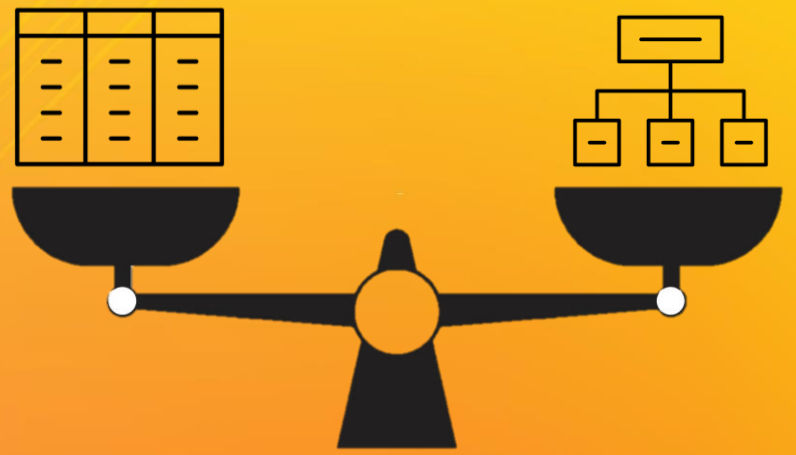
Databases in Cloud



SQL vs No-SQL Databases

Or

Relational vs Non-relational

Let’s cook Banana bread

Banana bread

STEPS

1. Preheat the oven to 350°F (175°C), and butter a 4x8-inch loaf pan.
2. In a mixing bowl, mash the ripe bananas with a fork until completely smooth. Stir the melted butter into the mashed bananas.
3. Mix in the baking soda and salt. Stir in the sugar, beaten egg, and vanilla extract. Mix in the flour.
4. Pour the batter into your prepared loaf pan. Bake for 50 minutes to 1 hour at 350°F (175°C), or until a tester inserted into the center comes out clean.
5. Remove from oven and let cool in the pan for a few minutes. Then remove the banana bread from the pan and let cool completely before serving. Slice and serve.

1 HOUR EASY ONE LOAF

INGREDIENTS

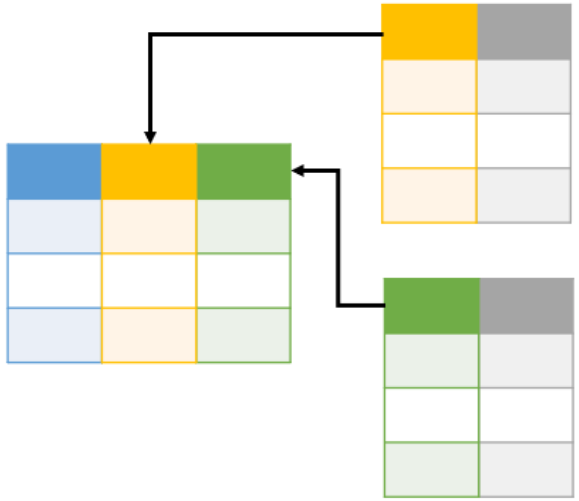
- 2 to 3 very ripe bananas, peeled
- 1/3 cup melted butter, unsalted or salted
- 1 teaspoon baking soda
- Pinch of salt
- 3/4 cup sugar
- 1 large egg, beaten
- 1 teaspoon vanilla extract
- 1&1/2 cups of all-purpose flower

Grocery shopping in a supermarket

SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph

Rows and Columns



Document

```
{
  “Id”: “1”,
  “FullName”:
   {
     “first”: “Jane”,
     “last”: “Doe”
   }
  “Year”: “2022”,
}
```

How storage evolved?

- In early days of computing Storage was very costly

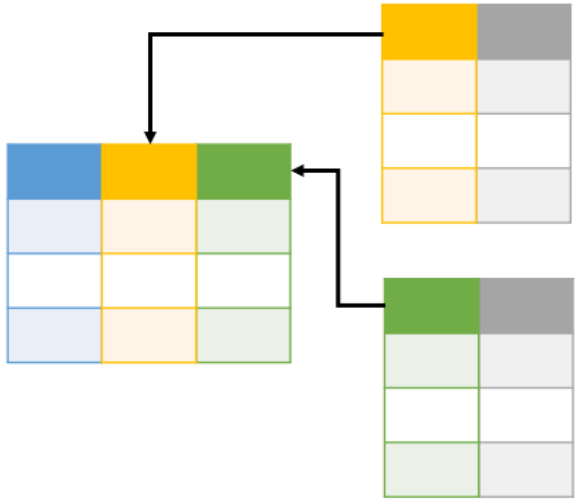
Image Source:
https://www.reddit.com/r/computerscience/comments/ak27u0/ibm_5mb_hard_drive_1956/

Image Source:
https://www.thessdreview.com/featured/micron-c200-microsd-card-review-1tb-as-high-capacity-becomes-the-norm-in-microsd/

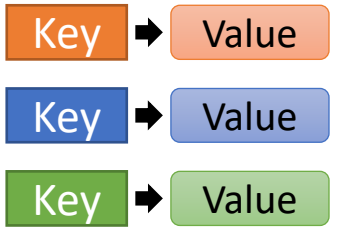
SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph

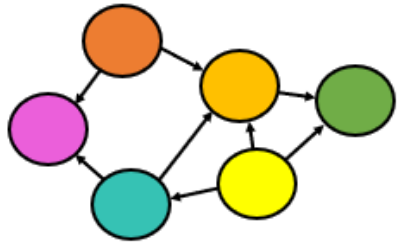
Rows and Columns



Key Value



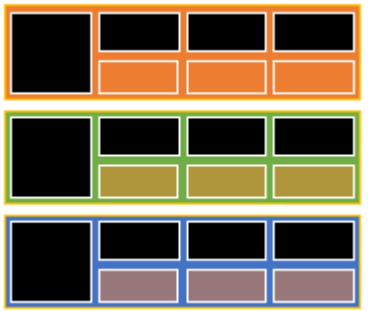
Graph



Document

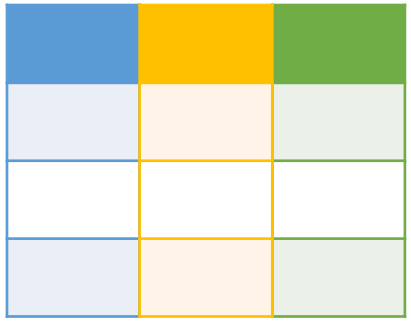
```
{
  “Id”: “1”,
  “FullName”:
   {
     “first”: “Jane”,
     “last”: “Doe”
   }
  “Year”: “2022”,
}
```

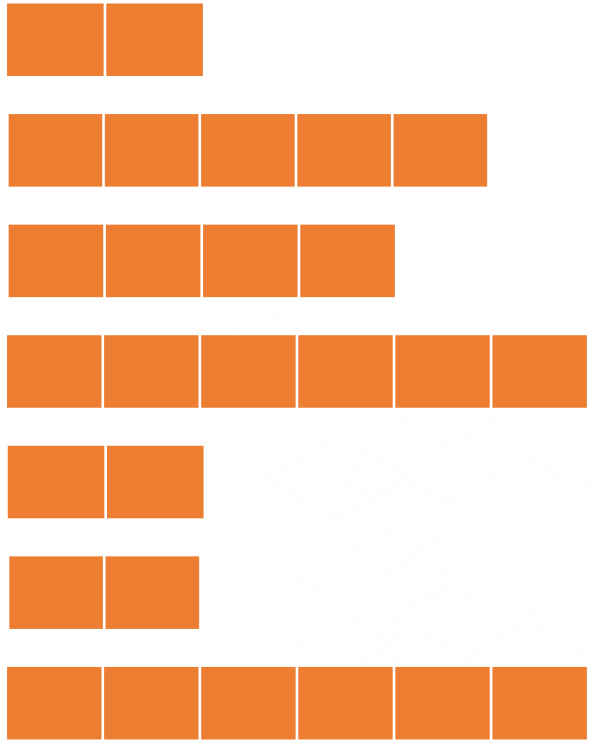
Wide Column



SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic





SQL vs. NoSQL Databases

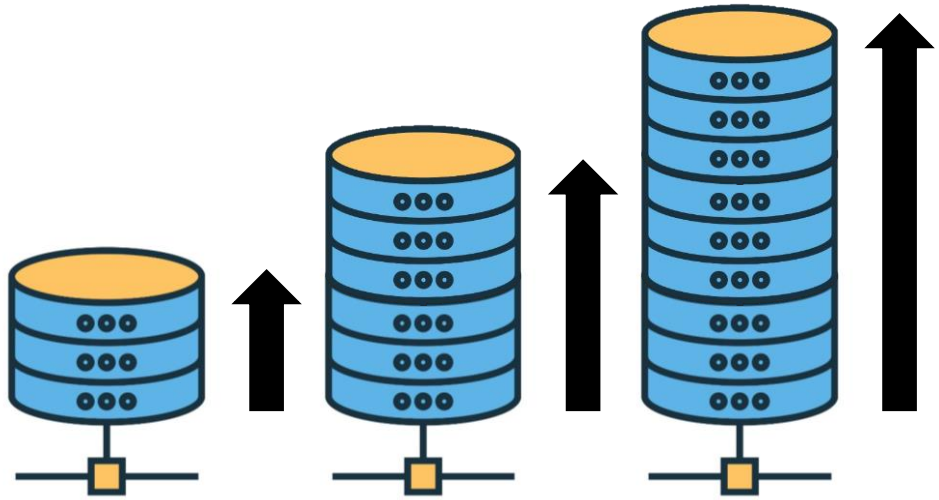
	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents

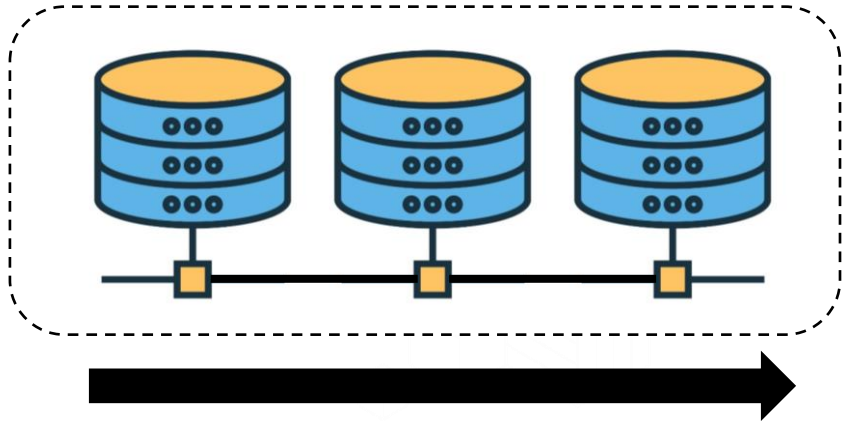
```
/*  Return all of the songs by an artist Elvis  */
SELECT * FROM Music
WHERE Artist= ‘Elvis’;
```

```
/*  Return all of the songs by an artist Elvis  */
{
   TableName: “Music”,
   KeyConditionExpression: “Artist =
:a”,
   ExpressionAttributeValues:
   {
    “:a”: “Elvis”
    }
}
```

SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scaling	Vertical	Horizontal



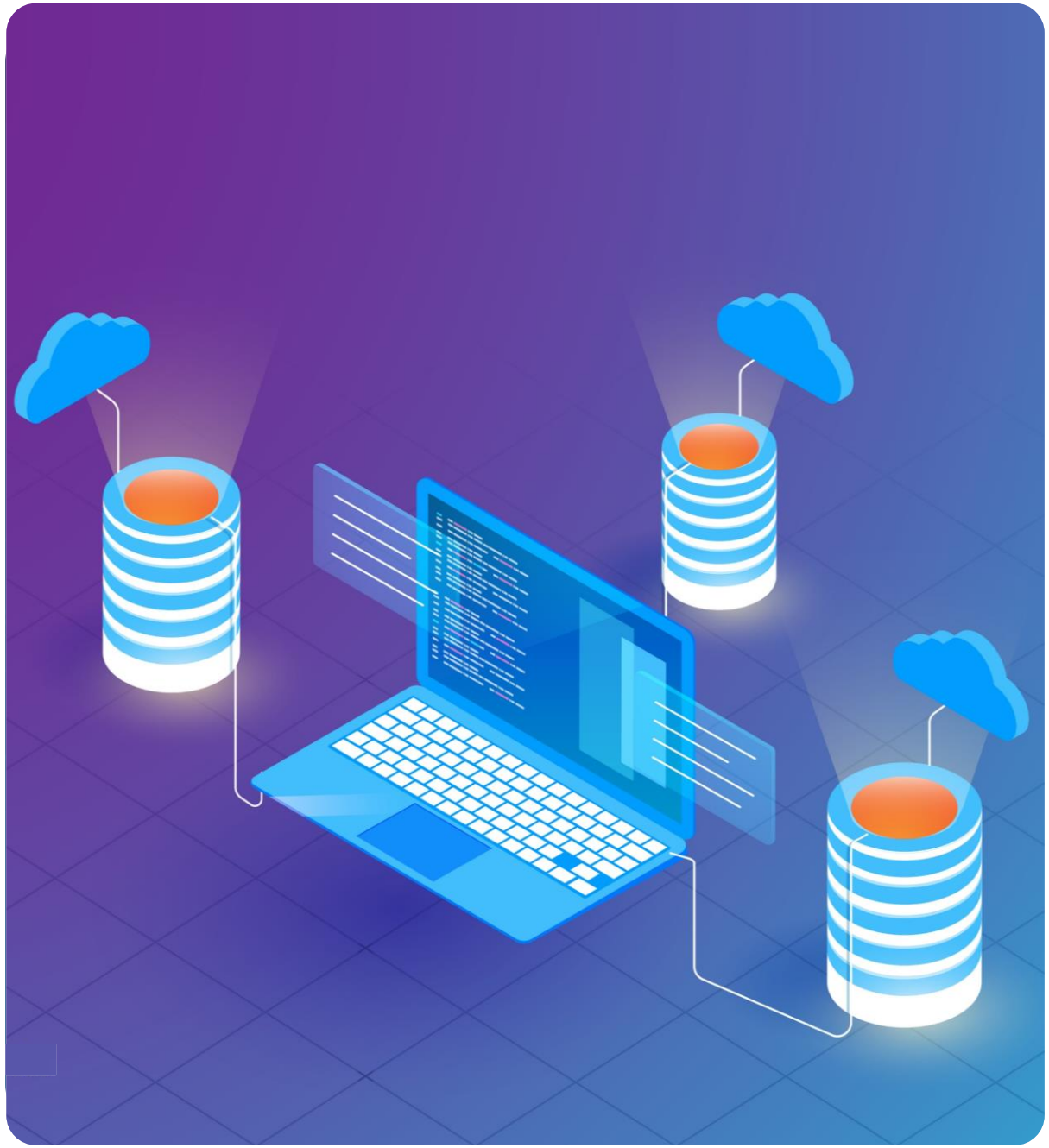


SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scaling	Vertical	Horizontal
Transactions	Supported	Support varies

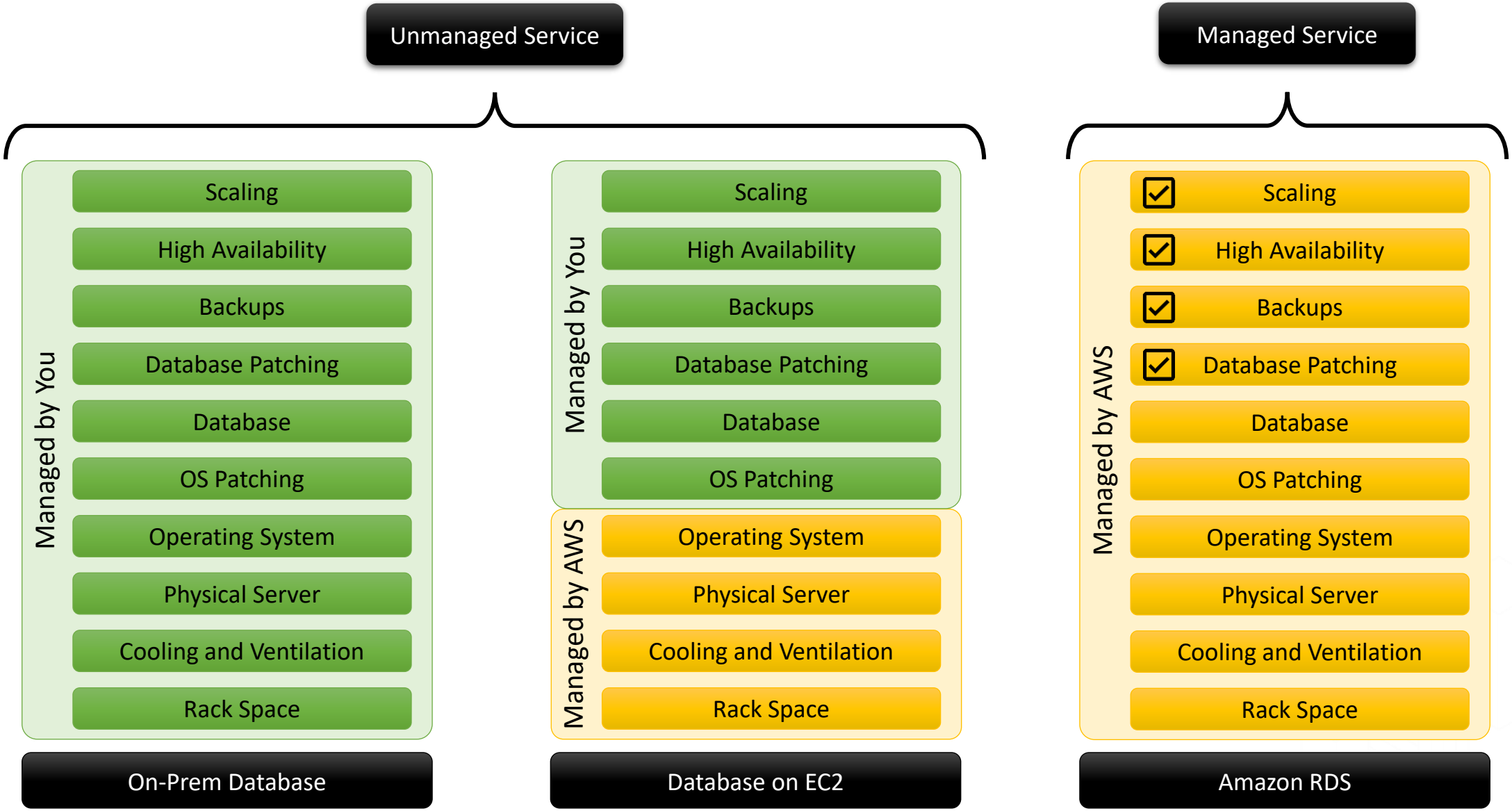
A	C	I	D
Atomicity Transitions are all or nothing	**Consistency** Only valid data is saved	**Isolation** Transactions do not affect each other	**Durability** Written data won't be lost

BA	S	E
Basically Available System does guarantee availability	**Soft state** System may change over time	**Eventual consistency** system will become consistent over time



Amazon Relational Database Service (RDS)

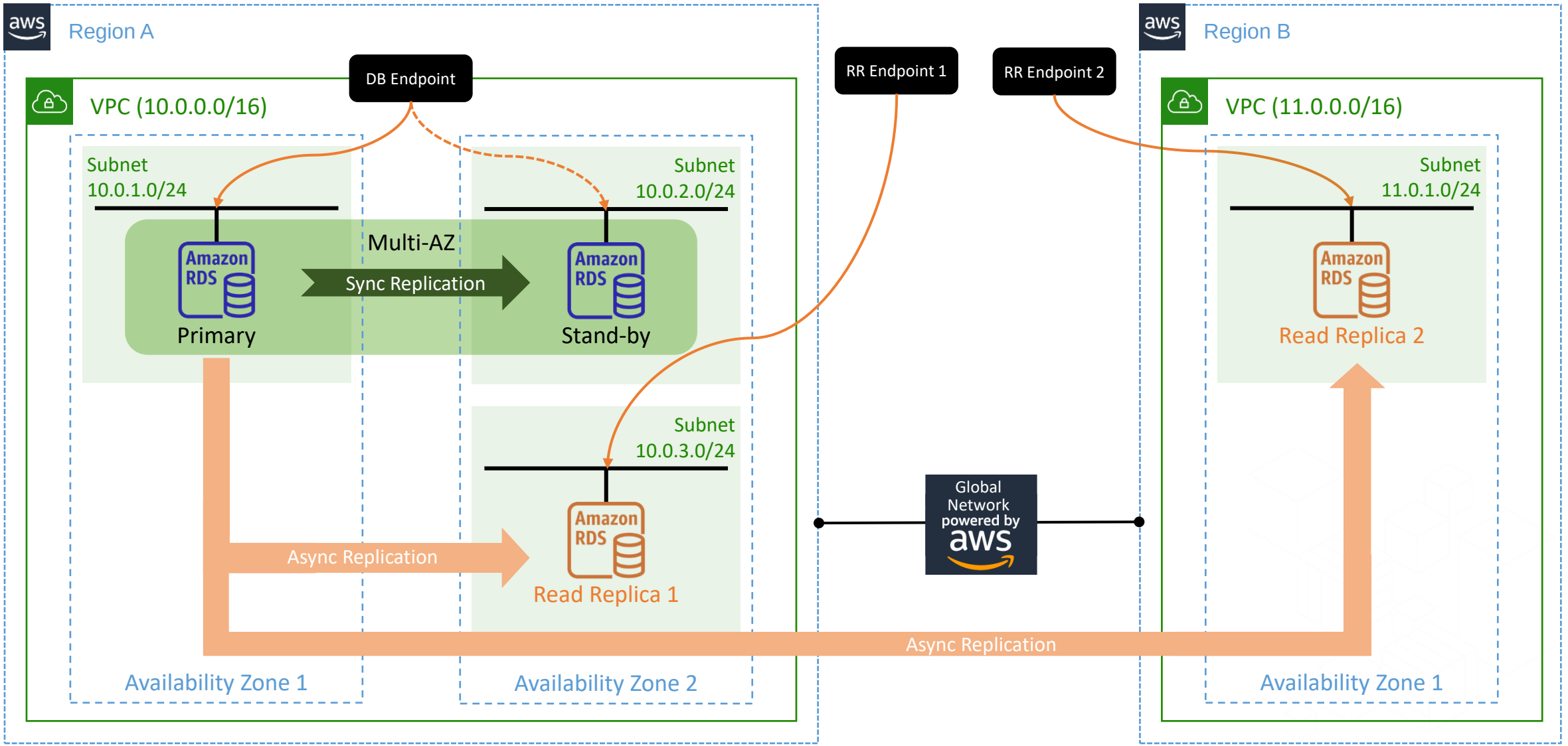
Running and Maintaining Databases



Amazon RDS

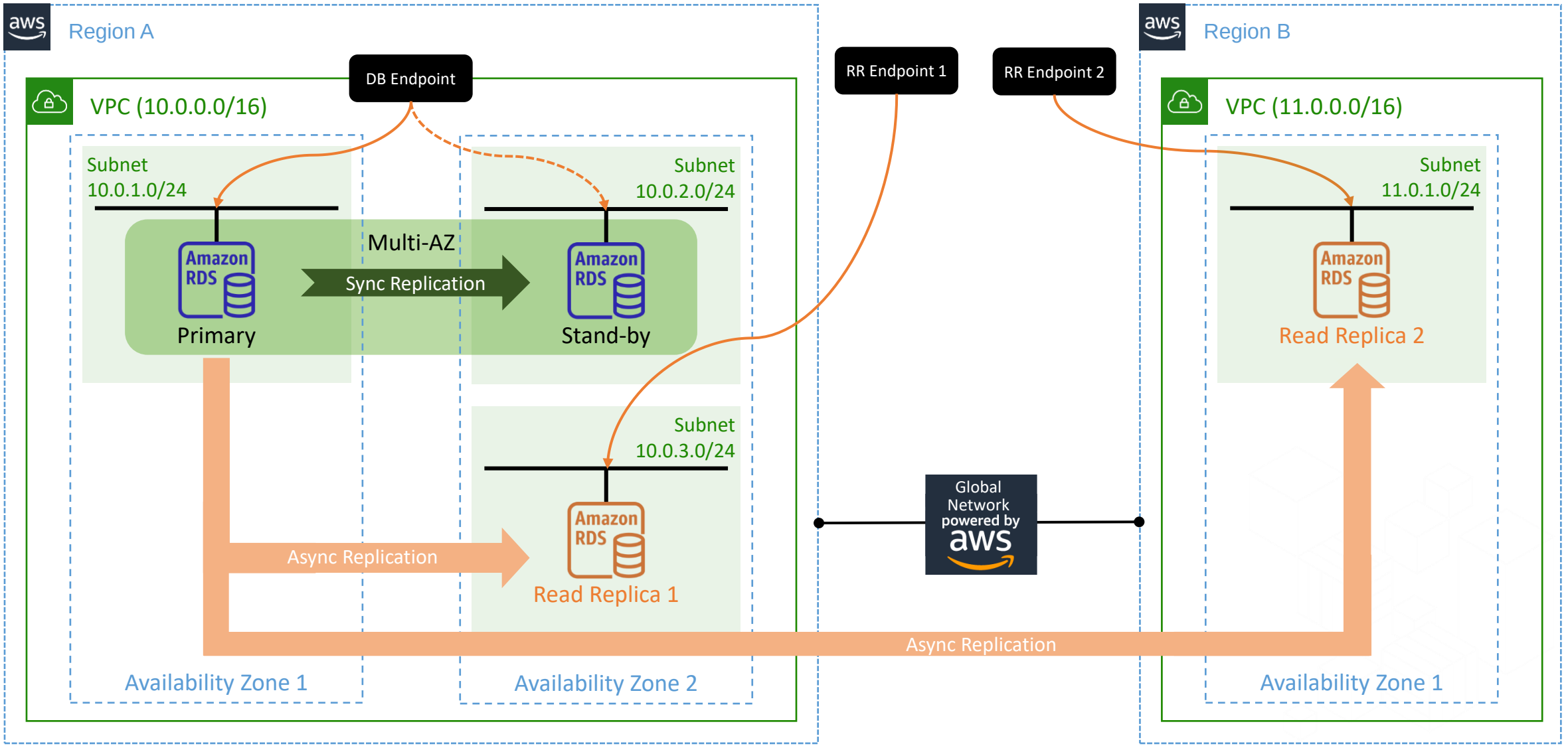
- Choice of different database engine
  - MySQL / PostgreSQL / Maria DB
  - MS SQL Server / Oracle
  - Amazon Aurora
- Supports High Availability and Read-Replica
- Snapshots can be copied across region
- Can migrate databases using Database Migration Service (DMS)
- Pricing
  - On-demand or Reserved Instance

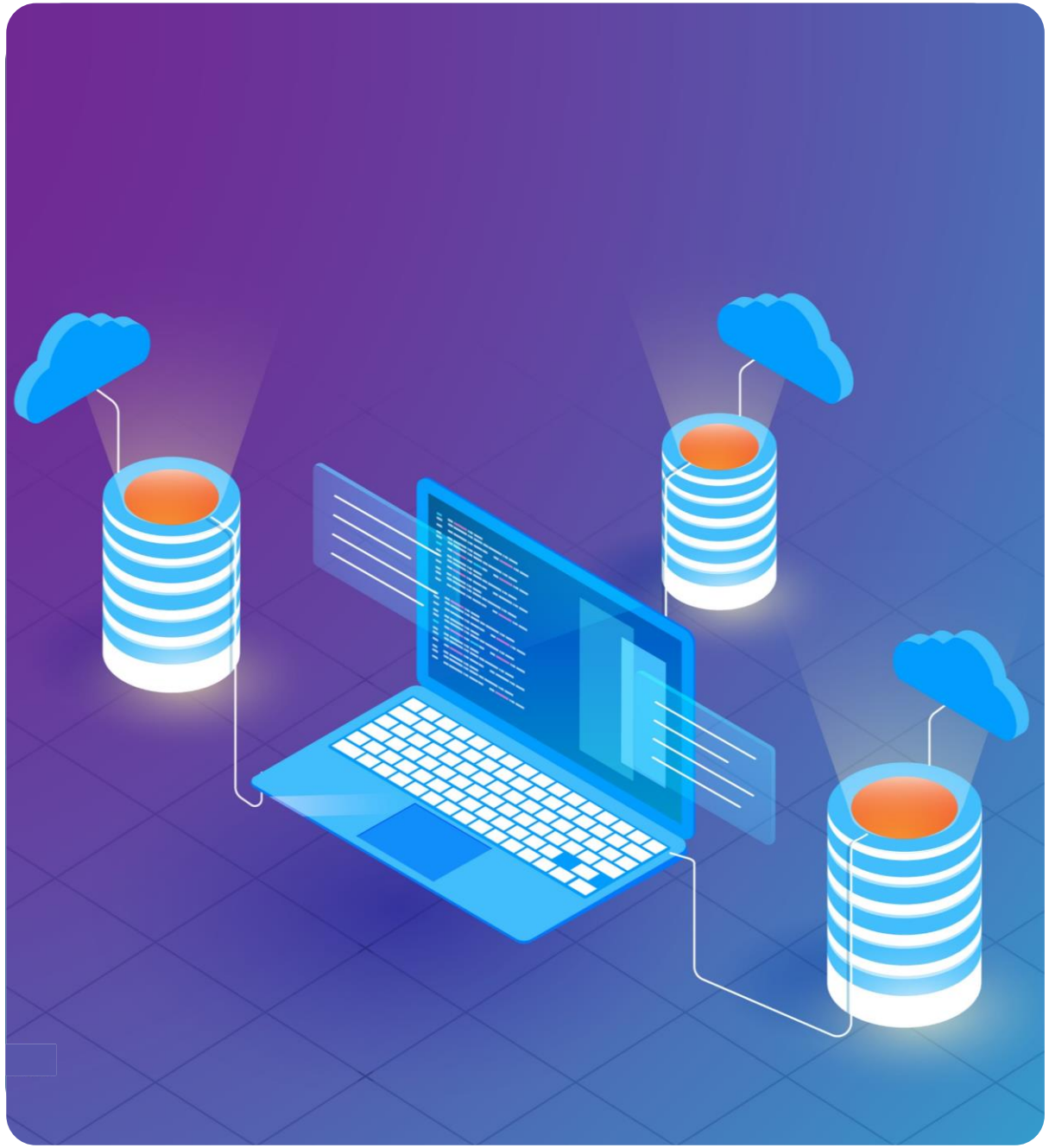
RDS – Multi-AZ and Read Replica



Amazon RDS
Multi-AZ and Read Replica

RDS – Multi-AZ and Read Replica





Amazon Aurora

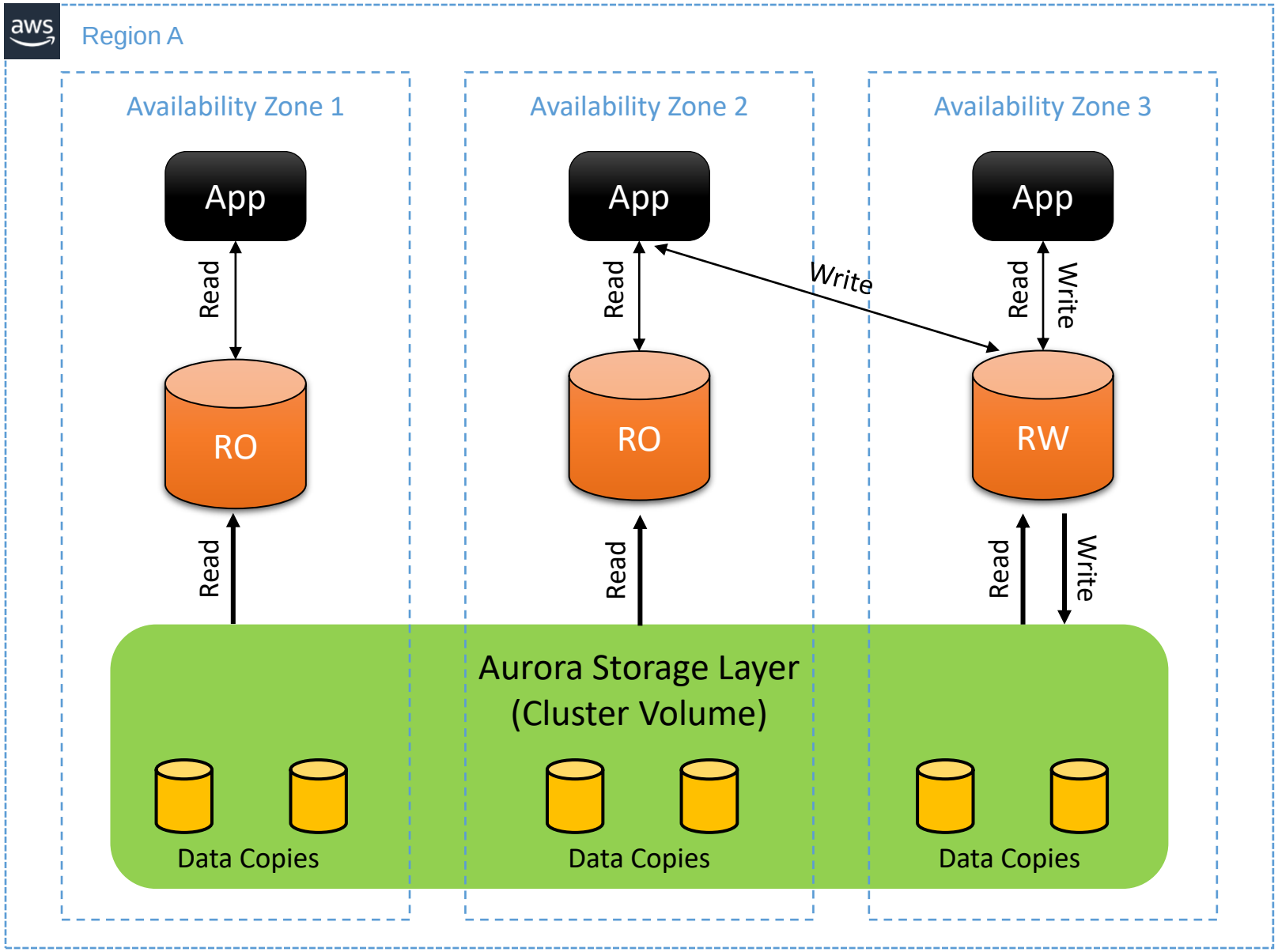
Why Amazon Aurora?

- Commercial Databases – Oracle, MS SQL Server
  - Features/Performance ++++
  - Cost $$$$
- Open Source Databases – MySQL, MariaDB, PostgreSQL
  - Features/Performance ++
  - Cost $$
- Amazon Aurora
  - Features/Performance ++++
  - Cost $$$

Amazon Aurora Flavours

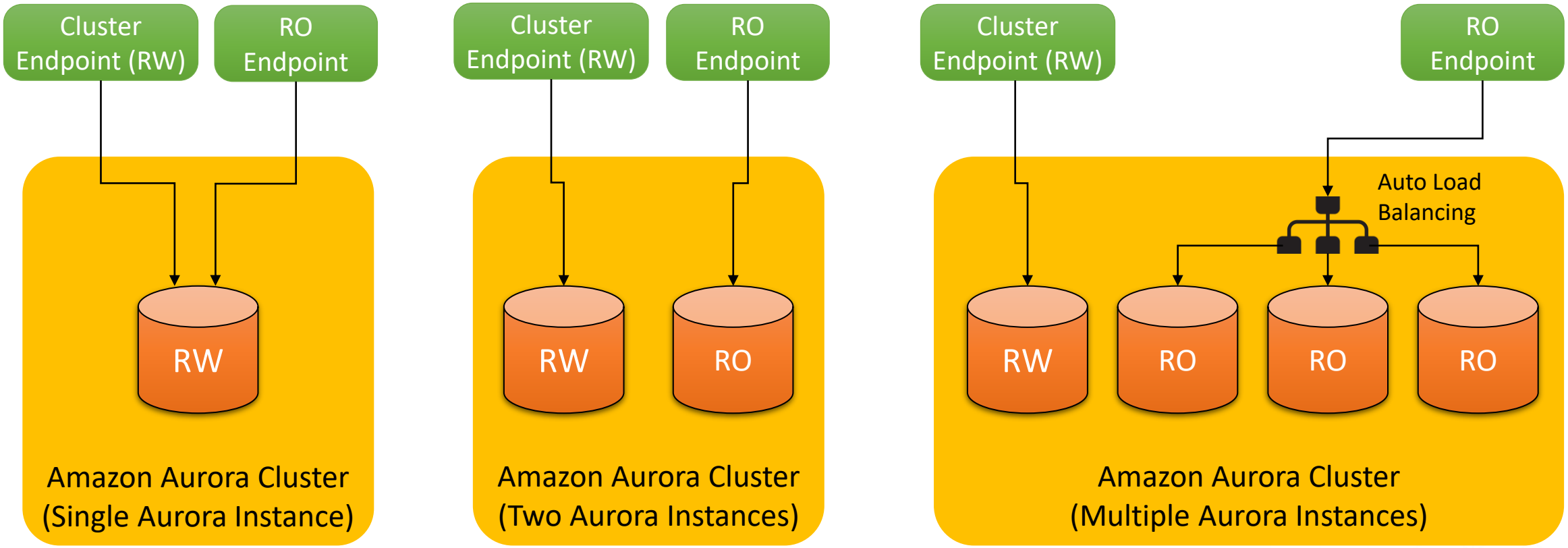
- Single-Master
- Multi-Master
- Global Database
- Serverless v2 and v1

Amazon Aurora



- Separates compute and storage layers
- Drop in compatibility with MySQL and PostgreSQL
- Six-way replication across three AZs
- Up to 15 read replicas with replica lag under 10-ms
- Automatic monitoring with failover

Amazon Aurora Cluster – Single Master



- In a single-master cluster, a single DB instance performs all write operations and any other DB instances are read-only. If the writer DB instance becomes unavailable, a failover mechanism promotes one of the read-only instances to be the new writer.

Reference:

FAQs

Category:

Database

Amazon Aurora

What?

- Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.
- It features a distributed, fault-tolerant, and self-healing storage system that is decoupled from compute resources.

Why?

- Aurora automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups while providing the security, availability, and reliability of commercial databases at 1/10th the cost.

When?

- Amazon Aurora is a great option for any enterprise application that can use a relational database.
- You need high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three AZs.

Where?

- Amazon Aurora is a regional service, it automatically maintains six copies of your data across three AZs.
- Cross-region Aurora replicas can be setup using either physical or logical replication. Physical replication uses Amazon Aurora Global Database, logical replication uses binlog for MySQL and PostgreSQL replication slots for PostgreSQL

Who?

- Amazon Aurora is fully managed by RDS and it automatically and continuously monitors and backs up your database to Amazon S3, enabling granular point-in-time recovery.
- Customer can scale the compute resources allocated to your DB Instance by changing your DB Instance class.

How?

- You choose Aurora as the DB engine option when setting up new database servers through Amazon RDS.
- After launching an Aurora instance, you can connect to it using any database client that supports MySQL or PostgreSQL.

Created by:

Ashish Prajapati

How much?

- For provisioned Aurora, you can choose On-Demand Instances and pay for your database by the hour with no long-term commitments or upfront fees, or choose Reserved Instances for additional savings.
- Aurora storage is billed in per GB-month increments, while I/Os consumed are billed in per million request increments.

Reference:

FAQs

Category:

Database

Amazon Aurora Serverless

What?

- Amazon Aurora Serverless is an on-demand, autoscaling configuration for Amazon Aurora. It automatically starts up, shuts down, and scales capacity up or down based on your application's needs.
- Aurora Serverless v2 is available for Aurora MySQL-Compatible and PostgreSQL-Compatible editions.

Why?

- Manually managing database capacity can take up valuable time and can lead to inefficient use of database resources. With Aurora Serverless, you create a database, specify the desired database capacity range, and connect your applications.
- It supports the full breadth of Aurora features, including global database, Multi-AZ deployments, and read replicas.

When?

- You want to run your database on AWS provisioning and managing database capacity.
- You have intermittent, infrequent, or unpredictable bursts of requests and want your database to automatically scale capacity to meet the needs of the application's peak load and scale back down when the surge of activity is over.

Where?

- Aurora Serverless is a Regional service. A Multi-AZ Aurora DB cluster has compute capacity available more than one AZ.
- Amazon Aurora Global Database allows a single Amazon Aurora database to span multiple AWS Regions.
- The storage for each Aurora DB cluster consists of six copies of all your data, spread across three AZs.

Who?

- There is no database capacity for you to manage. Amazon Aurora Serverless automatically starts up, scales compute capacity to match your application's usage, and shuts down when it's not in use.
- You can upgrade or switch existing clusters to use Aurora Serverless v2.

How?

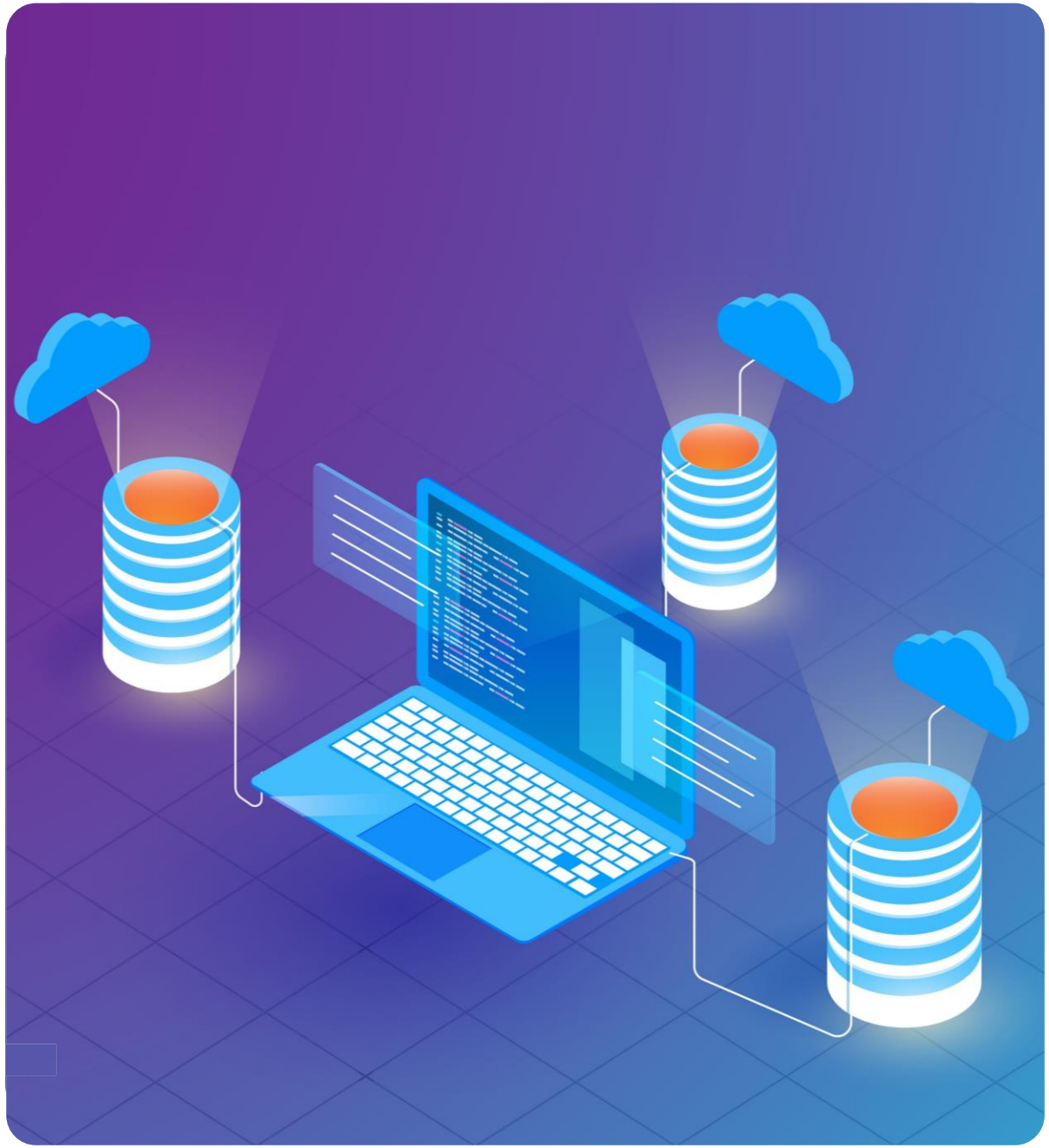
- Create a database, specify the desired database capacity range (minimum and maximum amount of resources needed), and connect your application. Aurora automatically adjusts the capacity within the range based on your application’s needs.

Created by:

Ashish Prajapati

How much?

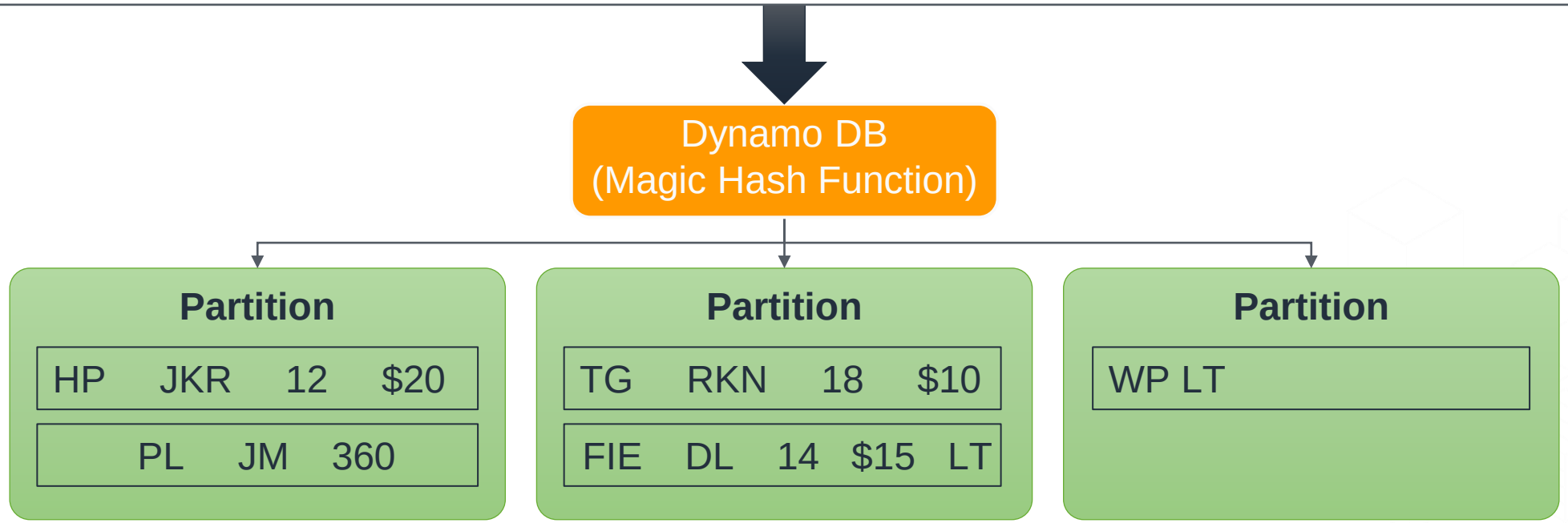
- Aurora Serverless measures database capacity in Aurora Capacity Units (ACUs) billed per second when the database is active.
- 1 ACU has approximately 2 GiB of memory with corresponding CPU and networking. Amazon Aurora database storage consumption is billed in per GB-month increments, and I/Os consumed are billed in per million request increments.



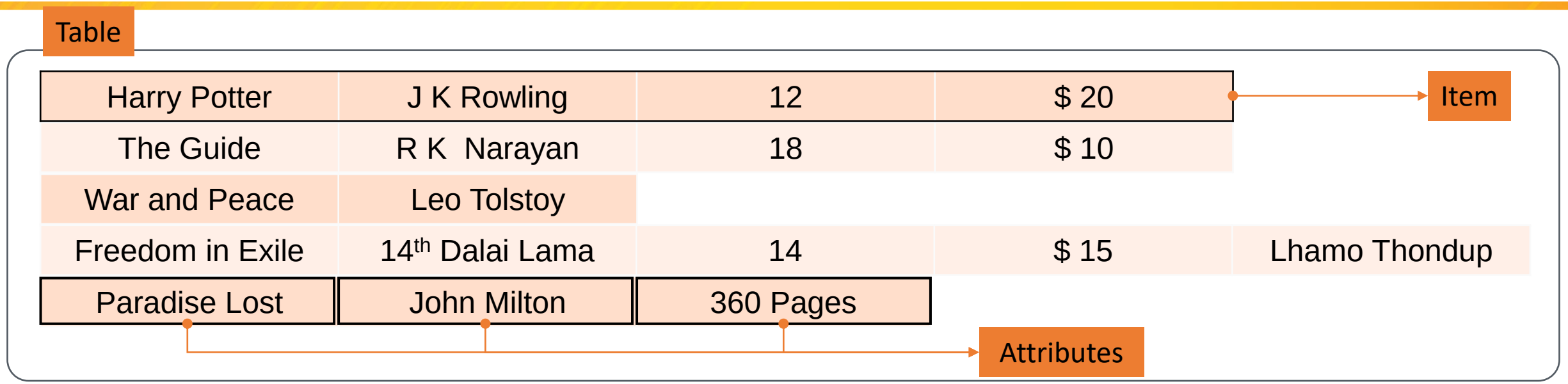
Amazon Dynamo DB

Dynamo DB

Harry Potter	J K Rowling	12	$ 20	
The Guide	R K Narayan	18	$ 10	
War and Peace	Leo Tolstoy			
Freedom in Exile	14th Dalai Lama	14	$ 15	Lhamo Thondup
Paradise Lost	John Milton	360 Pages		

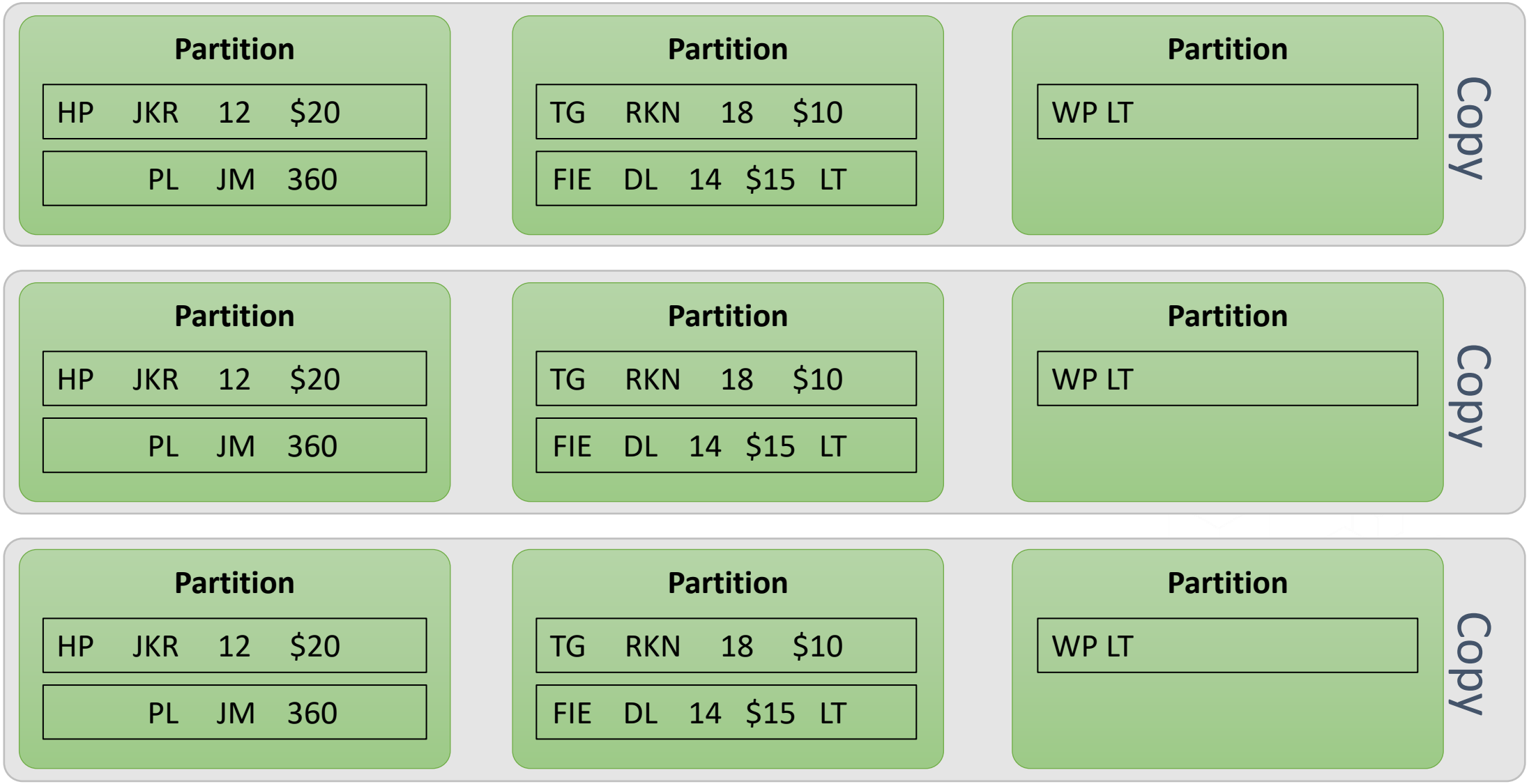


Dynamo DB – Tables, Items, Attributes

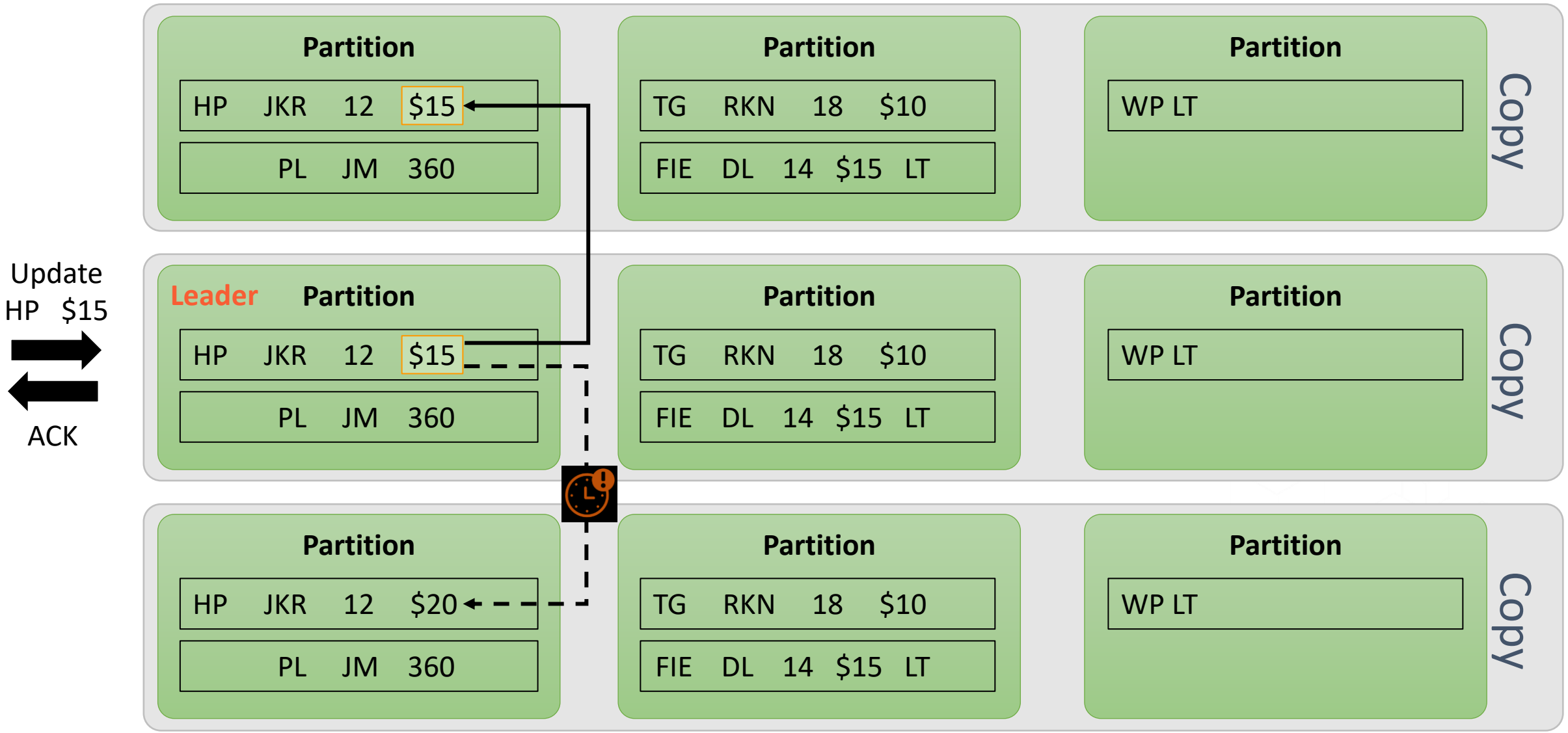


- **Tables** – Similar to other database systems, DynamoDB stores data in tables. A *table* is a collection of data.
- **Items** – Each table contains zero or more items. An *item* is a group of attributes that is uniquely identifiable among all of the other items.
- **Attributes** – Each item is composed of one or more attributes.
  An *attribute* is a fundamental data element, something that does not need to be broken down any further.

Dynamo DB Partitions



Write/Update Operation

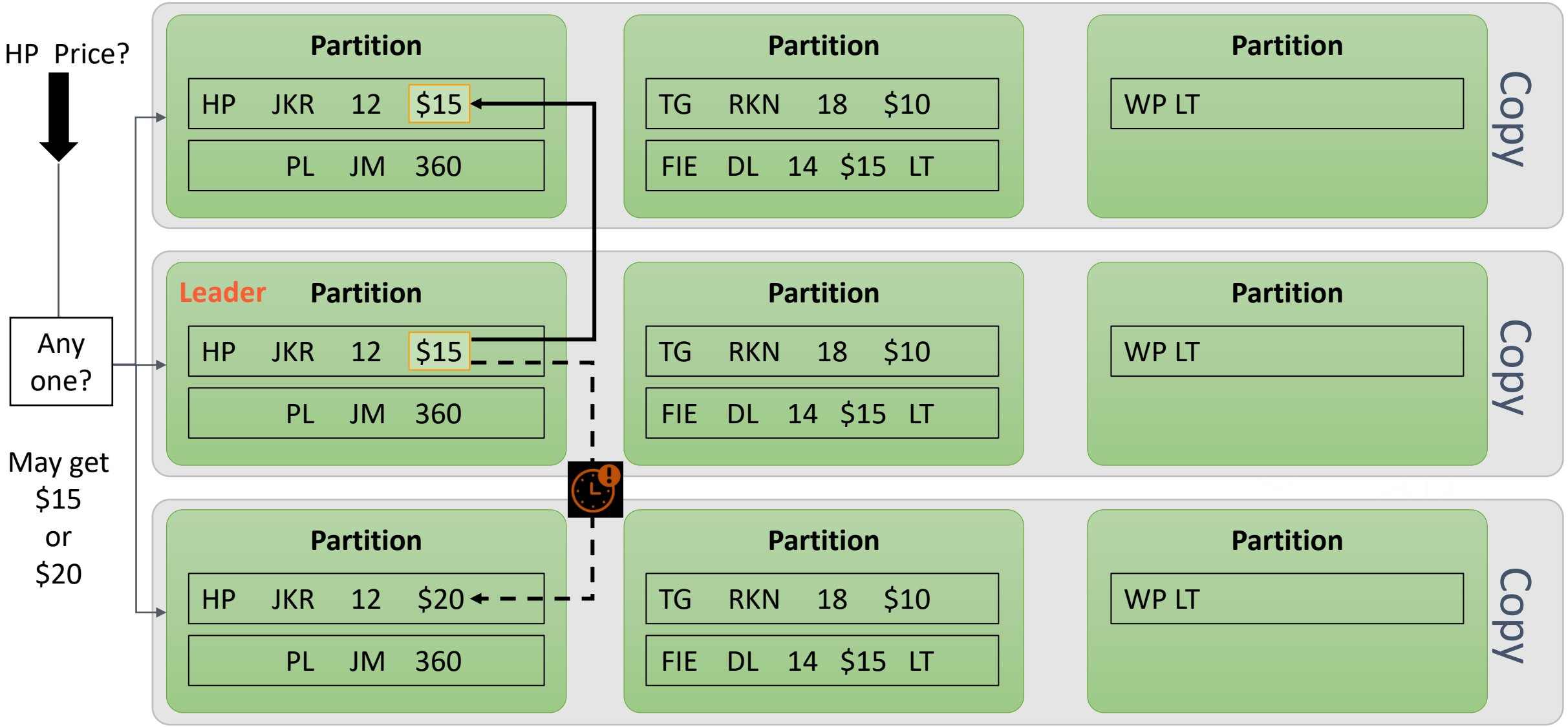


Paying for a dinner

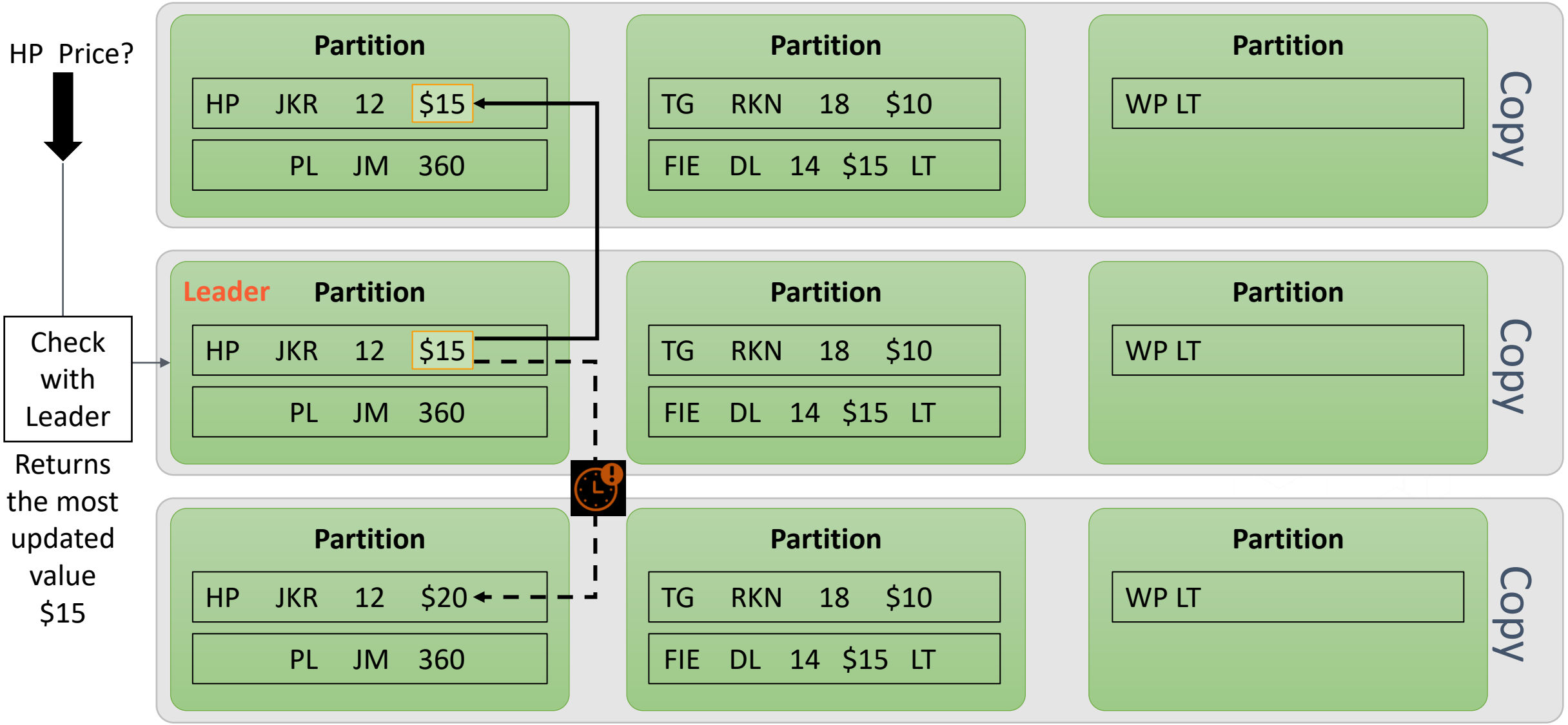
Pay by Cash

Pay by Credit Card

Read - Eventual Consistency



Read - Strong Consistency



Read Operation

- Read (GetItem)
  - Eventual Read
    - Default
  - Strong Read
    - Have to specify
- **ConsistentRead**
  - Determines the read consistency model: If set to true, then the operation uses strongly consistent reads; otherwise, the operation uses eventually consistent reads.
  - Type: Boolean
  - Required: No

Request Syntax

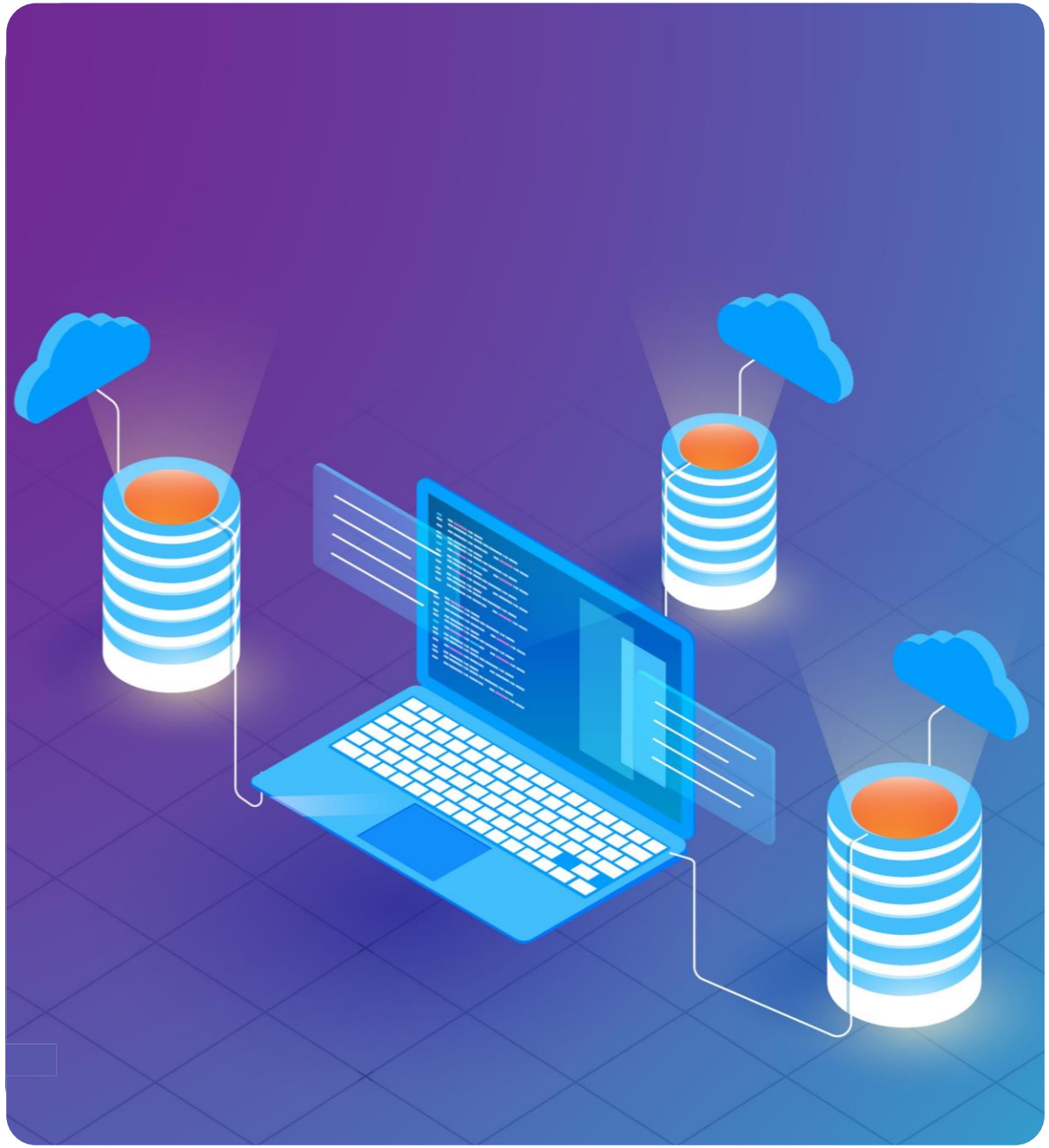
```
{
   "AttributesToGet": [ "string" ],
   "ConsistentRead": boolean,
   "ExpressionAttributeNames": {
      "string" : "string"
```

Attributes Example

```
{
   "Artist": "No One You Know",
   "SongTitle": "My Dog Spot",
   "AlbumTitle": "Hey Now",
   "Price": 1.98,
   "Genre": "Country",
   "CriticRating": 8.4
}
```

```
{
   "PersonID": 102,
   "LastName": "Jones",
   "FirstName": "Mary",
   "Address": {
         "Street": "123 Main",
         "City": "Anytown",
         "State": "OH",
         "ZIPCode": 12345
   }
}
```

- Most of the attributes are scalar, which means that they can have only one value. Strings and numbers are common examples of scalars.
- Some of the items have a nested attribute (Address). DynamoDB supports nested attributes up to 32 levels deep.



Partition Key and Sort Key

Partition Key and Sort Key

Partition Key – Title

Primary Key = Partition Key

Harry Potter	J K Rowling	12	$ 20	
The Guide	R K Narayan	18	$ 10	
War and Peace	Leo Tolstoy			
Freedom in Exile	14th Dalai Lama	14	$ 15	Lhamo Thondup
Paradise Lost	John Milton	360 Pages		
Harry Potter	J K Rowling	15	$ 25	

2nd Edition

Title – Not Unique, so can't be used as Partition Key

Harry Potter	1st Edition	J K Rowling	12	$ 20
Harry Potter	2nd Edition	J K Rowling	15	$ 25

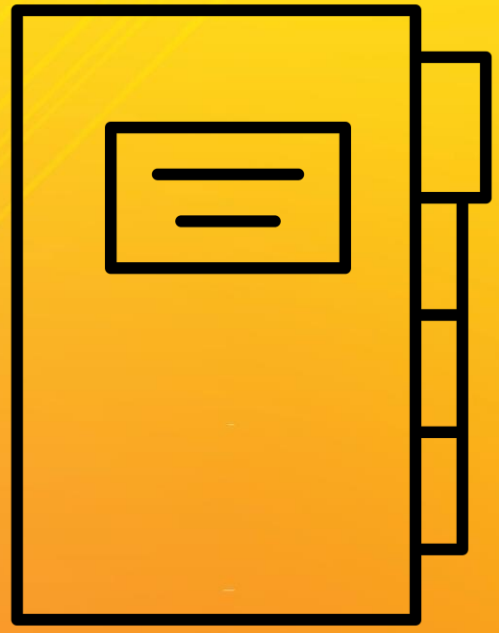
Partition Key + Sort Key

Primary Key = (Partition Key + Sort Key)

Partition Key - Examples

- Use high-cardinality attributes. These are attributes that have distinct values for each item, like `emailid`, `employee_no`, `customerid`, `sessionid`, `orderid`, and so on.

- Use composite attributes. Try to combine more than one attribute to form a unique key, if that meets your access pattern. For example, consider an orders table with `customerid#productid#countrycode` as the partition key and `order_date` as the sort key, where the symbol `#` is used to split different field.



Secondary Indexes

Why we need indexes?

Partition Key – Title

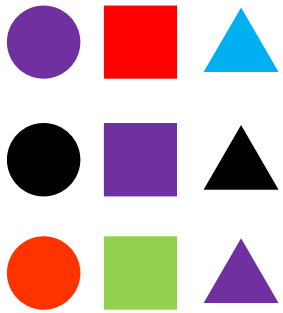
Sort Key – Author

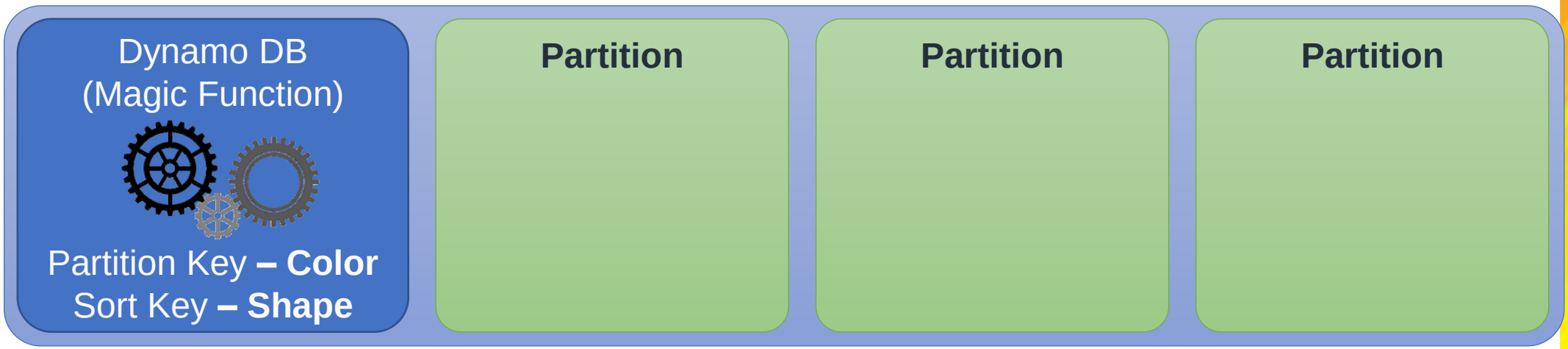
Primary Key = Partition Key + Sort Key

Harry Potter	J K Rowling	12	$ 20	
The Guide	R K Narayan	18	$ 10	
War and Peace	Leo Tolstoy			
Freedom in Exile	14th Dalai Lama	14	$ 15	Lhamo Thondup
Paradise Lost	John Milton	360 Pages		

- Questions?
  - Which book has 18 chapters?
  - Give me all the books of price less than $15?

Dynamo DB Table – Geometry

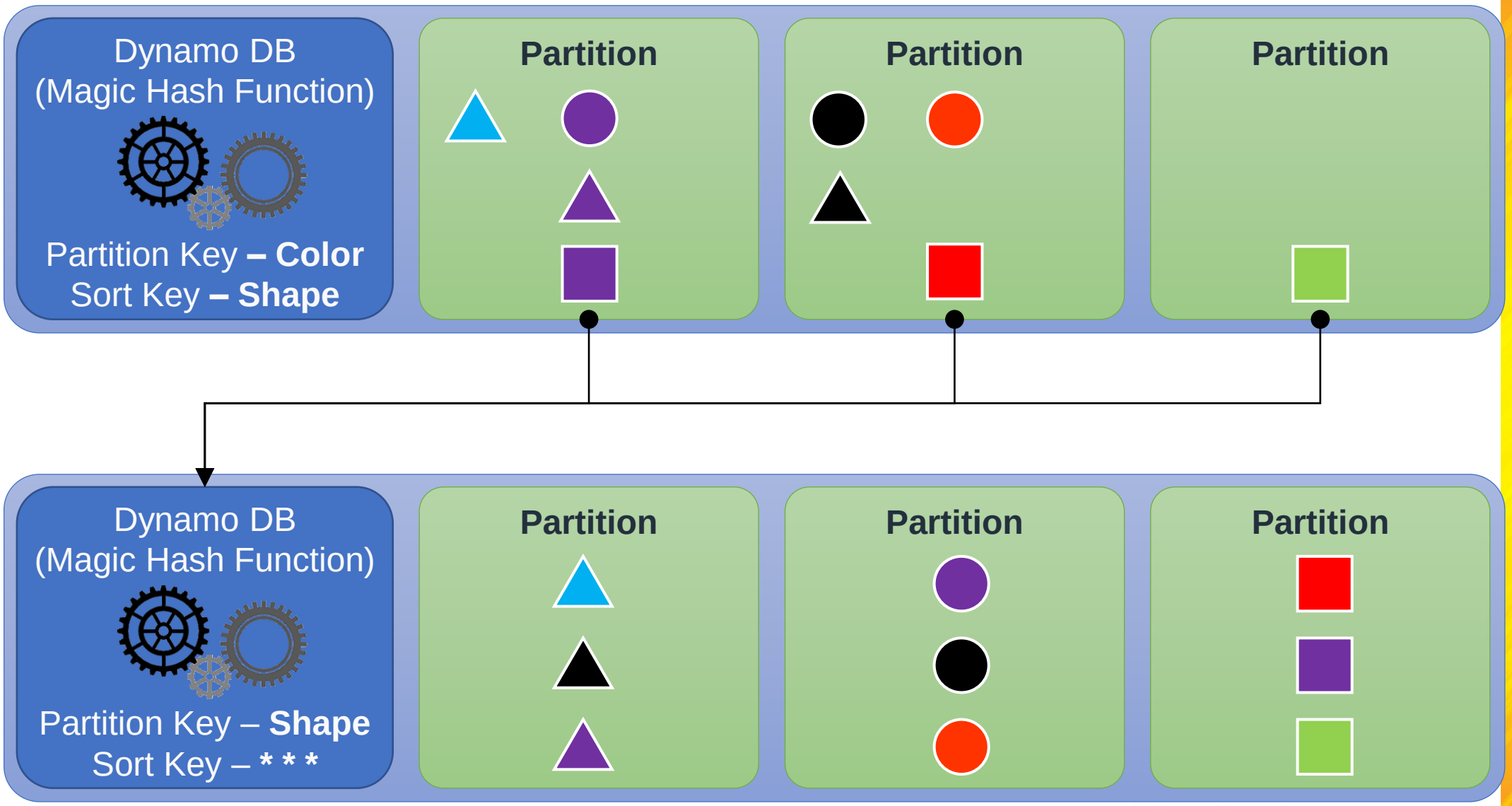




- Query
  - How many Purple Color Shapes we have?
  - Give me details of Purple Color Square?
  - How many Squares we have?
    - Current Primary Key is not efficient for this question. That's where Global Secondary Index is useful.

Global Secondary Index

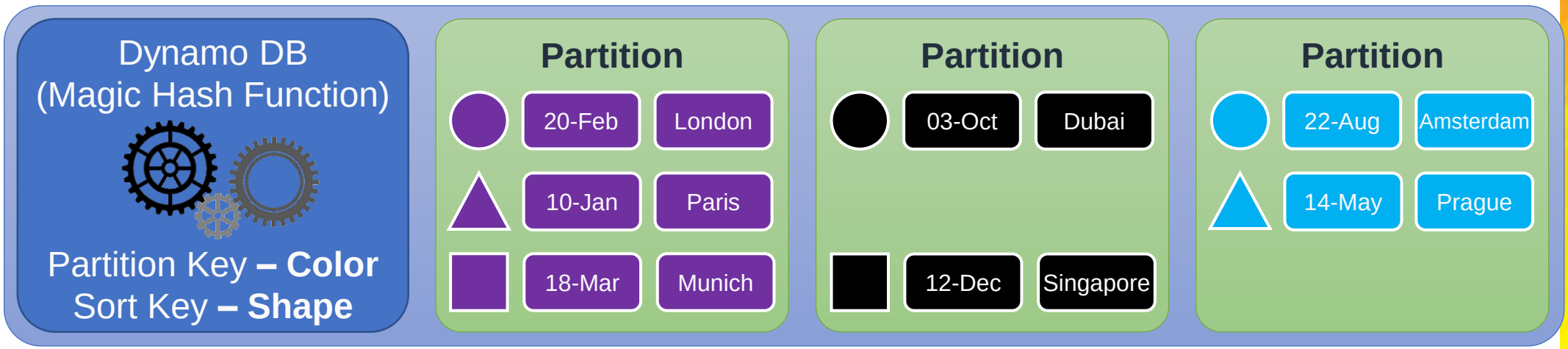
Dynamo DB Table
Geometry



GSI = Shape

Local Secondary Index

Dynamo DB Table
Geometry



- Query
  - How many Purple Color items for London?
  - Give me details of Purple Color items on 18-Mar?
- A single partition may have lots of items and we need an efficient way to find specific items. That's where Local Secondary Index is useful.

Indexes

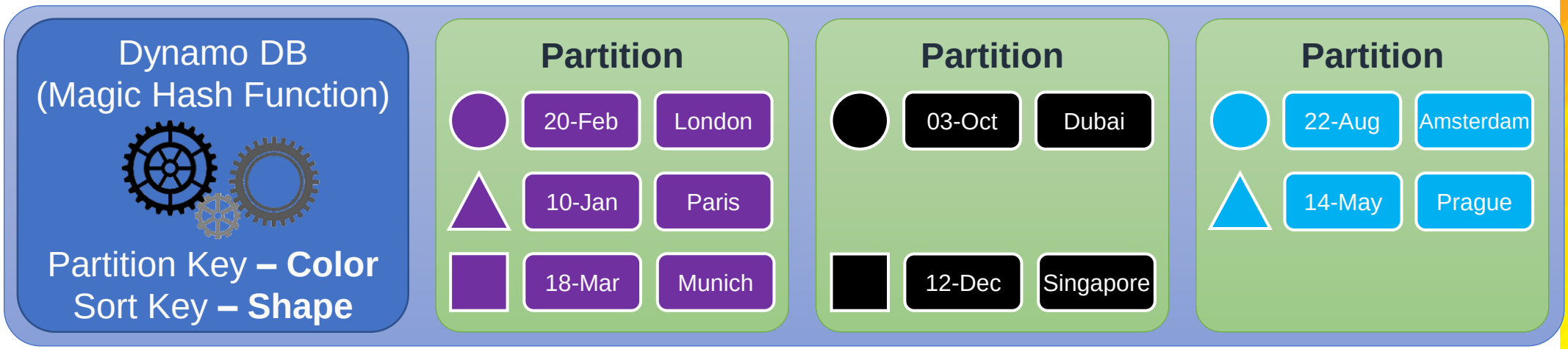
- DynamoDB supports two kinds of indexes:

- Global secondary index (GSI) – An index with a partition key and sort key that can be different from those on the table.

- Local secondary index (LSI) – An index that has the same partition key as the table, but a different sort key.

Local Secondary Index

Dynamo DB Table
Geometry



Partition Key + Sort Key (of your choice)

LSI 1	Color + Date
LSI 2	Color + City

GSI vs LSI

Compare	Global Secondary Index (GSI)	Local Secondary Index (LSI)
Queries	Across all partitions	In a single partition
Size Limit	No size limitations	Can’t exceed 10 GB
Provisioned throughput	Separate from table	Shares with the tables
Read Consistency	Only Eventual	Strong or Eventual
Maximum	20	5
Creation	Anytime	Only with table creation
Deletion	Anytime	Only with table deletion

Query vs Scan

Partition Key – Title

Primary Key = Partition Key

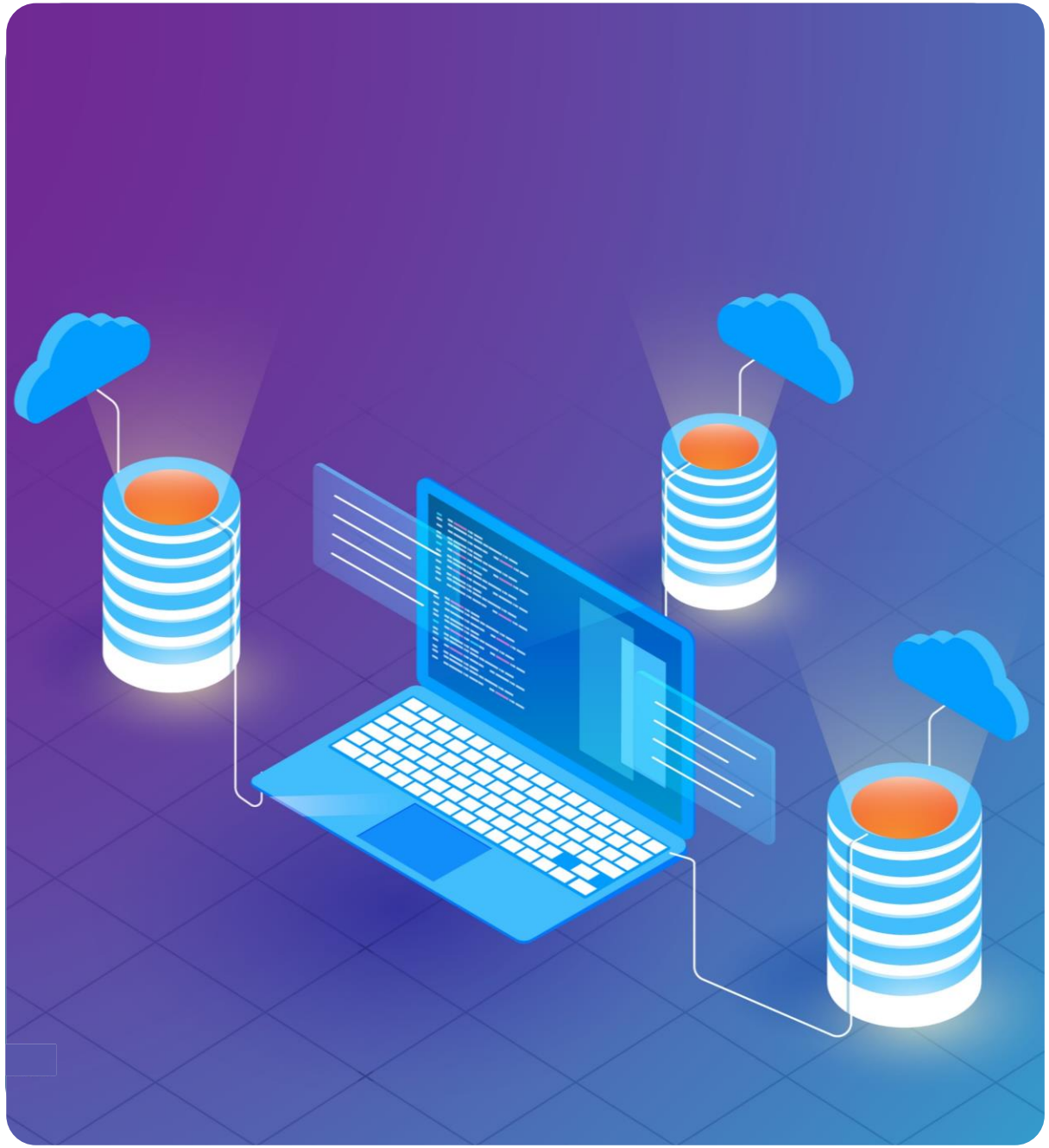
Harry Potter	J K Rowling	12	$ 20	
The Guide	R K Narayan	18	$ 10	
War and Peace	Leo Tolstoy			
Freedom in Exile	14th Dalai Lama	14	$ 15	Lhamo Thondup
Paradise Lost	John Milton	360 Pages		

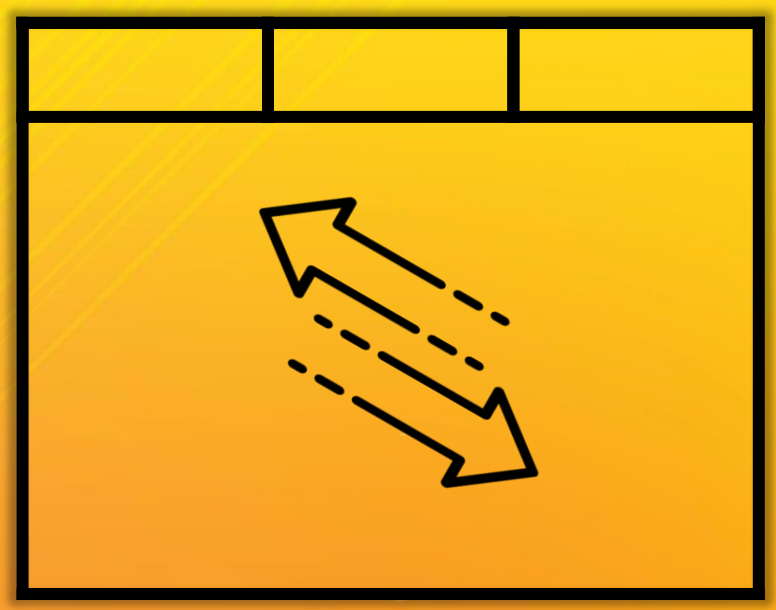
- Give me the book with title – War and Peace?
  - Query (Faster)

- Give me the book whose author is John Milton?
  - Scan (Slower)

Details about Amazon Dynamo DB

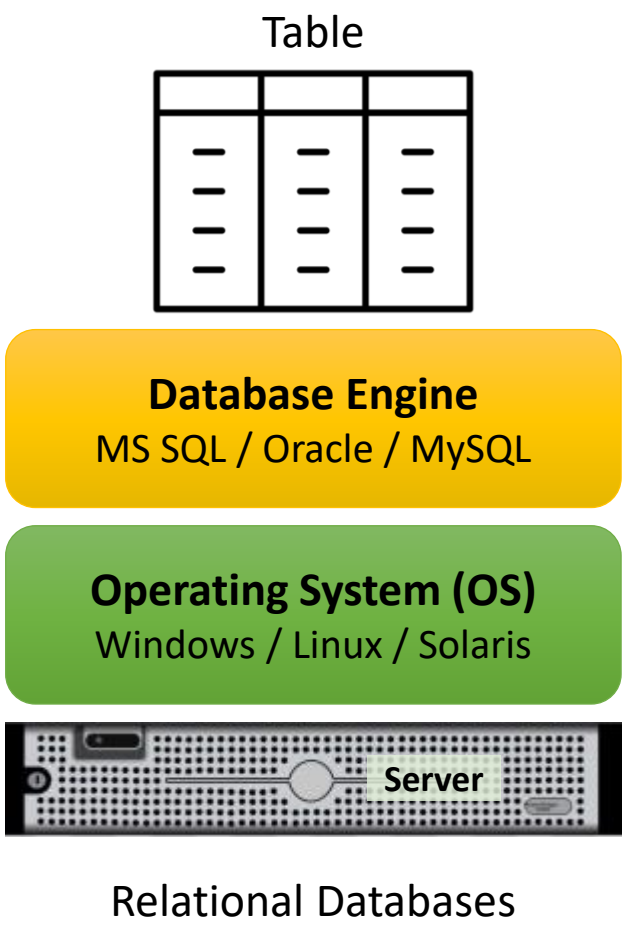
Primary Key	Simple Primary Key (Using only a Partition Key) Composite Primary Key (Using both a Partition and Sort Key)
Secondary Indexes	Global Secondary Index (GSI) • An index with a partition key and a sort key that can be different from base table Local Secondary Index (LSI) • An index that has the same partition key as the base table, but a different sort key
Capacity Modes	Provisioned - Suited for predictable or steady state workloads On-Demand - Suited for new or unpredictable workloads
Read Capacity Unit	One RCU represents one strongly consistent read request, or two eventually consistent read requests, for an item up to 4 KB
Write Capacity Unit	One WCU represents one write for an item up to 1 KB
Global tables	A fully managed solution for deploying multi-region, multi-master databases
Streams	Allows you to trigger Lambda function when data is modified in a table
Read Consistency	Eventually consistent - Data is returned immediately but data can be inconsistent Strongly Consistent - Will always read from the leader partition. Data will never be inconsistent but latency may be higher

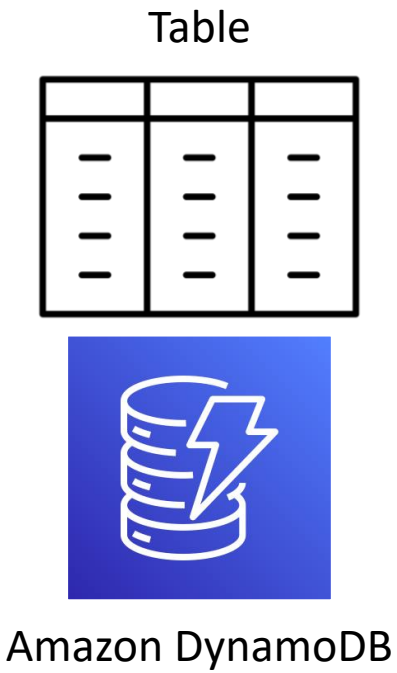




Dynamo DB
Read/Write Capacity

Dynamo DB Performance





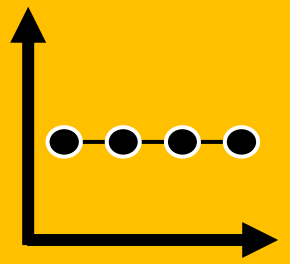
- Dynamo DB Table performance is controlled by configuring:
  - Read Capacity Unit (RCU)
  - Write Capacity Unit (WCU)

- One RCU = one strongly consistent read per second, or two eventually consistent reads per second, for an item up to 4 KB in size.

- One WCU = one write per second for an item up to 1 KB in size.

Dynamo DB capacity modes

- Amazon DynamoDB has two read/write capacity modes for processing reads and writes on your tables:

	Provisioned Mode 	**On-Demand Mode** 
What?	Provision the capacity (RCU/WCU) to run at a specific limit	No limit scaling, serving thousands of requests per second without capacity planning
Charges	Pay for provisioned capacity (whether you use it or not)	Pay only for read and write you perform
Benefit	Controls cost and supports capacity reservation	Instantly accommodates your workload as traffic ramps up or down
Suitable for	Steady state and predictable traffic	Random and unpredictable traffic
Floor and ceiling	Can be setup using Auto Scaling	Can scale to zero, no ceiling