NoSQL: Amazon DynamoDB Basics

Presented by: Vincent Itucal

What is DynamoDB?

- Amazon’s NoSQL Database Engine
- Optimized for key-value lookups
- For very high performance workloads with predictable performance (important for OLTP use cases)
- Fully Managed
- Pay what you use model

SQL		NoSQL
Relational	Model	Non-relational
Structured Tables	Data	Semi-structured
Strict Schema	Flexibility	Dynamic Schema
ACID	Transactions	Mostly BASE, with some ACID
Strong	Consistency	Eventual to Strong
Consistency Prioritized	Availability	Basic Availability
Vertical (Better Machines)	Scale	Horizontal by Data Partitioning(Adding More Machines)

Model, Structure and Flexibility

Anatomy of a DynamoDB item

- An item is the core unit of data in DynamoDB
- Comparable to a row in a relational DB
- Core aspects of items:
    - Attributes
    - Primary Keys
    - Attribute Types

Core aspects of items

- Primary Keys
  - Simple Primary Key:
    - Partition (Hash) Key
    - e.g. client_ids, txn_ids
  - Composite Primary Key:
    - Partition (Hash) Key + Sort (Range) Key
    - Usually for use cases in getting a number of items in a partition that satisfies a condition on the sort key
- Attributes Types
  - String, Number, Binary, Boolean, Null, List, Map

Secondary Indexes

Under the hood setting secondary indexes will create a “copy” of the same table but with a different set of keys (primary key or a different sort key)

- Local Secondary Indexes (LSI)
  - Different Sort Key
- Global Secondary Indexes (GSI)
  - Different Partition Key and Sort Key

Flexibility - Schemaless

- Introducing new fields does not need the current table to be recreated for it to handle logic moving forward
- If desire to load the items of the table with a schema in mind, it is usually done in your code’s logic instead

Transactions, Consistency, Availability and Scale

Transactions, Consistency and Availability

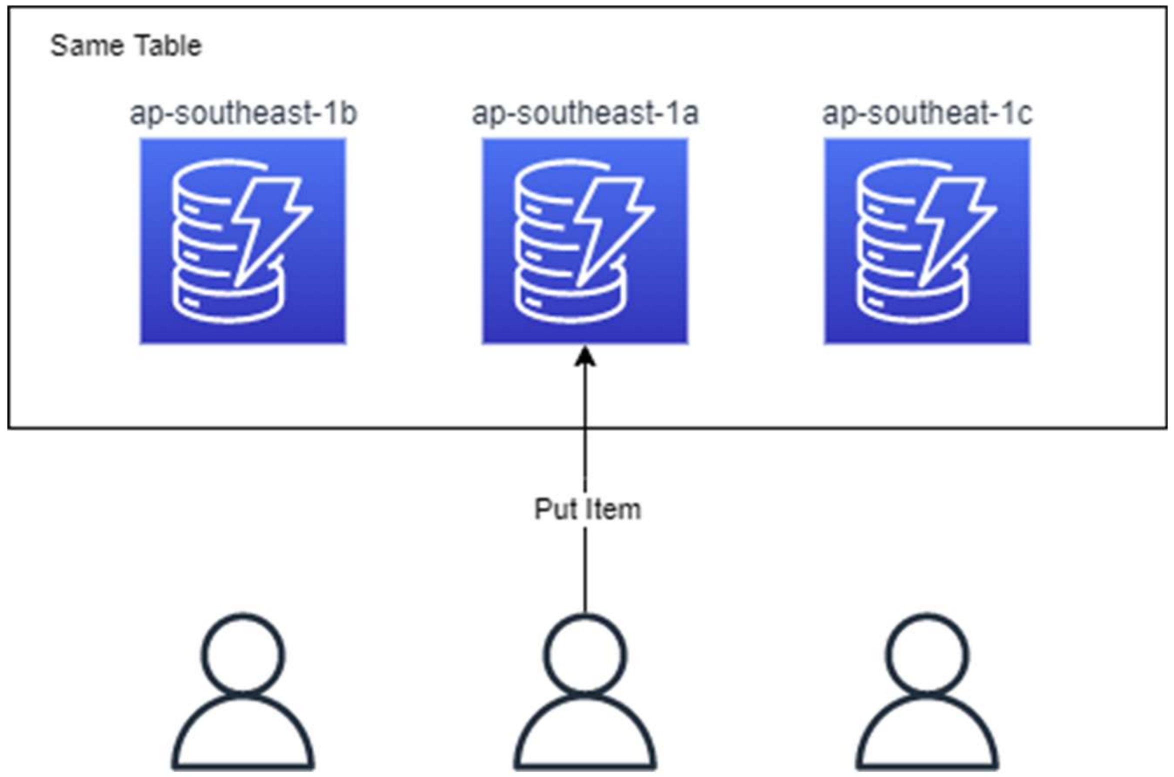
Relational Databases

- Mostly ACID compliant
  - Atomicity, Consistency, Isolation, Durability

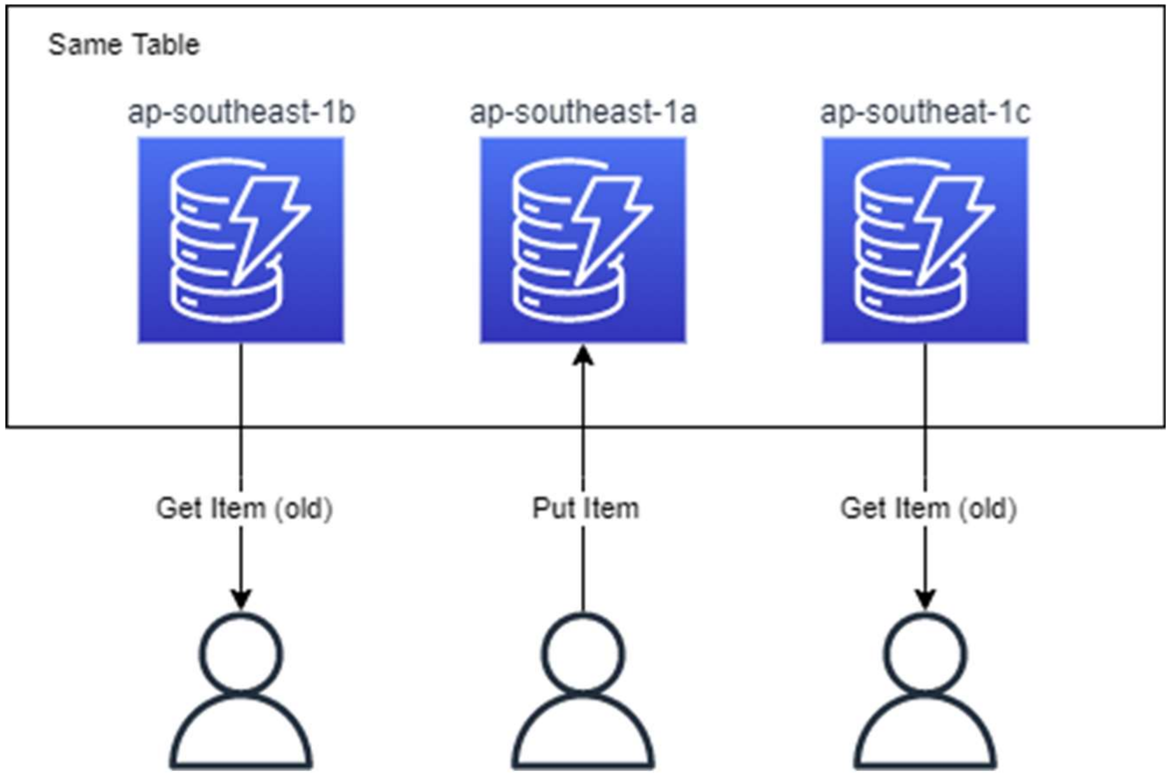
DynamoDB

- Supports ACID compliance with DynamoDB transactions
- But is mostly BASE
  - Basically Available
  - Soft State
  - Eventually Consistent

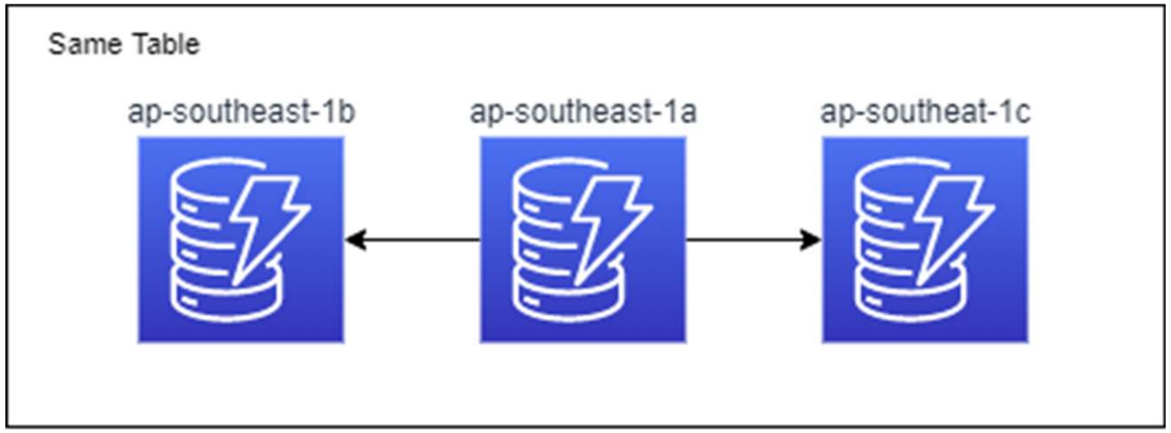
Transactions, Consistency and Availability



Transactions, Consistency and Availability

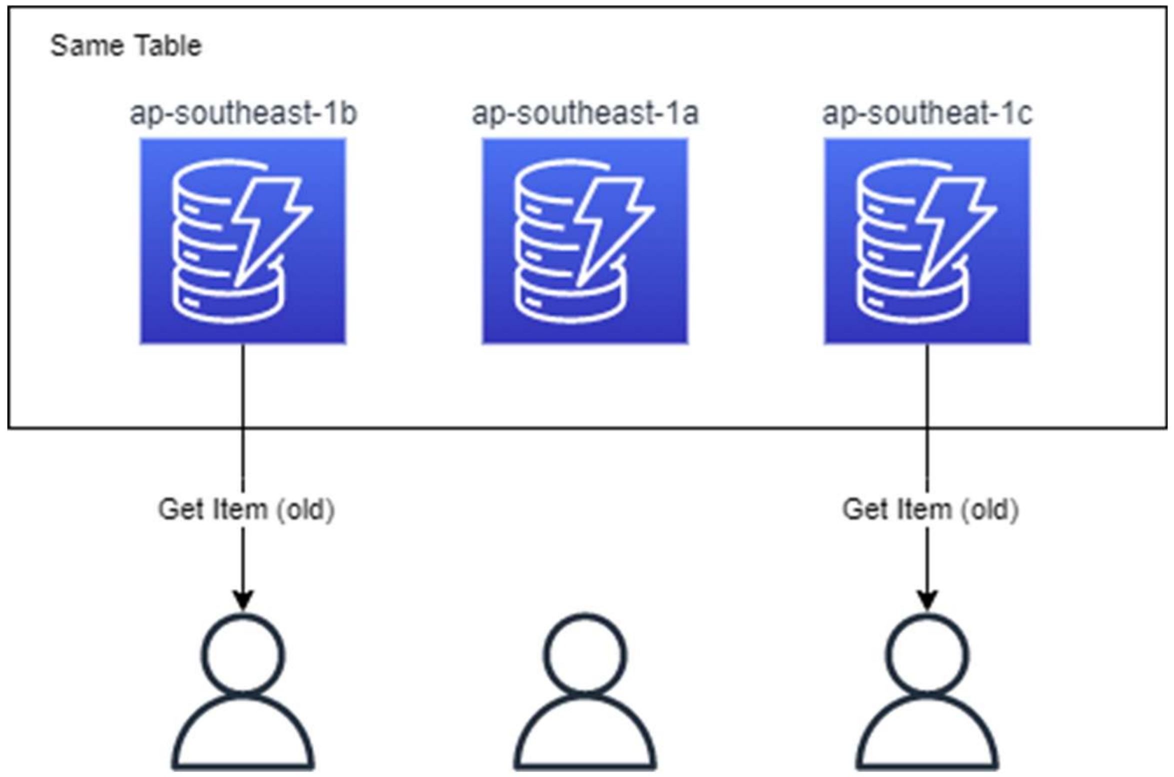


Transactions, Consistency and Availability



Data writes are eventually consistent
usually within ~1 second or less

Transactions, Consistency and Availability



Scalability

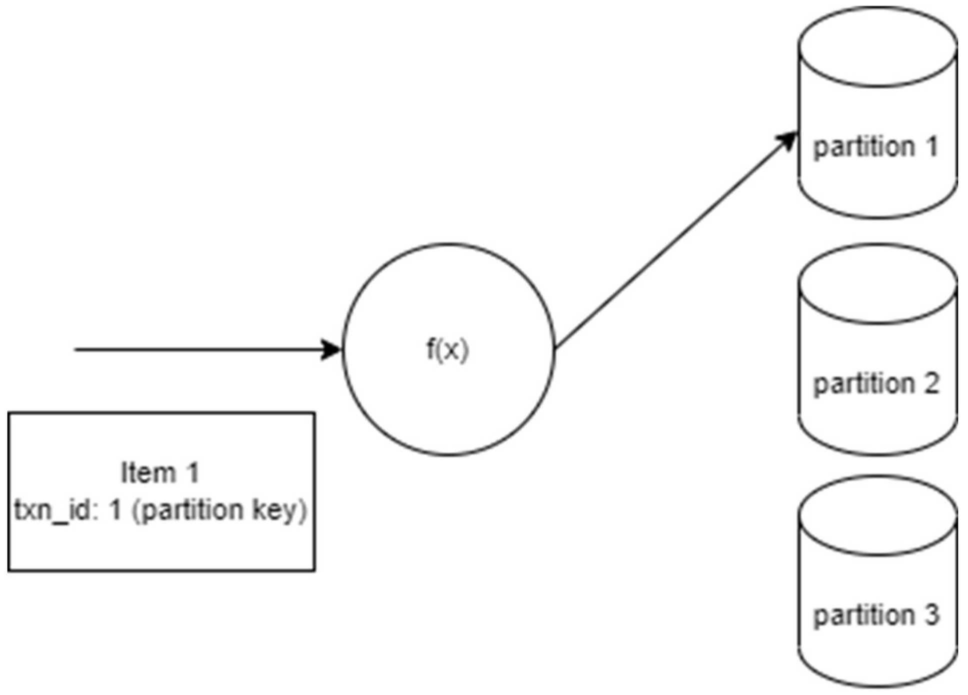
SQL

- Scales vertically
- Improved by better machines

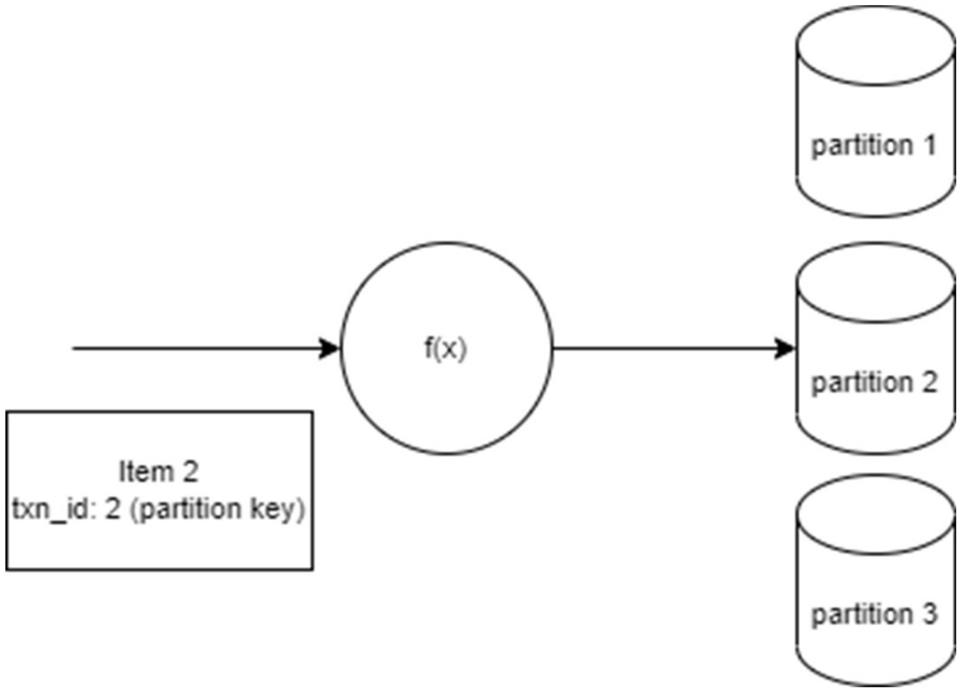
DynamoDB

- Scales horizontally
- By Data Partitioning

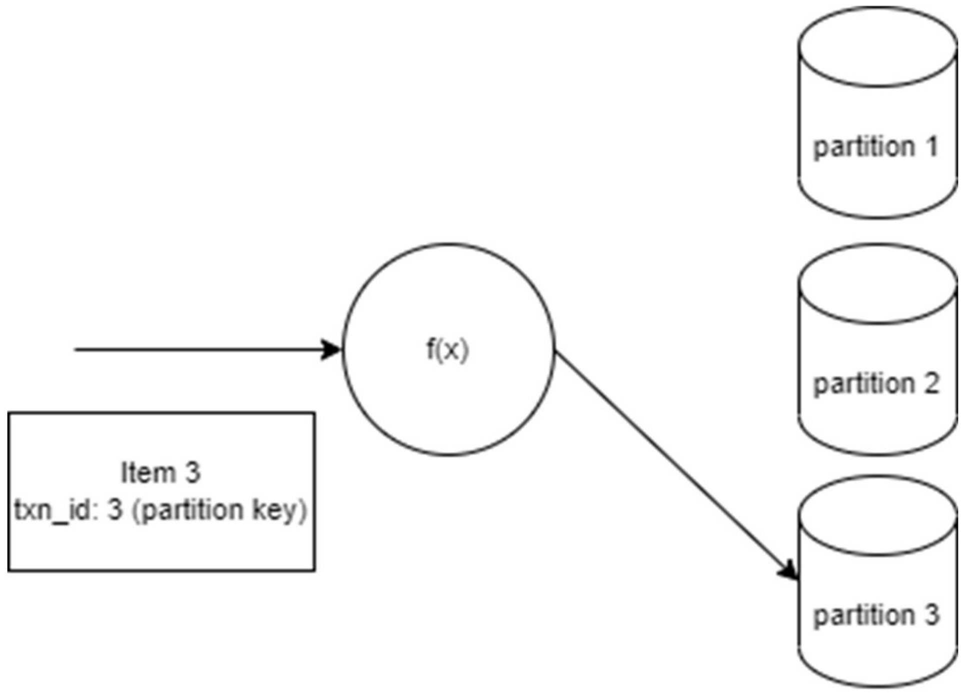
Scalability



Scalability

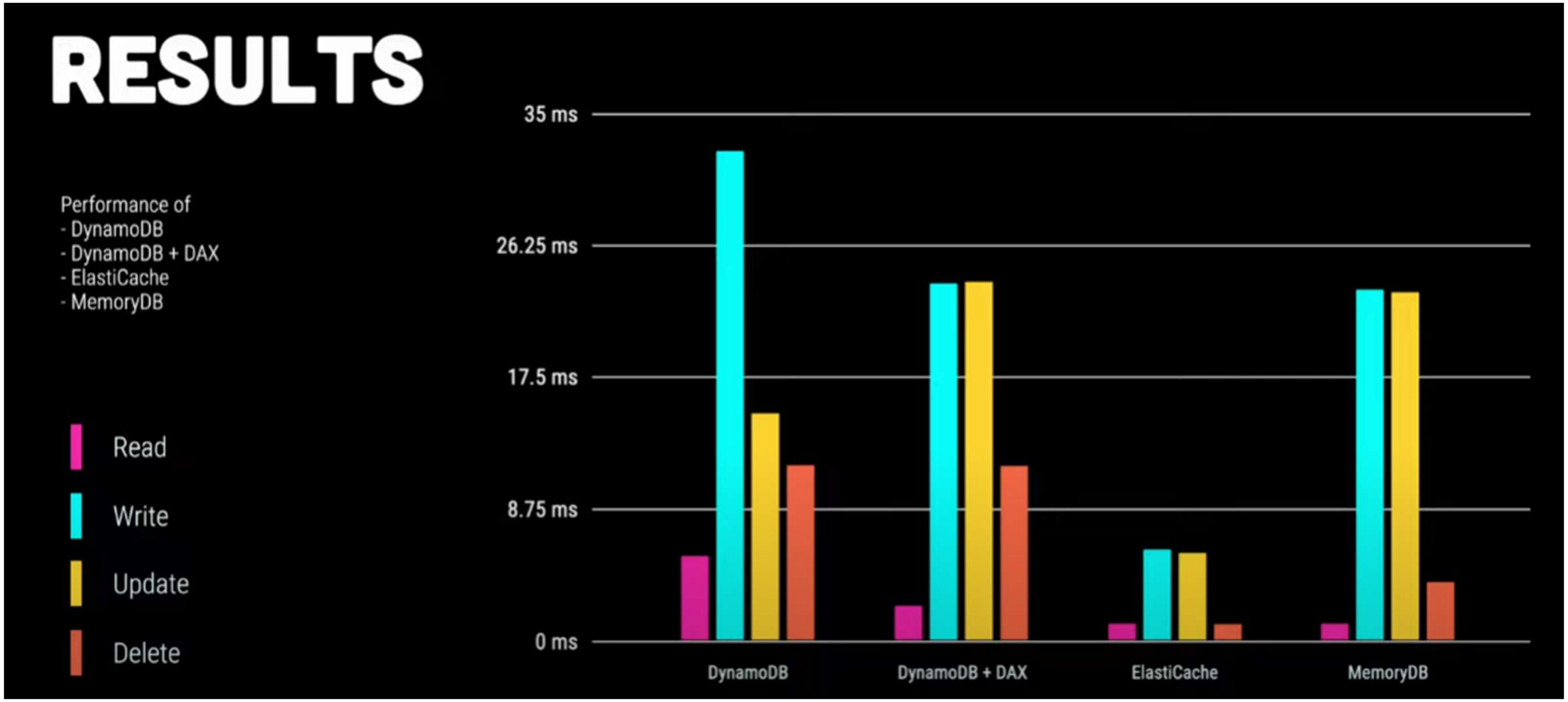


Scalability

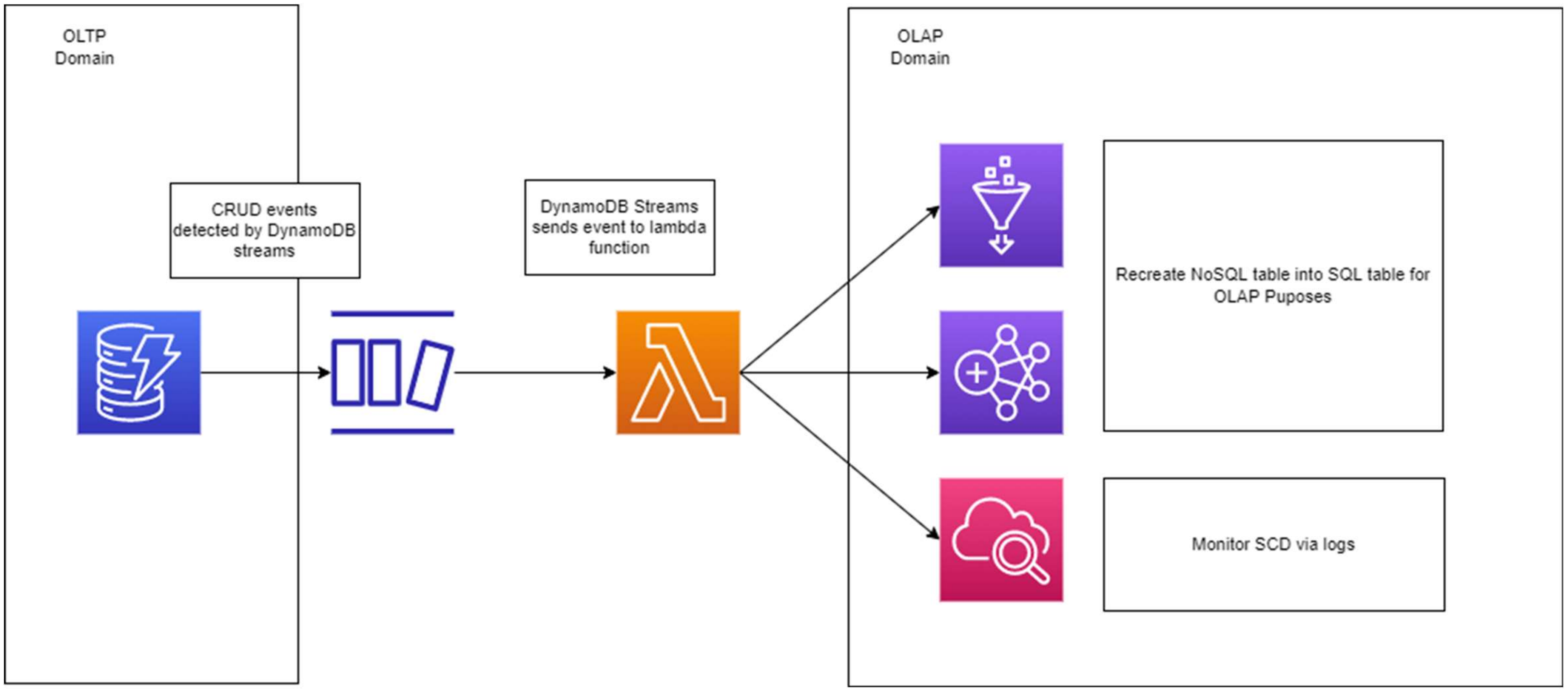


Other Features

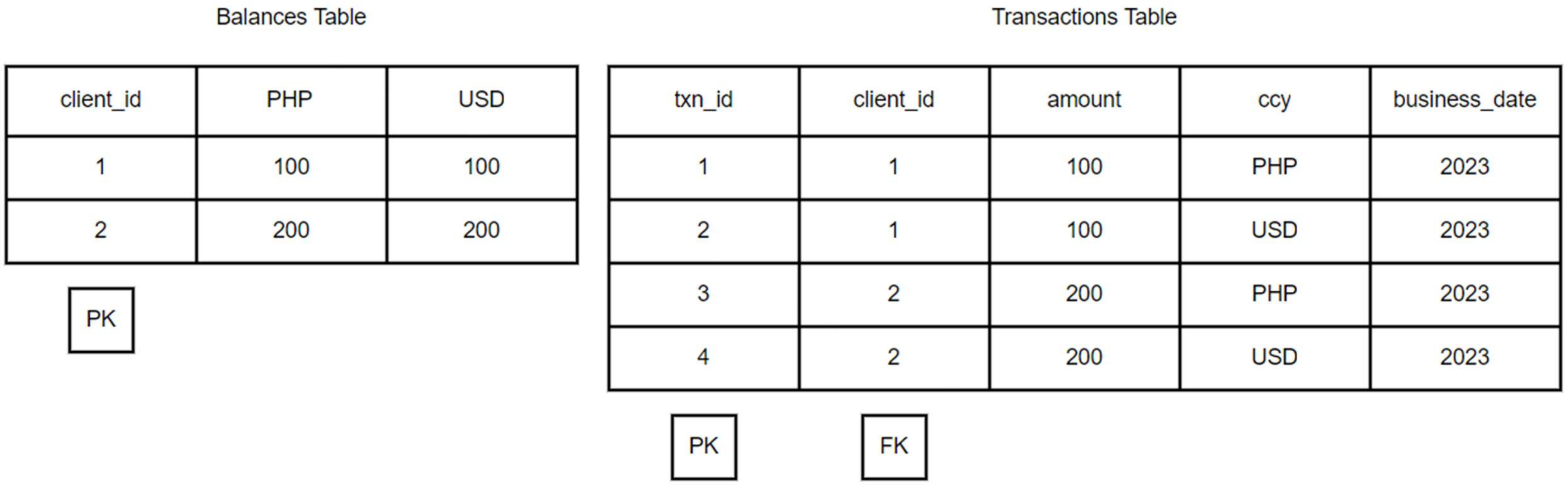
DynamoDB Accelerator (DAX) for caching



DynamoDB Streams


OLTP Domain
CRUD events detected by DynamoDB streams
DynamoDB Streams sends event to lambda function
OLAP Domain
Recreate NoSQL table into SQL table for OLAP Puposes
Monitor SCD via logs

SQL



Balances Table

client_id	PHP	USD
1	100	100
2	200	200

PK

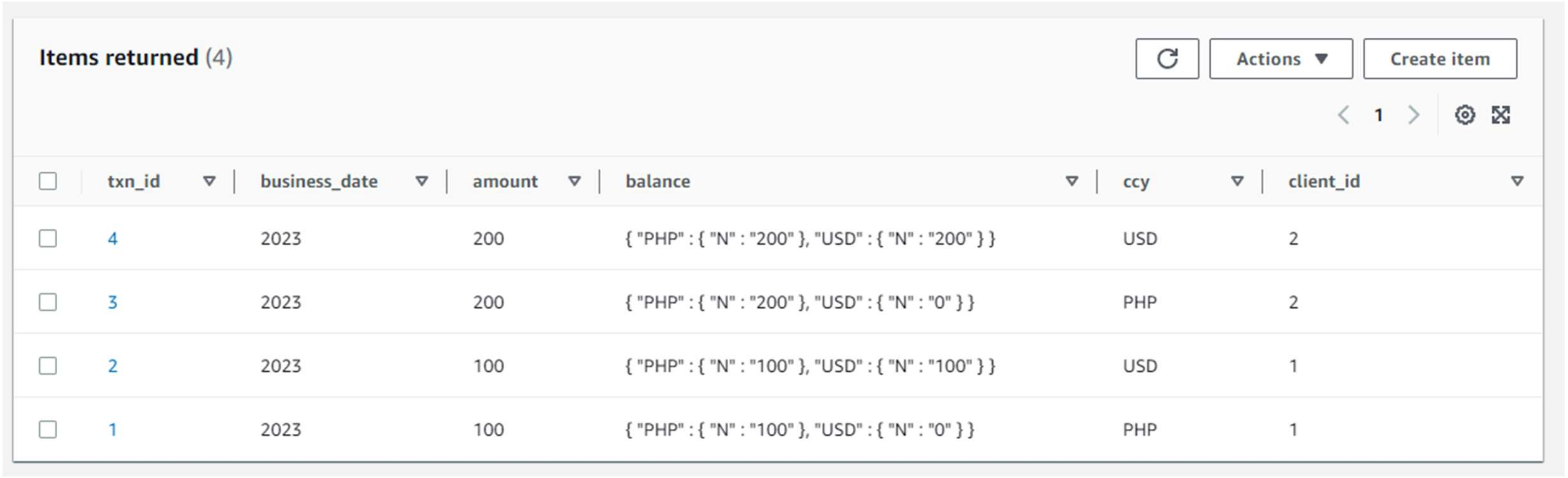
Transactions Table

txn_id	client_id	amount	ccy	business_date
1	1	100	PHP	2023
2	1	100	USD	2023
3	2	200	PHP	2023
4	2	200	USD	2023

PK

FK

DynamoDB



txn_id	business_date	amount	balance	ccy	client_id
4	2023	200	{ "PHP" : { "N" : "200" }, "USD" : { "N" : "200" } }	USD	2
3	2023	200	{ "PHP" : { "N" : "200" }, "USD" : { "N" : "0" } }	PHP	2
2	2023	100	{ "PHP" : { "N" : "100" }, "USD" : { "N" : "100" } }	USD	1
1	2023	100	{ "PHP" : { "N" : "100" }, "USD" : { "N" : "0" } }	PHP	1

Demo Time

- IaaC with AWS Cloudformation when deploying DynamoDB table
- Brief preview of AWS Codepipeline with code build and cloudformation
- Basic Operations (Scan, Get, Put, Delete, Query)
- Convert to pandas/spark dataframe
- From All Transactions Table, get a view of transactions for client 1