Couchbase N1QL

The Power of SQL Meets the Flexibility of JSON

Couchbase

Key Concepts

N1QL = JSON + SQL

N1QL is the first query language to combine the complete flexibility of JSON with the full power of SQL.

Fully Compatible with SQL

You can apply your knowledge of SQL to NoSQL. If you know SQL, you will know N1QL.

Joins to Leverage Relationships

Build a flexible JSON data model consisting of multiple documents that can be queried via JOINS to leverage the relationships among data.

Access to N1QL Your Way

Use N1QL directly, or access via the framework of your choice – no learning curve required.

No Impedance Mismatch

Apps process N1QL results as streams of JSON documents – no impedance mismatch, and no need for a complex translation layer.

Real Time Business Intelligence

N1QL has full compatibility with the SQL ecosystem via connectors and standard JDBC / ODBC drivers. Plug in to your BI or Reporting tool of choice.

Couchbase Server with N1QL enables you to build applications with more speed, less code, and greater agility.

The Bottom Line: N1QL is a comprehensive and declarative query language that combines the power and familiar syntax of SQL with the flexibility of JSON data model. This combination makes it faster and easier for developers to build web, mobile and IoT enterprise applications on top of our scalable database. Couchbase Server is the first NoSQL database to successfully combine high performance, scale, and comprehensive query with ease of development.

N1QL Drives Developer Agility

Couchbase N1QL is a comprehensive and declarative query language that leverages the flexibility of JSON and the power of SQL. It dramatically increases developer agility, as it enables you to query and transform semi-structured JSON data in any manner your application requires. This includes the ability to build a flexible JSON data model consisting of related documents that can be queried via **JOINS**, and to **NEST** or **UNNEST** data to query or transform complex documents. Additionally, N1QL is accessible via a developer's preferred development framework – whether LINQ, Spring, Ottoman (our object-document mapping [ODM)] framework for Node.js), or anything else. There's no learning curve. Since the application processes query results directly as streams of JSON documents, there's no longer an impedance mismatch, and no need for a complex translation layer.

Why Was N1QL Necessary?

Quite simply, there was a need for a more powerful NoSQL query solution. Relational databases, with SQL, have been the historical standard, but their rigid, tabular structure is ill suited to support the scale and semi-structured data used by most web, mobile, and IoT applications. Document databases such as Couchbase Server, which support the JSON data format, are more flexible and scalable – but until now, their adoption has been limited by the lack of a powerful query language. N1QL changes that by extending SQL – recognized by virtually every developer in the world – to JSON, the industry standard data model for web, mobile and IoT applications.

Why SQL for NoSQL?

SQL is proven and powerful. It has been the database industry's standard query language for more than 40 years; millions of developers around the world are building scalable, enterprise applications today using SQL either directly or indirectly through application development

> NoSQL systems have proven their value in the enterprise with ease of development, performance, and scalability, but developers still need a query language that lets them build applications that require complex queries on semi-structured data. UCSD defined SQL++ to provide the industry with specifications for a SQL backwards-compatible declarative language that works on semi-structured data. N1QL is consistent with our specification – it gives developers a fully declarative and SQL-compatible query language to build applications that leverage the agility of JSON. We think N1QL will propel NoSQL adoption just as SQL originally propelled Relational Database Management System adoption.
>
> – Yannis Papakonstantinou,
> University of California,
> San Diego Computer Science and Engineering Professor

Key Concepts

Developers Love N1QL

Develop faster by querying data with a language that is easy to understand as it is based on SQL

Develop with less code by leveraging N1QL to express complex query logic instead of writing and executing it within your application

Develop with greater agility by creating new indexes and queries without needing to change the data model

Develop in the language and framework of your choice with options for asynchronous and reactive data access

2440 West El Camino Real | Ste 600
Mountain View, California 94040

1-650-417-7500

www.couchbase.com

frameworks. By extending SQL to JSON, we can leverage all the SQL experience and capability and apply it to JSON, and by leveraging existing SQL constructs, N1QL will be familiar and easy for developers to adopt.

N1QL Connects the SQL Ecosystem to NoSQL

N1QL is also fully compatible with the SQL ecosystem. N1QL further benefits the enterprise by making access to data stored in Couchbase Server easy and efficient. N1QL has full compatibility with the SQL ecosystem via connectors and standard JDBC / ODBC drivers. This allows enterprises for the first time to connect popular ETL, Reporting, and BI tools to Couchbase Server. Companies like Databricks, Looker, Simba Technologies, Informatica, Tableau and Metanautix are all partnering with Couchbase to provide deeper, supported integrations.

Sample Queries/Syntax

Here's an example of a query that finds flights between Seattle-Tacoma International Airport (SEA) and Orlando International Airport (MCO). It searches for routes and schedule information based on the source and destination airports and the requested dates.

```
SELECT   a.NAME, s.flight, s.utc, r.sourceairport,
         r.destinationairport, r.equipment
FROM     `travel-routes` r
UNNEST   r.schedule s
JOIN     airlines a
ON KEYS  r.airlineid
WHERE    r.sourceairport='SEA'
AND      r.destinationairport='MCO'
AND      s.day=6
ORDER BY s.utc
```

Two items of note in this N1QL query are the JOIN and UNNEST commands. N1QL provides JOIN functionality, something previously not possible in a document database. And UNNEST is a powerful feature available in the Couchbase Query API. UNNEST enables you to flatten, the results returned in the SELECT statement. In the data model for the travel application, each route document contains a nested collection of schedule documents. To avoid a complicated JSON parsing code pattern for the return results, you can UNNEST the schedule documents so they become root level fields in the returned results.

About Couchbase

Couchbase delivers the world's highest performing NoSQL distributed database platform. Developers around the world use the Couchbase platform to build enterprise web, mobile, and IoT applications that support massive data volumes in real time. The Couchbase platform includes Couchbase Server, Couchbase Lite - the first mobile NoSQL database, and Couchbase Sync Gateway. Couchbase is designed for global deployments, with configurable cross data center replication to increase data locality and availability. All Couchbase products are open source projects. Couchbase customers include industry leaders like AOL, AT&T, Bally's, Beats Music, BSkyB, Cisco, Comcast, Concur, Disney, eBay, KDDI, Nordstorm, Neiman Marcus, Orbitz, PayPal, Rakuten / Viber, Tencent, Verizon, Wells Fargo, Willis Group, as well as hundreds of other household names. Couchbase investors include Accel Partners, Adams Street Partners, Ignition Partners, Mayfield Fund, North Bridge Venture Partners, and West Summit.