The Rise of Distributed SQL

Jim Hatcher, Solutions Engineer
hatcher@cockroachlabs.com
Dec 6, 2023

Agenda

Relational Databases

NoSQL - Cassandra

Distributed SQL - CockroachDB

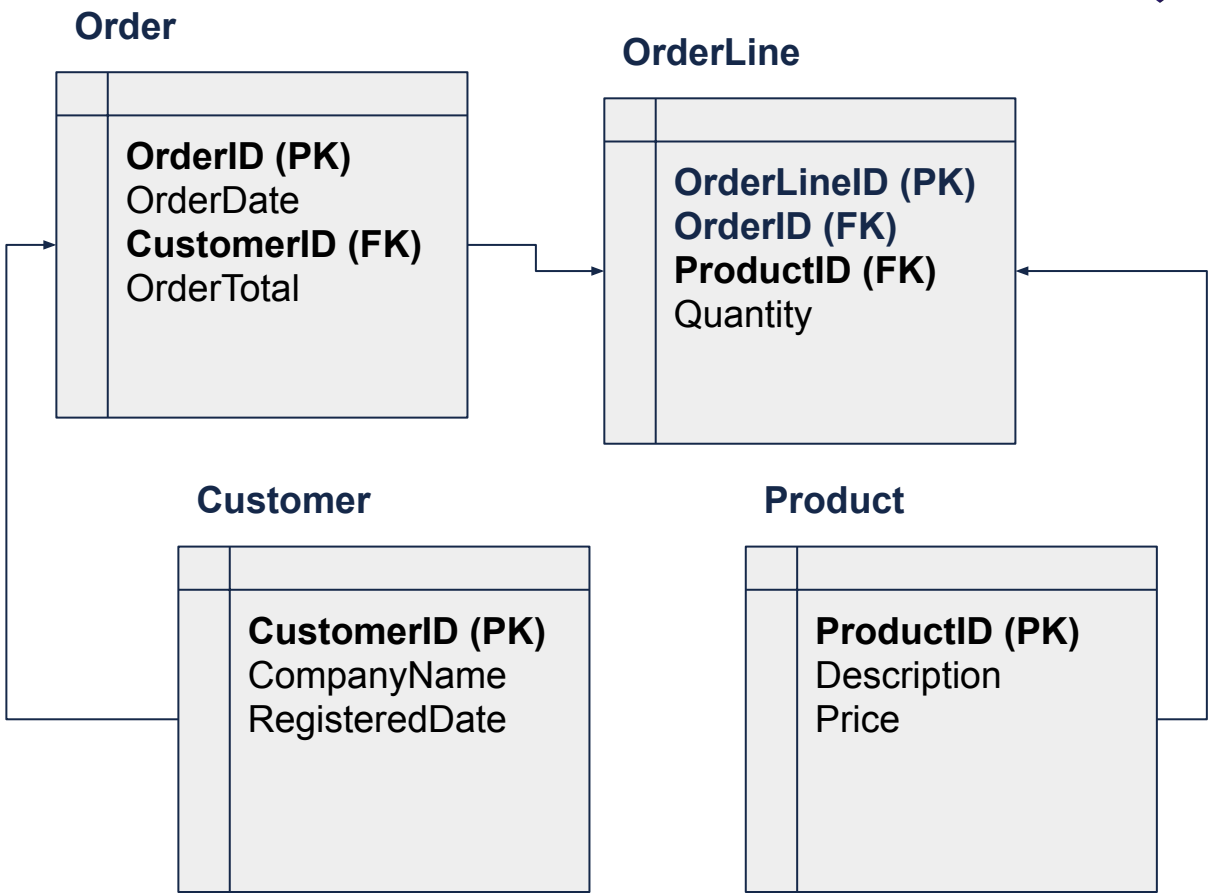
RDBMS – Data Modeling

Well-defined Schema
Structured data; strong typing

3rd Normal Form (3NF)
i.e., every piece of data is stored one time

Referential Integrity
Use foreign keys to enforce data integrity across tables

Constraints
Unique, null, FK, custom



RDBMS - ACID Compliance

Atomicity
All the data is committed or none of the data is committed.

Consistency
Writes do not violate defined rules (constraints, FKs, etc.)

Isolation
Concurrent transactions behave according to the rules of the isolation level.

Durability
Writes get written to disk and will survive a server crash.

```
BEGIN;

  INSERT INTO Order ( OrderID, OrderDate,
    CustomerID, OrderTotal )
  VALUES ( 1000, '2023-01-07', 9999, 357.00 );

  INSERT INTO OrderLine ( OrderLineID, OrderId,
    ProductID, Quantity )
  VALUES ( 20001, 1000, 3001, 4 );

  INSERT INTO OrderLine ( OrderLineID, OrderId,
    ProductID, Quantity )
  VALUES ( 20002, 1000, 3002, 3 );

COMMIT;
```

RDBMS – SQL Queries

SQL
Declarative language for specifying what data you want to read or what data you want to write

JOINs
Let you combine data from different tables

```
SELECT
  O.OrderID, O.OrderDate, C.CompanyName,
  P.Description, P.Price, OL.Quantity
FROM Order O
INNER JOIN OrderLine OL
  ON O.OrderID = OL.OrderID
INNER JOIN Product P
  ON OL.ProductID = OL.ProductID
INNER JOIN Customer C
  ON O.CustomerID
WHERE
  C.CompanyName = 'Acme'
  AND
  O.OrderDate BETWEEN '2023-01-01' AND
   '2023-02-01'
```

RDBMS – Query Performance Tuning

Cost-Based Optimizer
Create efficient query plans based on statistics kept about data in the tables

Indexes
Separate data structures that can be used by the query planner to speed up query execution

```
CREATE INDEX ON Customer ( CompanyName );
CREATE INDEX ON Order ( OrderDate );
```

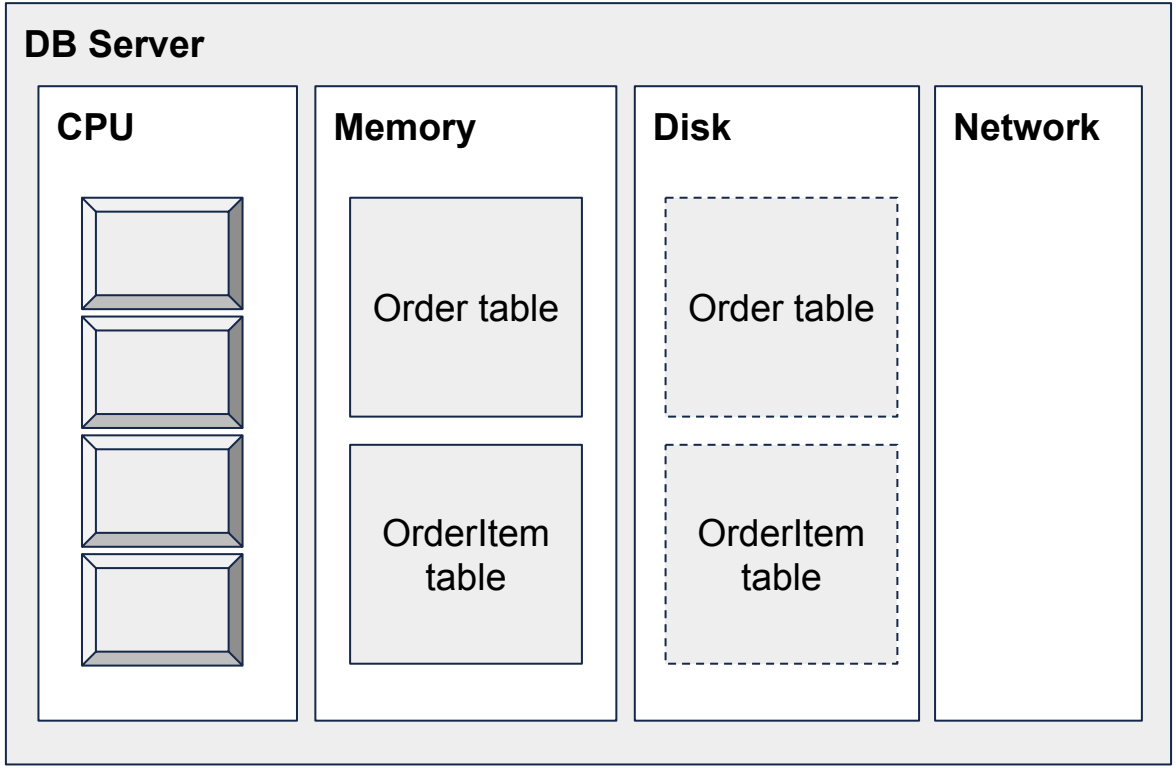
RDBMS – Relies on a single memory space

Locking
Updates to data require locking records

Scaling Model
Increase DB capacity via vertical scaling (i.e., add more resources to the server)

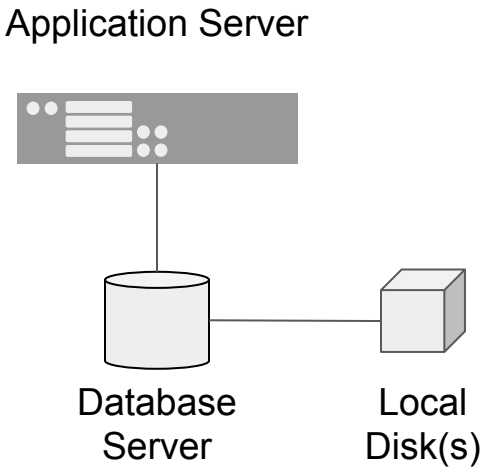
Scale Ceiling
At a certain size, you can't add more CPU, RAM, or disk

Hardware
Expensive, exotic



RDBMS – High Availability

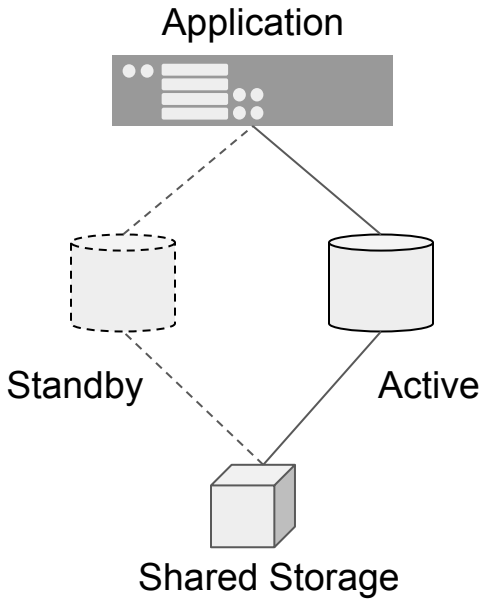
Survive Disk Outage
RAID



RDBMS – High Availability

Survive Disk Outage
RAID

Survive Hardware Outage
Standby Server



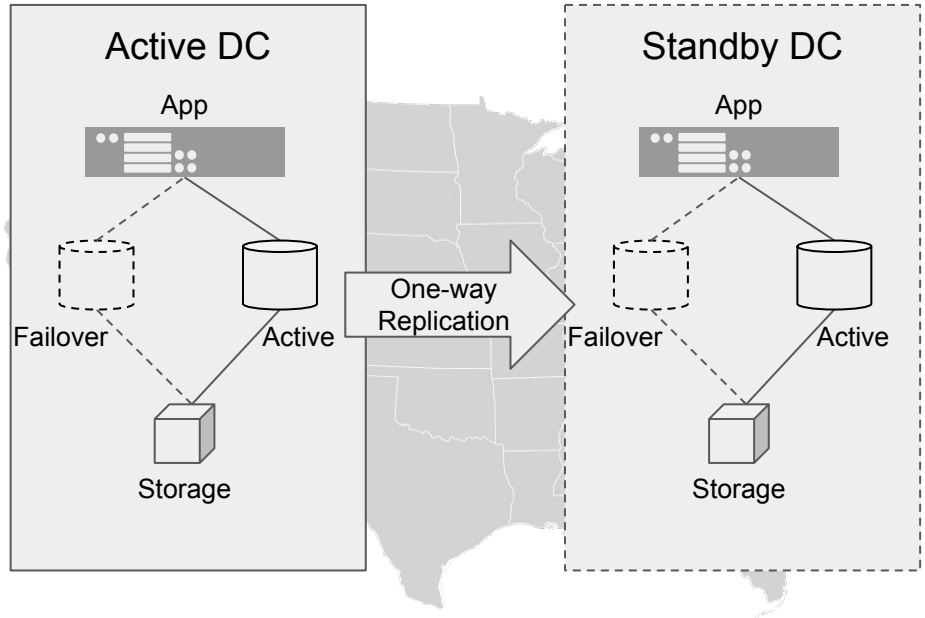
RDBMS – High Availability

Survive Disk Outage
RAID

Survive Hardware Outage
Standby Server

Survive DC Outage
Passive Standby

Active/Passive
Doable



RDBMS – High Availability

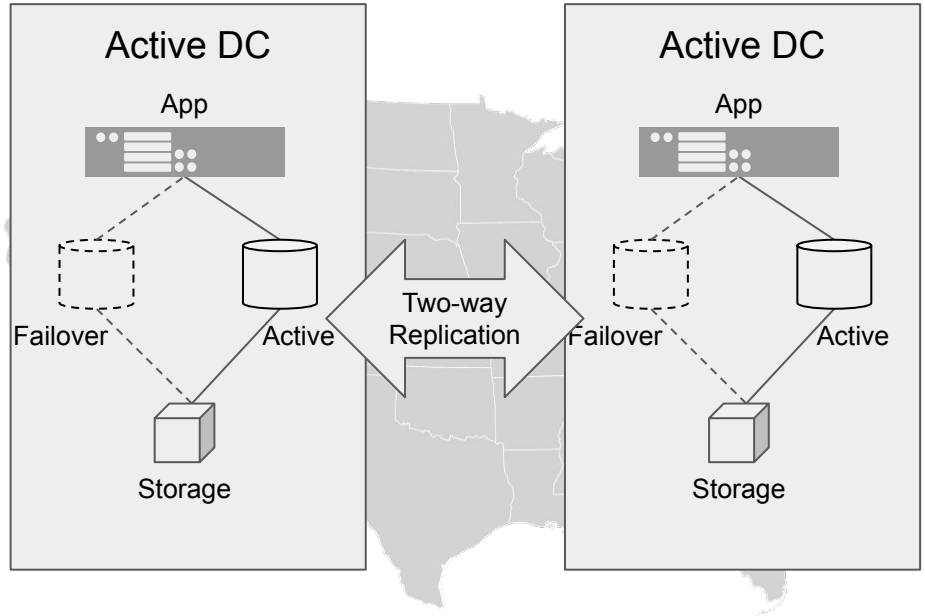
Survive Disk Outage
RAID

Survive Hardware Outage
Standby Server

Survive DC Outage
Passive Standby

Active/Passive
Doable

Active/Active
Hard, expensive



Scorecard

	Data Model	ACID	Lang	Tuning	Scaling	HA	Cloud Native
RDBMS	3NF 💖	Yes 💖	SQL 💖	Indexes 💖	Vertical 👎	Tricky 👎	No 👎
Cassandra							
CockroachDB							

Cassandra - Data Modeling

Well-defined Schema
Structured data; strong typing

Denormalization
Embrace it

Table per Query
Know query patterns ahead of time and model a table for each

Constraints
Enforce outside of database

Consistency
Enforce outside of database

OrderSummary

OrderID (PartKey) **OrderLineId (ClustKey)** OrderDate CustomerID OrderTotal CompanyName RegisteredDate ProductID Description Price Quantity

CompanyOrders

CustomerID (PartKey) **OrderID (ClustKey)** OrderDate OrderTotal CompanyName RegisteredDate

Cassandra - ACID Compliance

Atomicity
Limited to writes to the same partition key value

Consistency
Limited constraints available
Strong consistency in local region
Eventually consistent to other regions

Isolation
Last write wins

Durability
Writes get written to disk and will survive a server crash.

```
--write to the same partition in one batch
BEGIN UNLOGGED BATCH;
INSERT INTO CompanyOrders ( CustomerID, OrderID, …
VALUES ( 1000, 1, … );
INSERT INTO CompanyOrders ( CustomerID, OrderID, …
VALUES ( 1000, 2, … );
APPLY BATCH;

--use lightweight transaction
UPDATE OrderSummary
SET OrderTotal = 100
WHERE OrderID = 10
IF OrderTotal = 90;
```

Cassandra - CQL Queries

SQL
Declarative language for specifying what data you want to read or what data you want to write

JOINs
Not supported

Query by Keys
Filter on defined keys

SQL
Limited support for ORDER BY / GROUP BY

```
SELECT
  OrderID, OrderDate, CompanyName,
  Description, Price, Quantity
FROM OrderSummary
WHERE
  OrderID = 17;

SELECT
  CustomerID, OrderID, OrderDate, OrderTotal,
  CompanyName
FROM CompanyOrders
WHERE
  CustomerID = 59;
```

Cassandra - Query Performance Tuning

Table per query
Create a table with a specific set of keys to handle every query

Indexes
Limited practical use

```
CREATE TABLE CustomerOrder (
  CustomerID int,
  OrderID int,
  OrderDate timestamp,
  OrderTotal decimal,
  CompanyName text,
  RegisteredDate timestamp,
  PRIMARY KEY ( CustomerID, OrderID )
);

SELECT … FROM CustomerOrder
WHERE CustomerID = X;
SELECT … FROM CustomerOrder
WHERE CustomerID = X AND OrderID = Y;
```

Cassandra - Distributed

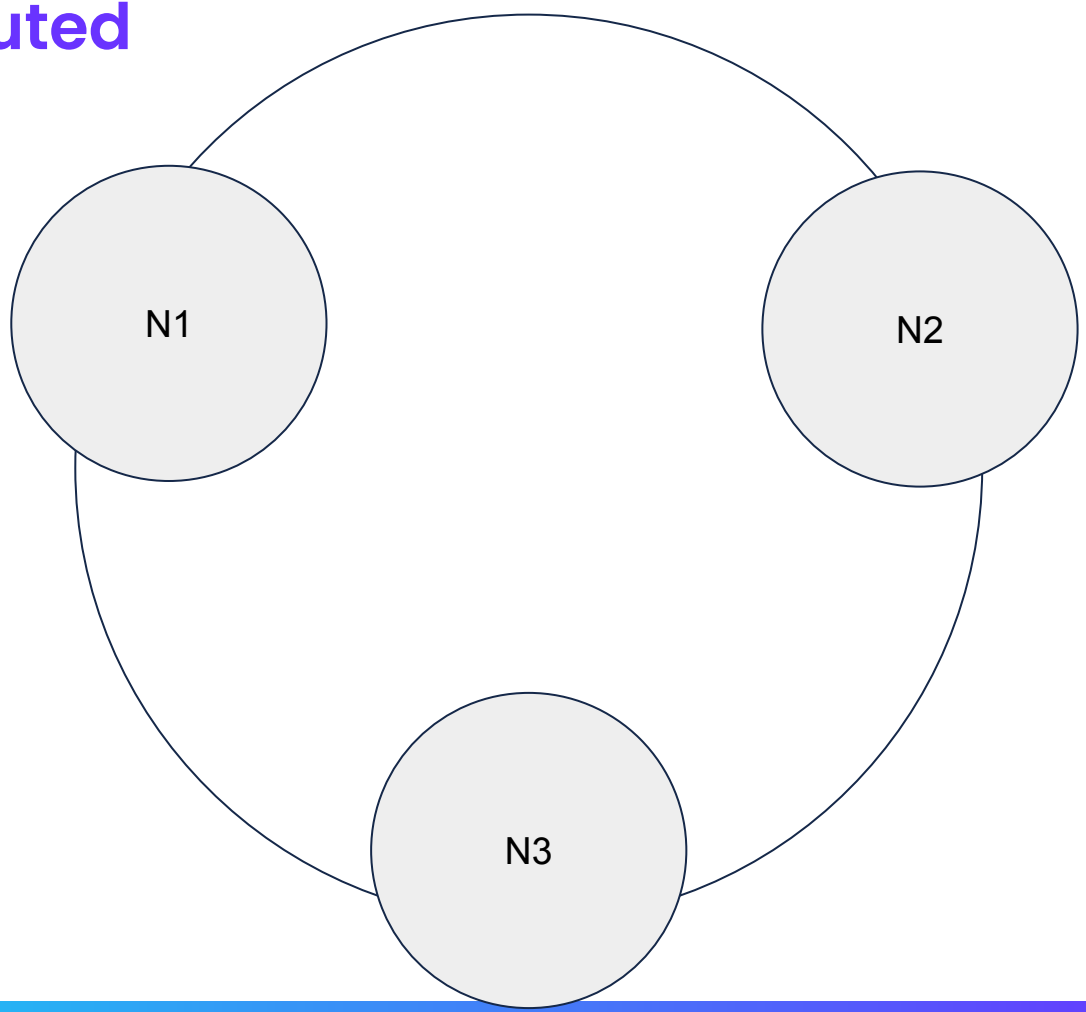
Partitioning
Hash partitioning

Replication Factor
Specify per data center

Scaling Model
Horizontal (i.e., scale out)

Hardware
Commodity

Resiliency
Can operate with nodes down

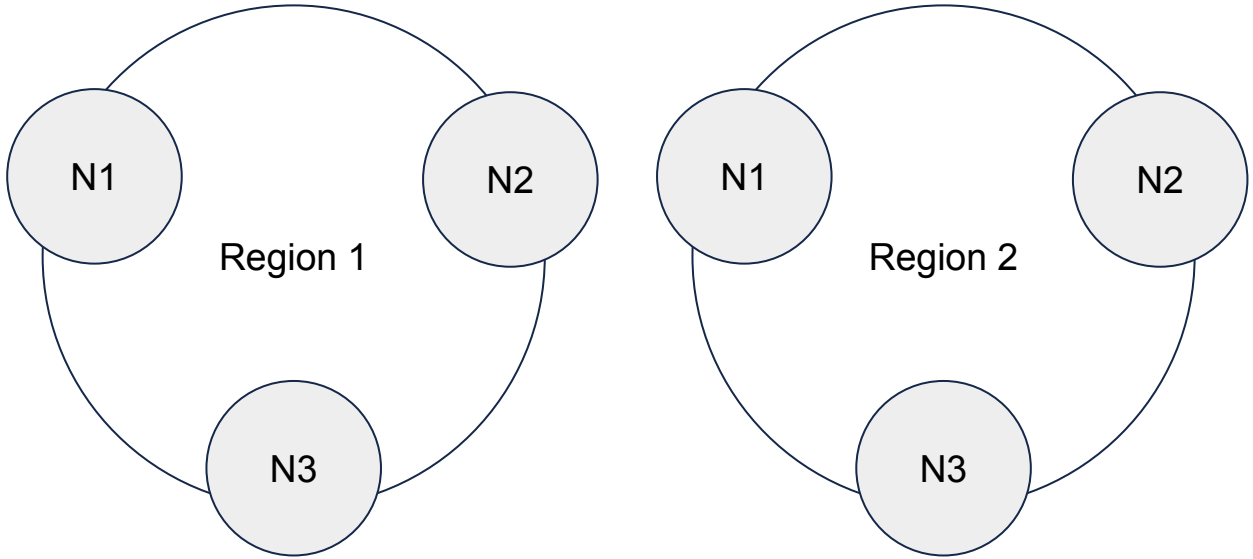


Cassandra - High Availability

Multi Region
Supported

Active / Active
Supported out-of-the-box with last-write wins scheme

Consistency
Strong consistency in the local region; eventual consistency to other regions

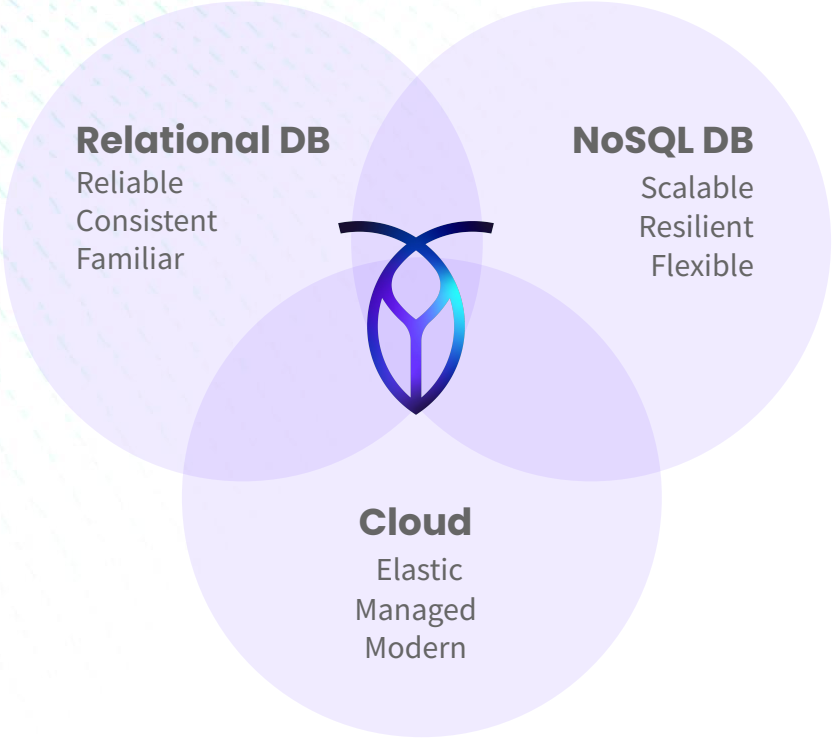


Scorecard

	Data Model	ACID	Lang	Tuning	Scaling	HA	Cloud Native
RDBMS	3NF 💖	Yes 💖	SQL 💖	Indexes 💖	Vertical 👎	Tricky 👎	No 👎
Cassandra	Denorm 👎	Limited 👎	CQL	Add table 👎	Vert/Hor 💖	Flexible 💖	Yes 💖
CockroachDB							

CockroachDB

Distributed SQL



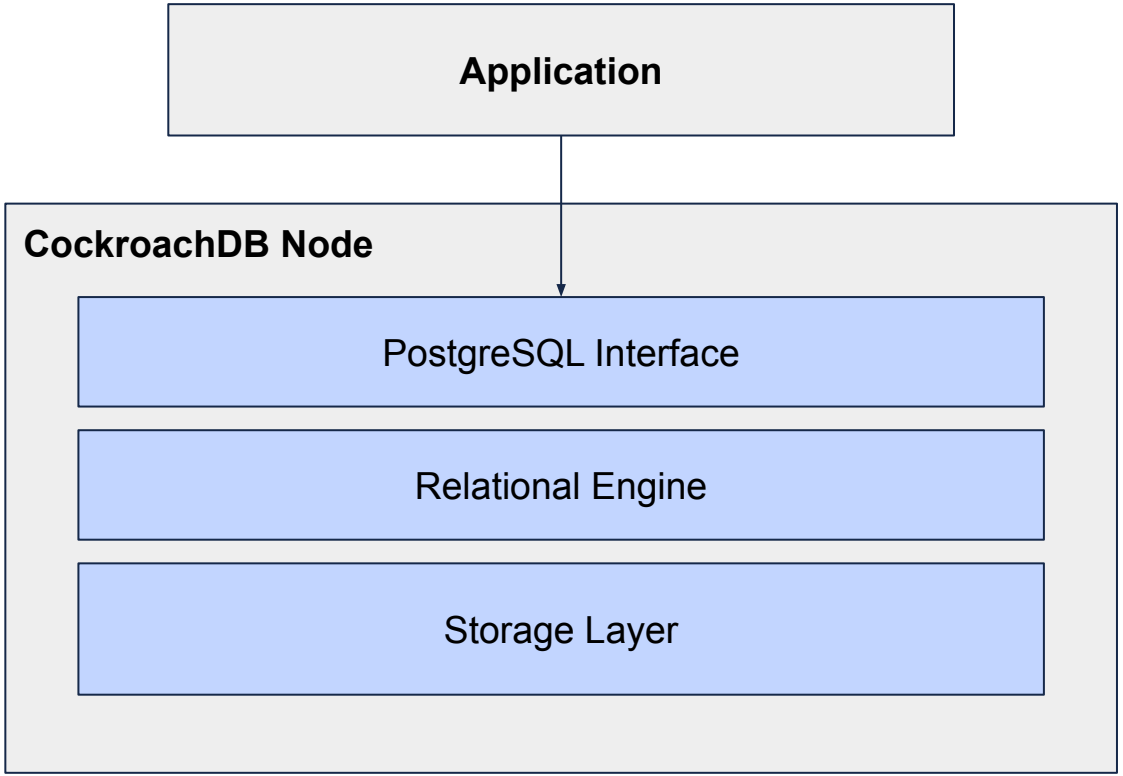
CockroachDB - Architecture

Postgres Interface
Postgres wire compatible

Relation Engine
Written from the ground up to be locality aware

Storage Layer
“Pebble” Key-Value Store

Workload
Designed for transactional workloads



CockroachDB - Distributed

Partitioning
Range partitioning

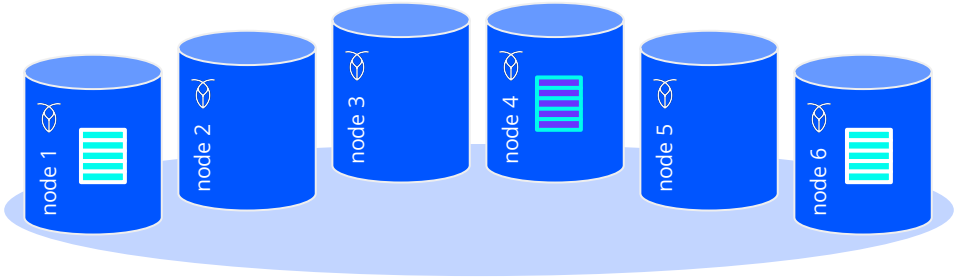
Replication Factor
Specify per cluster

Scaling Model
Horizontal (i.e., scale out)

Consensus
RAFT Protocol

Hardware
Commodity

Resiliency
Can operate with nodes down (need a quorum of replicas to read/write)

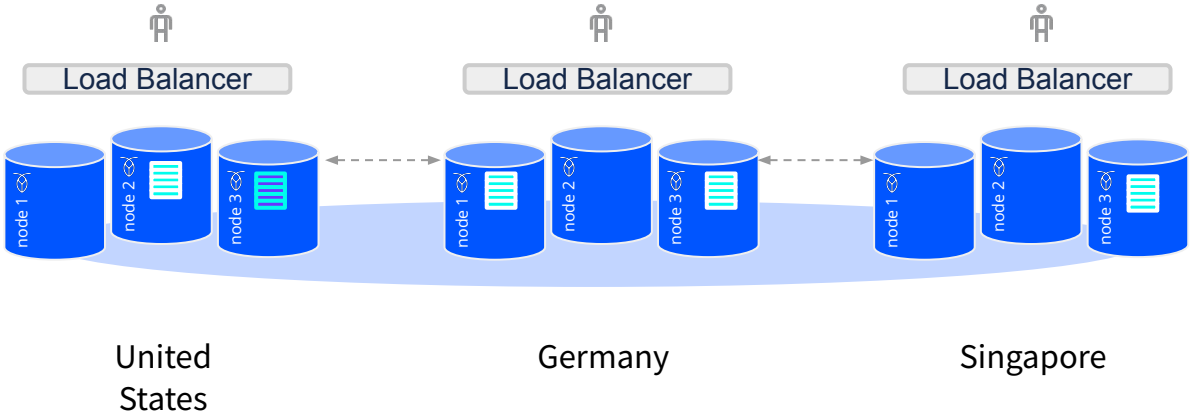


CockroachDB – High Availability

Multi Region
Supported

Active / Active
Supported out-of-the-box

Consistency
Strong consistency across the whole cluster



Scorecard

	Data Model	ACID	Lang	Tuning	Scaling	HA	Cloud Native
RDBMS	3NF 💖	Yes 💖	SQL 💖	Indexes 💖	Vertical 👎	Tricky 👎	No 👎
Cassandra	Denorm 👎	Limited 👎	CQL	Add table 👎	Vert/Hor 💖	Flexible 💖	Yes 💖
CockroachDB	3NF 💖	Yes 💖	SQL 💖	Indexes 💖	Vert/Hor 💖	Flexible 💖	Yes 💖

Questions?