
AWS
re:Invent

DAT402-PT1

Amazon DynamoDB advanced design patterns – Part 1

Rick Houlihan

Senior Practice Manager,
AWS DynamoDB

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Fundamentals of NoSQL

- Partition Overloading, Secondary Indexes

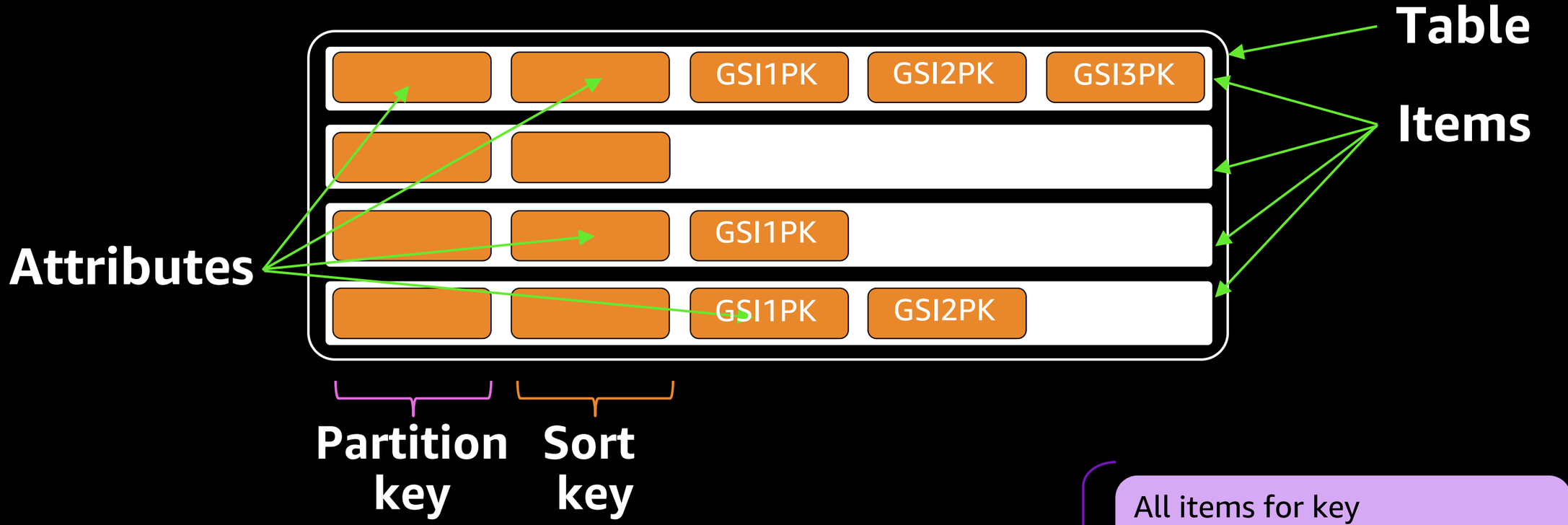
Design Patterns and Best Practices

- Global Tables and Summary Analytics
- Write Sharding for Selective Reads



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

NoSQL : An indexed object store



Mandatory
Key-value access pattern
Determines data distribution

Optional
Model 1:N relationships
Enables rich query capabilities

All items for key
==, <, >, >=, <=
“begins with”
“between”
“in”
sorted results
counts
top/bottom N values

Partition overloading

Use generic keys to facilitate heterogeneous partitions

Primary Key		Attributes			
PK	SK				
Customer_1	2019-11-29T08:31:28Z#O1	Source	Location	URL	CustomerType
		Online	US	www.amazon.com	Regular
	2019-11-29T08:31:28Z#O1#I1	ASIN	Status	Product	FCCID
		B07G6CQQYG	PROCESSING	BOOM 3	JNZS00170
	Customer_1	Login	Email	Name	Address
		jdoe	john@example.com	John Doe	23 5th Street, Anytown, USA

SELECT * WHERE PK=Customer_1 AND SK > 2019-10-29

Secondary indexes

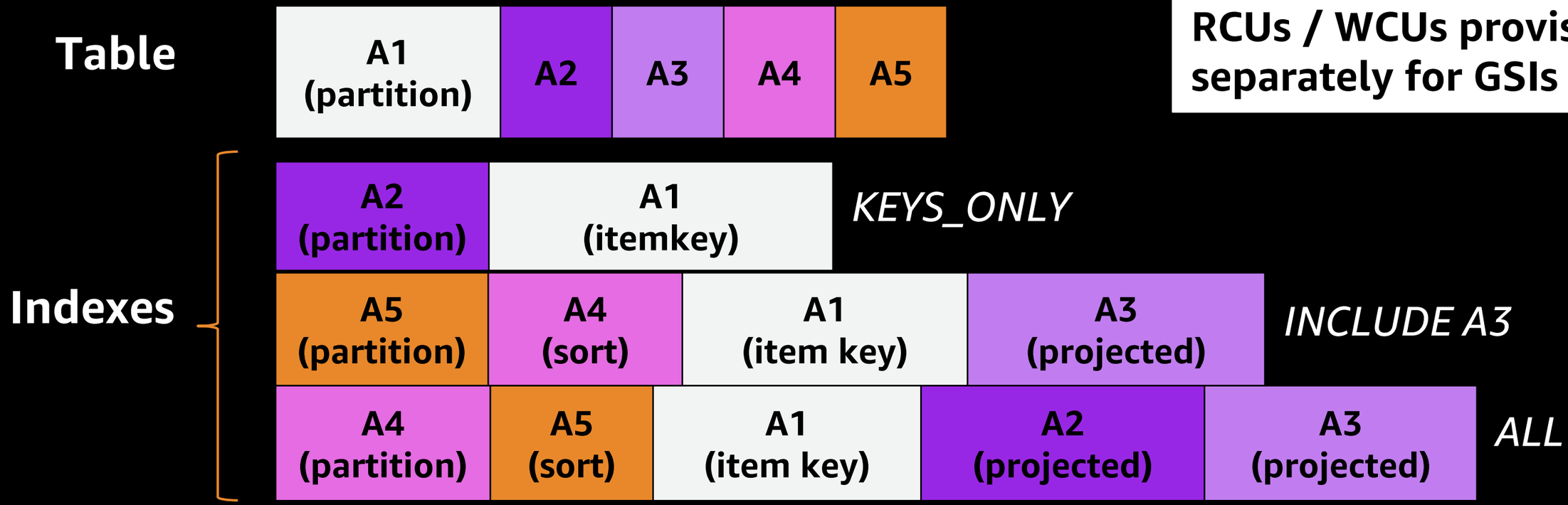
Online indexing

Support secondary access patterns

Index across all partition keys

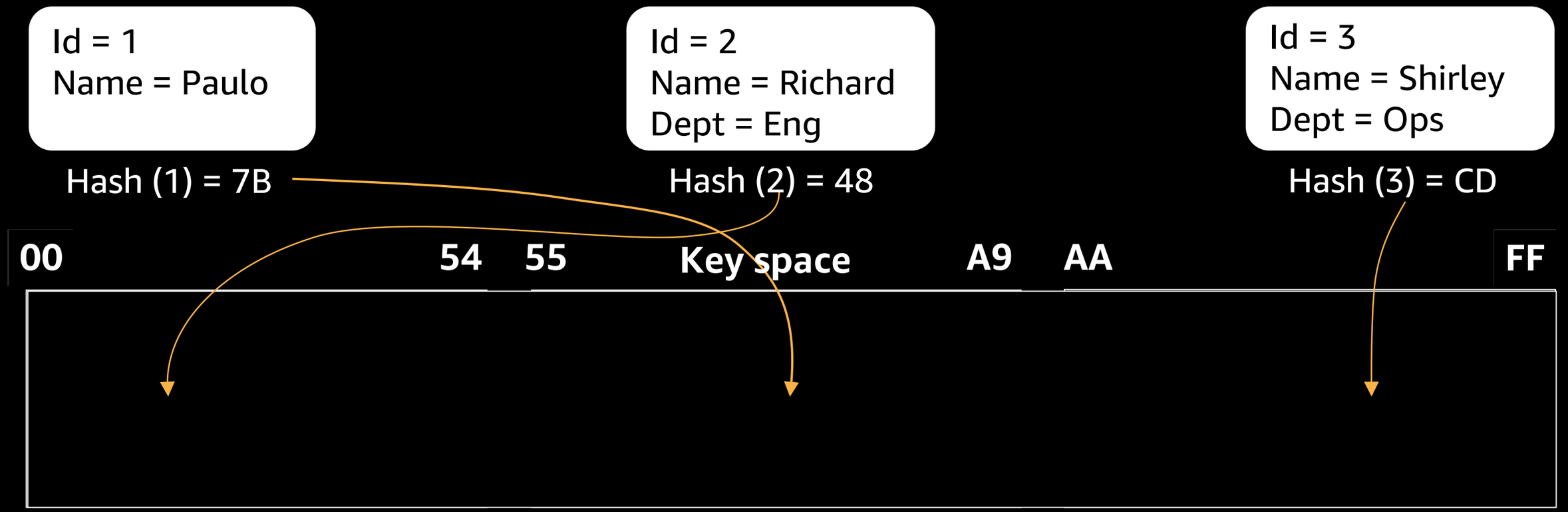
Use composite sort keys for compound indexes

RCUs / WCUs provisioned separately for GSIs



Partition / shard keys in NoSQL

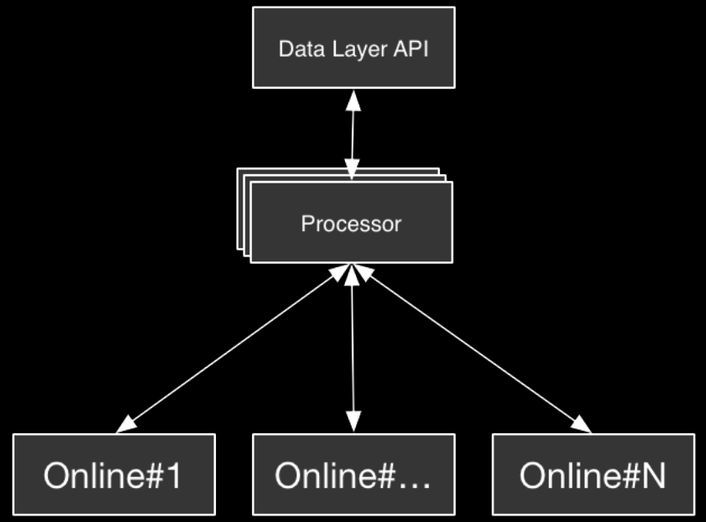
Partition / shard key is used for building an unordered hash index
Allows table to be partitioned for scale



Write sharding

Salt indexed keys to support high-density aggregations on GSIs

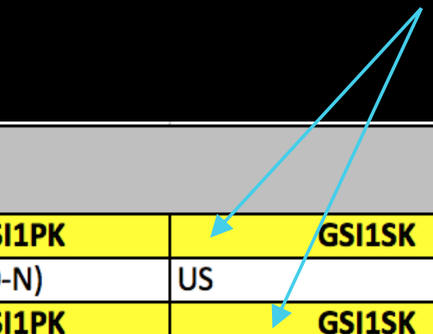
Primary Key		Attributes			
PK	SK				
Customer_1	2019-11-29T08:31:28Z#O1	Source	Location	Store	CustomerType
		Online#(0-N)	US	www.amazon.com	Regular
	2019-11-29T08:31:28Z#O1#I1	ASIN	Status	Product	FCCID
		B07G6CQQYG#(0-N)	PROCESSING	BOOM 3	JNZS00170
	Customer_1	Login	Email	Name	Address
		jdoe	john@example.com	John Doe	23 5th Street, Anytown, USA



- Abstract partitioning from clients behind an API
- Write across many partitions
- Use parallel processes to increase read throughput

Index overloading

Use generic keys once more to use indexes for multiple access patterns



Primary Key		Attributes			
PK	SK				
Customer_1	2019-11-29T08:31:28Z#O1	**GSI1PK**	**GSI1SK**	Store	CustomerType
		Online#(0-N)	US	www.amazon.com	Regular
	2019-11-29T08:31:28Z#O1#I1	**GSI1PK**	**GSI1SK**	Product	FCCID
		B07G6CQQYG#(0-N)	PROCESSING	BOOM 3	JNZS00170
	Customer_1	Login	Email	Name	Address
		jdoe	john@example.com	John Doe	23 5th Street, Anytown, USA

Index overloading

SELECT * WHERE PK=ONLINE#0 AND SK=US
…
SELECT * WHERE PK=ONLINE#N AND SK=US

Primary Key		Attributes			
GSI1PK	GSI1SK				
Online#(0-N)	US	PK	SK	Store	CustomerType
		Customer_1	2019-11-29T08:31:28Z#O1	www.amazon.com	Regular
B07G6CQQYG#(0-N)	PROCESSING	PK	SK	Product	FCCID
		Customer_1	2019-11-29T08:31:28Z#O1#I1	BOOM 3	JNZS00170

SELECT * WHERE PK=B07G6CQQYG#0 AND SK=PROCESSING
…
SELECT * WHERE PK=B07G6CQQYG#N AND SK=PROCESSING

High Velocity Aggregations



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Trade Aggregation Service

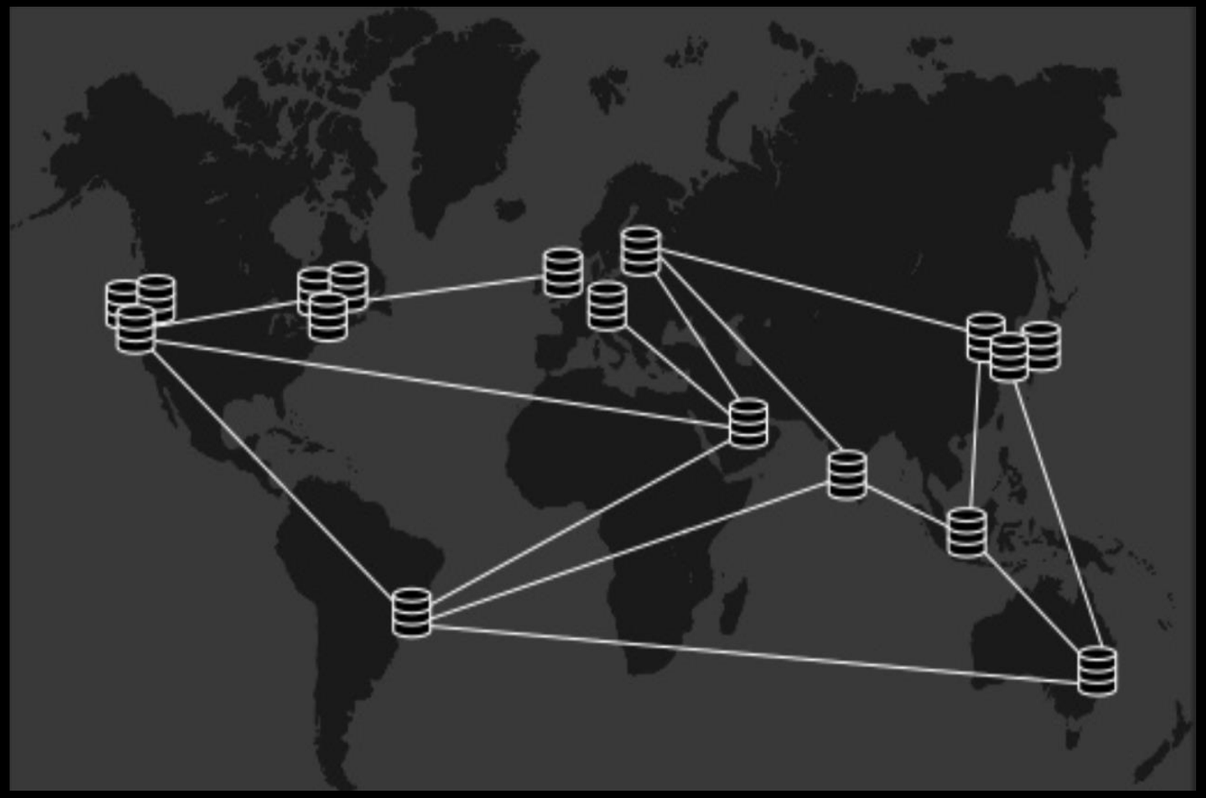
Scale Requirements

- Thousands of securities, millions of transactions
- High volume securities trade at up to 10K tps
- Trades across all securities peak at 100K tps
- Data availability across regions within 5 seconds

Four Access Patterns

- Get transaction by ID
- Get all transactions for security
- Get trailing 30-day summary for security
- Get 30-day summaries for all securities

DynamoDB Global Replication

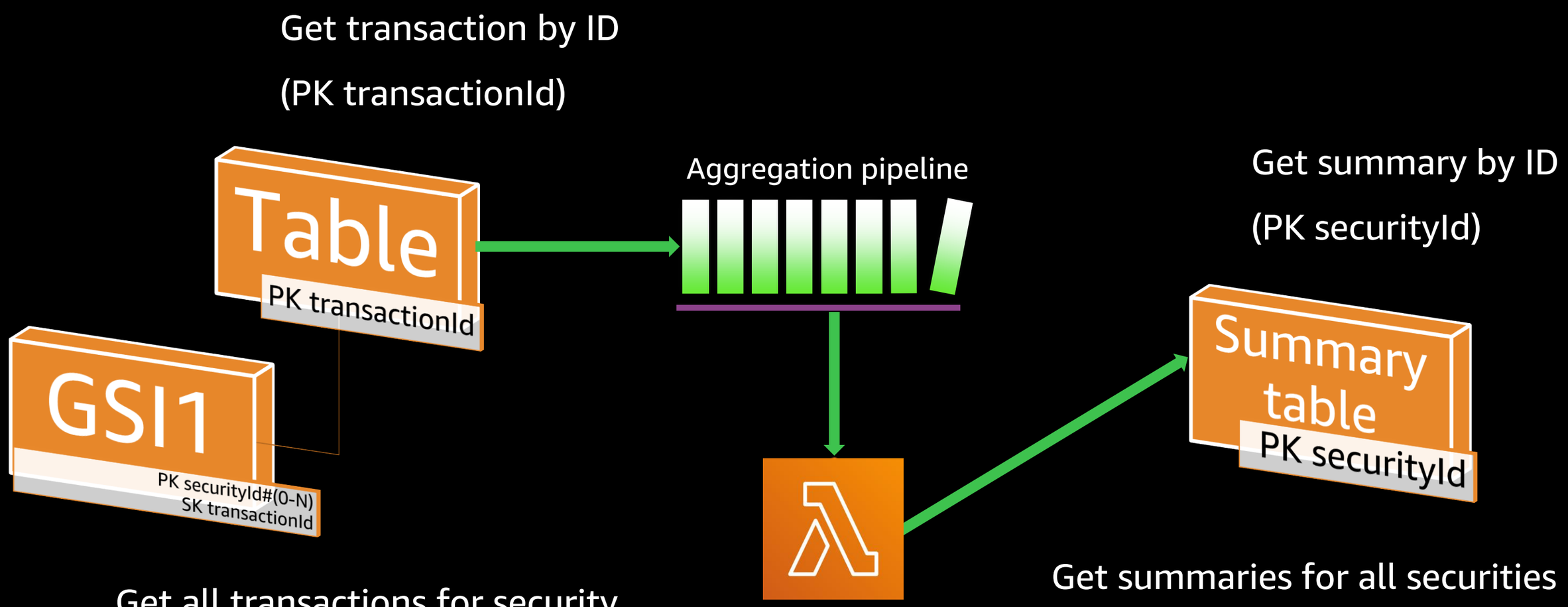


Fully managed, active-active, multi-region

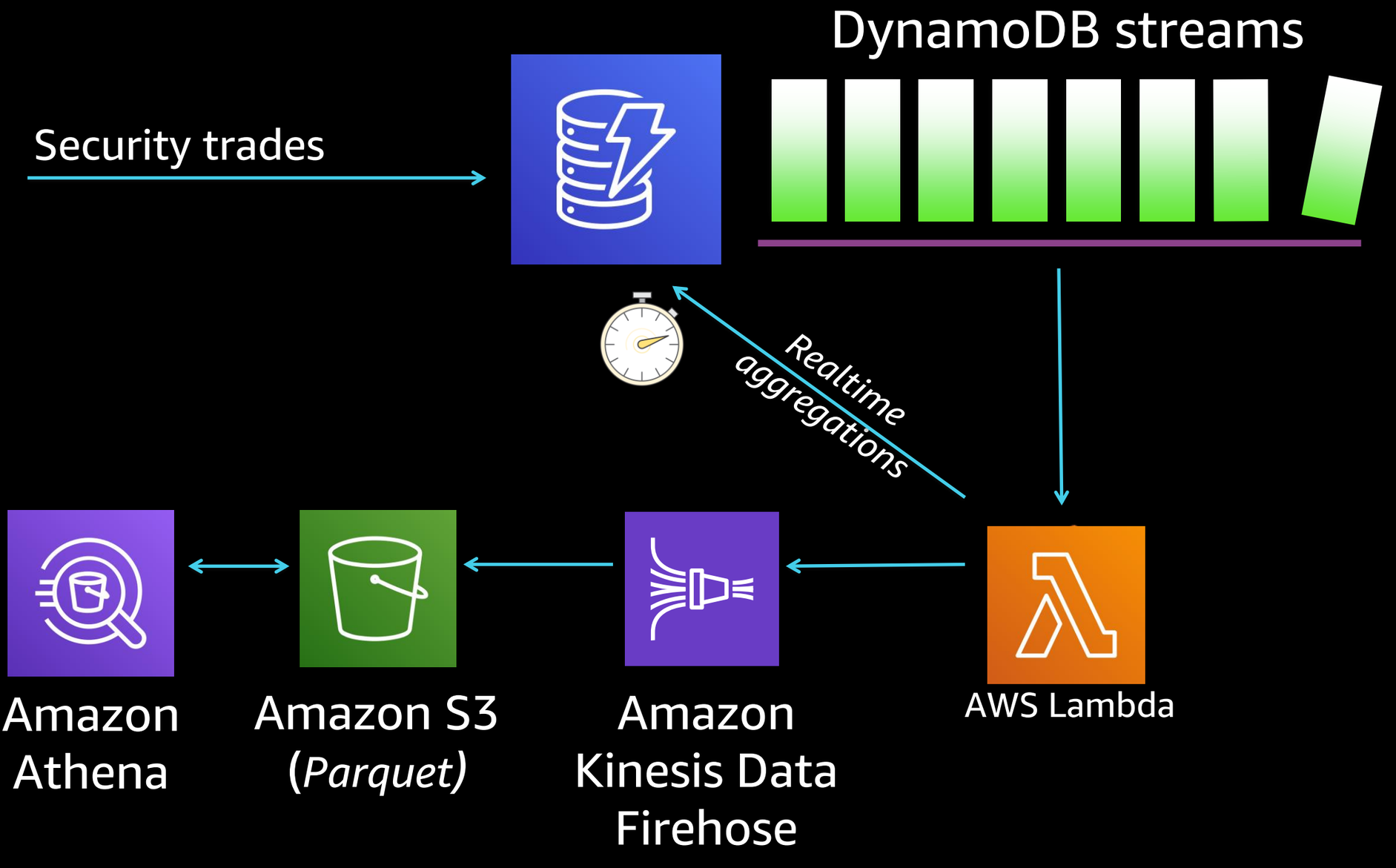
Cross region replication in less than 2 seconds

Deploys in minutes from the AWS CLI, Cloud Formation, your custom code, or the DynamoDB service console.

Table design



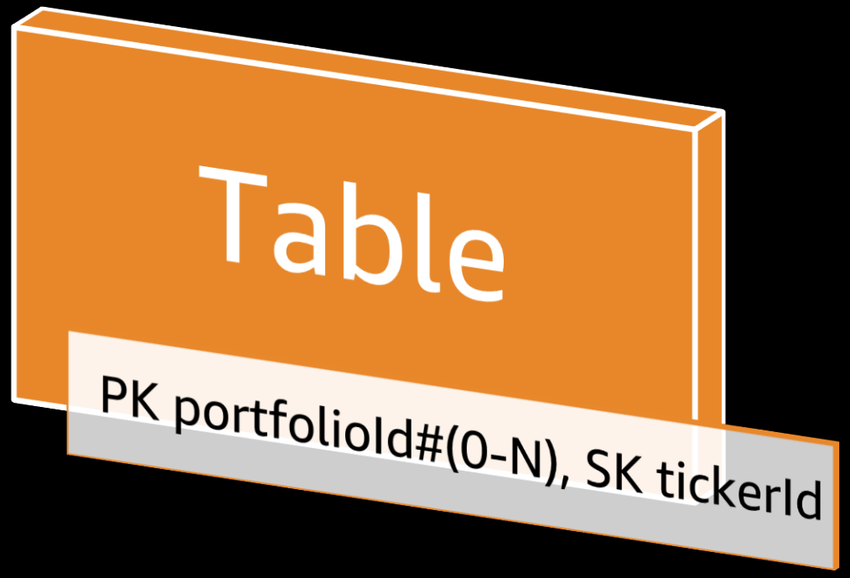
Summary analytics



Optimized Table Schema (GSI removed)

Get transactions by ID

PK = securityId#Math.abs(transactionID.hashCode() % 100), SK = transactionID



Get all trades for securityID

PK = securityID#(0-100)

Shaping the Shard Key

Use sort key to identify shard



Transaction Item:

```
new Item().withString("PK", String.format("%s#%d", security, Math.abs(transId.hashCode() % 100))
        .withString("SK", transId).withString("type", "transaction").withMap("data", data)
```

Summary Item:

```
new Item().withString("PK", security).withString("SK", region).withString("type", "summary")
        .withMap("dailyTotals", totals)
```

Keep watching...

DAT402–PT 2

Amazon DynamoDB advanced design patterns – Part 2

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Related sessions

DAT305-PT1

Data Modeling with Amazon DynamoDB – Part 1

DAT305-PT2

Data Modeling with Amazon DynamoDB – Part 2

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.