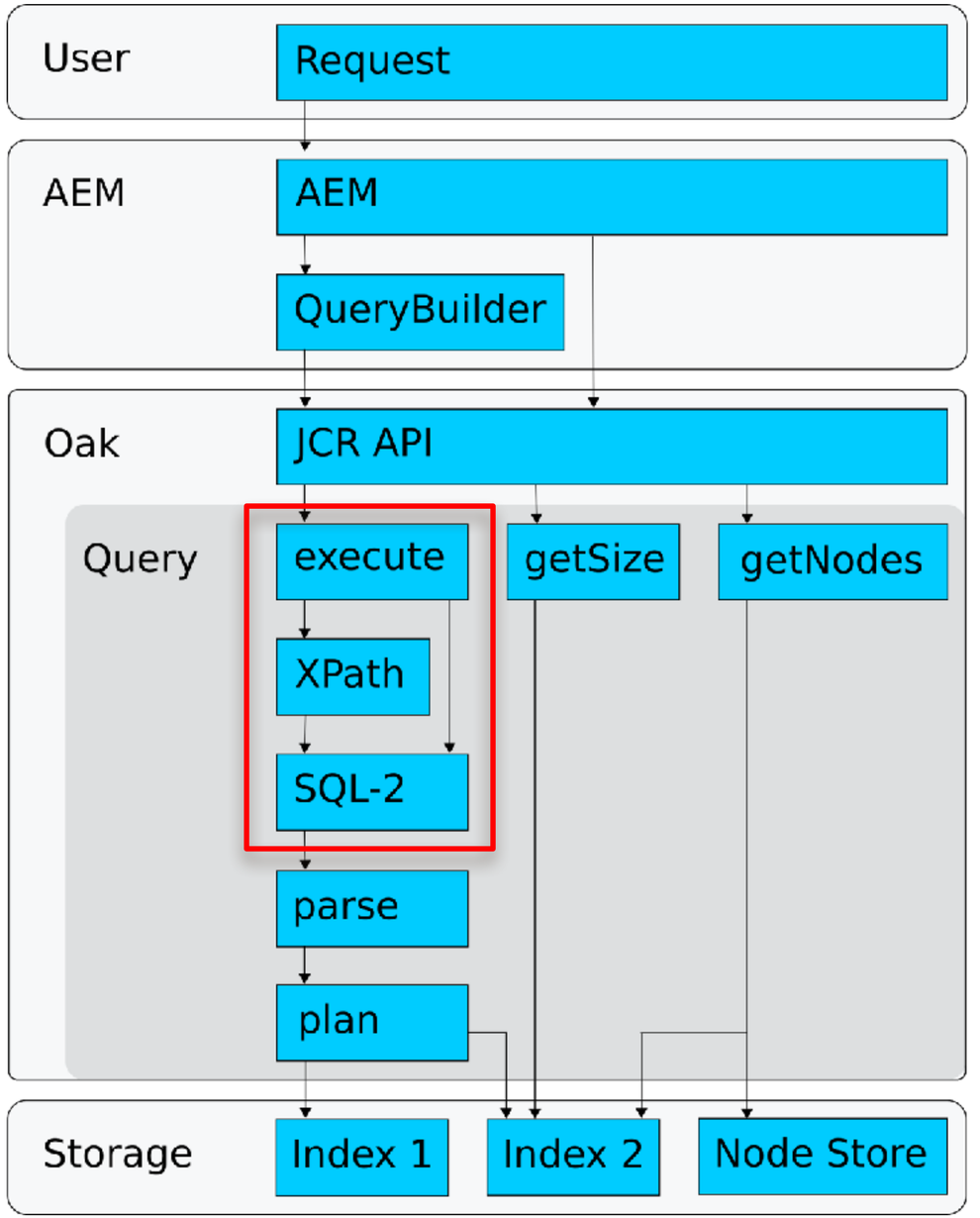
A single slash means filtering on a specific child node, while two slashes means filtering on a descendant node.

“*” means any node name, and any node type.

Adobe

New Syntax

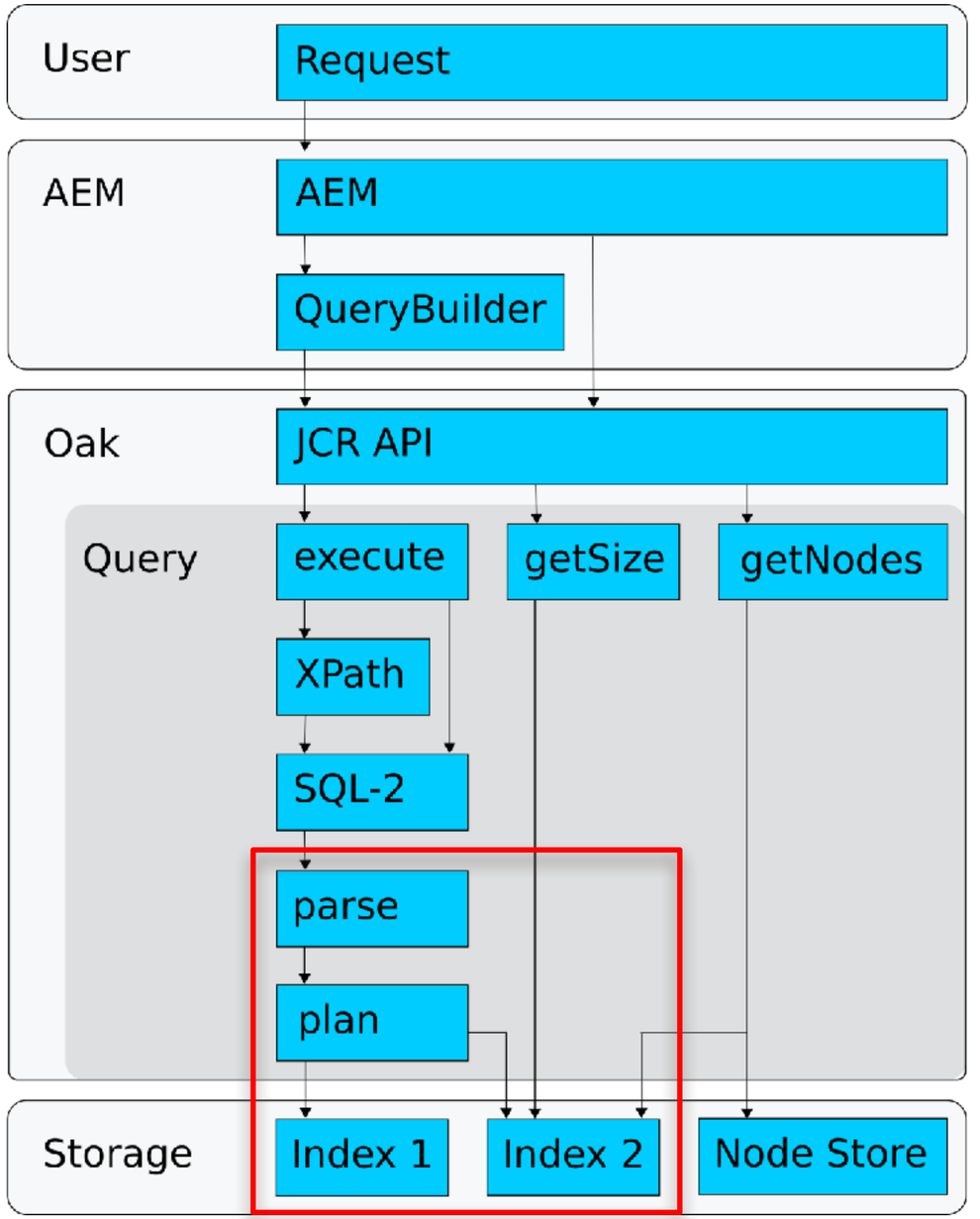
- New in Oak 1.8 / AEM 6.4:
  - **`fn:coalesce(@a, @b)`**
  - **`query option(index tag x)`**
  - index-based excerpts on properties (OAK-7151)



Adobe

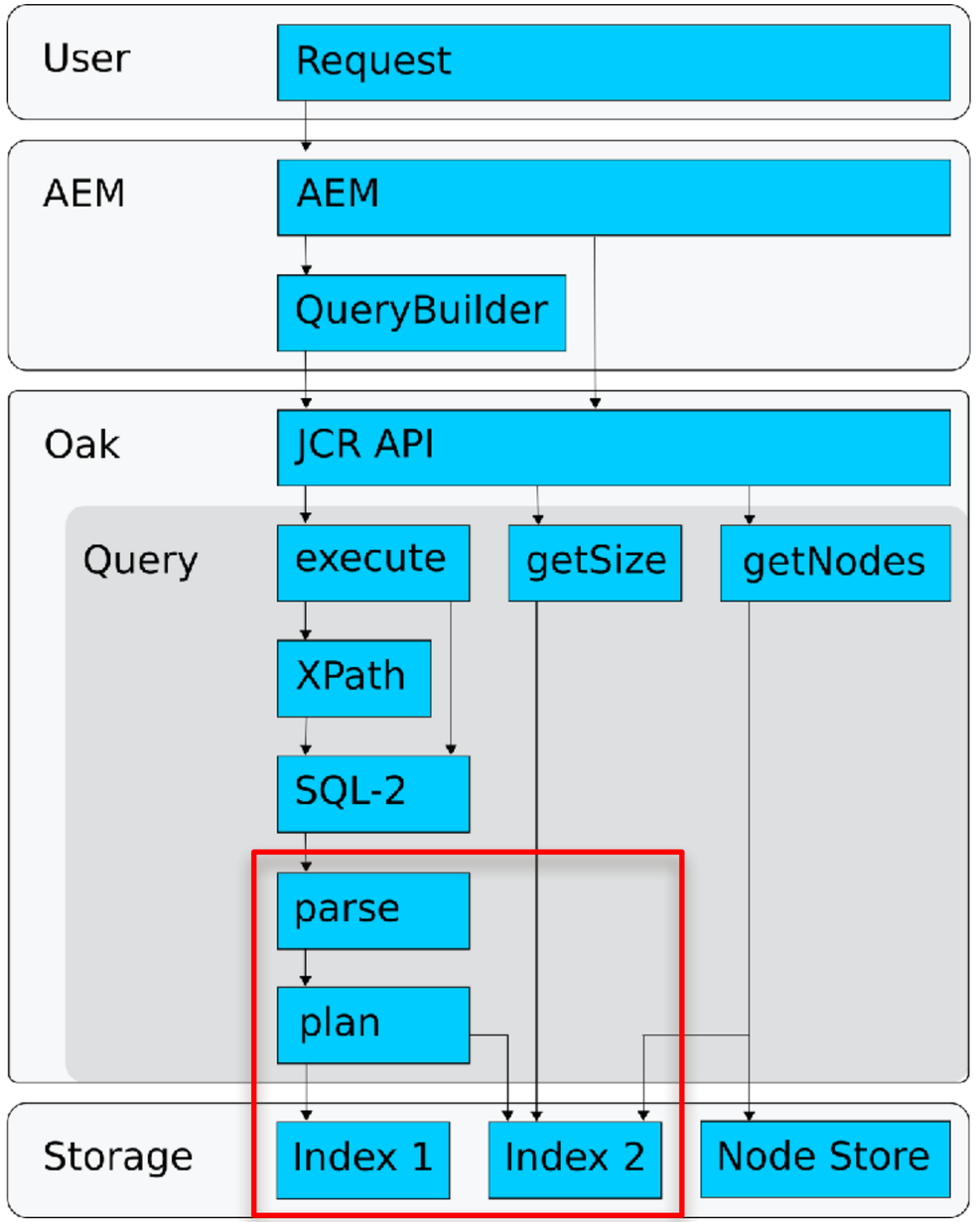
Query Preparation

- After parsing, possibly multiple plans are evaluated
  - "[a] inner join [b]" checks if [a] or [b] should be processed first, depending on the expected cost
  - when joining many, a greedy algorithm is used
  - for "a=1 or b=2", the cost is compared to converting to "union"
  - for each case, indexes estimate the cost



Cost Estimation

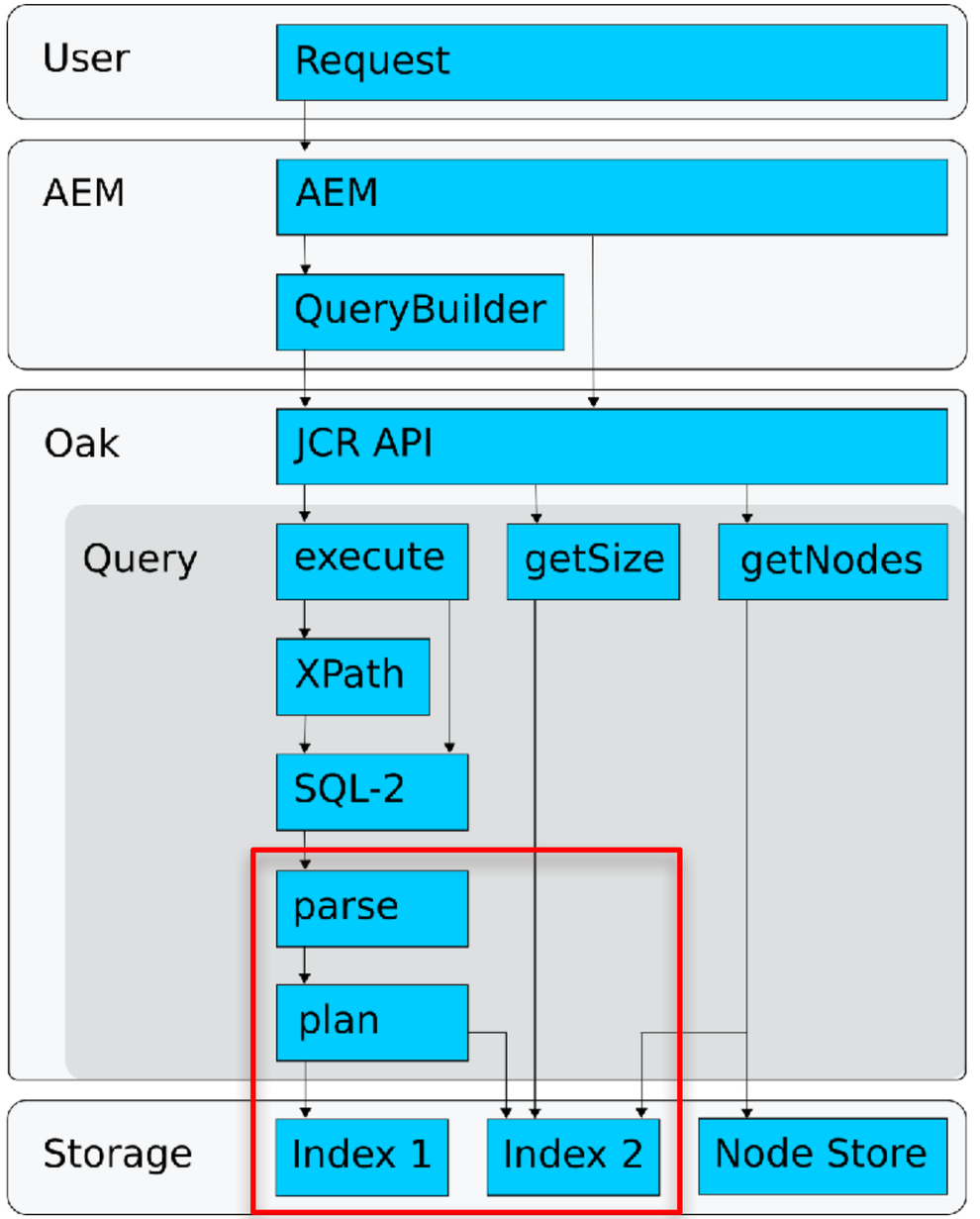
- Indexes estimate the cost
  - Join: 1 index per selector
  - Union: 1 index per subquery
  - Each index sees a "flat" query
  - Indexes may only apply some conditions; may sort; may evaluate path restrictions
  - Estimated node count is taken into account
  - Cost overrides may be used



Recommendations

- Always use a path restriction
- Avoid OR / UNION queries (ordering by score)

http://jackrabbit.apache.org/oak/docs/query/query-troubleshooting



Adobe

Query Planning

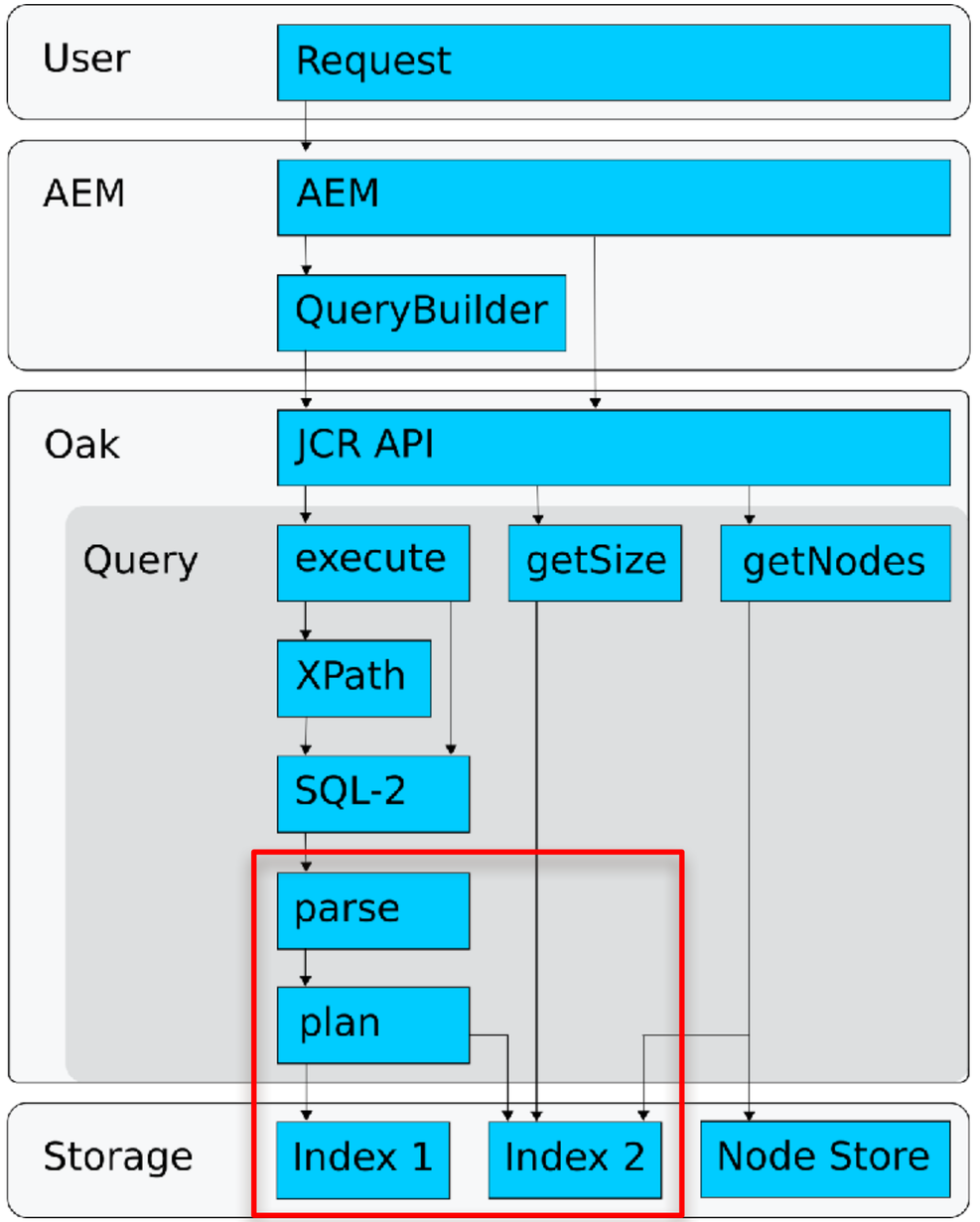
- The best index(es) are picked
  - The query plan shows index(es) used

Language *

XPath

Query *

```
/jcr:root/content/we-retail/us/en//element(*, cq:Page)
[(jcr:contains(., 'abc') or jcr:contains(@jcr:title, 'abc')
or jcr:contains(@jcr:description, 'abc'))]/rep:excerpt(.)
```



Adobe

Query Planning

- Possibly multiple indexes

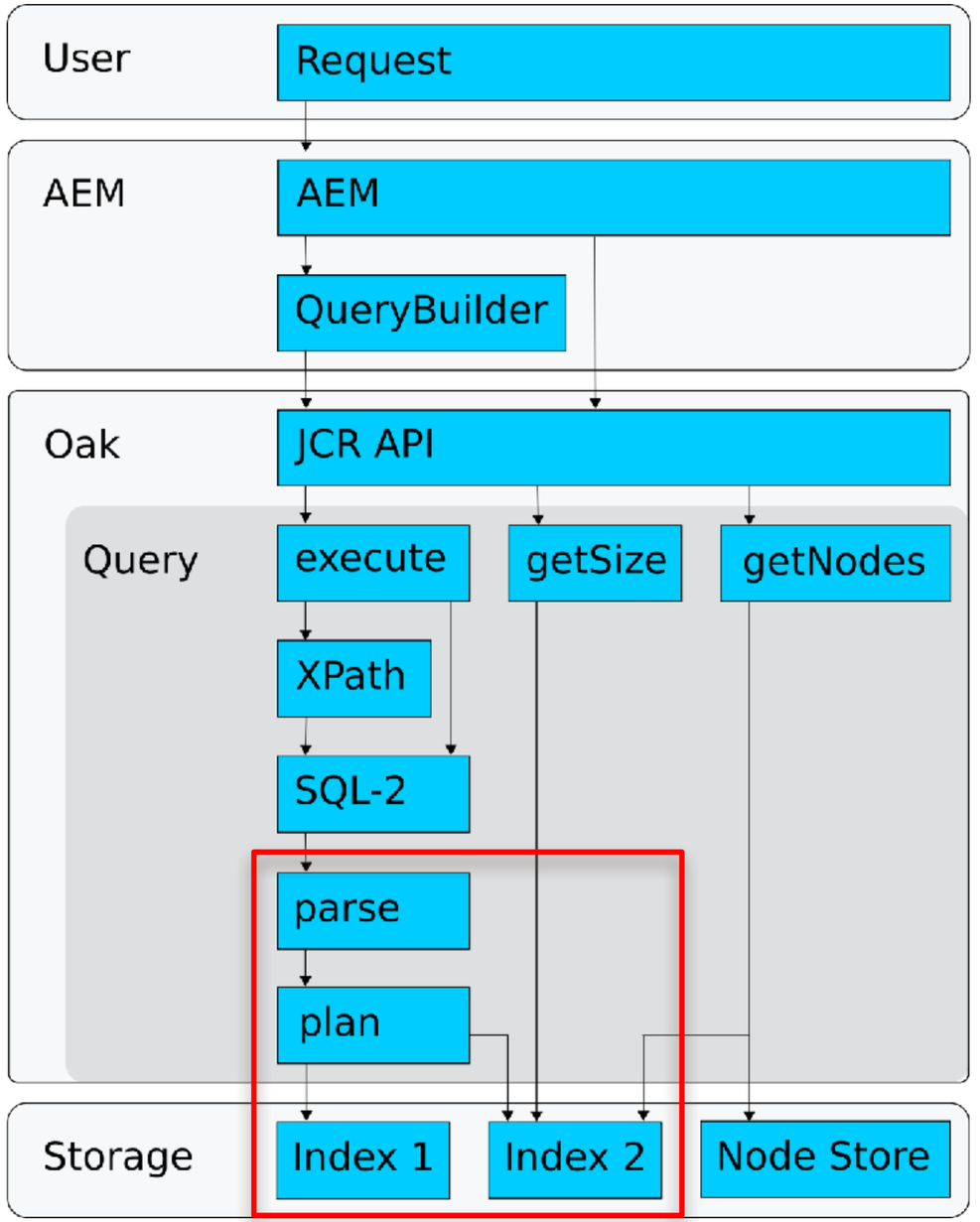
```
/jcr:root/content/we-retail/us/en//element(*, cq:Page)
[(jcr:contains(., 'abc') or jcr:contains(@jcr:title, 'abc')
or jcr:contains(@jcr:description, 'abc'))]/rep:excerpt(.)
```

Indexes Used

```
cqPageLucene(/oak:index/cqPageLucene)
lucene(/oak:index/lucene)
lucene(/oak:index/lucene)
```

Execution Plan

```
[cq:Page] as [a] /* lucene:cqPageLucene(/oak:index/cqPageLucene) +:fulltext:abc +:ancestors:/content/we-retail/us/en ft:("abc") where (isdescendantnode([a], [/content/we-retail/us/en])) and (contains([a].[*], 'abc')) */ union [cq:Page] as [a] /* lucene:lucene(/oak:index/lucene) full:jcr:title:abc ft:(jcr:title:"abc") where (isdescendantnode([a], [/content/we-retail/us/en])) and (contains([a].[jcr:title], 'abc')) */ union [cq:Page] as [a] /* lucene:lucene(/oak:index/lucene) full:jcr:description:abc ft:(jcr:description:"abc") where (isdescendantnode([a], [/content/we-retail/us/en])) and (contains([a].[jcr:description], 'abc')) */
```



Adobe

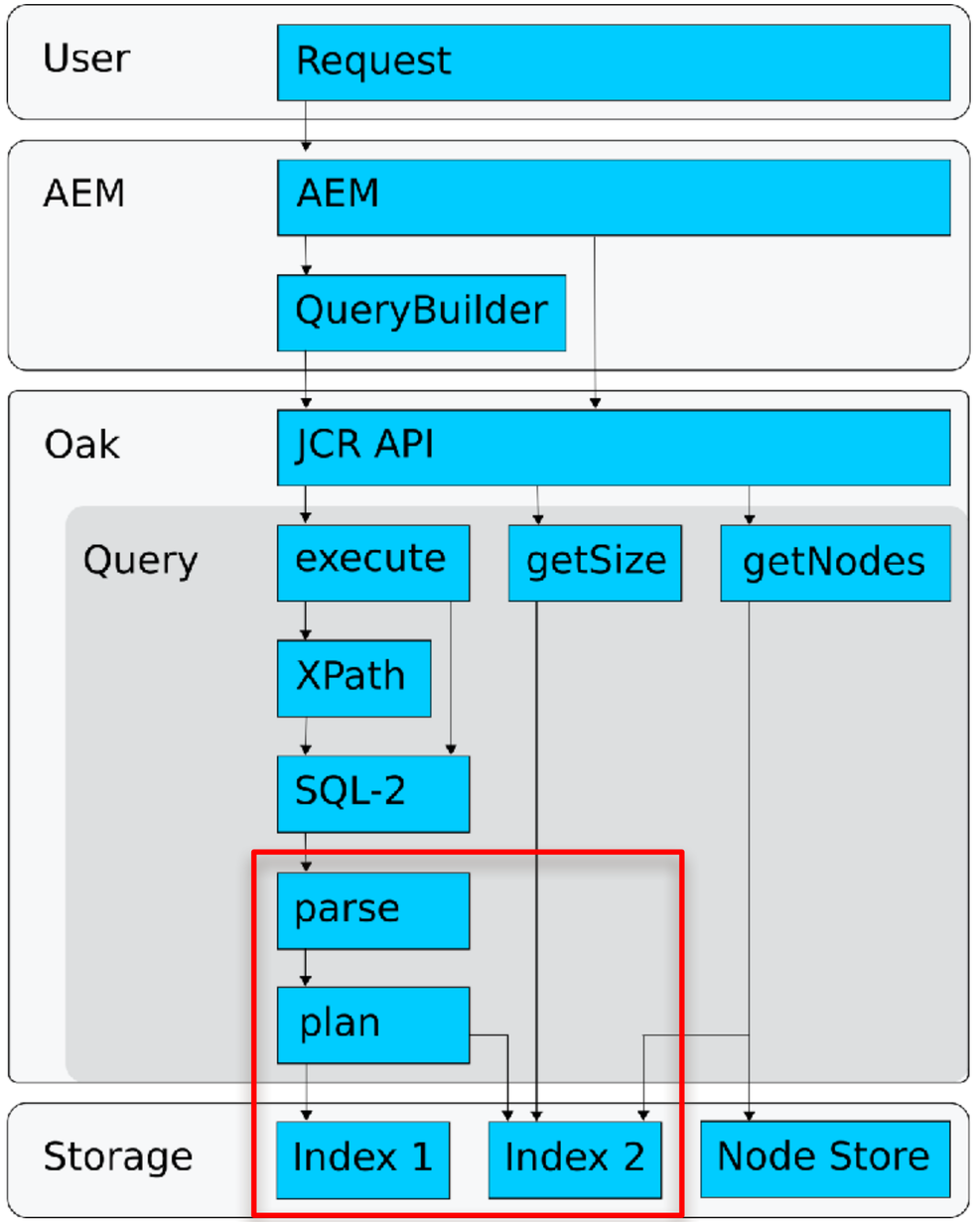
Index Tags

- Index Tags allow to filter indexes

```
/jcr:root/content//element(*, nt:file)
[jcr:contains(., 'test')]
option(index tag b)
```

```
/oak:index/cqPageLucene/tags = b

/oak:index/acmePageLucene/tags = [b, c]
```



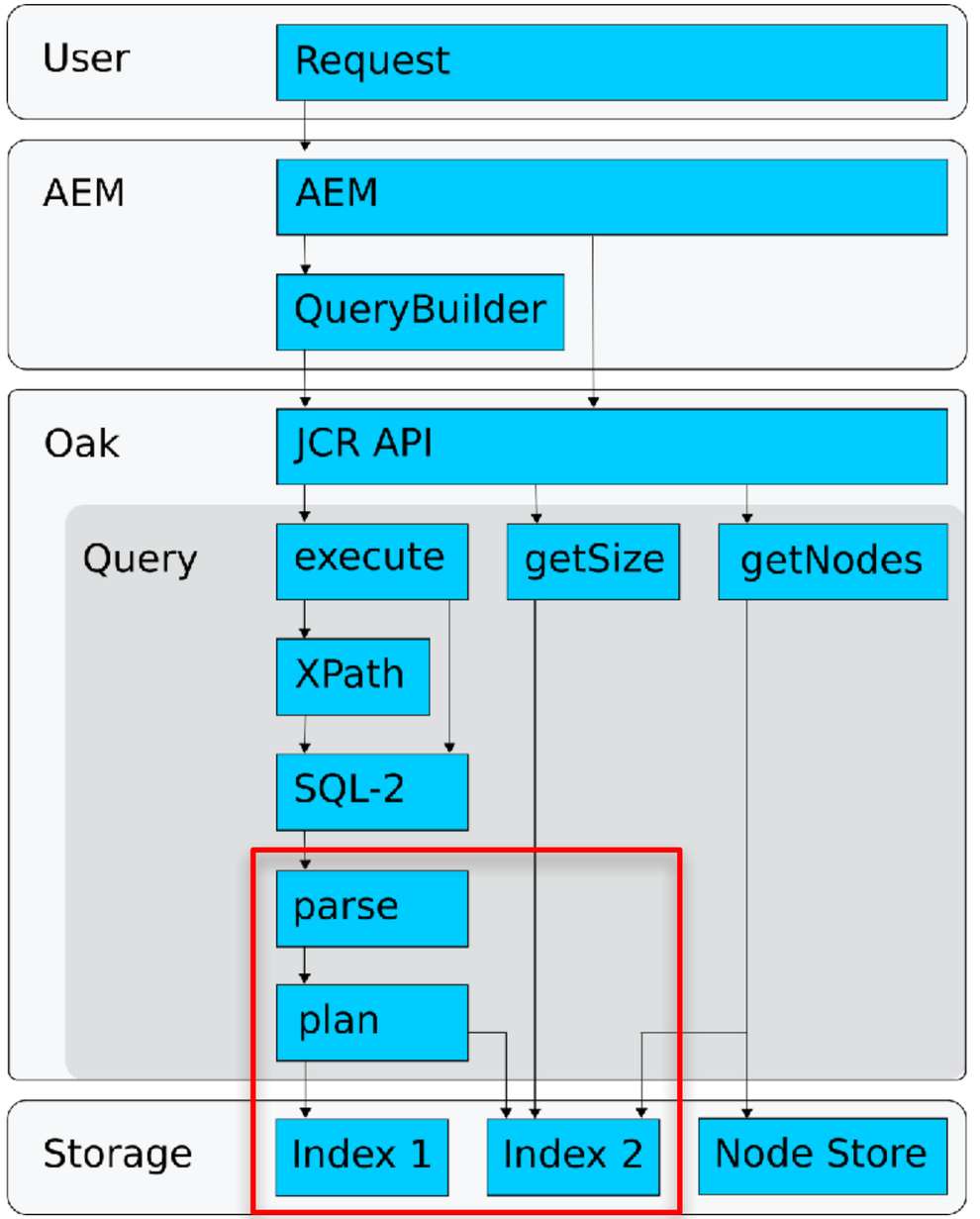
Traversal Warning

- Warning in the log

```
*WARN*
Traversal query (query without index):
...; consider creating an index
```

- If known to have few nodes:

```
/jcr:root/content/abc/def//*
option(traversal ok)
```



Prefer "Contains" Over "Like"

CONTAINS

- Case insensitive
- Uses the fulltext index

```
jcr:contains(., 'hello')
```

- Syntax: Apache Lucene
  https://wiki.apache.org/jackrabbit/EncodingAndEscaping
- Filter special characters

LIKE

- Case sensitive
- Index used: `jcr:like(@n, 'x%')`
- No index: `jcr:like(@x, '%x%')`
- Syntax:

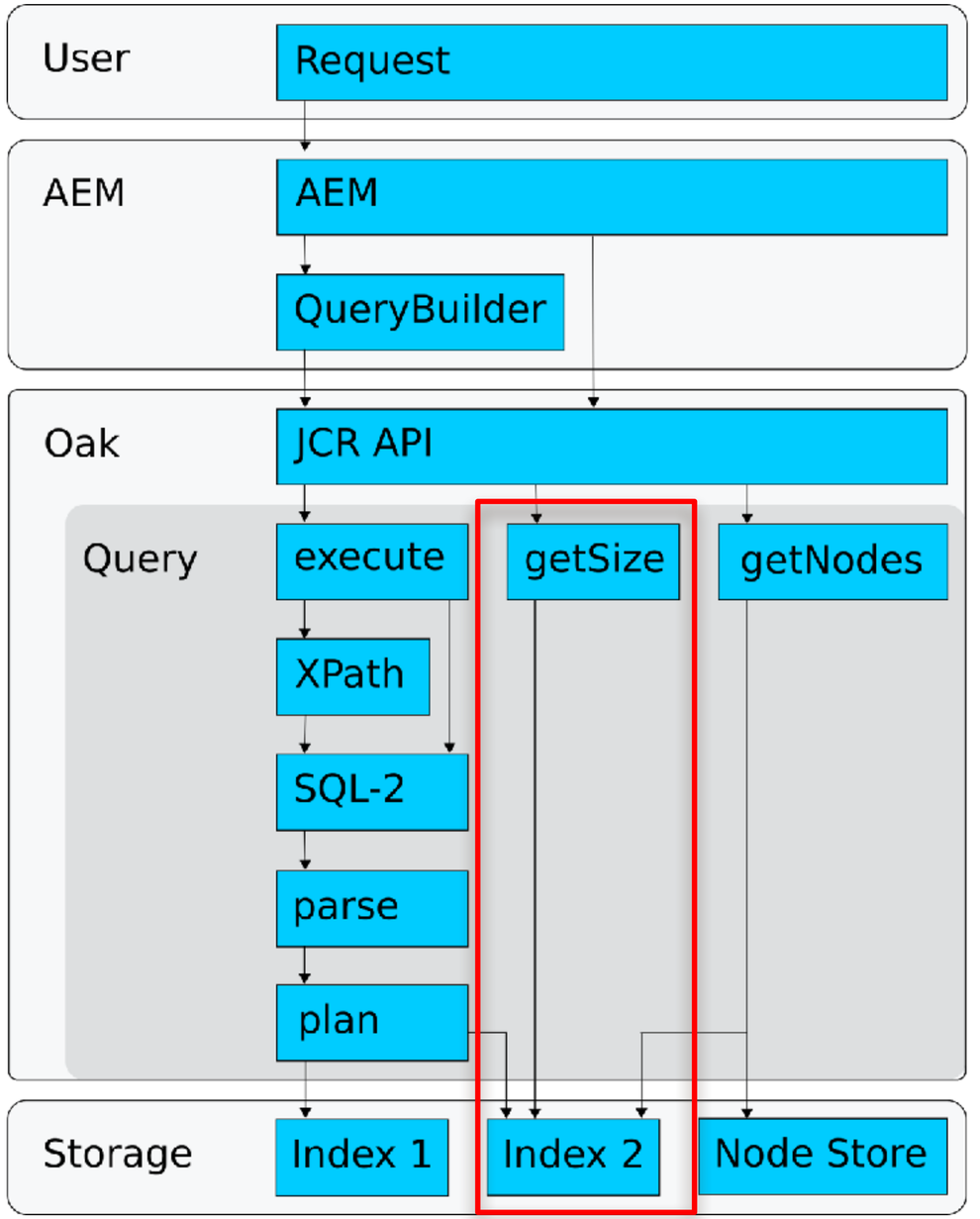
  any one character: `%`

  any characters: `_`

  escape: `\`

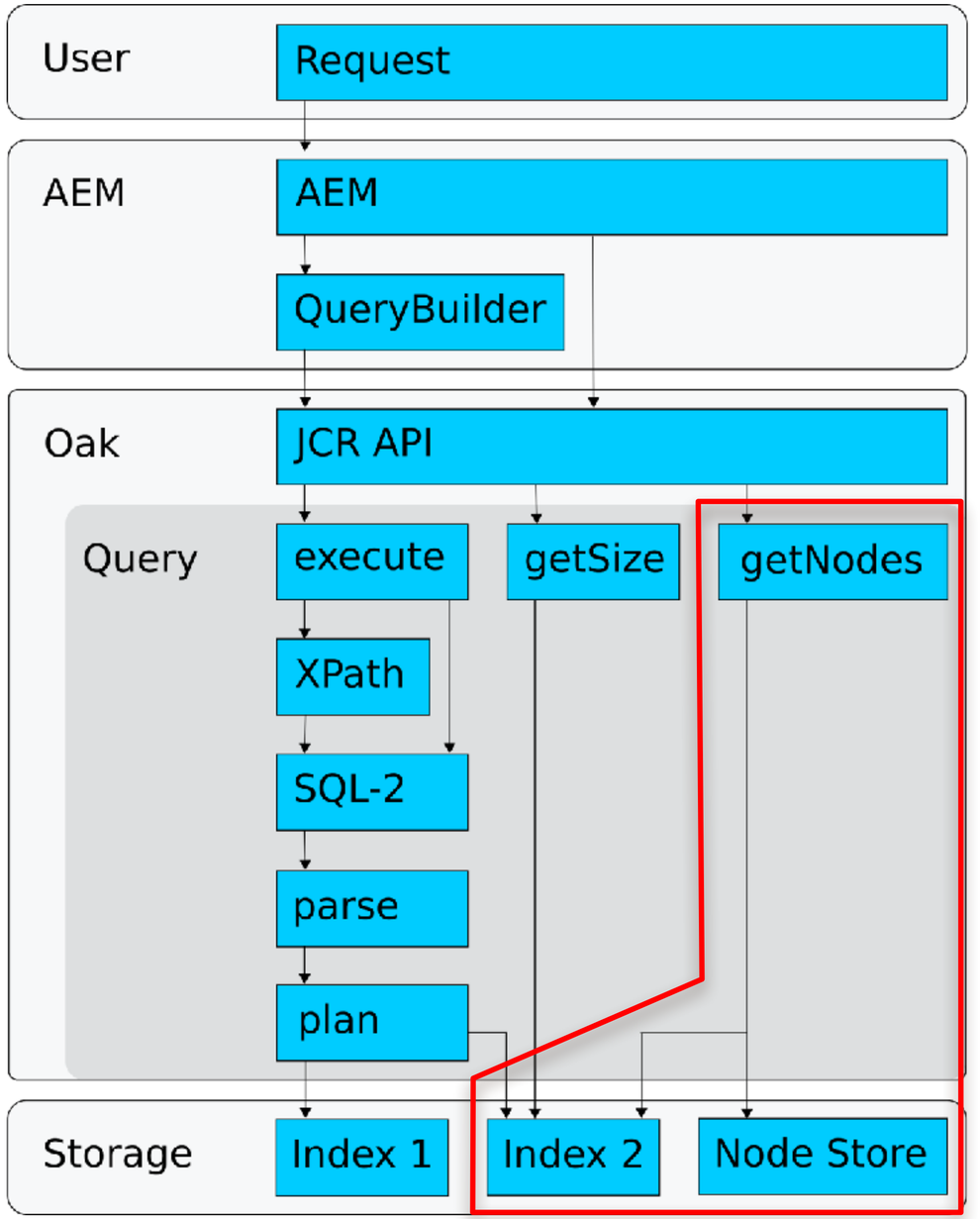
Query Result Size

- getSize asks the index, if "fast result size" is enabled
  - If the index supports this feature
  - Access rights are not checked
  - The QueryBuilder uses this if "p.guessTotal=true"- otherwise it reads all nodes (can be many)
- otherwise traverses



Result Iteration

- Iterating reads entries from the index, then the nodes, then filters (applies unindexed conditions)
  - ACLs are not checked in the index
  - A query may use multiple indexes

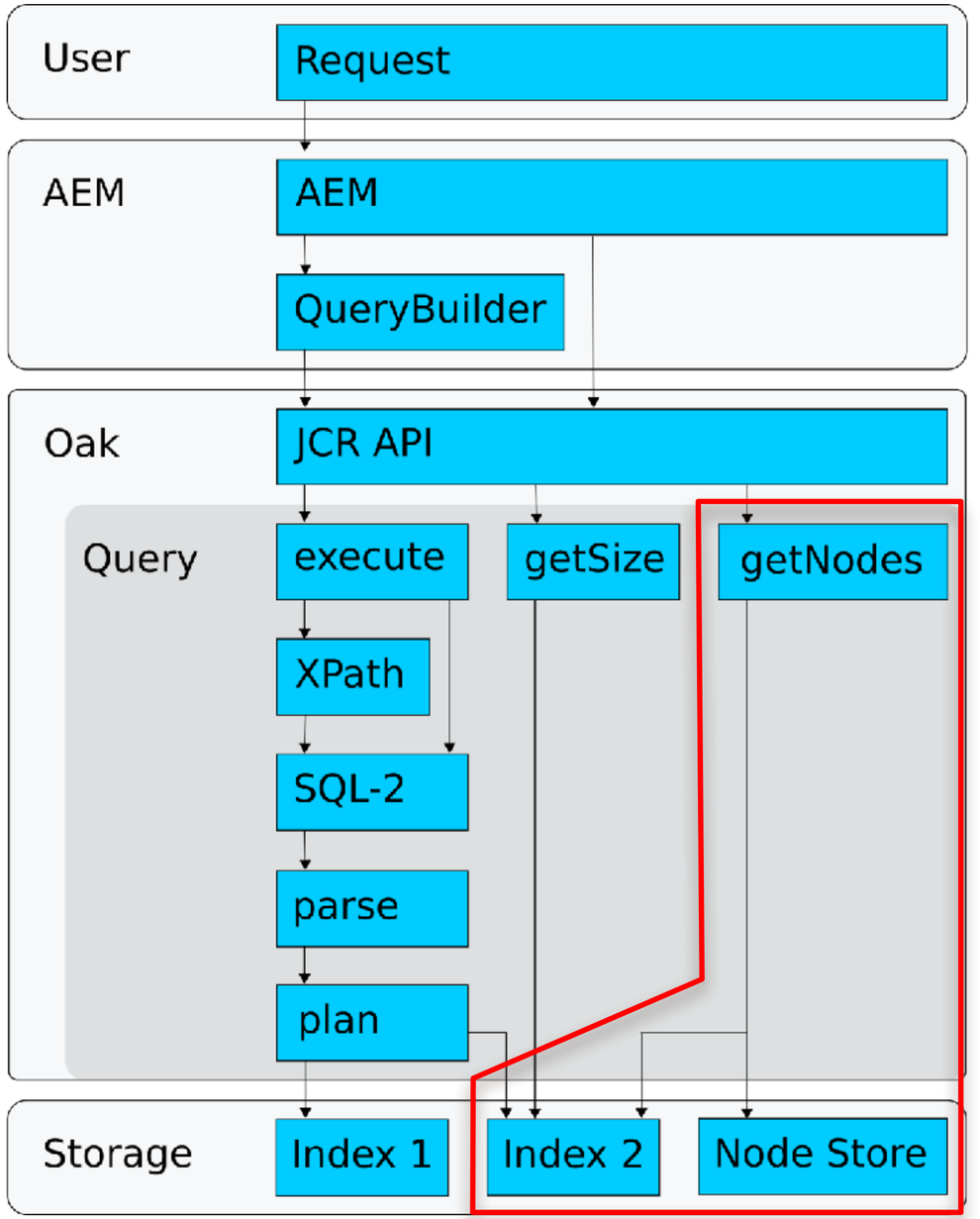


Result Iteration

- If many nodes are read, a warning is logged

```
*WARN*
Traversed 10000 nodes ...;
consider creating an index
or changing the query
```

- Change or create index, or change query to resolve

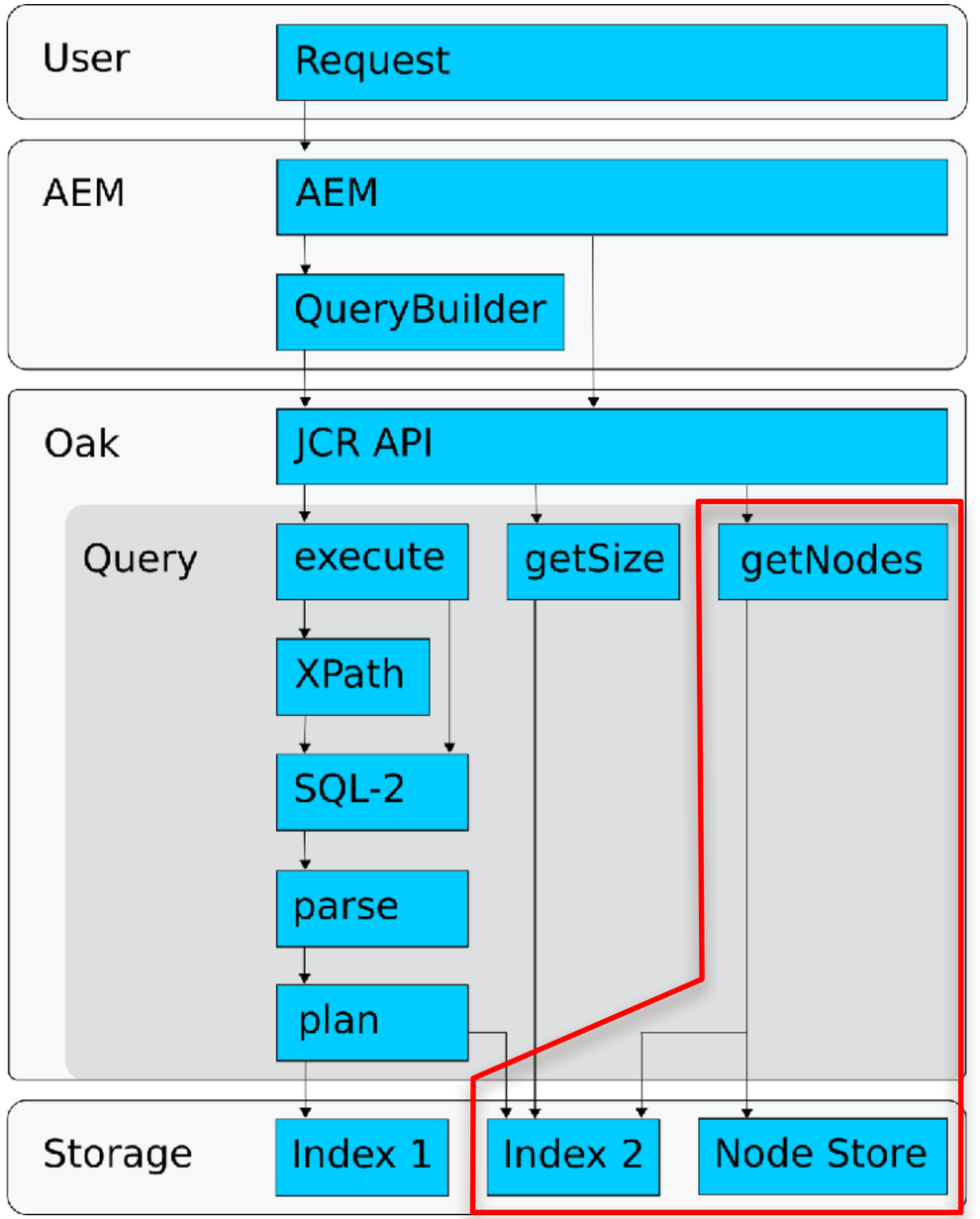


Result Iteration

- If too many nodes are read, iteration may fail:

```
*ERROR*
The query read more than x nodes...
processing was stopped.
```

- If no other way, temporarily change LimitReads / LimitInMemory

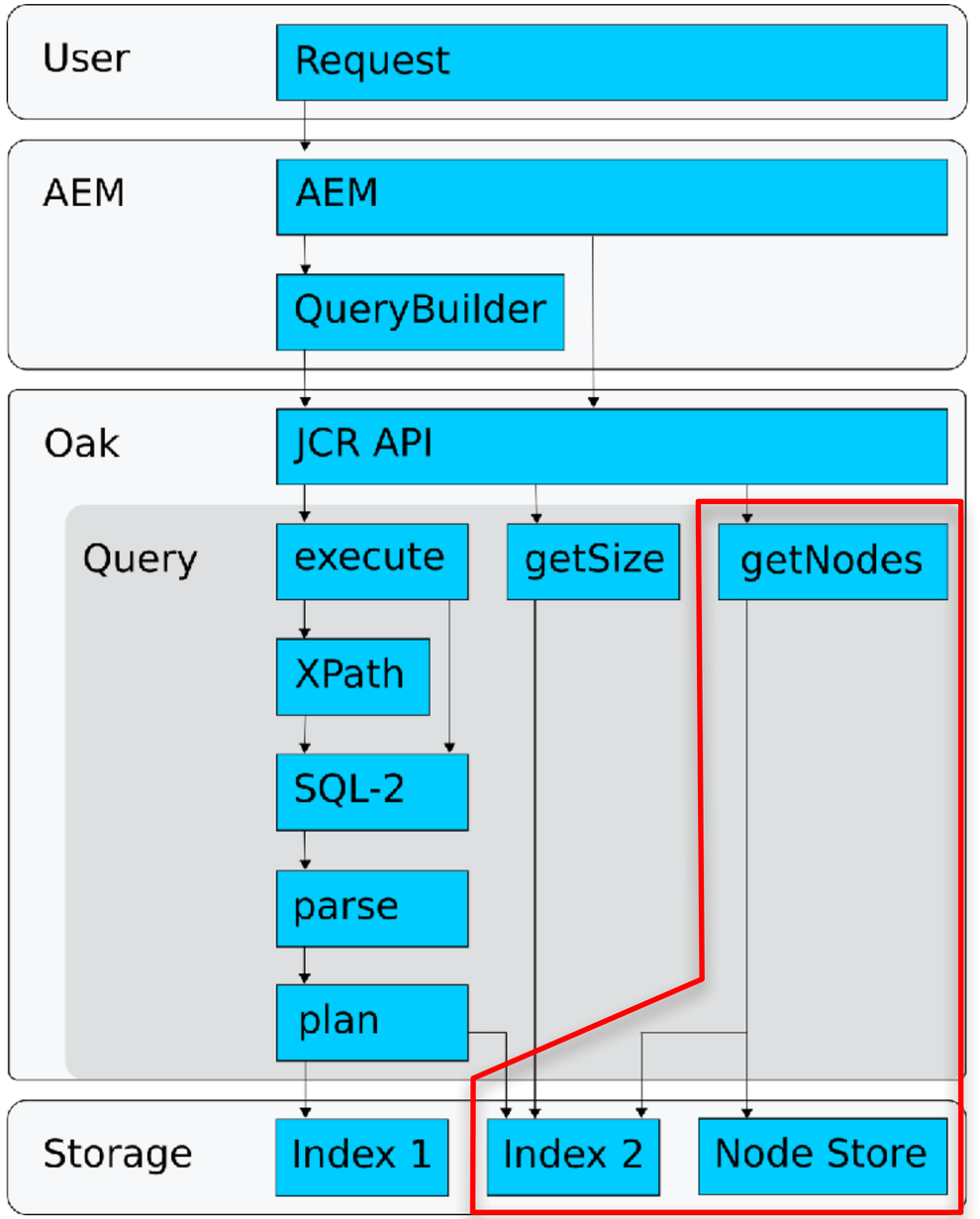


Excerpts

- Uses the Lucene Index: node excerpts and specified properties

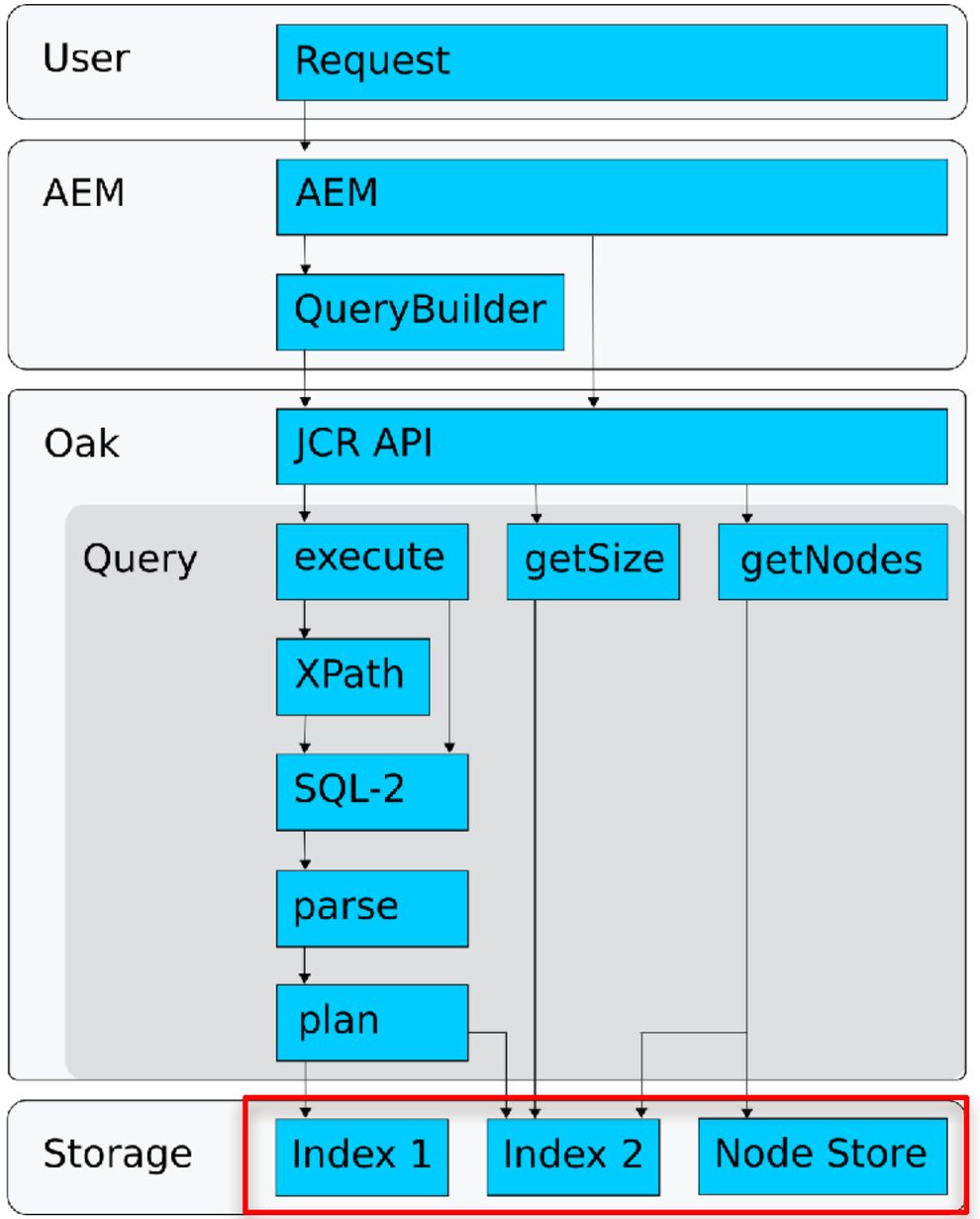
```
/jcr:root/content//*
[jcr:contains(., 'test')]/
(rep:excerpt(@jcr:title)|rep:excerpt(.))
```

- SimpleExcerptProvider otherwise (limitations; index config ignored)



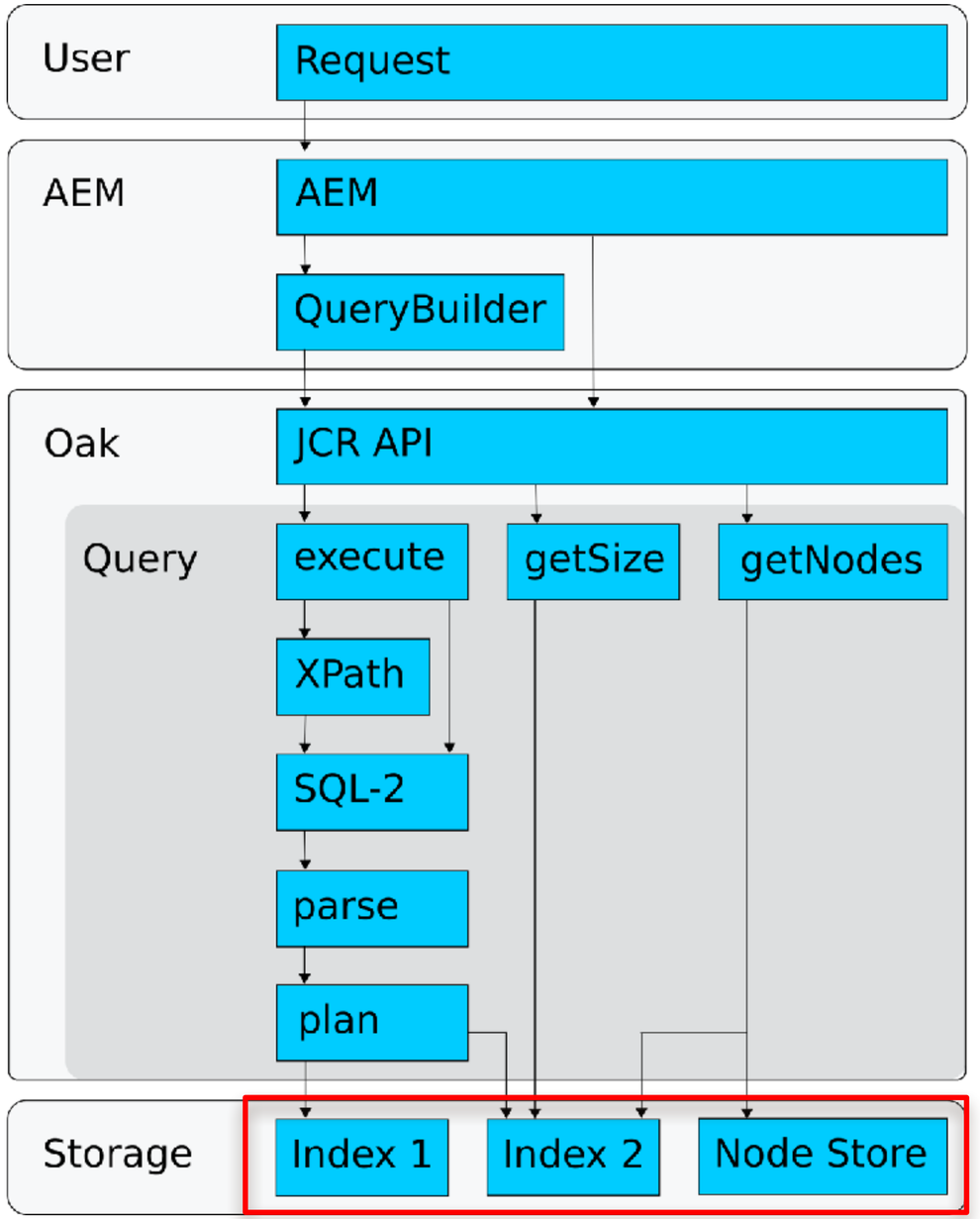
Index Types

- Property Index
  can be unique
  includes NodeType index, Reference index
  Ordered index is deprecated
- Lucene Index
  Lucene compatVersion 1: query time aggregation
  disabling copyOnRead / copyOnWrite not supported
- Traversal (index by path)



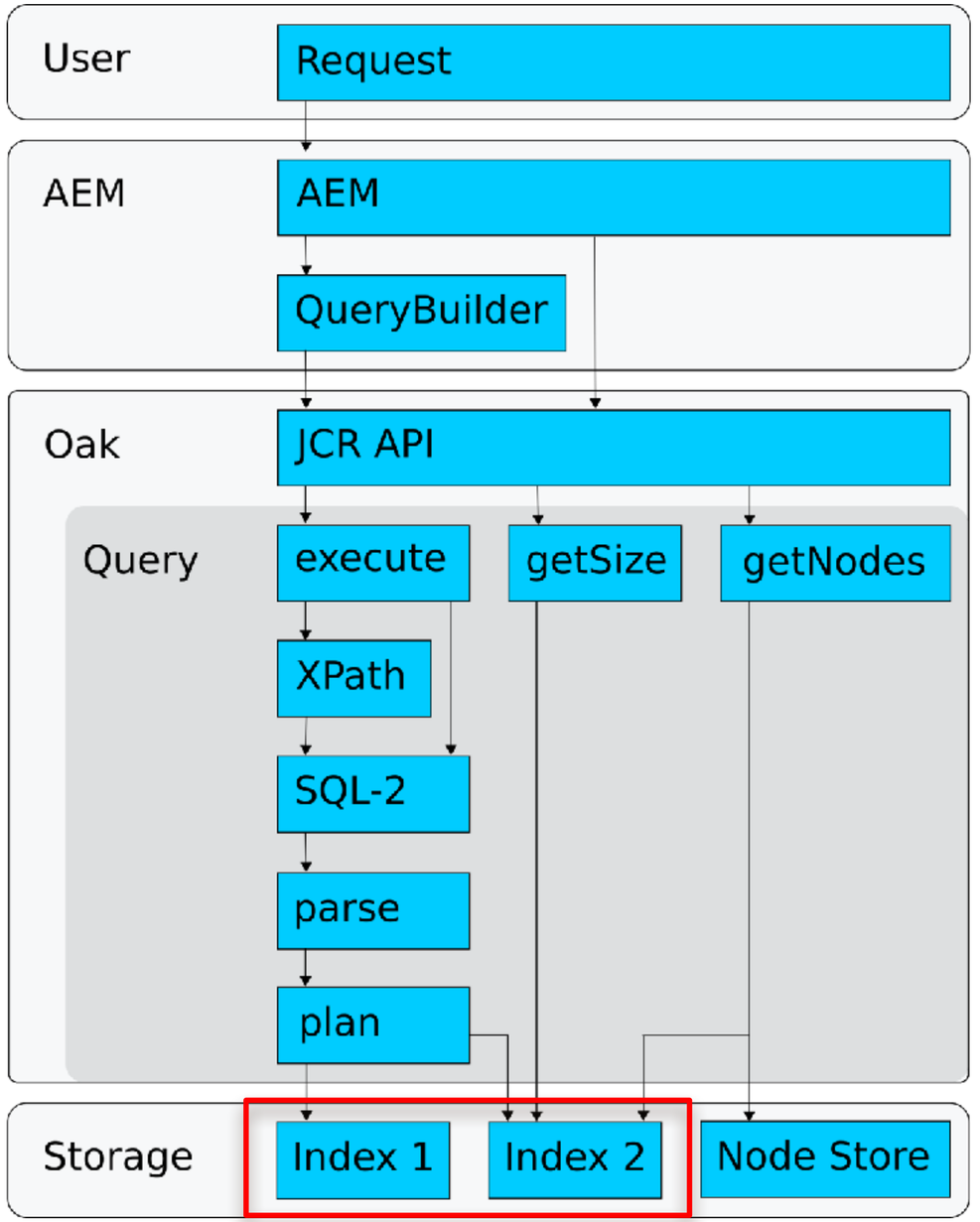
Lucene Index

- excludedPaths / includesPaths
  should match queryPaths
  otherwise some results may be missing
- evaluatePathRestrictions
  enabled: makes the index larger
  disabled: slows down queries (see also OAK-6123)



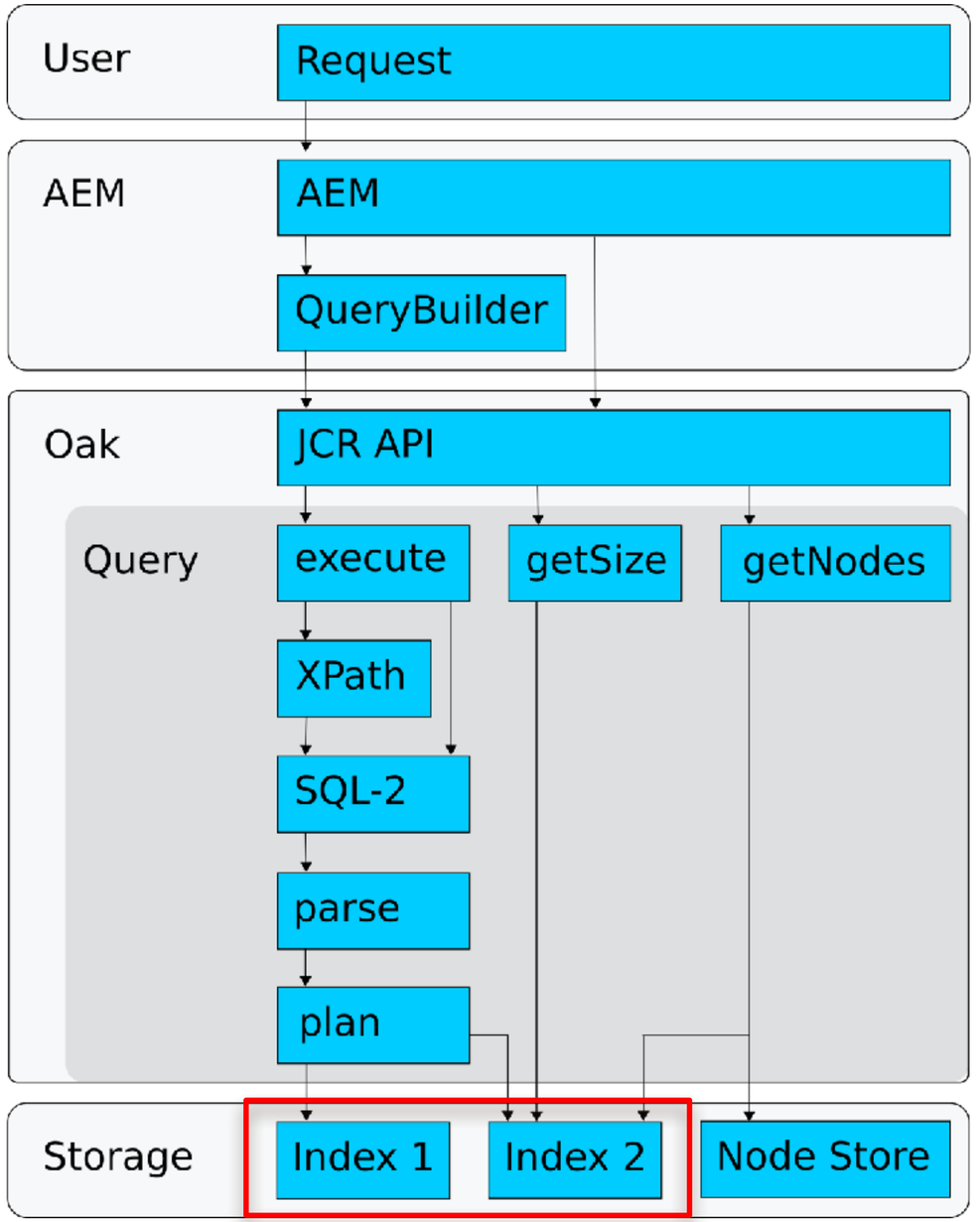
Index Types

- Synchronous: results reflect content
  changes from other cluster nodes are visible later
- Asynchronous: updated later
  NRT: near-real time; updated by each cluster node
- Hybrid
  index asynchronous, property is synchronous:
  /oak:index/ntBaseLucene/indexRules/nt:base/
  properties/vanityPath @sync=true



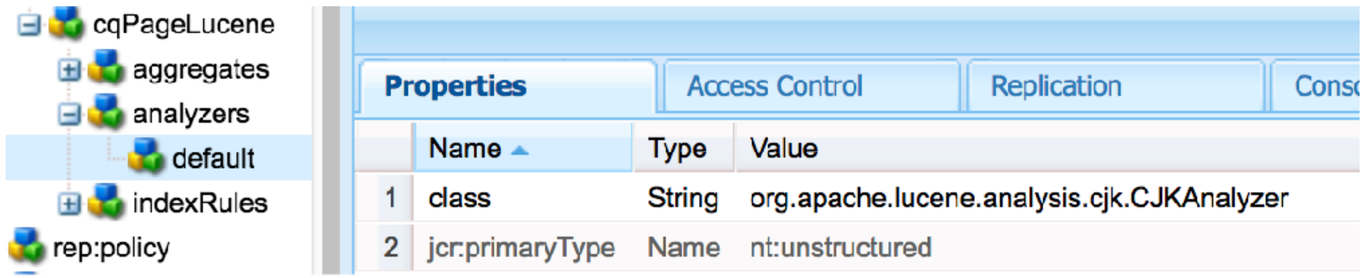
Async Testing

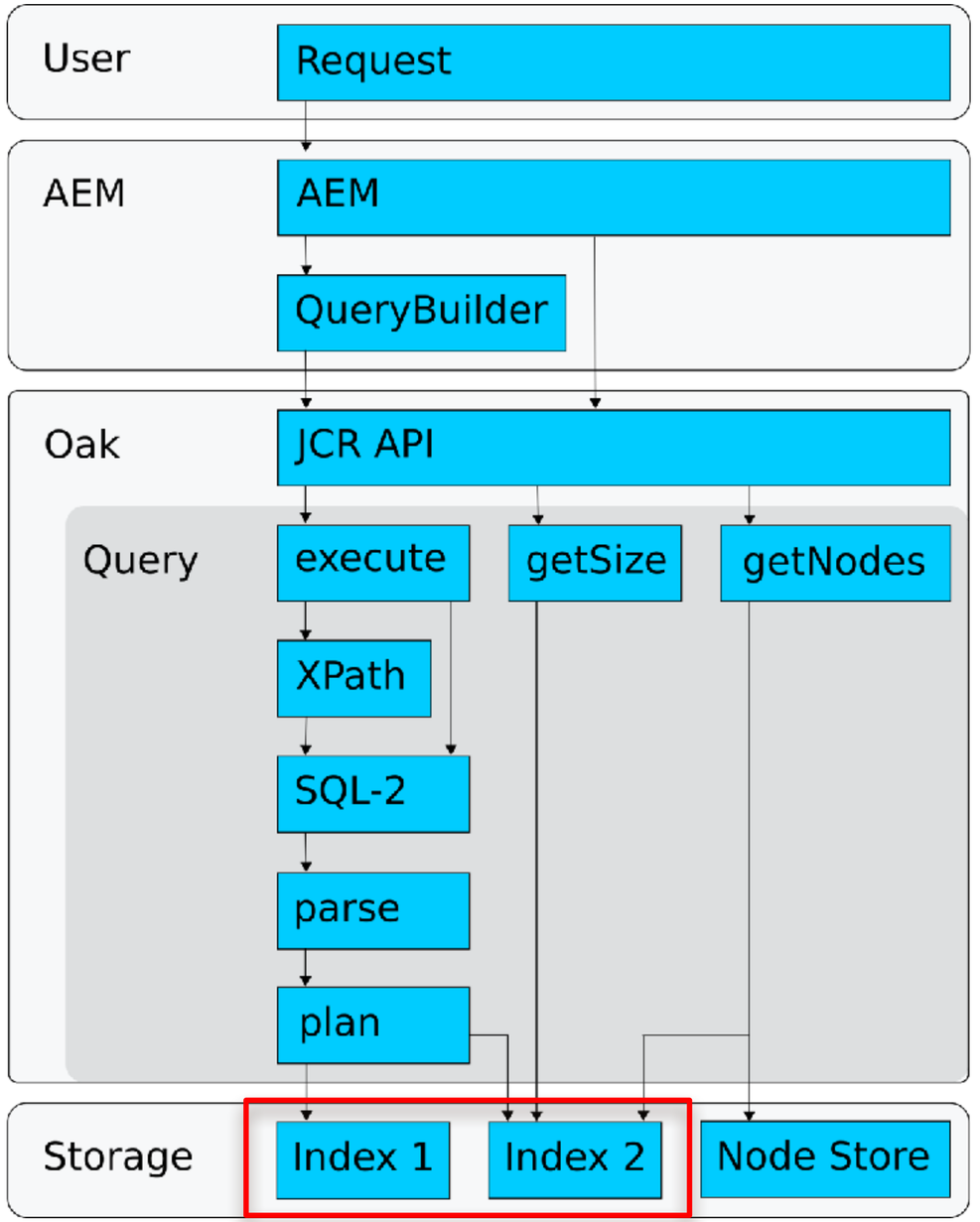
- Asynchronous indexes are harder to test
- Wait for indexing
  https://issues.apache.org/jira/browse/SLING-7169
  waitingIndexingDone



Fulltext Analyzer

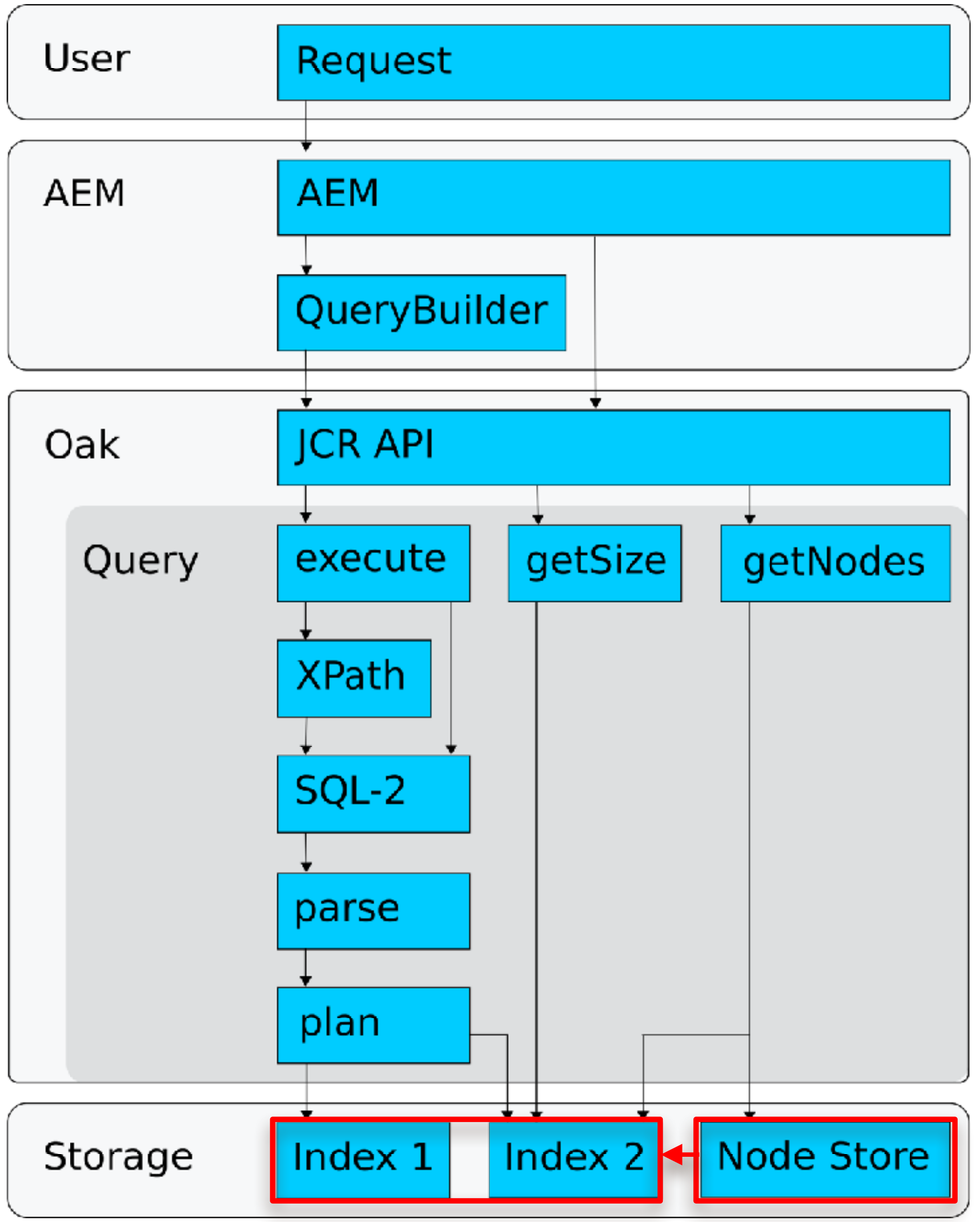
- Fulltext search uses Apache Lucene
- May require configuration
- See the Oak documentation
  http://jackrabbit.apache.org/oak/docs/query/lucene.html
- For Chinese, use the CJKAnalyzer





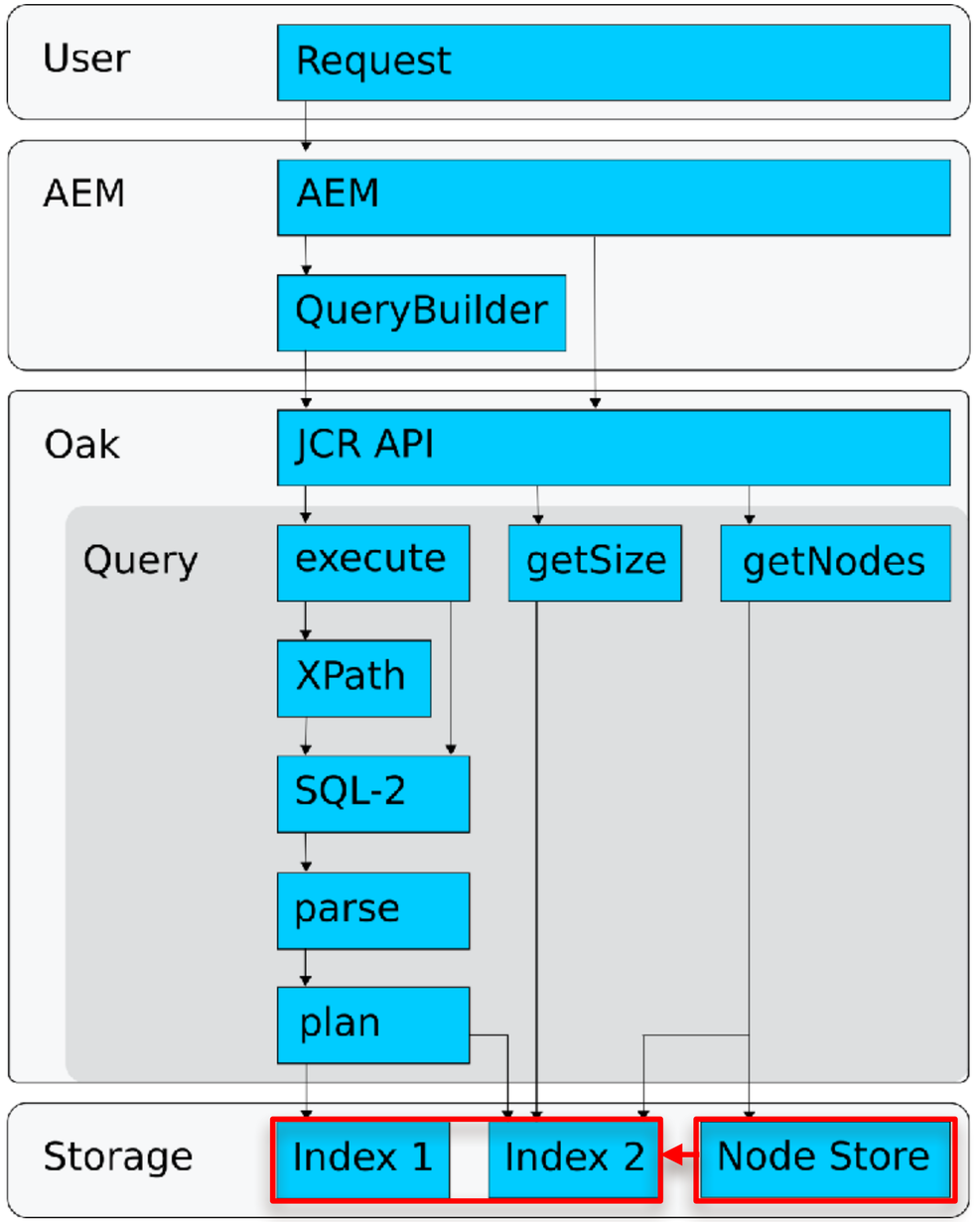
Lucene Indexing

- One cluster node indexes
- After a crash, indexing is paused (due to lease mechanism)
- Lucene binary cleanup task runs every night (AEM 6.4)



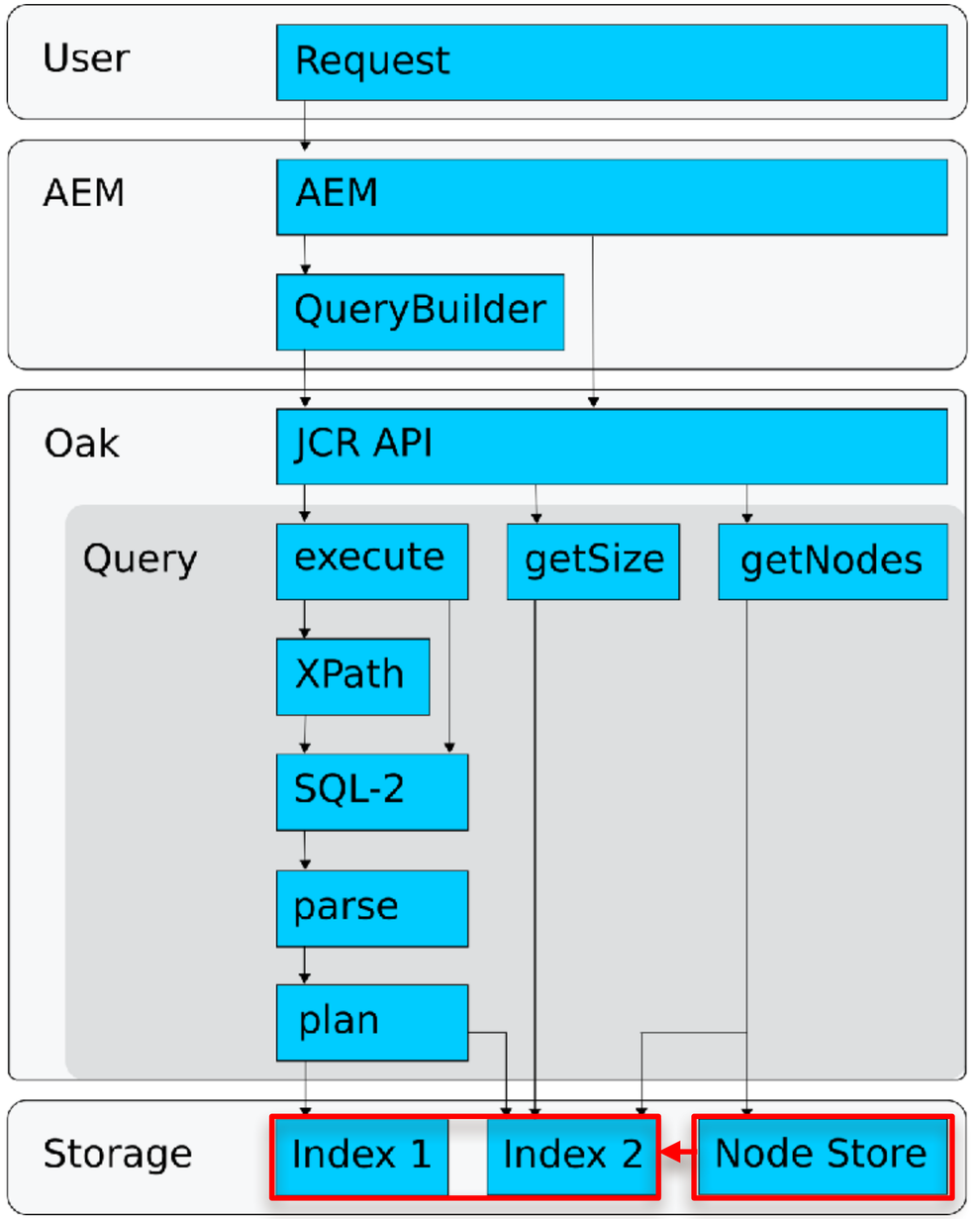
Text Extraction

- Is part of indexing
- Can be slow for large PDFs
- We have seen endless loops
- Protection against endless loops in AEM 6.4, but thread is not stopped and uses 100% CPU



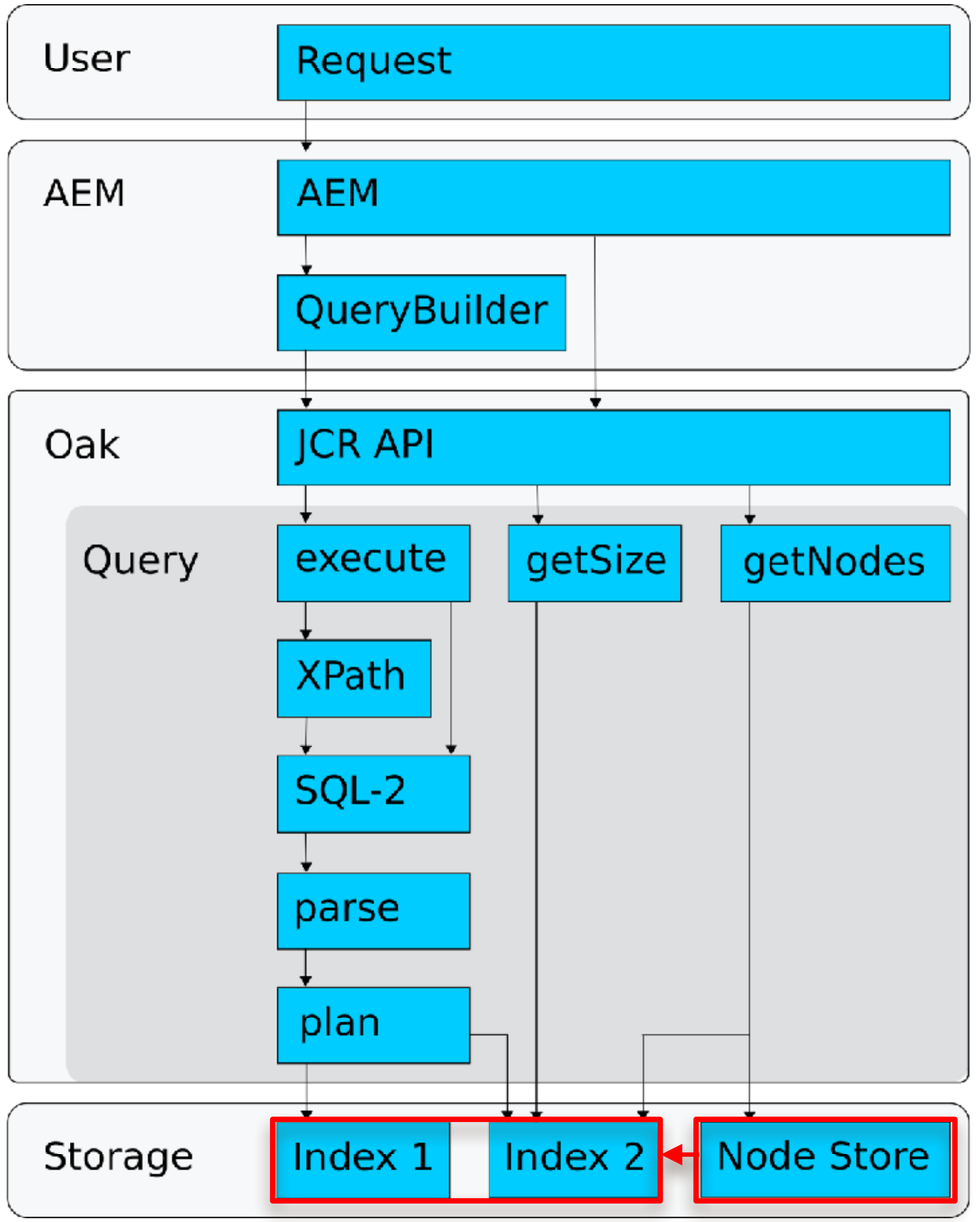
Reindexing

- Only needed in exceptional cases
  http://jackrabbit.apache.org/oak/docs/query/indexing.html#reindexing
- Can be very slow
- Old index is available during reindex
- oak-run (command line) may be much faster (specially MongoDB)
- Text extraction is slow

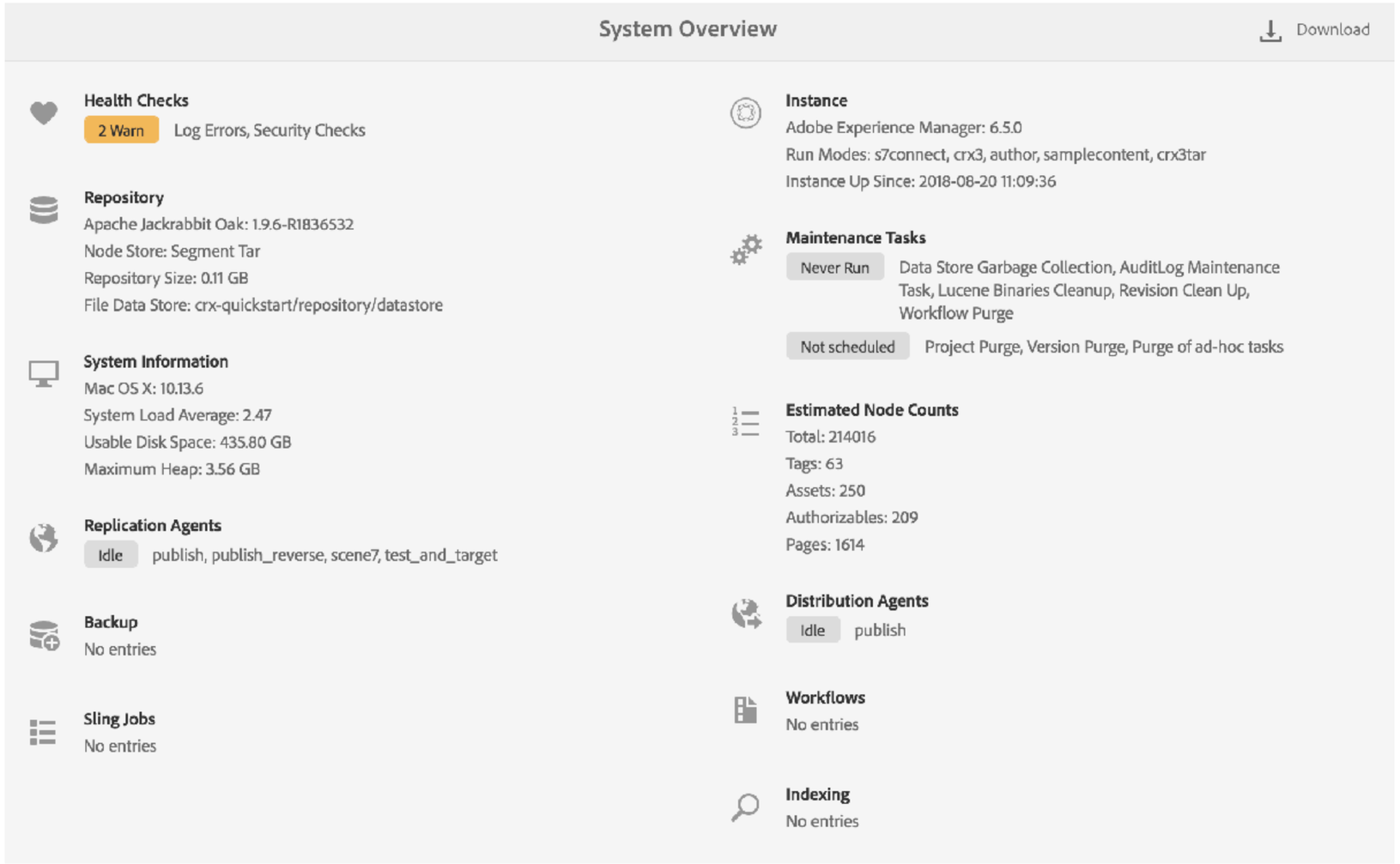


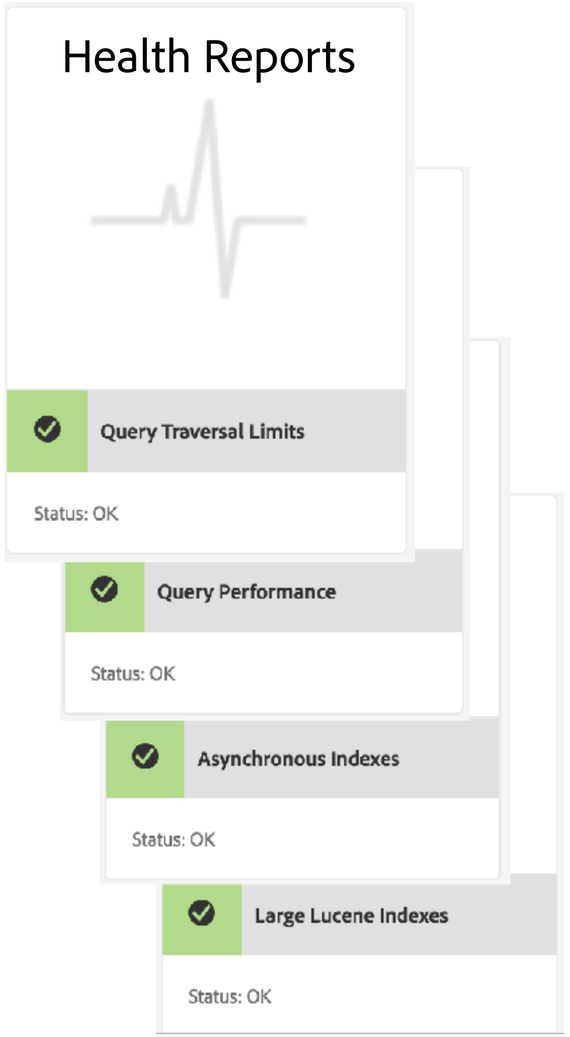
Text Pre-Extraction

- A key-value map with path=text
- Speeds up indexing
- Can be generated via oak-run
- Can be extracted from an index

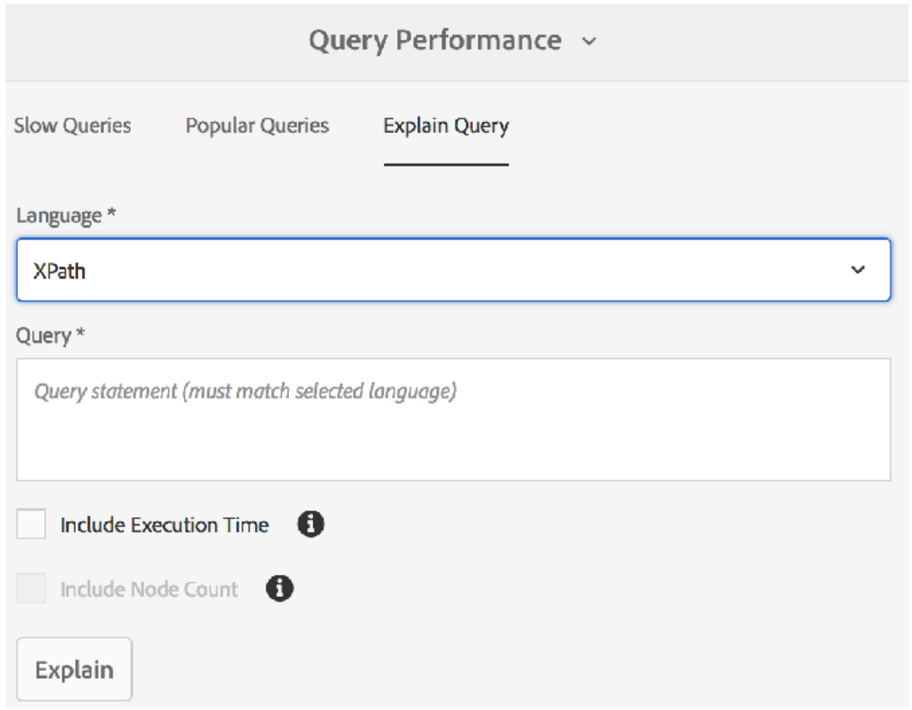


AEM 6.4 UI: Status Overview



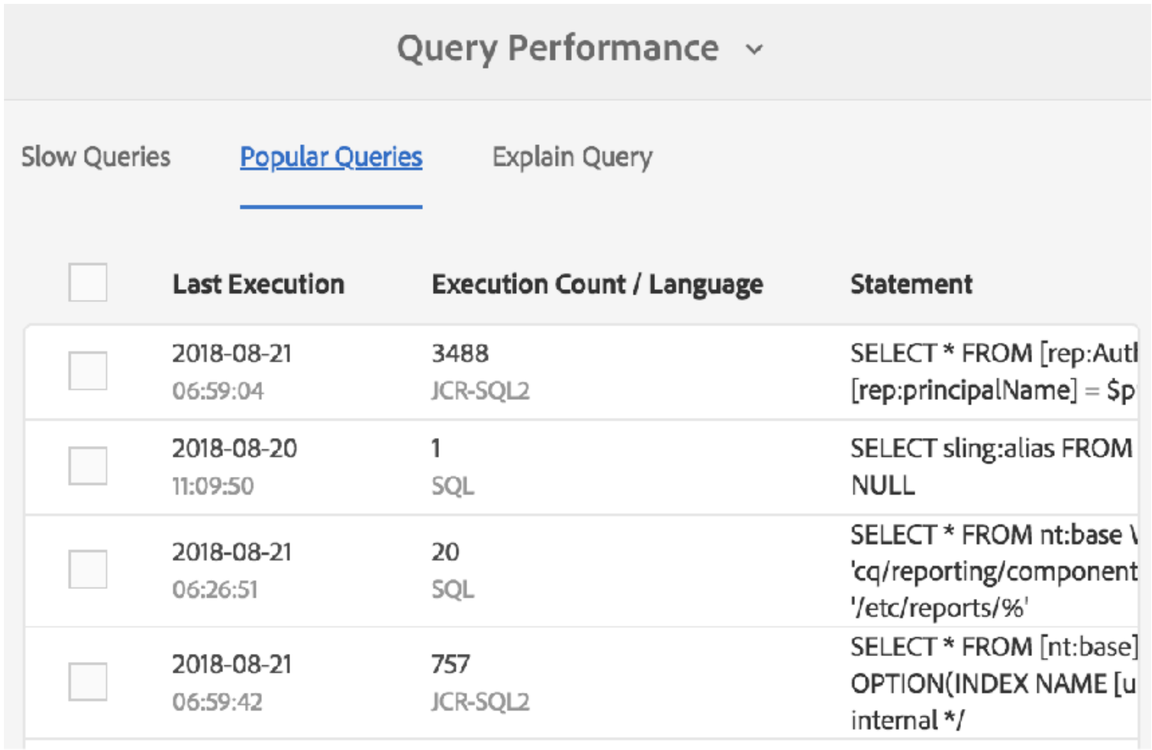


AEM 6.4 UI: Query Performance



More useful, and with QueryBuilder

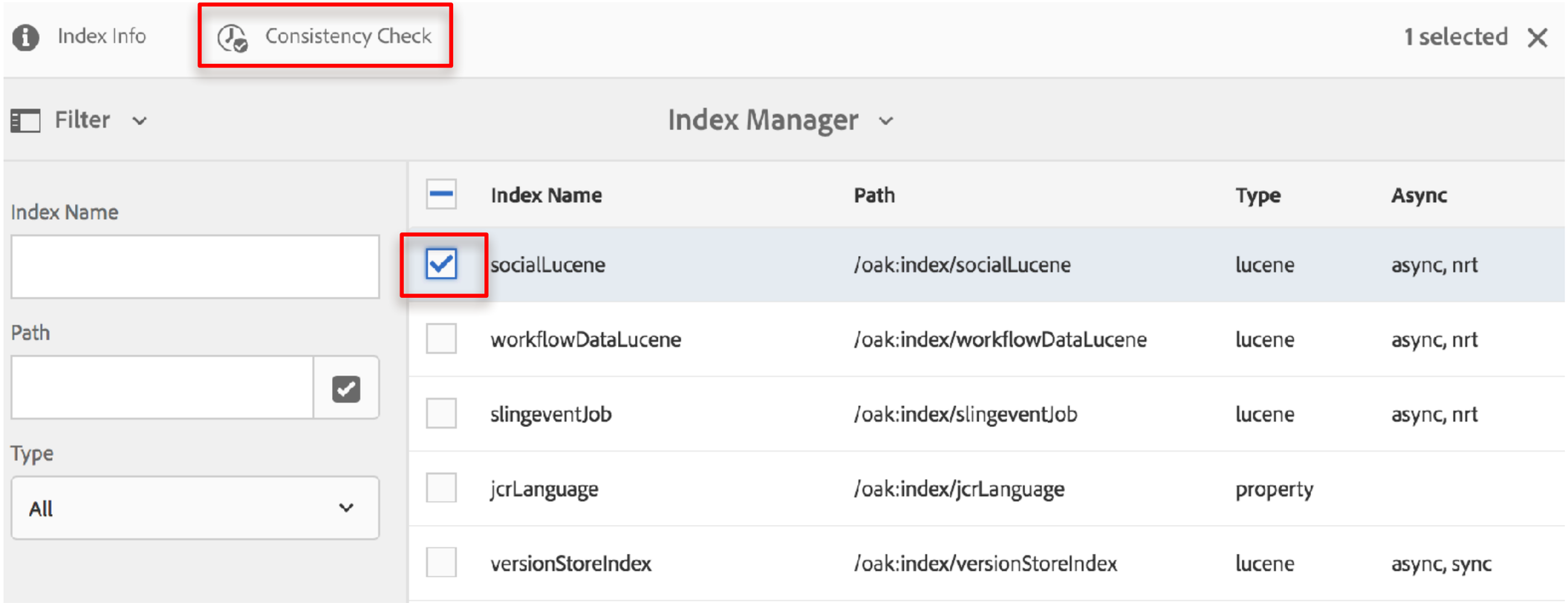
crx/de Query Tool: fixed in 6.4



Slow and popular queries in 6.4 shows accurate info

Adobe

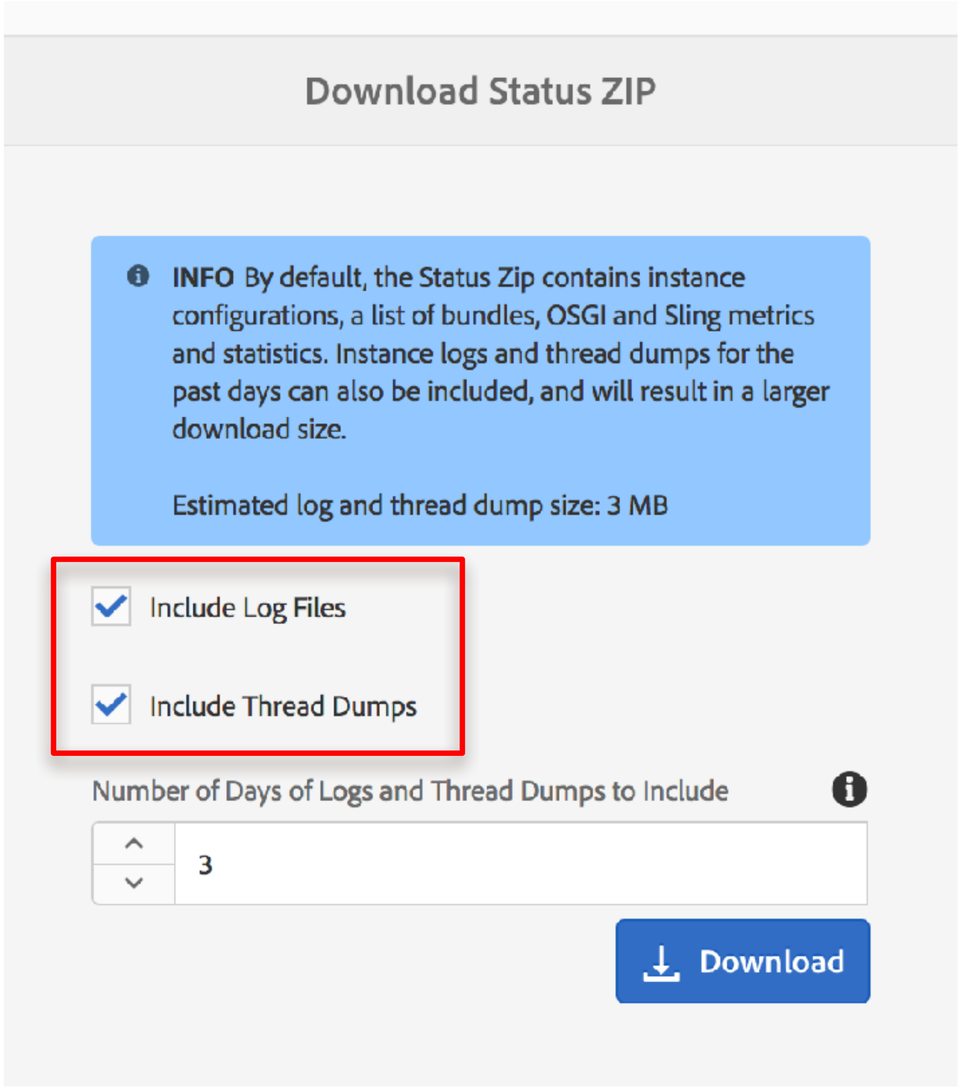
AEM 6.4 UI: Index Manager



Adobe

Error Reporting

- We need the query and query plan
  Debug log, or: Tools - Operations - Diagnosis - Query Performance - Explain Query
- Index config
  http://localhost:4502/oak:index.tidy.-1.json
  http://localhost:4502/content/oak:index.tidy.-1.json
- Configuration zip, log files
  http://localhost:4502/libs/granite/operations/content/diagnosistools/status.html



Related Presentations

- Oak and Queries (2014)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-oak-mongomk-and-queries.html
- Oak Lucene Indexes (2016)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-oak-lucene-indexes.html
- Indexing Best Practices and Troubleshooting (2017)
  https://helpx.adobe.com/experience-manager/kt/eseminars/ccoo-aem-indexing-recording.html
- Query Builder (2017)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-search-forms-using-querybuilder.html
- AEM Indexing and JCR Query (2017)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/aem-indexing-jcr-query.html
- Solr as an Oak index for AEM (2018)
  https://helpx.adobe.com/experience-manager/kt/eseminars/gems/Solr-as-an-Oak-index-for-AEM1.html

MAKE IT AN EXPERIENCE