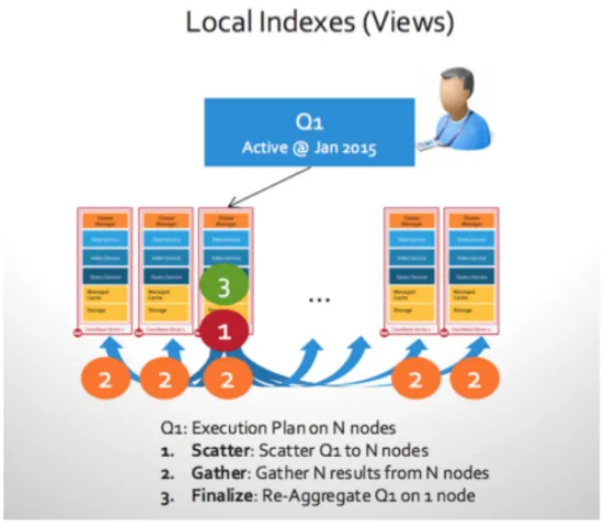
The Couchbase Blog
Couchbase, the NoSQL Database

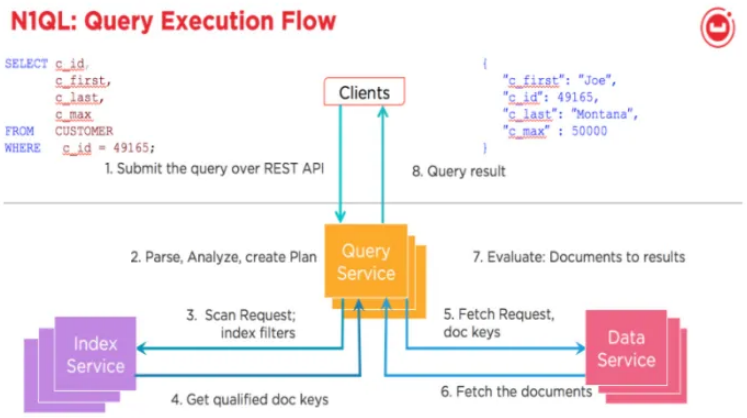
Comparing Couchbase Views with Couchbase N1QL & Indexing.

Keshav Murthy on December 4, 2017

As Couchbase data platform evolved, services like N1QL and GSI Indexing handled the use cases Couchbase VIEWS used to handle and much more. It's logical to ask the comparative question between them. Here is a table comparing both. This is intended for developers and architects familiar with both them and not as an introductory article. Use the links here to learn more and play with the respective features.



View Query Execution



N1QL Query Execution

Next →
Develop Web Applications w…

Topic	Couchbase Map-Reduce Views	Couchbase N1QL+GSI
Approach	Based on user defined map/r…	Based on declarative N1QL query (SQL

Approach	Based on user-defined map() and reduce() functions that operate on data in the background. Because map() and reduce() is written in Javascript, you can code complex logic within those functions.	Based on declarative N1QL query (SQL for JSON). Uses appropriate indexes to optimize execution and executed dynamically by orchestrating Query-Index-data services. N1QL enables easily writeable and readable queries for JSON. Because it's inspired by SQL, it's flexible, composable. Because, it's extended for JSON, it works on rich JSON data. Uses 4-valued boolean logic (true, false,NULL, MISSING)
More Info	Couchbase Docs: http://bit.ly/2jQrY11	1. http://query.couchbase.com 2. https://blog.couchbase.com/n1ql-practical-guide-second-edition/
Querying	Query based on 1. Single key 2. Set of keys 3. Start-End key 4. Start-End document keys 5. Group BY, Aggregation 6. Pagination	**Query Statements** 1. SELECT 2. INSERT 3. UPDATE 4. DELETE 5. MERGE 6. INFER 7. EXPLAIN **Query Operations:** 2. Set of keys

Next →
Develop Web Applications w…

		3. Range keys 4. Range of document keys 5. Arbitrarily complex predicates 6. INNER JOIN, LEFT OUTER JOIN 7. NEST, UNNEST 8. GROUP BY 9. Aggregation 10. Pagination (OFFSET, LIMIT) 11. Optimization 12. ORDER BY 13. HAVING 14. Subqueries (correlated, non-correlated) 15. Derived tables 16. SET operations: UNION, UNION ALL, EXCEPT, EXCEPT ALL, INTERSECT 17. Highly composable queries, meaning these operations can be simply combined with each other to express complex business questions and operations easily.
Indexing	Simple index for views.	1. Primary Index 2. Named primary index 3. Secondary index 6. Array Index

Next →
Develop Web Applications w…

		7. ALL array 8. ALL DISTINCT array 9. Partial Index 10. Adaptive Index 11. Duplicate Indices 12. Covering Index
Partitioning	Aligned to data partitioning.	Independent services. N1QL and GSI scales independent of Data service and each other.
Scale	Scales with data service	Independent scaling via Multidimensional scaling (MDS)
Fetch with document key	Because the data is partitioned on document key, fetches the document directly from the node	Specify the query via USE KEYS clause. Because the data is partitioned on document key, fetches the document directly from the node
Fetch with Index key	Scatter-Gather	Each index scan on a single node; Data on multiple nodes. Post processing in Query node
Range scan	Scatter-Gather	Index scan on a single node.

Next →
Develop Web Applications w…

Grouping,	Built-in with Views API	Built into N1QL

aggregation		
Caching	File system	Index buffer pool Data cache
Storage	Couchstore	Plasma storage engine (5.0 & above) Memory Optimized Index (4.5 and above) ForestDB (community)
Availability	Replica Based	5.0: Replicas 4.x: Equivalent Indexes
Query Latency (Simple queries)	10 milliseconds to 100 milliseconds	5 milliseconds+
Query Throughput (Simple queries)	3K to 4K queries per second	40K queries per second
	data service)	query services: MDS)

Next →
Develop Web Applications w…

Applicability	Aggregations, best of large scale aggregations for low and moderate latency requirements. Map-reduce operations on the data is done in the background as the data is modified.	Best for attribute based lookup, range scans, complex select-join-project-array Operations. Supports grouping, aggregation and ordering — these operations are done dynamically as part of query execution.
Application requirements	Report on well defined metrics Large scale aggregations Latency sensitive	Secondary key lookups Range Scans Operational aggregations Filtered queries Ad-hoc queries with complex predicates, joins, aggregations, app search, pagination, secondary key based updates.
Spatial	Supported via Spatial Views	Not directly. https://dzone.com/articles/speed-up-spatial-search-in-couchbase-n1ql
Consistency	Stale = UPDATE_AFTER Stale = OK	Unbounded (stale = OK) AT_PLUS (read your own writes)

Next →
Develop Web Applications w…

updates up to now(). Stale = False).

Tools	Web console	Web console, Developer workbench, Query monitoring, Query Profiling, Visual explain, INFER.

Posted in: Application Design, Best Practices and Tutorials, Couchbase Architecture, Data Modeling, N1QL / Query

Tagged in: couchbase, Indexing, MapReduce, N1QL, performance, Secondary Indexing, SQL

Posted by Keshav Murthy

Keshav Murthy is a Vice President at Couchbase R&D. Previously, he was at MapR, IBM, Informix, Sybase, with more than 20 years of experience in database design & development. He lead the SQL and NoSQL R&D team at IBM Informix. He has received two President's Club awards at Couchbase, two Outstanding Technical Achievement Awards at IBM. Keshav has a bachelors degree in Computer Science and Engineering from the University of Mysore, India, holds eight US patents.

All Posts Website

Leave a reply

Comment

Next →
Develop Web Applications w…