Couchbase review: A smart NoSQL database

Flexible, distributed document database offers an easy query language, mobile synch, independently scalable services, and strong consistency within a cluster

By Martin Heller
Contributing Editor, InfoWorld | August 12, 2019

Every medium to large business needs a database. Large multi-national businesses often need globally distributed databases, and when they use their database for financial or inventory applications they need strong consistency. Few databases can fill both needs.

Couchbase Server is a memory-first, distributed, flexible JSON document database that is strongly consistent within a local cluster. Couchbase Server also supports cross data center replication with eventual consistency across clusters.

Couchbase Lite is an embedded mobile database that works offline and synchronizes with Couchbase Sync Gateway when online. Sync Gateway synchronizes with Couchbase Server as well as with multiple Couchbase Lite instances.

Couchbase Server can be deployed on premises, in the cloud, on Kubernetes, or in hybrid configurations. It comes in both open source and enterprise versions.

The Couchbase Server query language, N1QL, is a SQL superset designed for JSON document databases, with extensions for analytics. Couchbase also supports key-value data access and full-text search.

Couchbase, the company behind the database, grew from the merger of Membase (maker of an in-memory cached clustered key-value database) and CouchOne (developers of the Apache CouchDB document database) in 2011. The new company started with the key-value layer, added the JSON document layer in 2012, and went on to add a mobile database in 2014, SQL-like queries in 2015, full-text search in 2017, and analytics in 2018.

Couchbase alternatives and competitors

Alternatives to Couchbase include MongoDB, another flexible document database; MongoDB combined with Redis for caching; Oracle Database, a high-end relational database; and SQL Server, Microsoft’s relational database offering. Relational database systems were designed for use on single, large servers, and it’s hard to scale them out. MongoDB was designed to do master-slave replication, which scales a little, but needs sharding to scale out well. Redis helps to speed up
MongoDB, but introduces another moving part, which can complicate management of the combined systems.

Other recent alternatives to Couchbase include CockroachDB, Azure Cosmos DB, Amazon Aurora, Aerospike, Amazon DocumentDB, and Amazon DynamoDB. I’ve discussed both the relational and NoSQL options in previous reviews.

Couchbase Server architecture

CouchbaseServer performs multiple roles: data service, index service, query service, security, replication, search, eventing, analytics, and management. These services can each be run on one or more nodes.

Couchbase Server has been designed around three basic principles: memory and network-centric architecture, workload isolation, and an asynchronous approach to everything.

Writes are committed to memory, then persisted to disk and indexed asynchronously without blocking reads or writes. The most-used data and indexes are transparently maintained in memory for fast reads. This heavy use of memory is good for latency and throughput, although it increases Couchbase’s RAM requirements.

Couchbase Server can scale each of its services independently, to make them more efficient. The query service can benefit from more CPU resources, the index service can use SSDs, and the data service can use more RAM. Couchbase calls this multi-dimensional scaling (MDS), and it is one of Couchbase Server’s distinguishing features. Asynchronous operations help Couchbase Server to avoid blocking writes, reads, or queries. The developer can balance durability and consistency against latency when needed.

The Couchbase JSON data model supports both basic and complex data types: numbers, strings, nested objects, and arrays. You can create documents that are normalized or denormalized. Couchbase Server does not require or even support schemas. By contrast, MongoDB doesn’t require schemas, but can support and enforce them if the developer chooses.

As I’ll discuss in more detail later on, you can access Couchbase Server documents through four mechanisms: key-value, SQL-based queries, full-text search, and JavaScript eventing. If your JSON documents have subdocuments or arrays, you can access them directly using path expressions without needing to transfer and parse the whole document. The eventing model can trigger on data changes (OnUpdate) or timers. In addition, you can access Couchbase Server documents through synchronization with Couchbase Mobile.

Couchbase Server is organized into buckets, vBuckets, nodes, and clusters. Buckets hold JSON documents. vBuckets are essentially shards that are automatically distributed across nodes. Nodes are physical or virtual machines that host single instances of Couchbase Server. Clusters are groups of nodes. Synchronous replication occurs between the nodes in a cluster.

Couchbase Server deployment options

You can install Couchbase Server on premises, in the cloud, and on Kubernetes. Couchbase Server Enterprise Edition is free for development and testing and available by subscription for production. The open source Couchbase Server Community Edition is free for all purposes. Aside from some omitted features, Couchbase Server Community Edition is API-compatible with Couchbase Server Enterprise Edition.

I created a cloud test drive session on Google Cloud Platform, which (after a five-minute deployment delay) gave me a three-node Couchbase Server cluster and a Sync Gateway node, all good for three hours. I needed about one hour to go through the four Couchbase tutorials, which gave me a feel for querying the server.

Couchbase query tools

You can query Couchbase Server using a key to retrieve the associated value, which can be a JSON document or a Blob. You can also query it with the SQL-like N1QL language or with a full-text search. Both N1QL and full-text queries go faster if the bucket has indexes to support the query.

N1QL

N1QL, pronounced “nickel,” looks very much like standard SQL, with extensions for JSON. I found it much easier to pick up than MongoDB’s aggregation pipeline, given that I’ve been using SQL for decades.
There are actually two similar variants of N1QL: one for the Couchbase Server Query service, and one for the Analytics service, which is an Enterprise Edition feature. N1QL for Analytics is based on SQL++.

Some of the N1QL extensions are USE KEYS, NEST, UNNEST, and MISSING. USE KEYS and USE HASH are query hints for JOINs. NEST and UNNEST pack and unpack arrays. MISSING is a JSON-specific alternative to NULL. IS NOT MISSING means that a specific value is present or NULL in a document. The keyword for values that are NOT MISSING and NOT NULL is KNOWN. N1QL queries can use paths, which also apply to full-text searches.

Full-text search
Couchbase supports external full-text search engines, such as Solr, but it also has its own Go-based, full-text search engine, Bleve. Bleve is included in Couchbase Mobile as well as Couchbase Server, and it supports most of the search syntaxes you’d expect.

Couchbase SDKs
All of the main Couchbase services are exposed for programming through the SDK. SDKs are available for C/C++, .Net (C#, F#, and Visual Basic .Net), Go, Java, Node.js, PHP, Python, and Scala.

In addition to the SDKs, Couchbase offers tight integration with several frameworks: Spring Data, .NET LINQ, and Couchbase’s own Ottoman Node.js ODM. For example, the following sample query uses Linq2Couchbase:

```java
var context = new BucketContext(ClusterHelper.GetBucket("travel-sample"));
var query = (from a in context.Query<Airline>()
            where a.Country == "United Kingdom"
            select a).
            Take(10);
query.ToList().ForEach(console.WriteLine);
ClusterHelper.Close();
```

Couchbase Mobile
Couchbase Mobile has two parts: Couchbase Lite, which runs on a mobile device, and Couchbase Sync Gateway, which runs on a server node. Couchbase Lite runs on iOS, Android, .Net, and Xamarin, and supports the Swift, Objective-C, Java, Kotlin, and C++ languages.

For example, the following Java code defines a query to run on Android:

```java
Database database = DatabaseManager.getDatabase();
Query searchQuery = QueryBuilder.builder().
  .select(SelectResultExpression.newExpression("airportname"))
  .from(DataSource.database(database)).
  .where(Expression.property("type").equals(String.valueOf("airport")))
  .and(Expression.property("airportname").like(String.valueOf("K")));
```

Couchbase benchmarks
While InfoWorld has not benchmarked Couchbase Server, a third-party (Altoros) has done so using the YCSB JSON and key-value tests and the TPCx-IoT test. The chart below is for the JSON document benchmark. As you can see, Couchbase Server outperformed both MongoDB and DataStax. You can
rerun these benchmarks yourself, as Altoros has supplied all of the required scripts.

Overall, Couchbase Server stacks up well as a NoSQL JSON document database with a SQL-like query language and a full-text search engine, and Couchbase Mobile extends the value proposition to mobile devices. Whether Couchbase makes sense for you depends on your application and requirements.

If you need the reliable schema structure of a relational database, or the connection-orientation of a graph database, then Couchbase will not do what you want. But if you need a globally scalable document database, then Couchbase is a good choice.

Cost: Couchbase Server Community Edition: Free. Couchbase Server Enterprise Edition: Annual subscriptions are priced by node and available at different price points depending on a node’s needed cores and RAM. Development and test nodes are free. Enterprise Edition cloud deployments are available by the hour, with typical software pricing of $0.662/node/hour on AWS for Couchbase Server and $1.641/node/hour for the Mobile Sync Gateway, with a standard template using four server nodes and two sync nodes initially, with autoscaling. Pricing is roughly comparable on Microsoft Azure and Google Cloud Platform. You can also bring your own license and pay only for the cloud resources.


—

Cost: Couchbase Server Community Edition: Free. Couchbase Server Enterprise Edition: Annual subscriptions are priced by node and available at different price points depending on a node’s needed cores and RAM. Development and test nodes are free. Enterprise Edition cloud deployments are available by the hour, with typical software pricing of $0.662/node/hour on AWS for Couchbase Server and $1.641/node/hour for the Mobile Sync Gateway, with a standard template using four server nodes and two sync nodes initially, with autoscaling. Pricing is roughly comparable on Microsoft Azure and Google Cloud Platform. You can also bring your own license and pay only for the cloud resources.


---

JSON Benchmark as run by Altoros, comparing Couchbase Server, MongodDB, and DataStax. As you can see, in this test Couchbase provided higher throughput and lower latency, which improved as the data and server were scaled.

---

**At a Glance**

**Couchbase Server 6.02**

Couchbase Server is a flexible, memory-first, distributed JSON document database that is strongly consistent within a local cluster. A mobile version, Couchbase Lite, can run offline and sync to the server when connected.

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scales both vertically and horizontally</td>
<td>There is no way to apply a schema</td>
</tr>
<tr>
<td>Can scale different services independently for maximum performance</td>
<td>Some of the best features are available only in Enterprise Edition</td>
</tr>
<tr>
<td>Mobile database can sync with server when connected and run independently when not connected</td>
<td>Eventually consistent across clusters</td>
</tr>
<tr>
<td>Document model (JSON) is flexible and doesn’t need a schema</td>
<td>Strongly consistent within a cluster</td>
</tr>
<tr>
<td>NoSQL is SQL-like and easy to learn</td>
<td></td>
</tr>
</tbody>
</table>

Martin Heller is a contributing editor and reviewer for InfoWorld. Formerly a web and Windows programming consultant, he developed databases, software, and websites from 1986 to 2010. More recently, he has served as VP of technology and education at Alpha Software and chairman and CEO at Tubifi.