LATENCY IN MULTI-REGION DEPLOYMENTS

LATENCY IN DATABASES

How fast can I **read** my data?

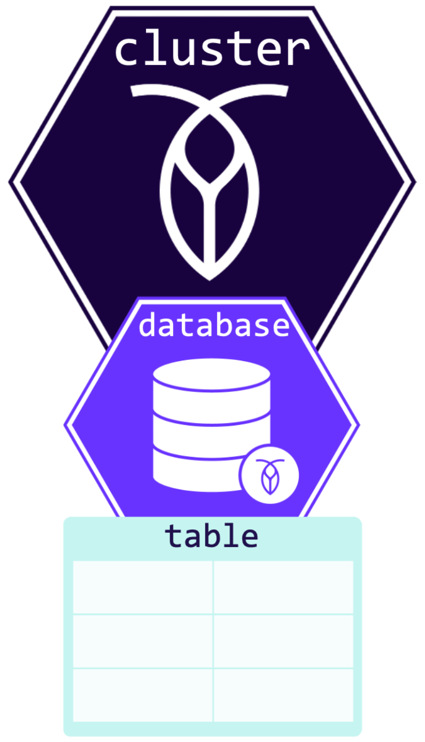
How fast can I **write** my data?

SOME REASONS FOR DATABASE LATENCY

- Slow executing SQL statements.
- Speed of network latency, directly related to where your data is located
  - Time it takes for read-data to travel to users.
  - Time it takes to write data *and reach consensus* .

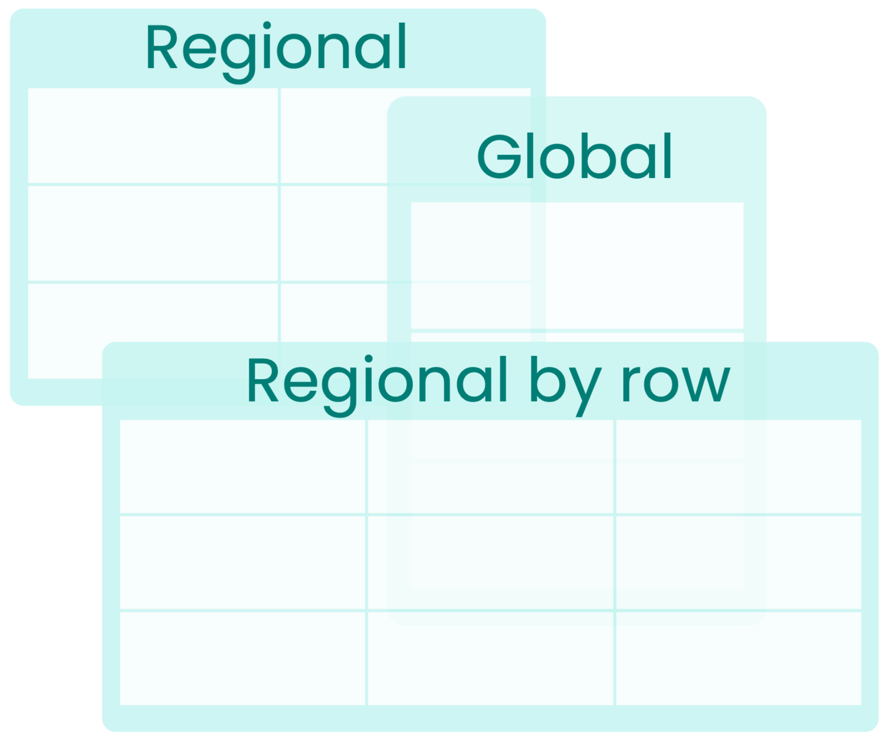
COCKROACHDB LOCALITY SETTINGS ADDRESS SPEED-OF-NETWORK LATENCY.

MULTI-REGION LOCALITY OVERVIEW



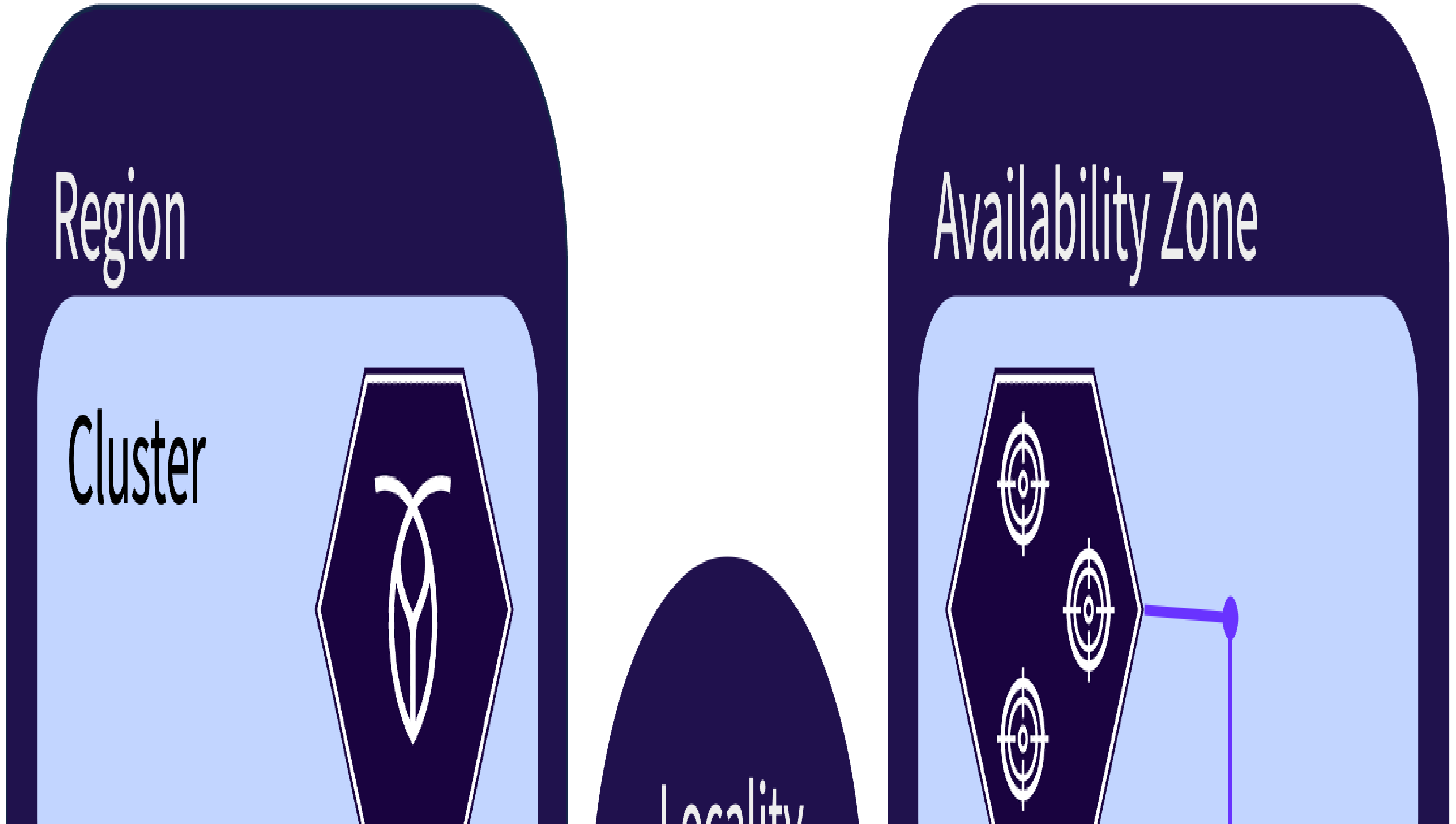
- In CockroachDB, latency is tuned using multi-region `locality` settings.
- Locality is configured at three levels - cluster, database, table.

TABLE LOCALITY



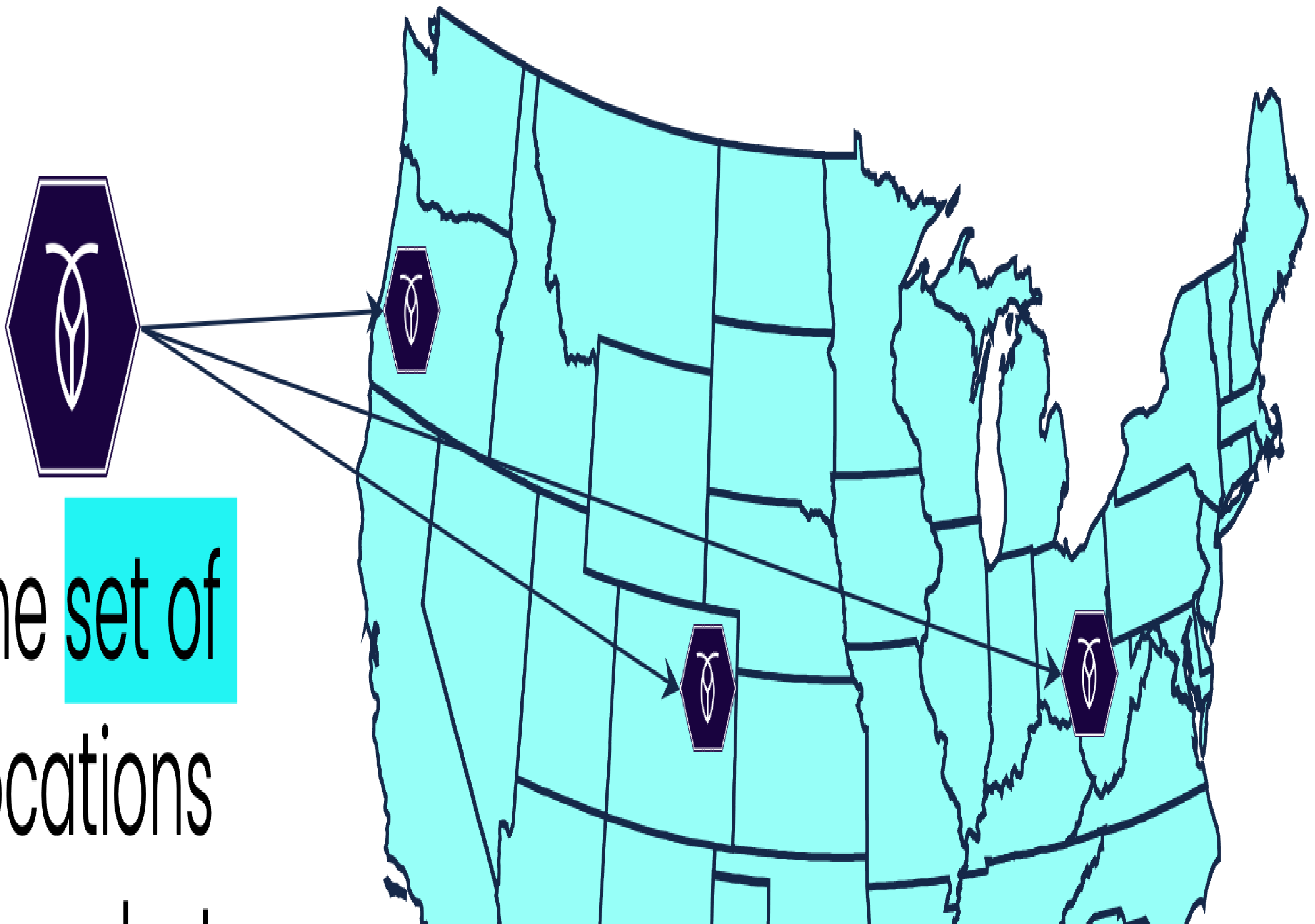
- **Regional tables** - optimized for fast reads and writes from the *home region*.
- **Regional by row tables** - optimized for fast reads and writes from the *region specified at the row level*.
- **Global tables** - optimized for fast **reads** from *all regions*.

COCKROACHDB LOCALITY TERMINOLOGY



REGION -- LOGICAL IDENTIFICATION OF HOW NODES AND DATA ARE CLUSTERED AROUND GEOGRAPHICAL LOCATIONS

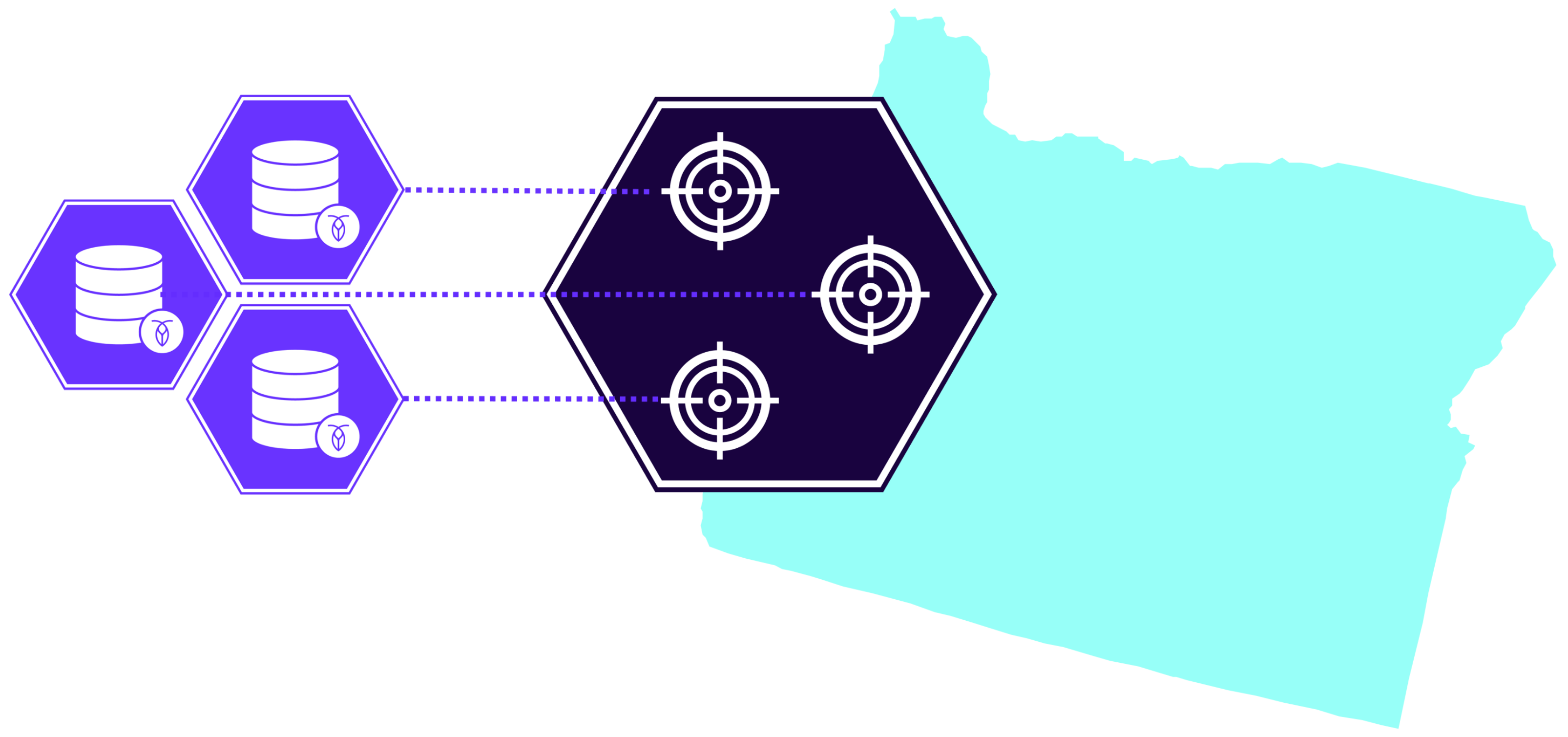
COCKROACHDB CLUSTER REGION



The set of

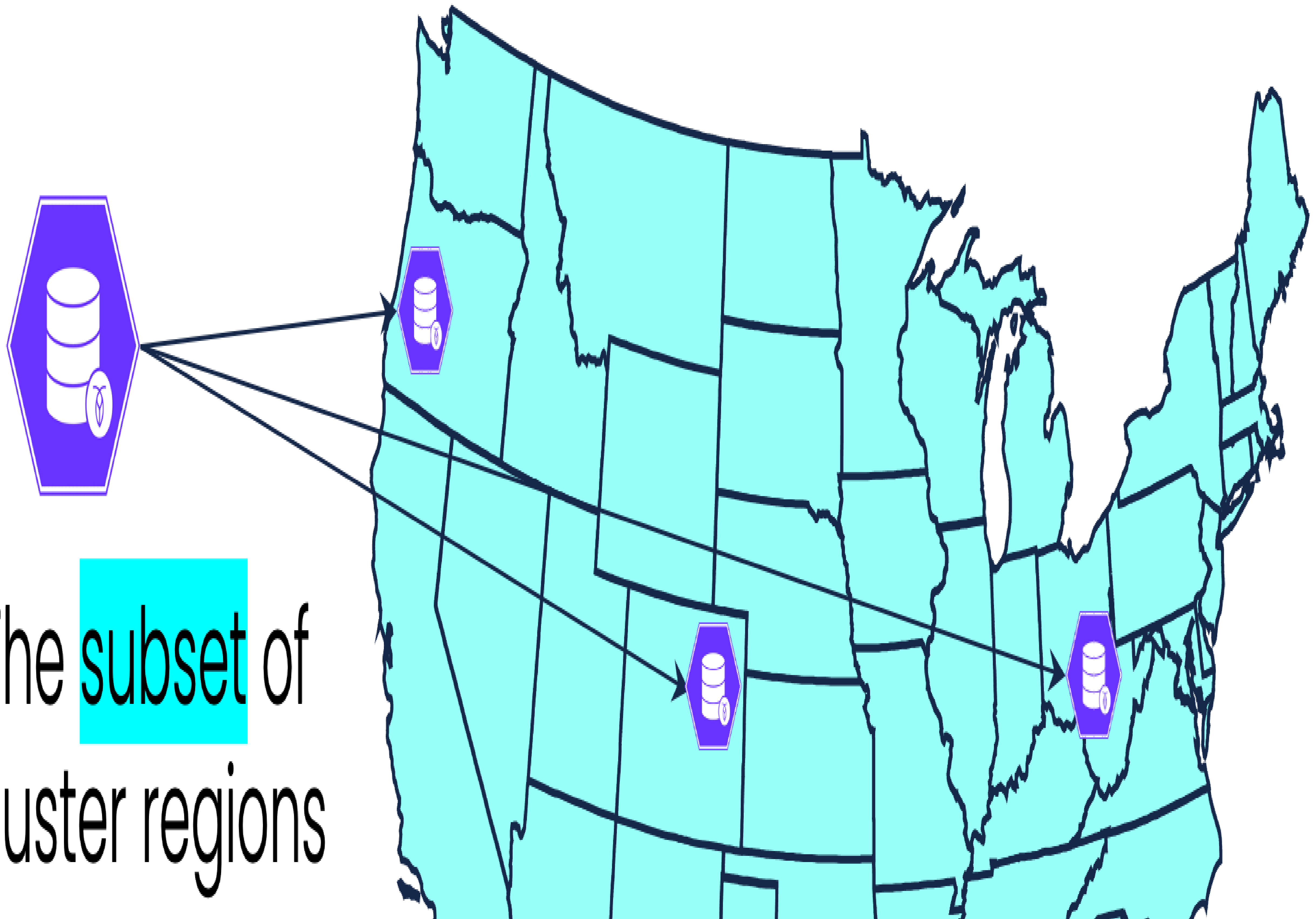
locations

COCKROACHDB AVAILABILITY ZONE



- A zone is a deployment area within a cluster region.
  - A cluster region is made up of a collection of zones.
- There can be multiple nodes in a single availability zone
  - *Recommendation*: place different replicas of your data in different availability zones.

COCKROACHDB DATABASE REGION



The subset of cluster regions

YOUR MULTI-REGION JOURNEY STARTS WITH CONFIGURING YOUR CLUSTER REGIONS

COCKROACHDB CLUSTER REGIONS

- Run CockroachDB on specified infrastructure (physical nodes)
- Define the locality of the nodes available for the databases to use

COCKROACHDB CLUSTER REGIONS

- Specifying a region with a region tier is required in order to enable CockroachDB's multi-region capabilities.
- CockroachDB spreads the replicas of each piece of data across as diverse a set of localities as possible, with the order determining the priority.

COCKROACHDB CLUSTER REGION LIMITATIONS

- Every node requires the same `locality` key-pair settings.
- In self-hosted mode, region locality is not be guaranteed.
  - CockroachDB does not check that a node's physical location matches its user-given locality label.
- You cannot add regions to a CockroachDB Dedicated cluster on the fly.

CONFIGURE CLUSTER REGIONS

cockroach
demo

CockroachDB
Self-hosted

CockroachDB
Dedicated

- Steps to configuring a cluster region depend on your environment

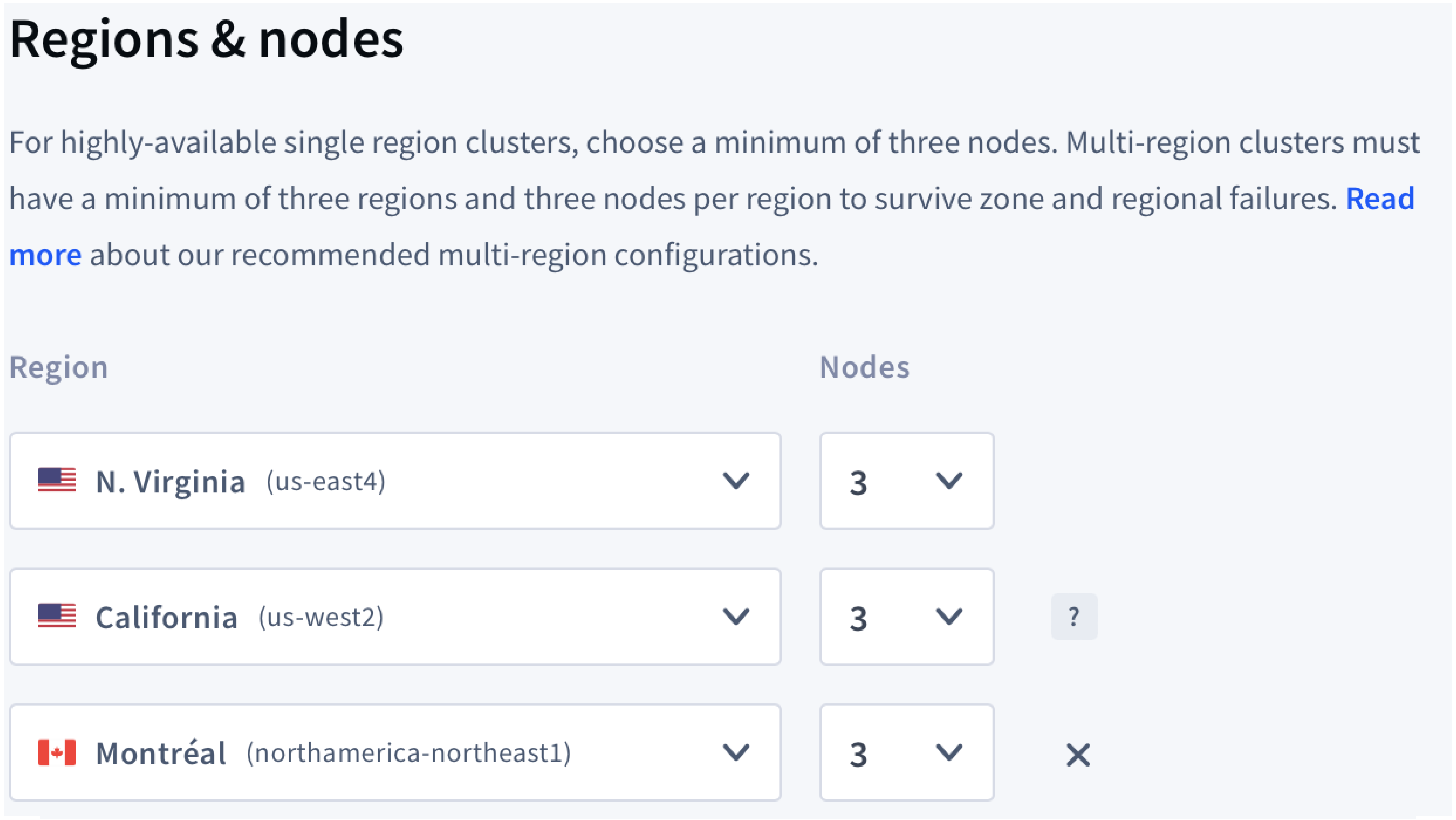
CONFIGURING CLUSTER REGION

- With demo use `cockroach demo`
- With CockrochDB self-hosted use `cockroach start`
- These should be followed by the `locality` flag and any other flags

```
cockroach start --locality= \
region=us-east-1,zone=us-east-1b: \
region=us-east-1,zone=us-east-1a: \
region=us-east-1,zone=us-east-1c: \
region=us-east-2,zone=us-east-2b: \
region=us-east-2,zone=us-east-2a: \
region=us-east-2,zone=us-east-2c: \
region=us-west-1,zone=us-west-1b: \
region=us-west-1,zone=us-west-1a: \
region=us-west-1,zone=us-west-1c \
# ... other required flags go here
```

Example using CockroachDB Self-hosted

CONFIGURING CLUSTER REGION USING COCKROACHDB DEDICATED



Regions & nodes

For highly-available single region clusters, choose a minimum of three nodes. Multi-region clusters must have a minimum of three regions and three nodes per region to survive zone and regional failures. Read more about our recommended multi-region configurations.

Region	Nodes
N. Virginia (us-east4)	3
California (us-west2)	3
Montréal (northamerica-northeast1)	3

AFTER CONFIGURING CLUSTER REGIONS, DATABASE REGIONS SHOULD BE CONFIGURED

DATABASES IN A MULTI-REGION CLUSTER THAT HAVE NOT YET HAD THEIR PRIMARY REGIONS SET WILL HAVE THEIR REPLICAS SPREAD AS BROADLY AS POSSIBLE FOR RESILIENCY.

MULTI-REGION DATABASES

- For database to be multi-region, it requires setting a primary region at minimum.
- When a primary region is added:
  - Tables will be rebalanced. All tables will have their voting replicas and leaseholders moved to the primary region by default.
  - Another name for this is *regional by table*.

MULTI-REGION DATABASES

- Once a database is configured as multi-region:
  - All data is stored within its assigned regions.
  - CockroachDB optimizes access to the database's data from the primary region.

CONFIGURING DATABASE REGIONS

- To add the first region, use the ALTER DATABASE ... PRIMARY REGION statement

```
ALTER DATABASE {database} SET PRIMARY REGION "us-east1";
```

- To add additional regions use the following syntax

```
ALTER database {database} ADD region "europe-west1";
```

- Regions can also be dropped from a database

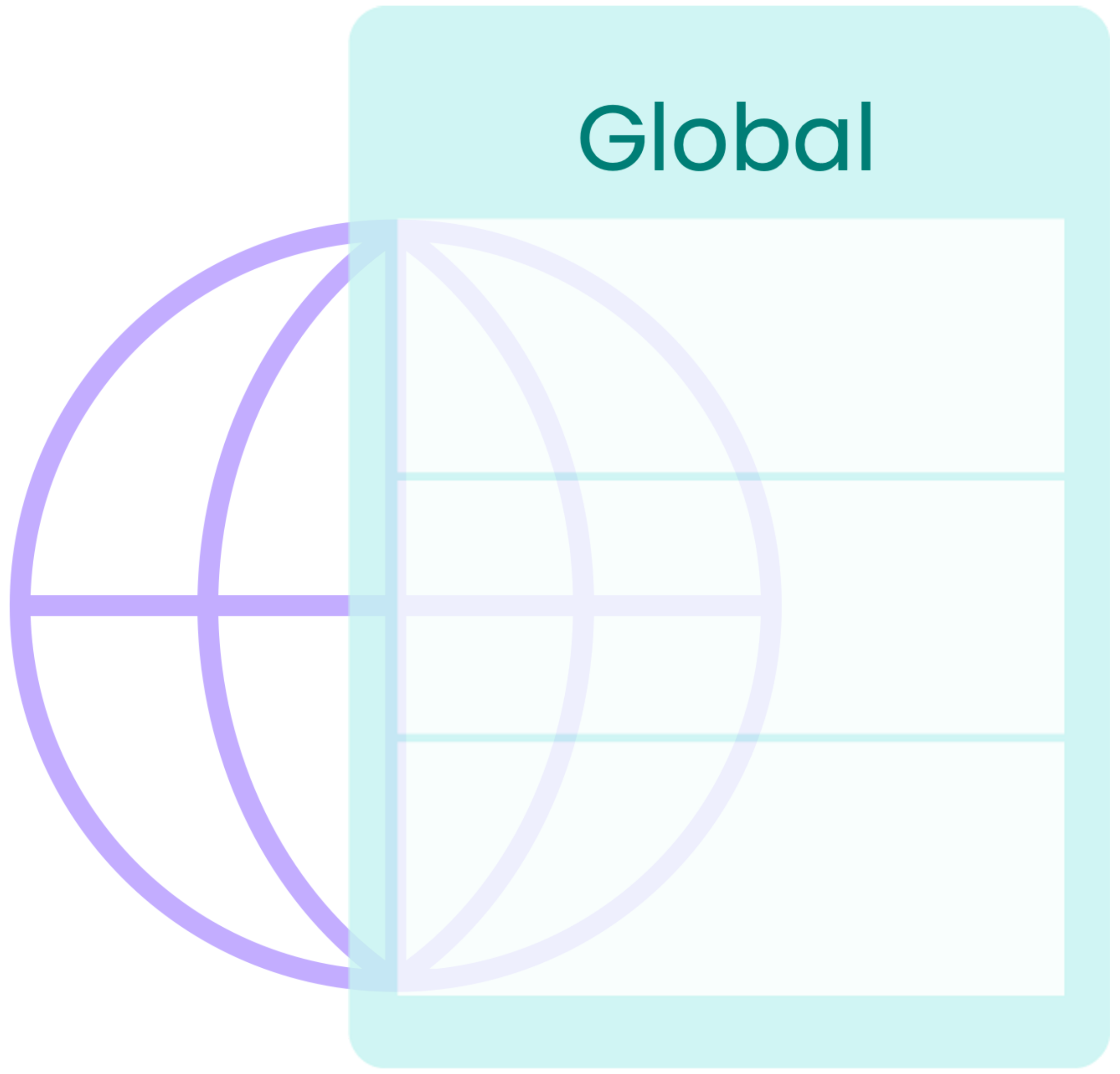
```
ALTER DATABASE {database} DROP REGION "europe-west1";
```

  - *The primary region can only be dropped if it's the last region remaining region*

THE *TABLE LOCALITY* INDICATES HOW COCKROACHDB OPTIMIZES ACCESS TO A TABLE'S DATA IN A MULTI-REGION CLUSTER.

GLOBAL TABLES

GLOBAL TABLE LOCALITY



- Tables optimized for fast reads from all regions.
- Global tables have slightly slower data write speeds.
- Global tables are useful in cases where data is read a lot more than it is written.

WHEN TO USE GLOBAL TABLES

- Reads cannot be stale for business or technical reasons (foreign key constraints).
- Your use case can accept higher write latency.
- Latency-sensitive reads cannot be tied to specific regions.

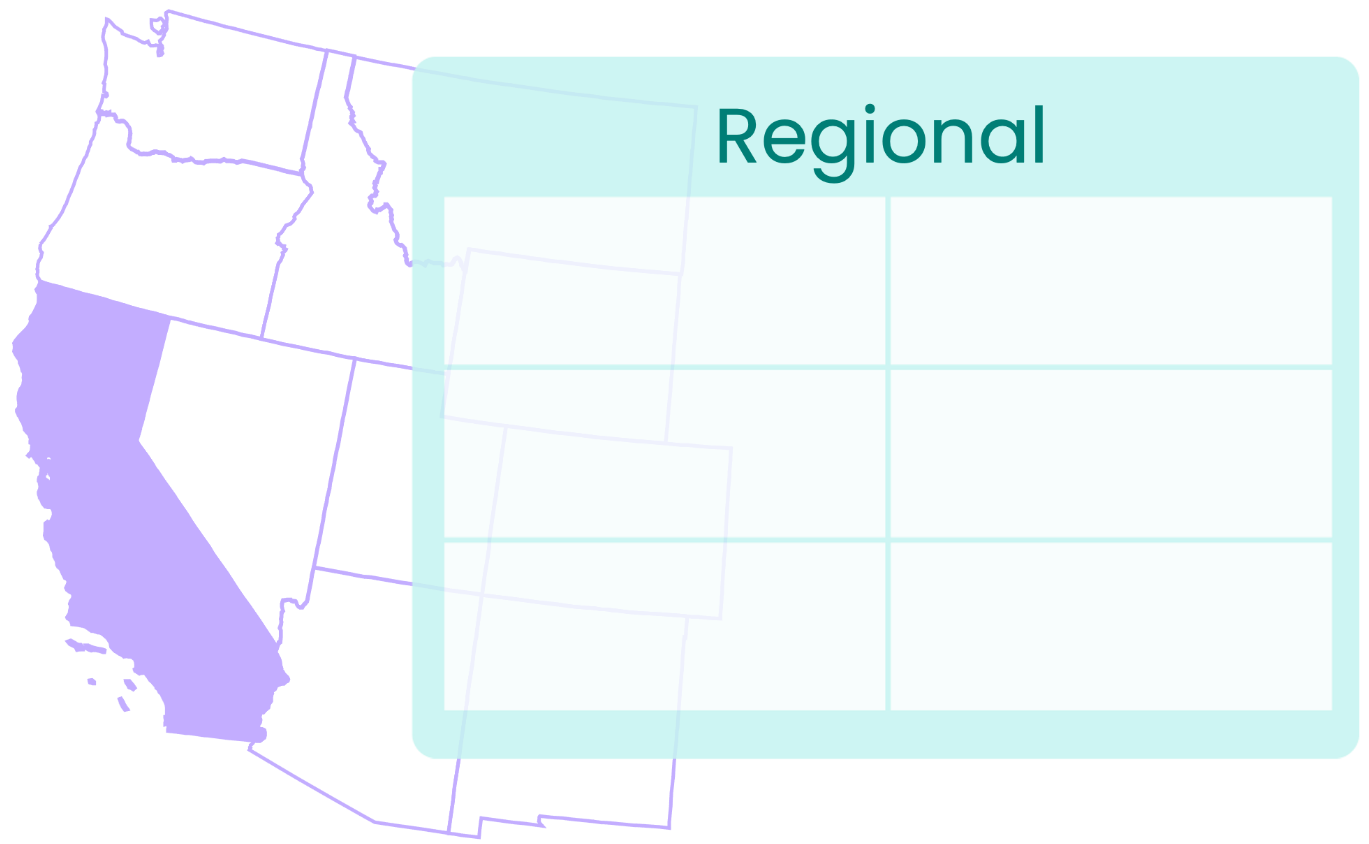
GLOBAL TABLES ARE A GOOD CHOICE FOR RARELY UPDATED LOOK-UP OR REFERENCE TABLES.

GLOBAL TABLE LOCALITY SYNTAX

```
ALTER TABLE {table} SET LOCALITY GLOBAL;
```

REGIONAL TABLES

REGIONAL TABLE LOCALITY



- Regional is the default locality for all multi-region database tables.
- A regional tables is optmized for fast reads and writes from its home region.
- Access to regional table data from other regions will be slower.

REGIONAL TABLE LOCALITY SYNTAX

- A regional table's home region defaults to the database's primary region. If you were to do this manually, the syntax would look like this:

```
ALTER TABLE {table} SET LOCALITY REGIONAL BY TABLE IN PRIMARY REGION;
```

- You can also set the locality without specifying the region. This will default again to the database's primary region.

```
ALTER TABLE {table} SET LOCALITY REGIONAL BY TABLE;
```

- To specify a non-default region, use the following syntax:

```
ALTER TABLE {table} SET LOCALITY REGIONAL BY TABLE IN "us-east-1";
```

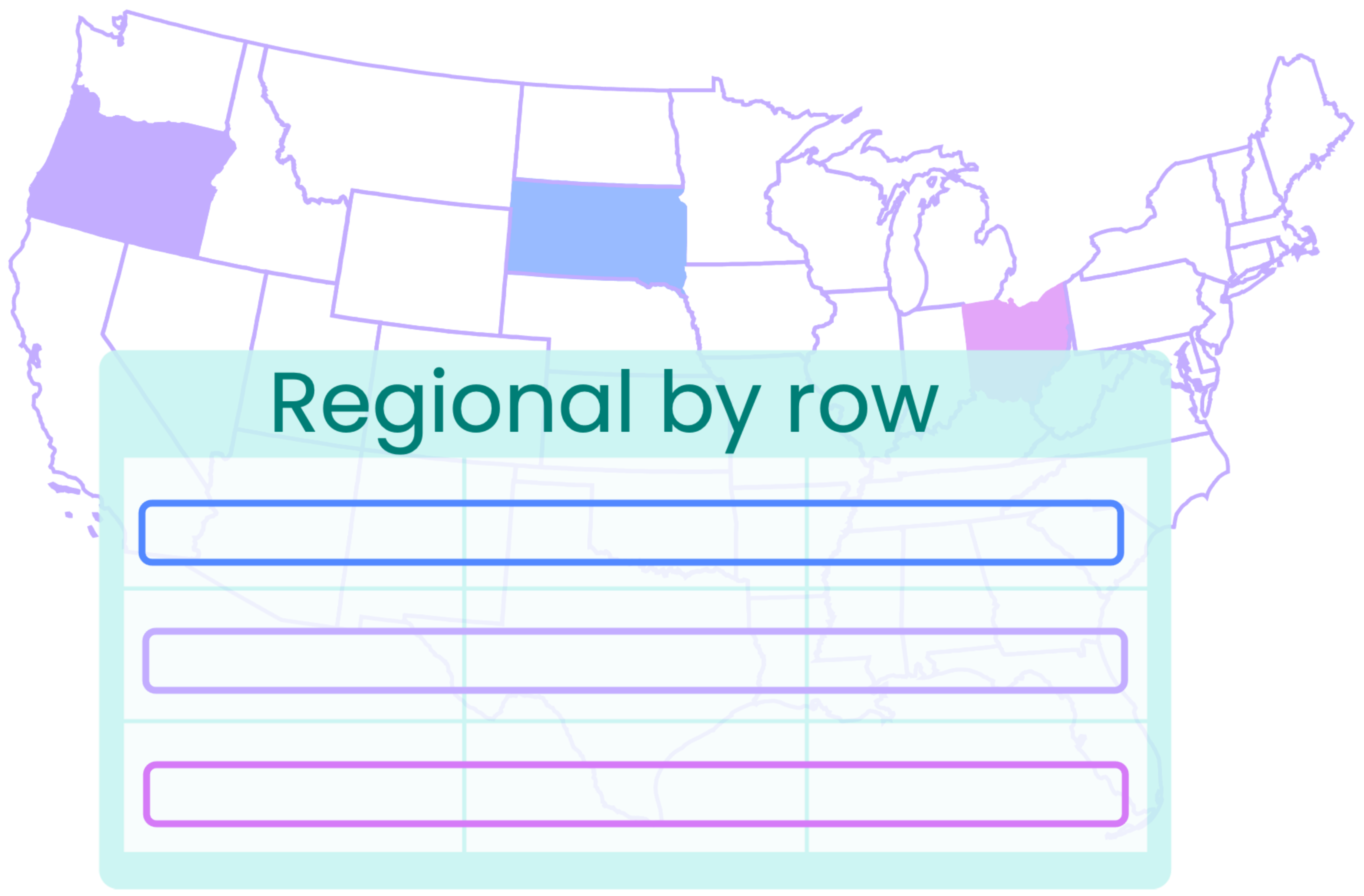
REGIONAL BY TABLE LIMITATIONS

- Regional by table is the default and may not meet your needs if:
  - You need fast reads from all regions.
  - You need fast reads and writes from more than one region.

REGIONAL BY ROW TABLES

WHEN YOU NEED FAST READS AND WRITES FROM MORE THAN ONE REGION.

REGIONAL BY ROW TABLE LOCALITY



- Assigns a region to each row
- Optimized for fast reads and writes from the region assigned to the row
- The table and its indexes are automatically partitioned on the region column.
- Each partition is optimized for access from a specific region.

REGIONAL BY ROW TABLE LOCALITY SYNTAX

- To make an existing table a regional by row table, use the following statement:

```
ALTER TABLE {table} SET LOCALITY REGIONAL BY ROW;
```

- This will create a hidden `crdb_region` column that represents the row's home region.
- You can use a column name other than crdb_region for the hidden column:

```
ALTER TABLE {table} SET LOCALITY REGIONAL BY ROW AS {column_name};
```

POPULATING THE `crdb_region` COLUMN

- For existing data, you can manually update the `crdb_region` column.
  - Example:

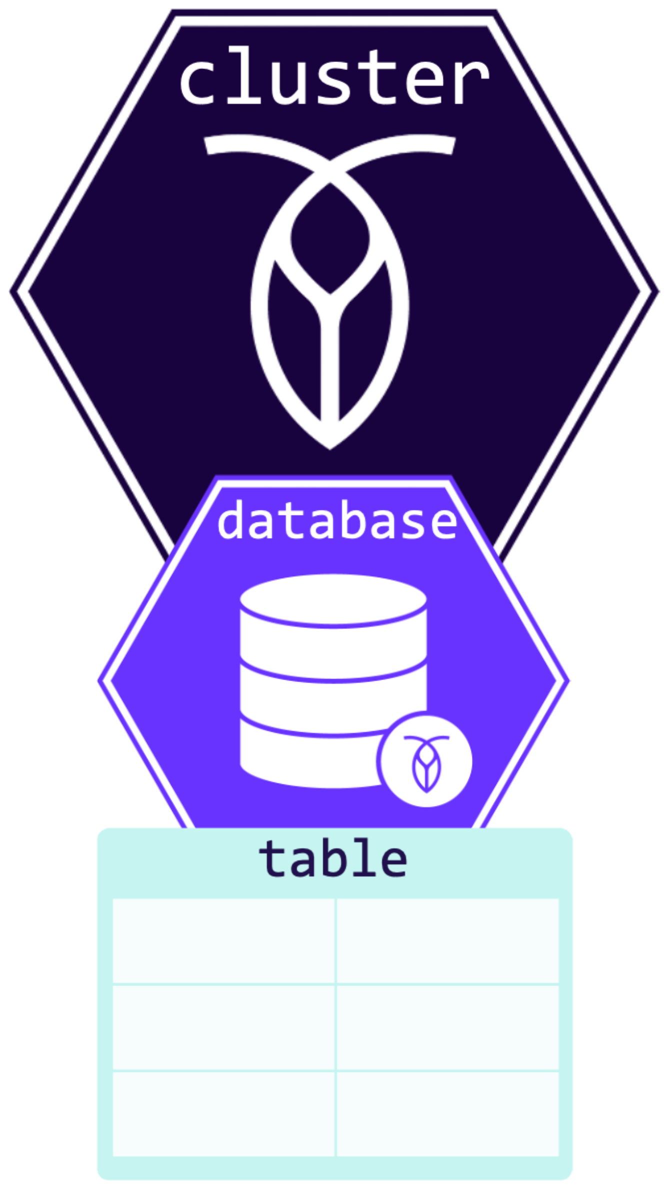
```
UPDATE {table} SET crdb_region = 'us-west' WHERE city IN (...)
```

- For new inserts there are two options for populating the `crdb_region` field.
  - CockroachDB can set it automatically using the region of the gateway node (`gateway_region()`)
  - You can manually insert the data using the following:

```
INSERT INTO {table} (crdb_region, ...) VALUES ('us-east', ...);
```

SUMMARY - LATENCY AND LOCALITY

- CockroachDB multi-region allows you to address latency related to where your data is located.
- The concept of locality is present at the cluster, database and table level.
- A primary region must be added to a database to make it multi-region.



SUMMARY - TABLE LOCALITY

- *Global* table locality optimizes for fast read access from all regions with slightly slower write access.
- The default table locality, *regional*, optimizes for fast access from a single region.
- *Regional by row* table locality optimizes for fast access from a single region *specified at the row level*.