SASI and Secondary Indexes
Hi!

- Computer Engineer
- Programming
- Electronics
- Math 😄 😄
- Physics
- Lego
- Meetups
- Animals
- Coffee
- GIFs
Let's talk about indexes
Regular index

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>&quot;Foo&quot;</td>
<td>&quot;<a href="mailto:foo@foo.com">foo@foo.com</a>&quot;</td>
</tr>
<tr>
<td>111</td>
<td>&quot;Bar&quot;</td>
<td>&quot;<a href="mailto:bar@bar.com">bar@bar.com</a>&quot;</td>
</tr>
</tbody>
</table>
You can search a user by ID and get his/her email.

ID → email
What happens if given an email you want to retrieve the user ID?
Let's have some Math:

\[ x \rightarrow y \]

\[ y \rightarrow x \]

Inverse function
In terms of matrices:

<table>
<thead>
<tr>
<th>ID</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>&quot;<a href="mailto:foo@foo.com">foo@foo.com</a>&quot;</td>
</tr>
<tr>
<td>111</td>
<td>&quot;<a href="mailto:bar@bar.com">bar@bar.com</a>&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>email</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;<a href="mailto:foo@foo.com">foo@foo.com</a>&quot;</td>
<td>110</td>
</tr>
<tr>
<td>&quot;<a href="mailto:bar@bar.com">bar@bar.com</a>&quot;</td>
<td>111</td>
</tr>
</tbody>
</table>
We would have to iterate over users, then compare the emails, then get the ID
But we can find the ID in analogical terms very easily!
What if we could have an extra index on a column to help on that?
<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>&quot;Foo&quot;</td>
<td>&quot;<a href="mailto:foo@foo.com">foo@foo.com</a>&quot;</td>
</tr>
<tr>
<td>111</td>
<td>&quot;Bar&quot;</td>
<td>&quot;<a href="mailto:bar@bar.com">bar@bar.com</a>&quot;</td>
</tr>
</tbody>
</table>

Regular index (key)

Secondary index (on column)
Secondary Indexes - The good parts

- Practical
- Easy to make
- Easy to understand
- Good performance for small sets of data
Secondary Indexes - The bad parts

- Slow for large sets of data
- They are applied locally, instead of globally
- In a ring with 5 instances, 1 query => 5 reads.
Several Strategies for the Secondary Index
References

- https://dzone.com/articles/cassandra-indexing-good-bad
- http://www.slideshare.net/edanuff/indexing-in-cassandra
- https://pantheon.io/blog/cassandra-scale-problem-secondary-indexes
Things are not lost
Apple opened their second index strategy
SSTableAttachedSecondaryIndex (SASI)

https://github.com/xedin/sasi
SASI through an example
cqlsh> CREATE KEYSPACE foo
    WITH replication = {
        'class': 'SimpleStrategy',
        'replication_factor': '1'
    };

cqlsh> USE foo;
CREATE TABLE bar ( id uuid, fname text, lname text, age int, created_at bigint, primary key (id)) WITH COMPACT STORAGE;

NOTE: COMPACT STORAGE IS MANDATORY!!11!!111 (for a while)
Creating the indexes
CREATE CUSTOM INDEX ON bar (fname)
USING 'org.apache.cassandra.db.index.SSTableAttachedSecondaryIndex'
WITH OPTIONS = {
'analyzer_class':
'org.apache.cassandra.db.index.sasi.analyzer.NonTokenizingAnalyzer',
'case_sensitive': 'false'
};

https://github.com/xedin/sasi/blob/master/src/java/org/apache/cassandra/db/index/sasi/analyzer/StandardAnalyzer.java
CREATE CUSTOM INDEX ON bar (lname)
USING 'org.apache.cassandra.db.index.SSTableAttachedSecondaryIndex'
WITH OPTIONS = {'mode': 'SUFFIX'};

Analyses by suffix.
'mode': 'SPARSE'

CREATE CUSTOM INDEX ON bar (created_at)
USING 'org.apache.cassandra.db.index.SSTableAttachedSecondaryIndex'
WITH OPTIONS = {'mode': 'SPARSE'};

Analyses ranges of time by timestamp
Understanding the basics of SASI architecture
SASI takes advantage of Cassandra's Architecture to build a powerful set of data structure that manages information between memory and disk as well.
Cassandra's key features to SASI

- Write-only
- Immutability
- Ordered data sets
memory => disk

SSTable \rightarrow \text{starts writing}

SASI \rightarrow \text{creates structures in memory}

SASI \rightarrow \text{flushes structures to disk}

SSTable \rightarrow \text{ends writing}
SASI

Indexing

Querying
Indexing

Indexed columns

SSTable

Index files by SASI

MEMORY
OnDiskIndexBuilder
Indexed columns

Index files by SASI

MEMORY
OnDiskIndexBuilder

Disk
OnDiskIndex

Optimised data structures
(List<ByteBuffer> and custom iterators)

Information of SSTable is turned into these structures
SASI

Indexing

Querying
Querying

Index files by SASI

QueryPlan
Analysis
Execution

RangeIntersectionIterator
RangeUnionIterator
Limitations

- Cluster must be configured to use a partitioner that produces LongTokens (Murmur3Partitioner). Does not work with ByteOrderedPartitioner.
- CQL3 requires COMPACT STORAGE
- Only Cassandra 2.0.x is supported. If you don't have plans to upgrade, you cannot use SASI properly
Conclusion
Powerful tool for Secondary Indexes
Let's study data structures and more Mathematics :)
References

- SASI repo https://github.com/xedin/sasi
- SecondaryIndexes Q/A https://wiki.apache.org/cassandra/SecondaryIndexes
- SASI on Cassandra https://issues.apache.org/jira/browse/CASSANDRA-10661
Special Thanks

- @PatrickMcFadin
- @wheresLINA
- @planetcassandra
- @lafp, @romulostores and @pedrofelipee (GIFs)
Thank you :) 

Questions?

hannelita@gmail.com
@hannelita
@planetcassandra
@datastax