

# Amazon DynamoDB

#### Deep dive on key features that drive business impact

Pete Naylor - DynamoDB Specialist SA - AWS

March 25, 2021





## **Topics of discussion**

- Elasticity throughput and storage
- Caching
- Backup/restore
- Export to Amazon S3
- PartiQL support (SQL statements)
- Multi-Region architectures
- Streaming of change data
- Materializing views in additional indexes
- Building on DynamoDB



# Elasticity – throughput and storage



## **Throughput modes**

Provisioned capacity mode (since service GA in 2012)

- Storage scales automatically.
- Provide throughput requirement as signal to the service.
- Auto scaling read and write capacity since June 2017.

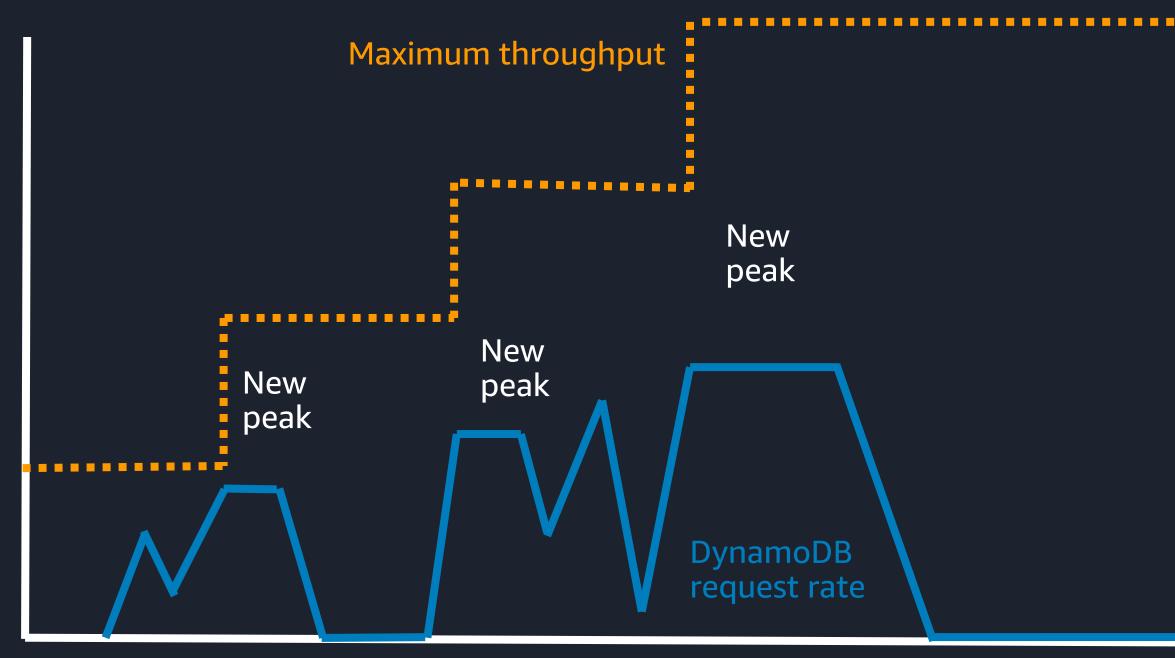
On-demand capacity mode (added November 2018)

- Storage scales automatically.
- Throughput scales instantly within 2x prior maximum consumption level pay per request.





### Partitions split to double your capability

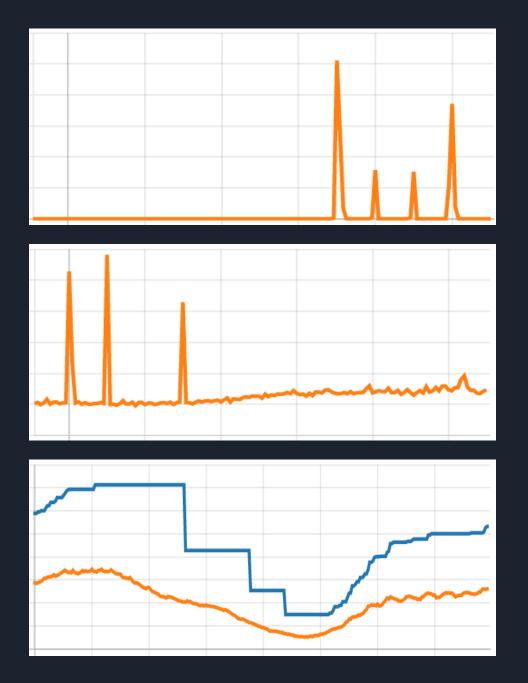


#### . . . . . . . . . . . . . . . . . .

#### Throughput maximum never descales



## **Choosing the right capacity mode**



Cost versus price

#### Engineering time and opportunity cost

#### Operational load and risk

#### Resource lifecycle and utilization

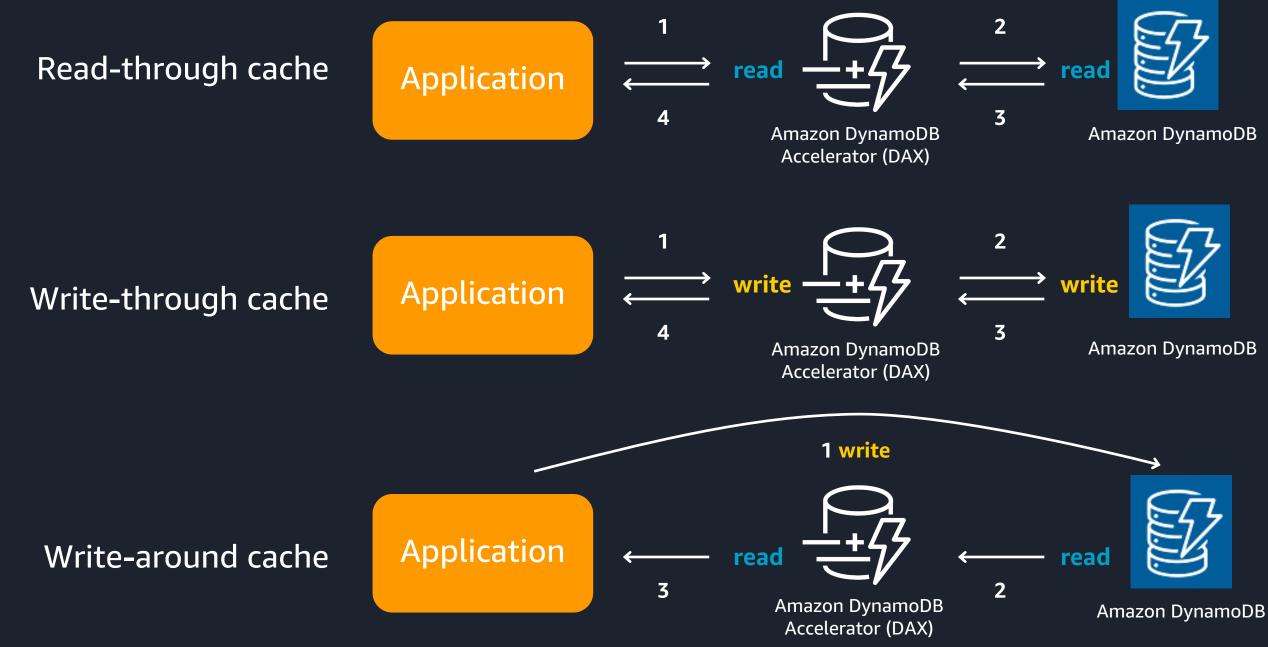




# Caching

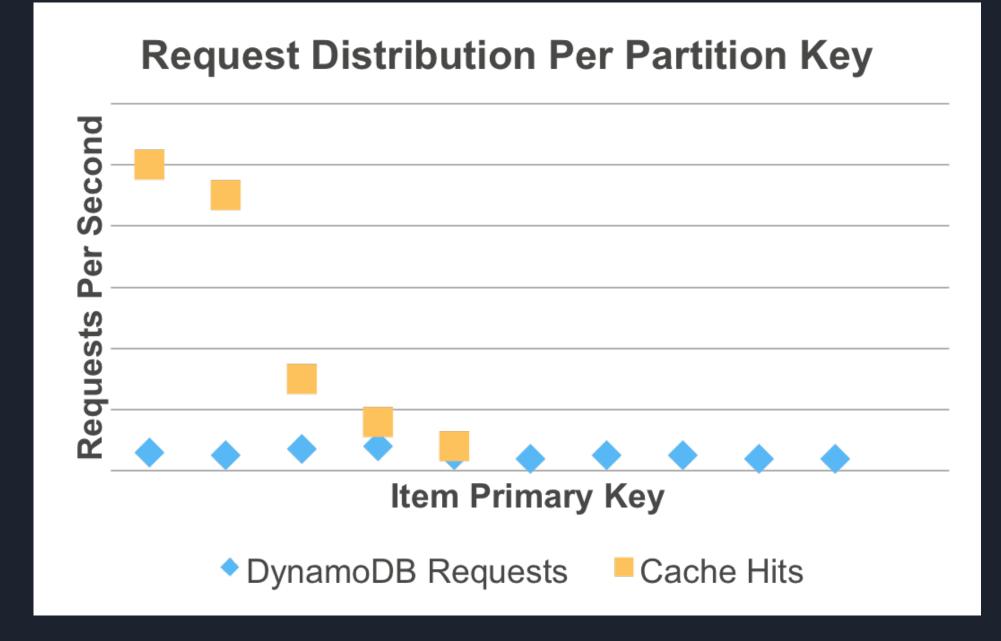


#### **DynamoDB Accelerator (DAX)**

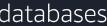




#### **DAX:** Reduced read unit consumption for DynamoDB







#### **DAX:** Business benefits



#### Lower latency – more delightful customer experience



#### Smoothing of unanticipated skew in data pressure



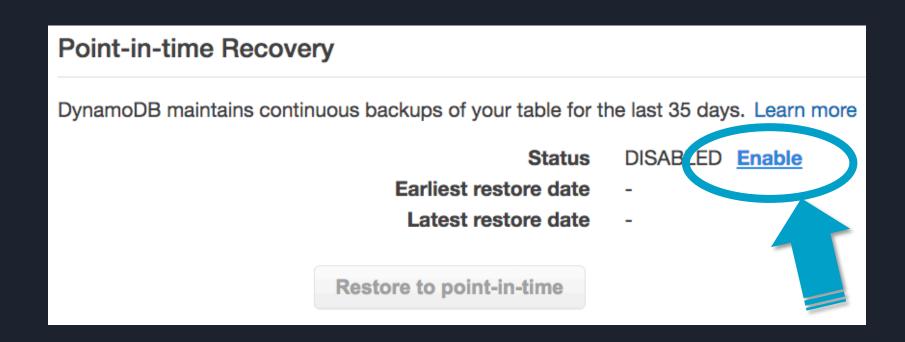
#### Less read units consumed – improved cost efficiency



# Backup/restore



## **Point-in-time recovery (PITR): properties**



- 35-day rolling window of protection
- Restore to a new table with second granularity
- No impact to ongoing operations
- Covers table deletion and corruption risks





## **Point-in-time recovery (PITR): restore scenarios**

Point-in-time Recove	ry						
New table name*	sense-of-relief						
Restore date and time	November 9, 2020 09 ~ : 58 ~ : 25 ~ UTC-8						
Earliest restore date	November 9, 2020 at 9:58:25 AM UTC-8						
Latest restore date	November 9, 2020 at 9:58:25 AM UTC-8						
	Restore entire table data						
	<ul> <li>Restored table will include all local secondary indexes and global secondary indexes.</li> </ul>						
	Restore without secondary indexes						
	<ul> <li>Restored table will exclude the local secondary indexes and global secondary indexes. Note: Restores can be faster and more cost efficient if you choose to exclude secondary indexes from being created.</li> </ul>						

- ulletdelete or data change
- Restore from known-good ullettimestamp to original table name
- Restore from known-good ullettimestamp to a new table
- Comparison and selective ulletreversion of any unwanted changes?

## Accidental (or malicious) table



## **Point-in-time recovery (PITR): peace of mind**



- DynamoDB often stores critical business data  $\bullet$
- Business continuity and disaster recover are hard and often not fully  $\bullet$ considered or tested
- Having PITR as a continuous backup is an easy first step for production  $\bullet$ tables that will give you a starting point for recovery





# **Export to Amazon S3**



## Exporting table data to S3

Streams	Exports					
Exports to S					View	Export to S3
<b>Q</b> Find export	ts				<	1 > 🤞
Export	ID	▽	Table name ⊽	Destination S3 bucket ⊽	Status $\bigtriangledown$	Start time (GMT+1)
0	n:aws:dynamodb:eu-west-1: table/MyTableToExpo 01603967139384-5a2e758	ort/	MyTableToExport	s3://my- dynamodb-export- bucket	) Exporting	Oct 29, 2020 11:25:39

You now can make a recovery from your PITR backup into S3 – simple API or use the console

- Consumes no read units no ulletload on your table
- Output to your S3 bucket in ullet(compressed) DynamoDB JSON format or Amazon Ion format
- Restore to any second in your ulletPITR window



# PartiQL support (SQL statements)



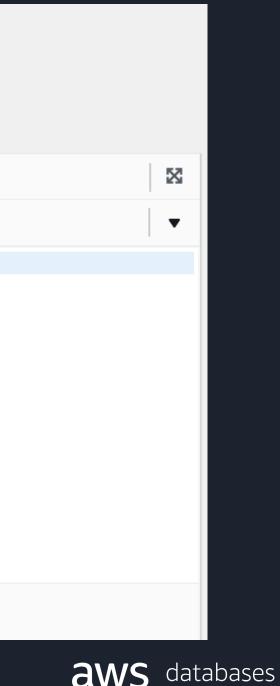
#### SQL-compatible statements for DynamoDB operations

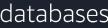
DynamoDB > PartiQL editor

#### PartiQL editor

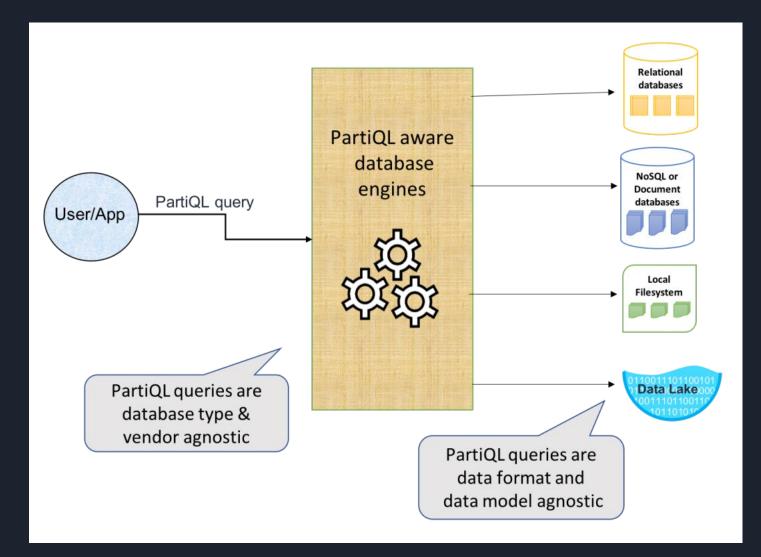
Operations performed using the PartiQL editor may incur charges. Learn more 🗹

Resources	С×	
	-	Query 9 $\times$ $\bigcirc$ Query 10 $\times$ +
<b>Q</b> Find tables		<pre>1 SELECT * FROM "composite" WHERE "pk" = 'abc123' AND begins_with("sk", 'inv#');</pre>
Tables (5) < 1	> ©	
▶ composite		
dynacount	•••	
<ul> <li>partiqltest</li> </ul>	•••	
simple	•••	
usertable	***	
		Run Clear





#### SQL-compatible statements for DynamoDB operations



PartiQL is an open source specification for a SQL-compatible query language supporting any level of data structure. Three new **DynamoDB API actions:** 

ExecuteStatement

BatchExecuteStatement

ExecuteTransaction

SQL verbs:

INSERT, UPDATE, SELECT, DELETE



# Multi-Region architectures



## **Global tables: Multi-Region architectures**



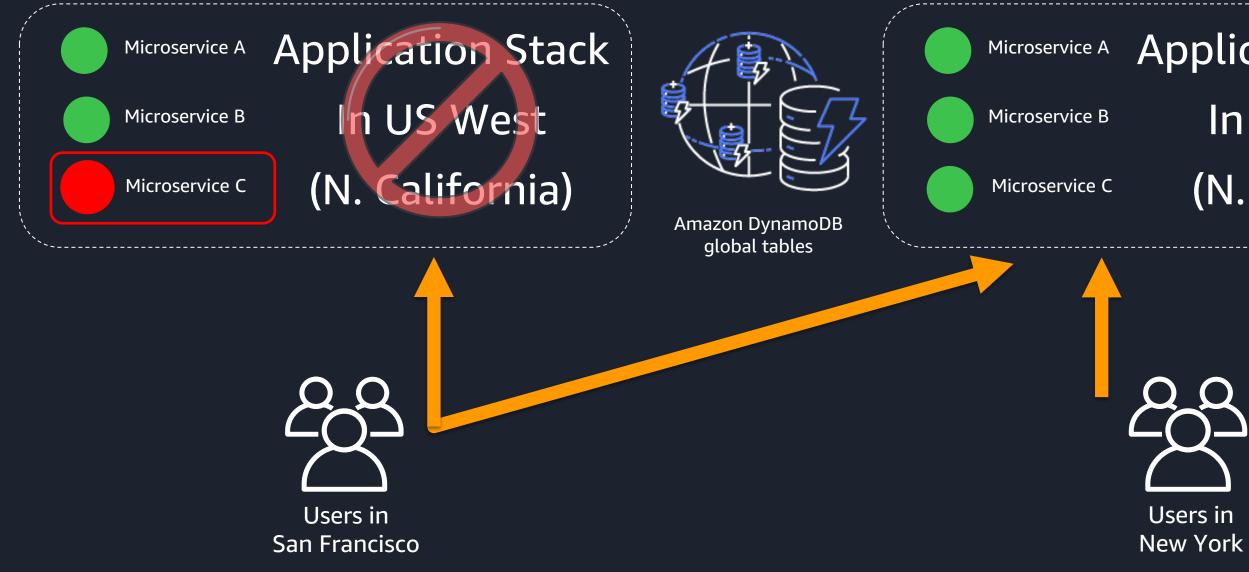
- Geographically 1. distributed customer base
  - ulletoperations
  - •
- 2. Business continuity / disaster recovery

## Low-latency data

Meet legal and data regulatory compliance



### Global tables: architecture beyond the data



## Application Stack In US East (N. Virginia)



databases

## **Global tables: traffic distribution choices**

- Reads and writes to a single Region (active-passive) ullet
- Local reads, with writes to a single "primary" Region ullet
- Local reads and writes (active-active) consider item "homing" ightarrow
- Manual and/or automatic failover ullet
- Amazon Route 53 health checks and ightarrowrouting policy – latency, geolocation
- Consider AWS Global Accelerator  $\bullet$ for fastest failover, lowest latency
- Amazon API Gateway can provide rate limiting and caching







databases

# Streaming of change data



#### **DynamoDB and change data capture**



#### DynamoDB table

#### Stream to an Amazon Kinesis data stream

#### Stream details

Kinesis Data Streams for DynamoDB captures item-level changes in your table, and replicate the changes to a Kinesis data stream. You then can consume and manage the change information from Kinesis. Charges apply

Destination Kinesis data stream				
Choose a stream		▼	Create new 🖸	C
he destination stream must be in the same AWS account and AN	WS Region as this Dyna	moDB	table. Ensure that it has su	fficient capacit
ccommodate streaming from this table. To learn more, see shar				

#### You now can capture change events direct to Amazon Kinesis Data Streams

AWS Lambda



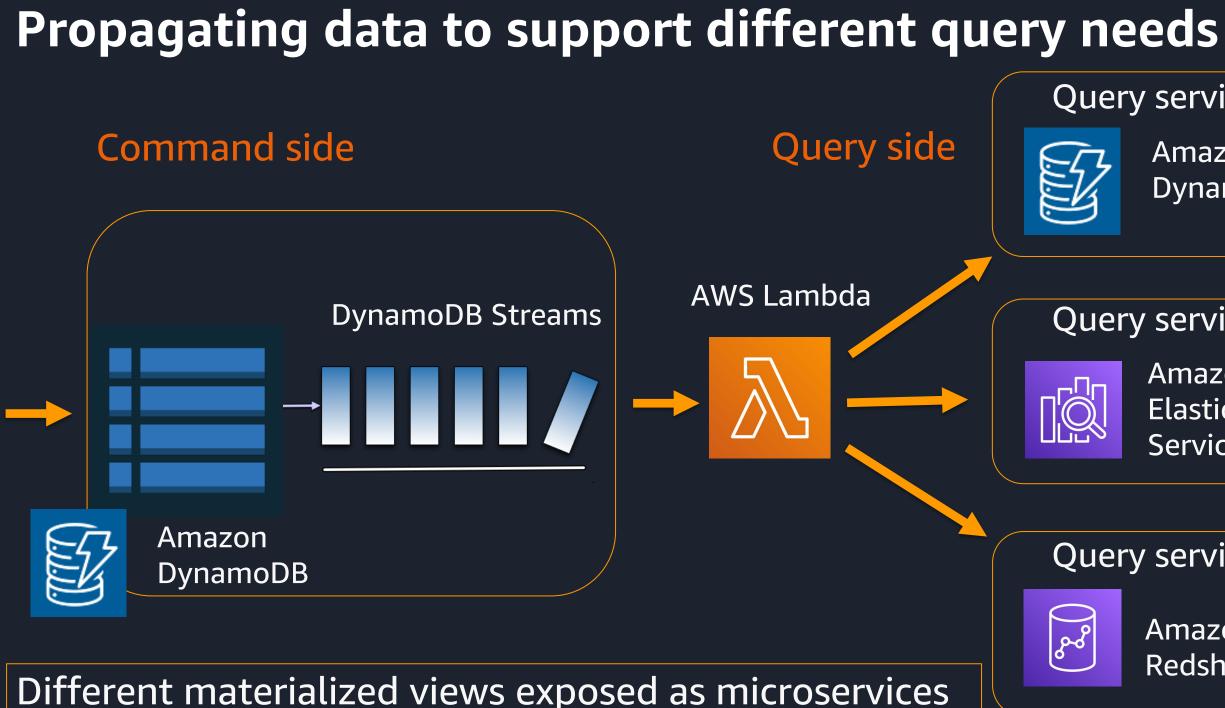
**Amazon Kinesis** Data Streams





# Materializing views in additional indexes





#### Query service 1

Amazon DynamoDB

#### Query service 2

Amazon Elasticsearch Service

#### Query service 3

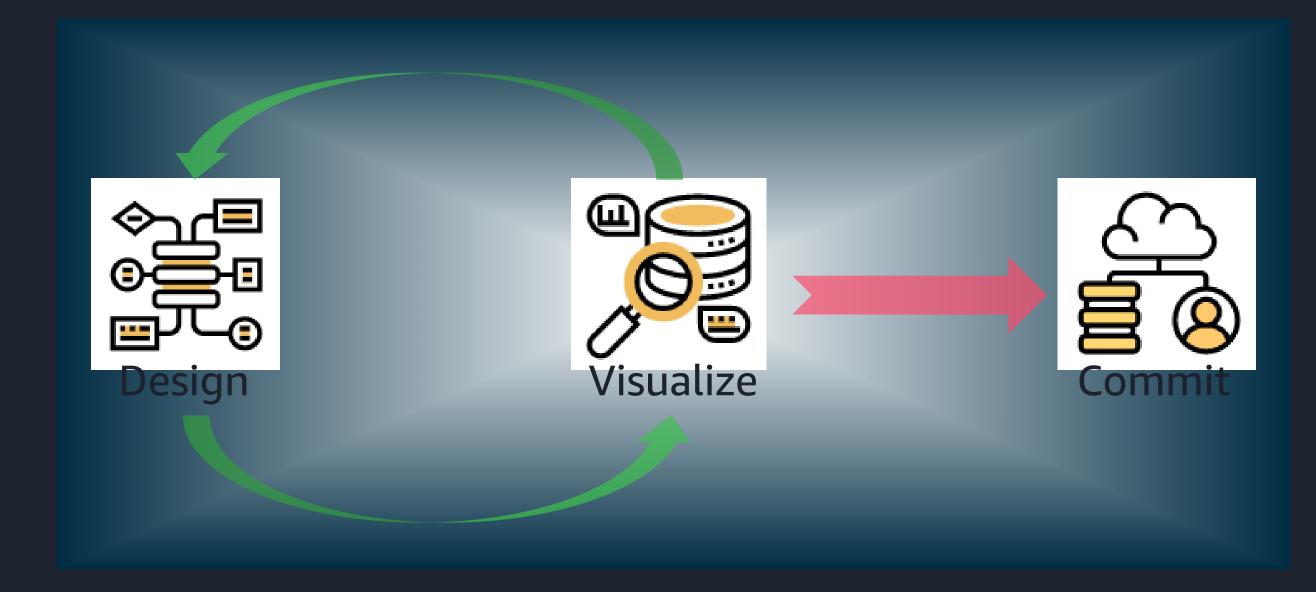
Amazon Redshift



# Building on DynamoDB



## NoSQL Workbench: data modeling flow





## NoSQL Workbench: collaborate and document

Visualizer	Aggregate view				
Data model 🕖					
OnlineBank 🗸	Primary key			A	
<ul> <li>▲</li> <li>▲</li> </ul>	Partition key: acct	Attributes			
Accounts     Update				bal	
				543.55	
Transactions Update				bal	
				228.42	
Aggregate view	Transactions				
Commit to Amazon DynamoDB	Primary key				
	Partition key: txid	Attributes			
	7d622075-f2f1-4dd4-8aaf-fb29e87c2b9a	time		desc	
	10022013-1211-4004-0881-102960162098	1590987629		\$73.00 from 12345 to 5432	







#### Pete Naylor DynamoDB Specialist SA



# Thank you!

# @DynamoDB – Twitter aws.amazon.com/dynamodb



## Implementing application workflows

- Workbench operation builder generates sample code for you (Java, JavaScript, Python)
- AWS Management Console, AWS CLI, and Amazon ulletCloudWatch
- ReturnConsumedCapacity igodol
- SDKs: Java, Python, .NET, Go, JavaScript, Node.js, ulletC++, Ruby, and PHP



#### **AWS Free Tier** 25 GB storage 25 provisioned RCU 25 provisioned WCU

## Resources

- **On-demand capacity**
- **DynamoDB** Accelerator (DAX)
- **Point-in-time recovery**
- **Global tables**
- NoSQL Workbench
- Advanced data modeling (YouTube)
- What's new with Amazon DynamoDB
- Amazon's purpose-built database journey
- **Caching challenges and strategies (Amazon Builders' Library)**
- **Demos for Devs (Twitch)**



