AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

DAT333

DynamoDB deep dive: Advanced design patterns

Rick Houlihan
Worldwide Technical Leader – NoSQL Services
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Brief history of data processing (why NoSQL?)
- NoSQL data modeling
- Normalized versus denormalized schema
- Amazon DynamoDB and the serverless ecosystem
- Modeling for real-world applications

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

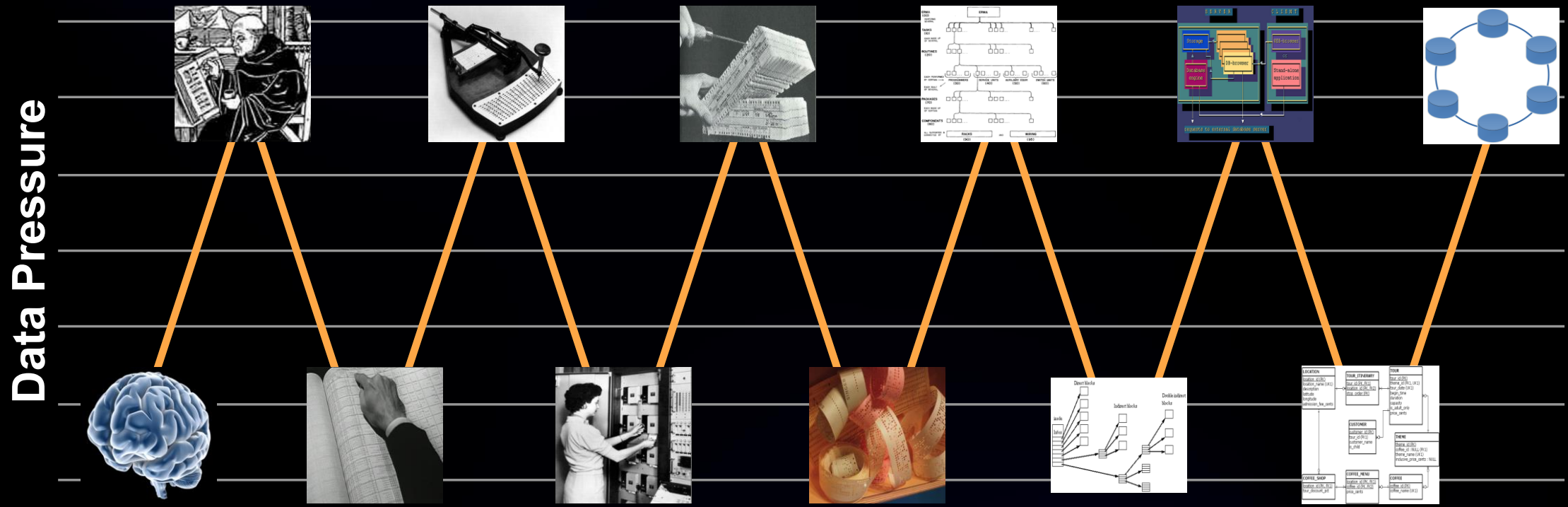
History of data processing

“History repeats itself because
nobody was listening the first time.”

Anonymous

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Timeline of database technology

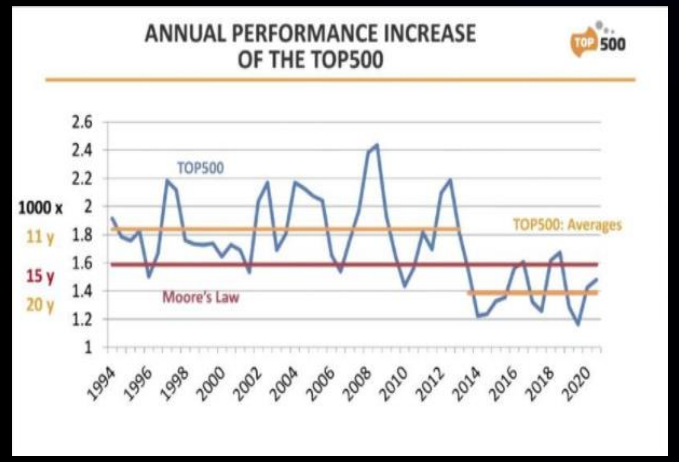


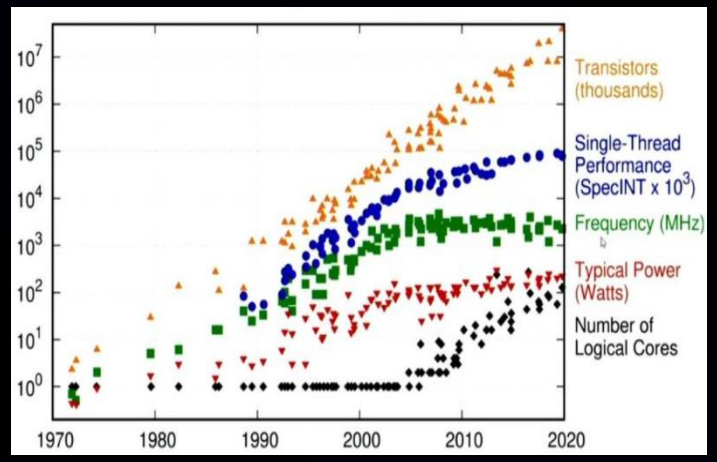
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

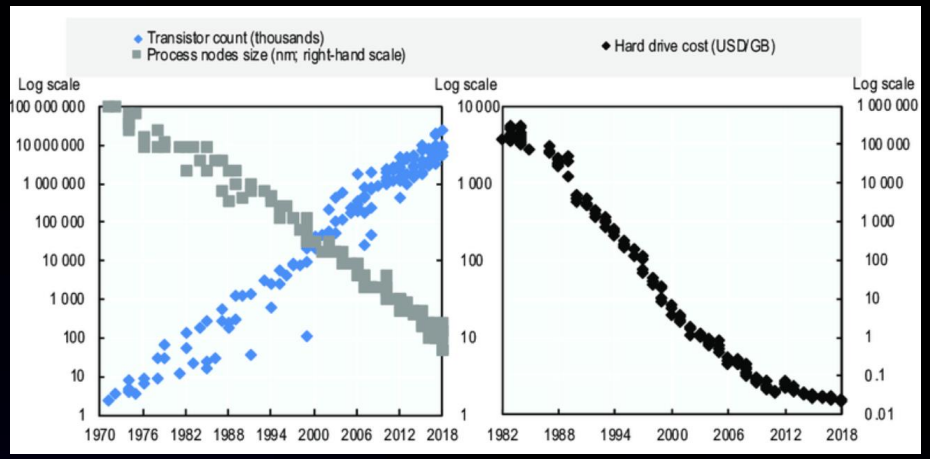
Moore’s Law no longer an enabler for RDBMS

CPU performance is flattening





Storage cost will drop



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

It’s not new thinking…

On normalization:

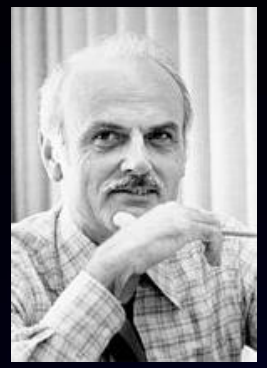
“The simplicity of the array representation which becomes feasible when all relations are cast in normal form is not only an **advantage for storage** purposes but also for **communication of bulk data** between systems which use widely different representations of the data.”

On denormalization:

“If the strong redundancies in the named set are directly reflected in strong redundancies in the stored set (or if other strong redundancies are introduced into the stored set), then, generally speaking, extra storage space and update time are consumed with a potential drop in query time for some queries and in load on the central processing units.”

On defining an ERD:

“Ideally, the variety of permitted data representations should be just adequate to cover the spectrum of performance requirements of the total collection of installations. Too great a variety leads to unnecessary overhead in storage and continual reinterpretation of descriptions for the structures currently in effect.”



A Relational Model of Data for Large Shared Data Banks

Edgar Frank "Ted" Codd
19 August 1923 – 18 April 2003

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why NoSQL?

SQL	NoSQL
Optimized for storage	**Optimized for compute**
Normalized/relational	Denormalized/hierarchical
Ad hoc queries	Instantiated views
Scale vertically	Scale horizontally
Good for OLAP	Built for OLTP at scale

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling DynamoDB

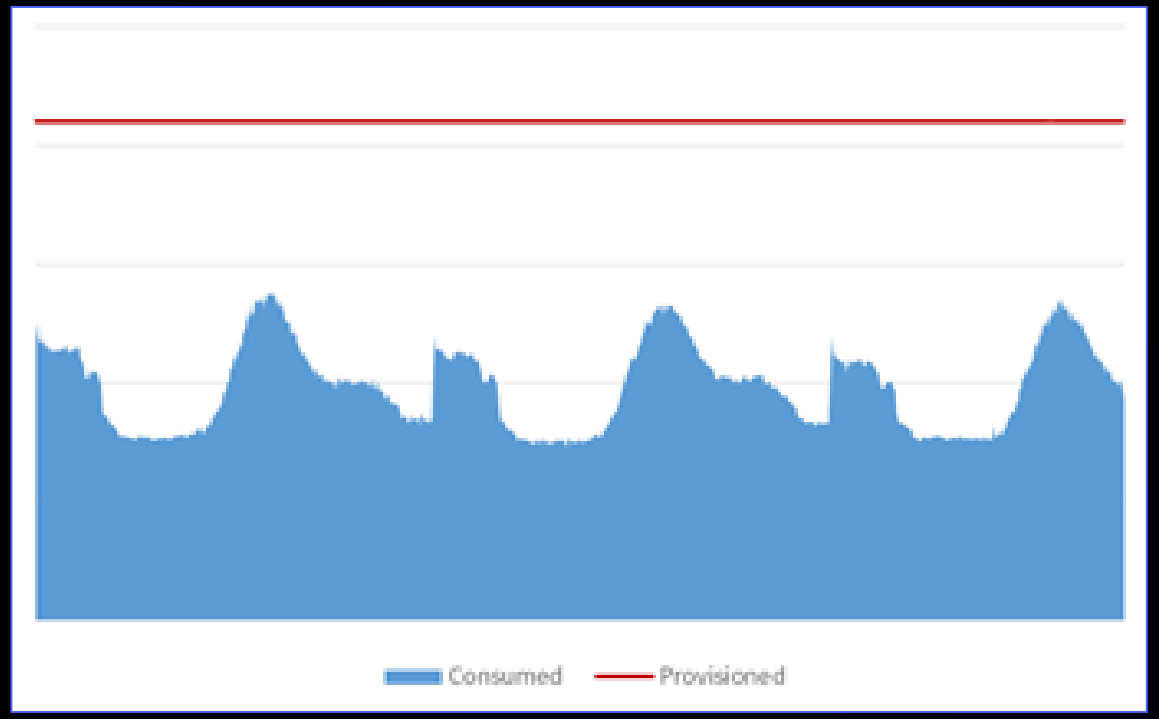
"We are stuck with technology when what we really want is just stuff that works."

Douglas Adams

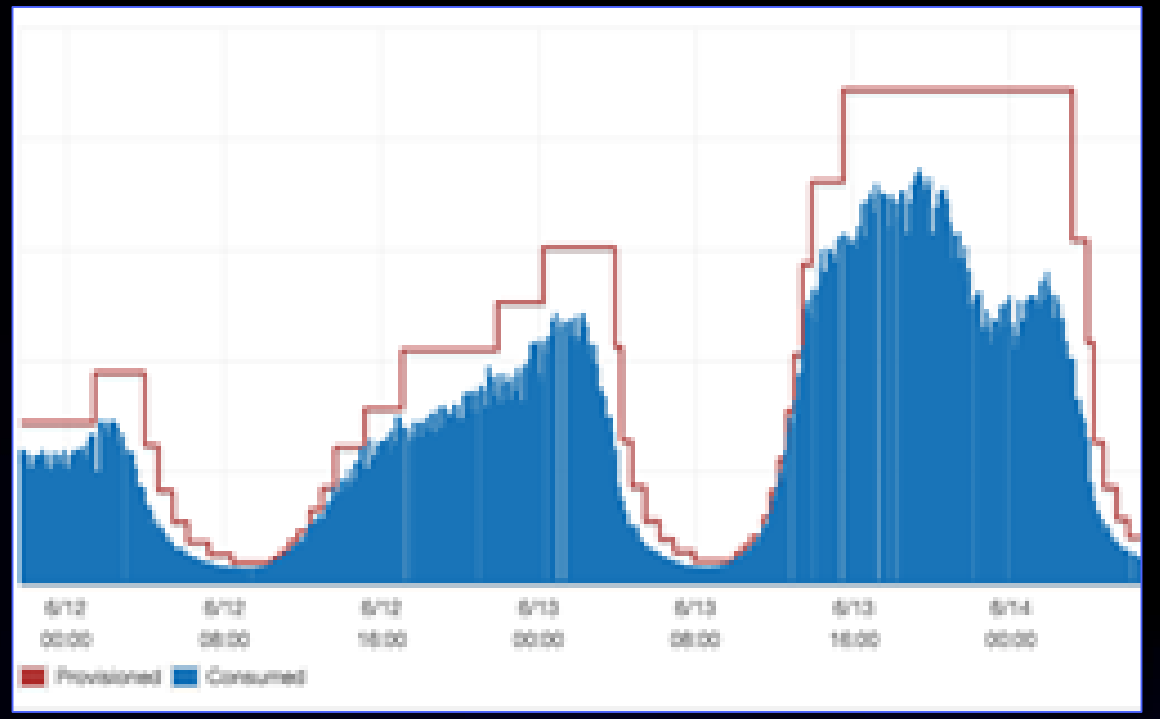
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Auto scaling

Throughput automatically adapts to your actual traffic



Without auto scaling



With auto scaling

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

On-demand capacity mode



Features

- No capacity planning, provisioning, or reservations – simply make API calls
- Pay only for the reads and writes you perform

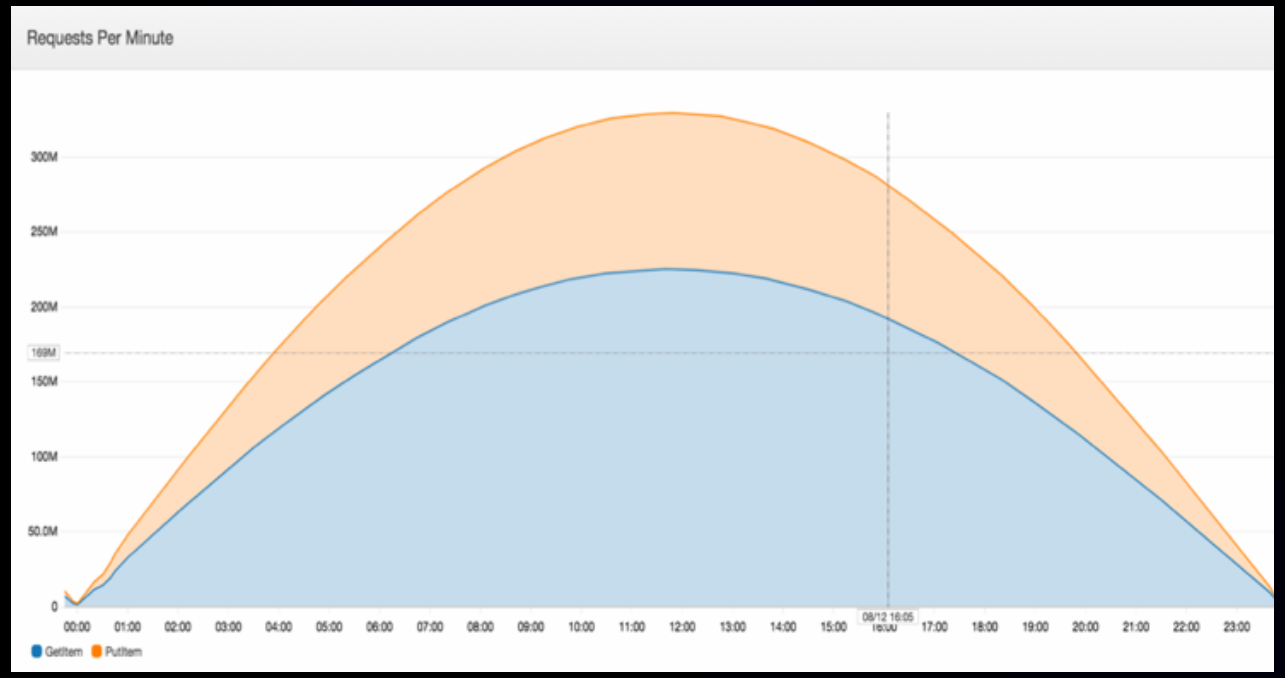
Key benefits

- Eliminates trade-offs of overprovisioning or underprovisioning
- Instantly accommodates your workload as traffic ramps up or down

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

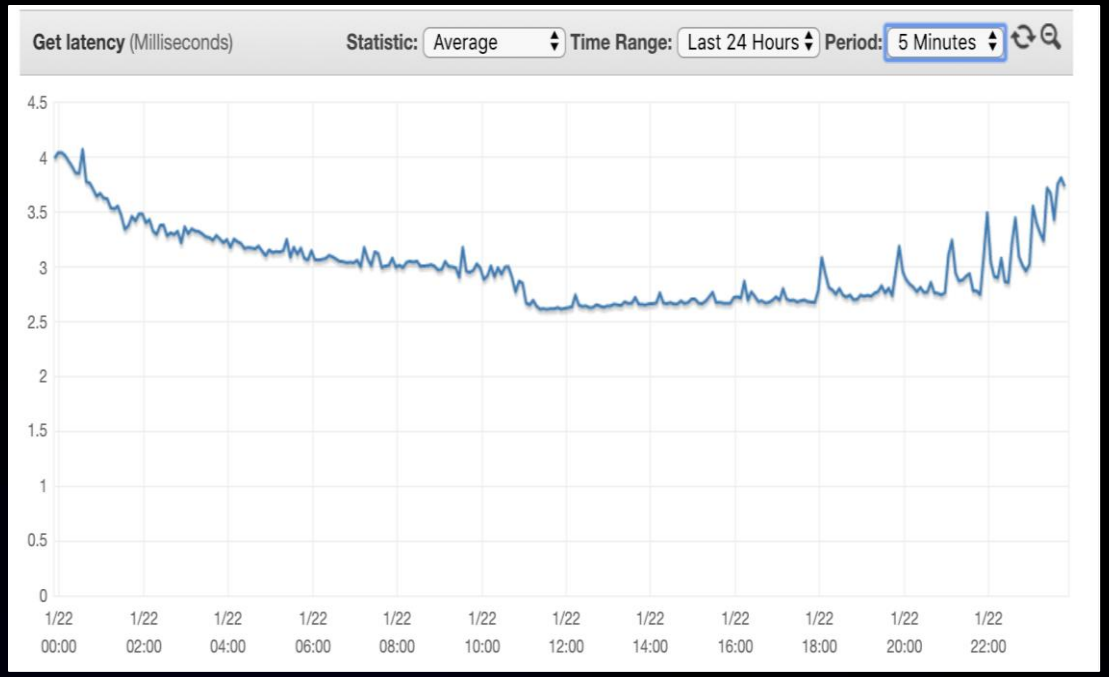
Performance at any scale

High request volume



Many **millions of requests** per second per table

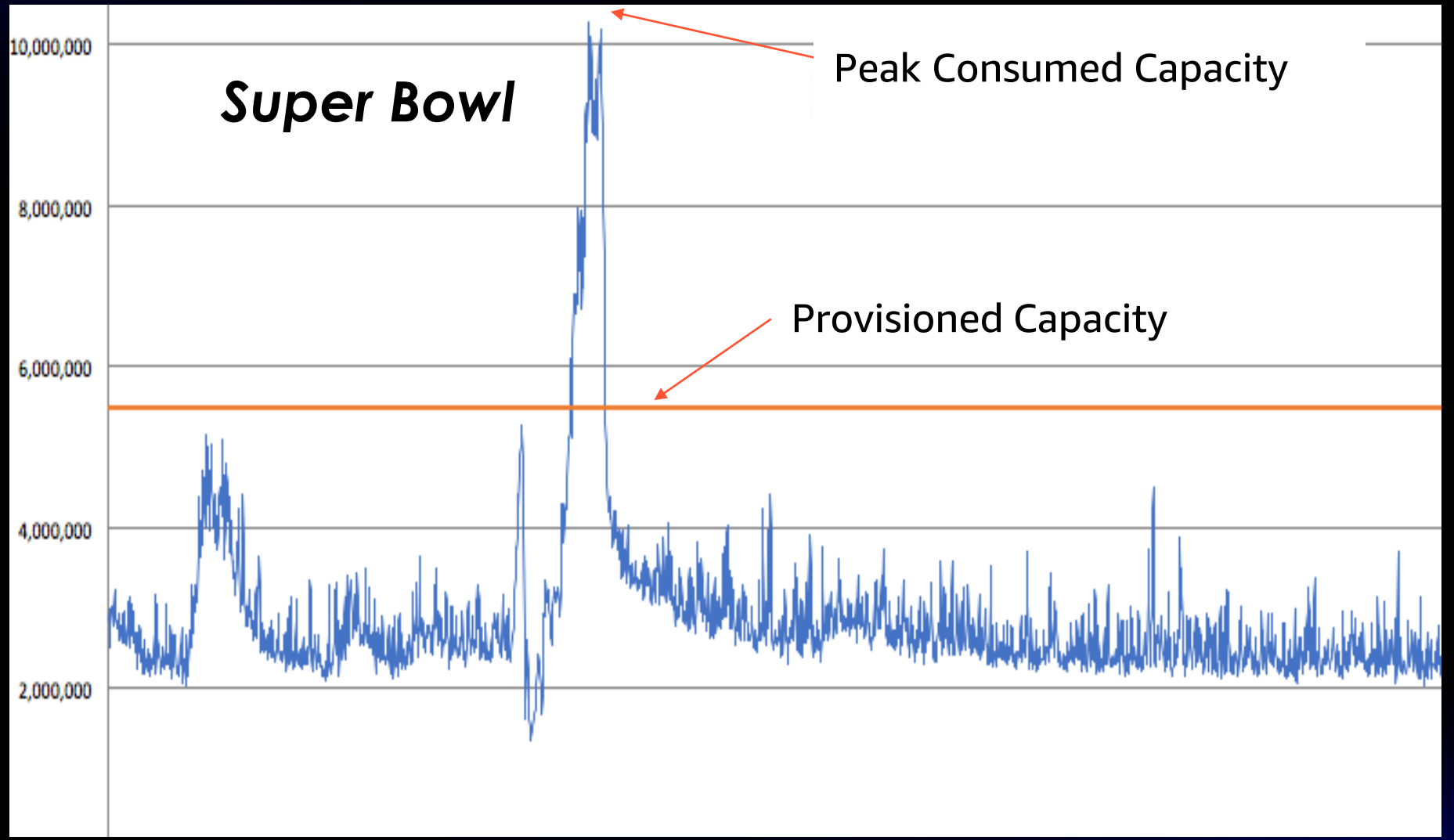
Consistent low latency



Millisecond variance

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

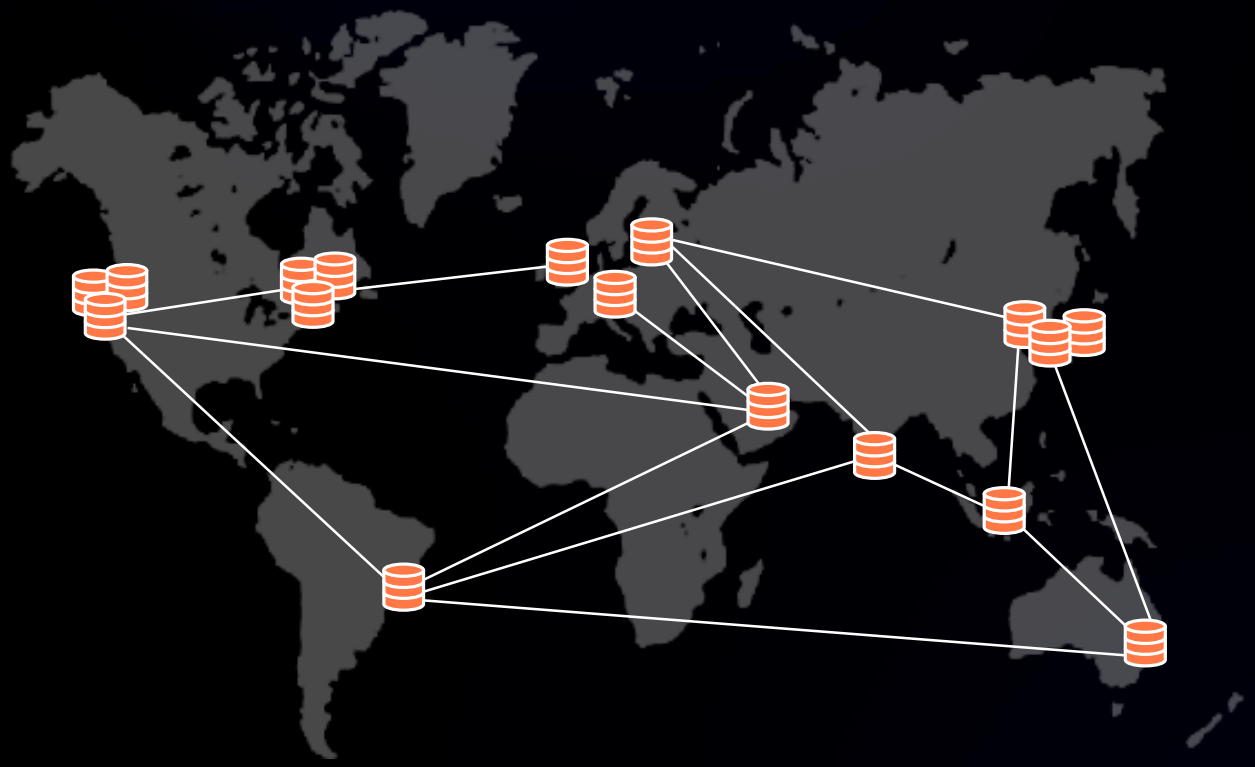
Global-scale events: Elastic is the new normal



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Global tables with multi-Region replication

Performance at scale



Build high-performance, globally distributed applications

Low-latency reads and writes to locally available tables

Multi-Region redundancy, resiliency, and 99.999% availability

Multi-active writes from any Region

Easy to set up and no application rewrites required

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

NoSQL fundamentals

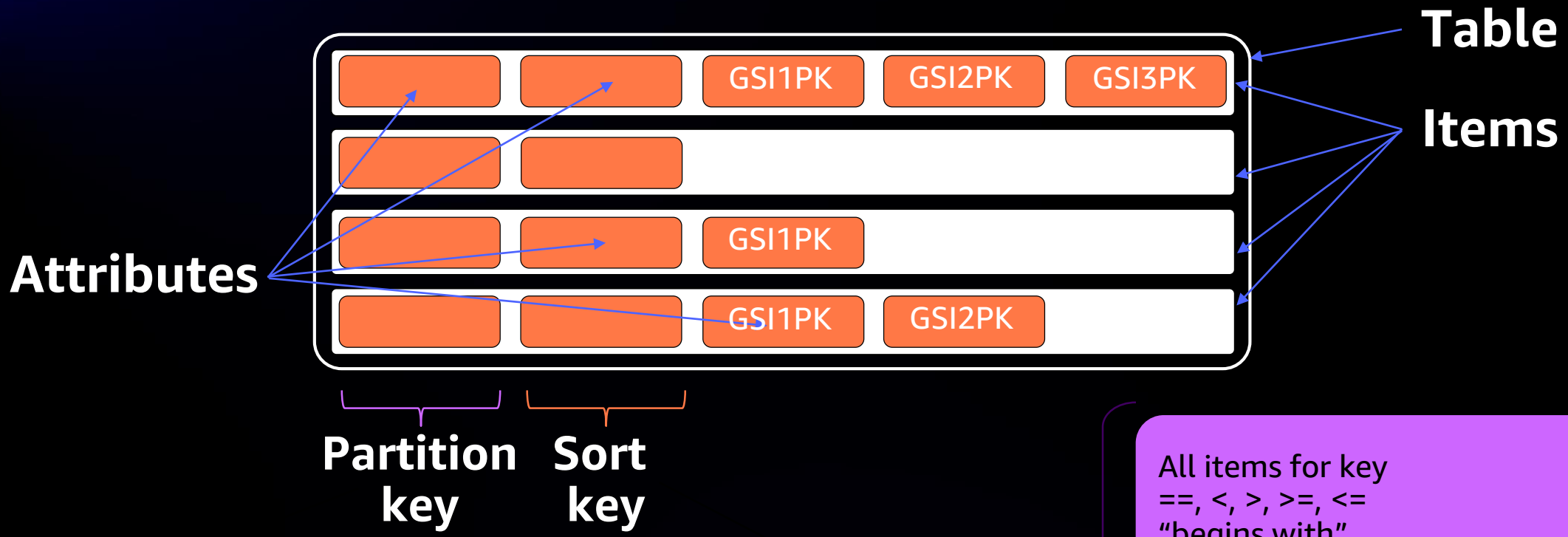
"Data are just summaries of thousands of stories – tell a few of those stories to help make the data meaningful."

Chip & Dan Heath

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Table



Mandatory
Key-value access pattern
Determines data distribution

Optional
Model 1:N relationships
Enables rich query capabilities

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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	123 5th Street, New York, NY

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

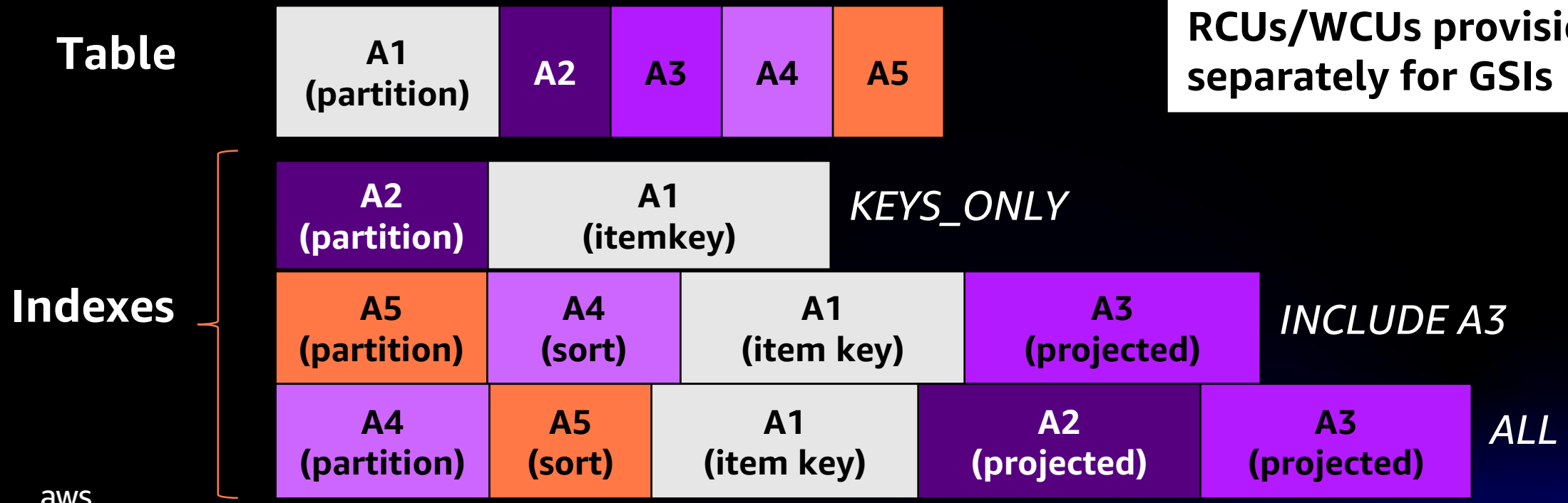
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Secondary indexes

Online indexing

- Support secondary access patterns
- Index across all partition keys
- Use composite sort keys for compound indexes

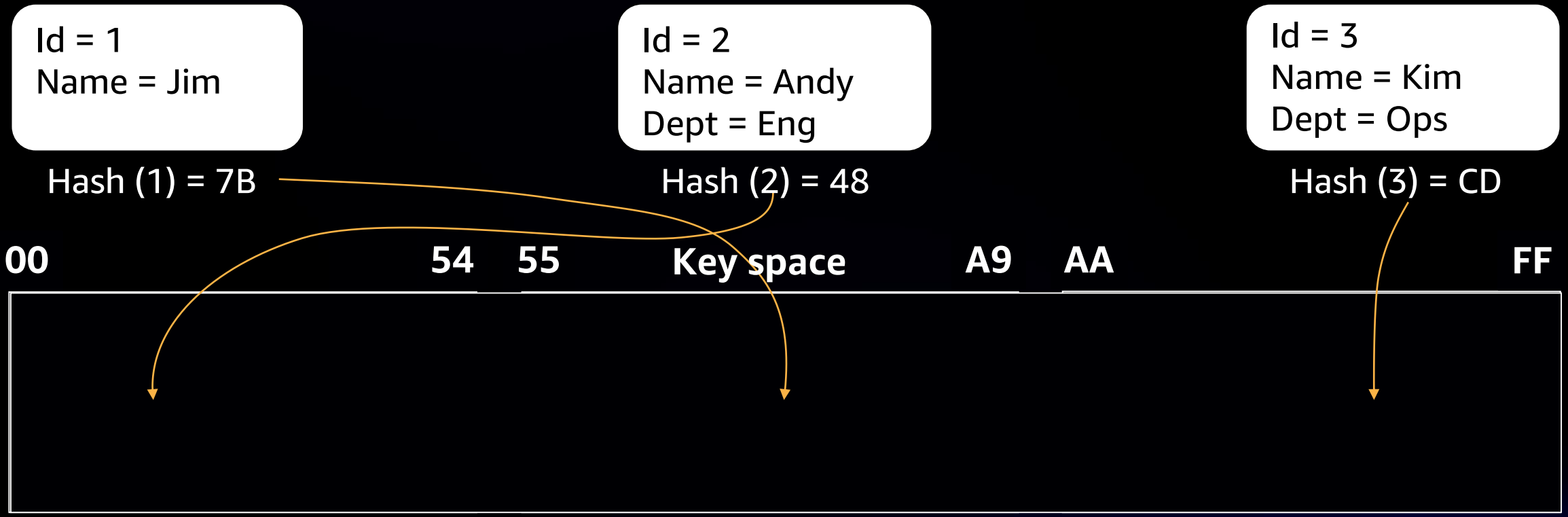
RCUs/WCUs provisioned separately for GSIs



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Partition/shard keys in NoSQL

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

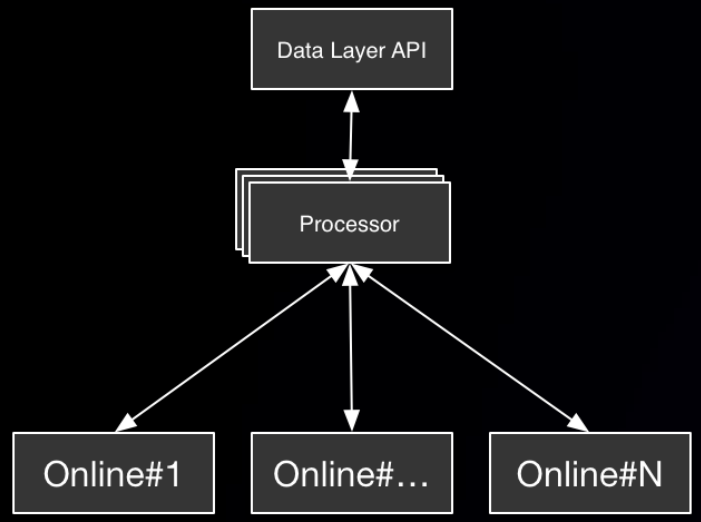


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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	123 5th Street, New York, NY

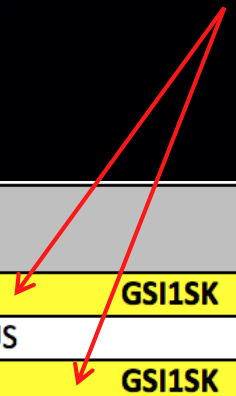


- Abstract partitioning from clients behind an API
- Write across many partitions
- Use parallel processes to increase read throughput

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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	123 5th Street, New York, NY

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

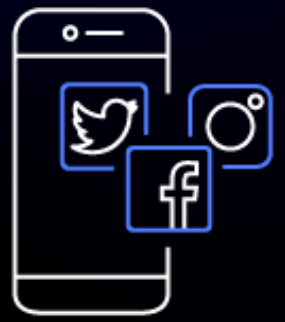
NoSQL data modeling

“If we have data, let’s look at data. If all we have are opinions, let’s go with mine.”

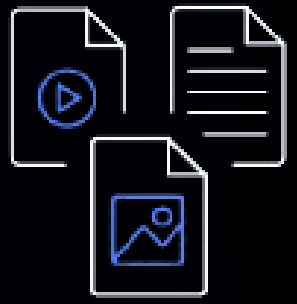
Jim Barksdale

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

It’s all about relationships



Social network



Document management



Process control



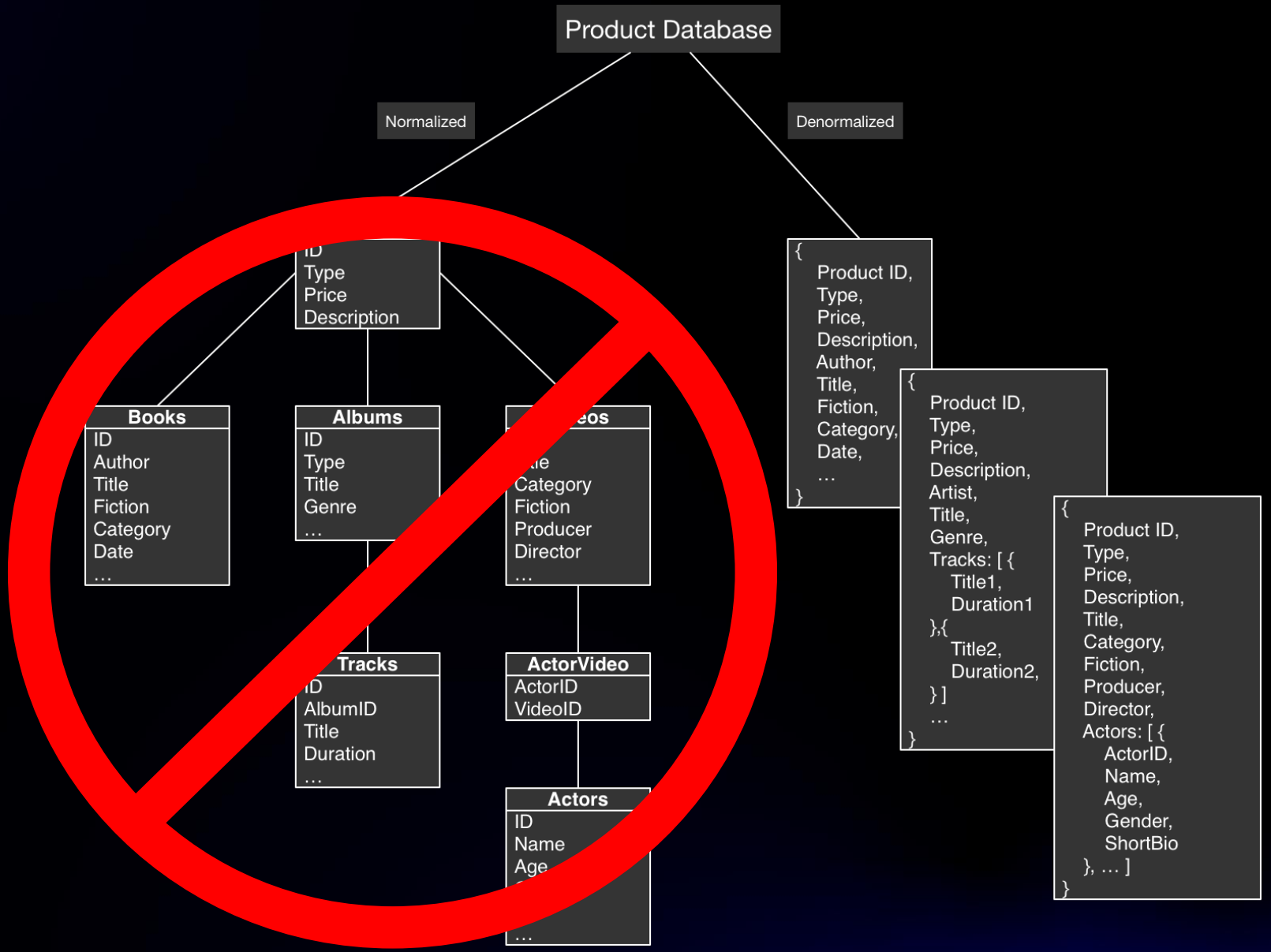
IT monitoring



Data trees

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

SQL vs. NoSQL design pattern



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ad hoc “joins” in SQL

```
SELECT * FROM PRODUCTS
INNER JOIN BOOKS ON
productId = productId
WHERE name = “Book Title”
```

```
SELECT * FROM PRODUCTS
INNER JOIN ALBUMS ON
productId = productId
INNER JOIN TRACKS ON
albumId = albumId
WHERE name = “Album Title”
```

```
SELECT * FROM PRODUCTS
INNER JOIN VIDEOS ON
productId = productId
INNER JOIN ACTORVIDEO ON
videoId = videoId
INNER JOIN ACTORS ON
actorId = actorId
WHERE name = “Movie Title”
```

productId	name	type	price
1	Frankenstein	Book	11.99
2	Dire Straits	Album	17.49
3	Big	Video	14.99
4	Jane Eyre	Book	10.99
5	The Dark Side of the Moon	Album	17.49
6	Saving Private Ryan	Video	18.99

bookId	productId	author	publisher	ISBN-10
1	1	Mary Shelley	Bantam	553212478
2	4	Charlotte Brontë	Wordsworth	1853260207

albumId	productId	artist	producer	releaseDate
1	2	Dire Straits	Muff Winwood	10/7/78
2	5	Pink Floyd	Pink Floyd	3/1/73

videoId	productId	writer	director	releaseDate
1	3	Ann Spielberg	Penny Marshall	6/5/88
2	6	Robert Rodat	Steven Spielberg	7/21/98

actorVideoId	videoId	actorId	character
1	1	1	Josh
2	2	1	Captain Miller
3	1	2	Susan
4	1	3	MacMillan

actorId	gender	name	birthDate
1	M	Tom Hanks	7/9/56
2	F	Elizabeth Perki	11/18/60
3	M	Robert Loggia	1/3/30

trackId	albumId	song	duration
1	1	Down to the Waterline	3:55
2	1	Water of Love	5:23
3	1	Setting Me Up	3:18
4	1	Six Blade Knife	4:10
5	1	Southbound Again	2:58
6	1	Sultans of Swing	5:47
7	1	In the Gallery	6:16
8	1	Wild West End	4:42
9	1	Lions	5:05
10	2	Speak to Me	1:13
11	2	Breathe	2:43
12	2	On the Run	3:36
13	2	Time	4:36
14	2	The Great Gig in the Sky	19:27
15	2	Money	6:23
16	2	Us and Them	7:49
17	2	Any Colour You Like	3:26
18	2	Brain Damage	3:49
19	2	Eclipse	2:03

Time Complexity

O(log(N)+log(M)+log(M)+log(M))

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modeled “joins” in NoSQL

SELECT * WHERE PK=“Book Title”

SELECT * WHERE PK=“Album Title”

SELECT * WHERE PK=“Movie Title”

Time Complexity

O(log(N))

Primary Key		Attributes			
PK	SK				
Frankenstein	Mary Shelley	Type	Price	Publisher	ISBN-10
		book	11.99	Bantam	553212478
Dire Straits	Dire Straits	Type	Price	Producer	ReleaseDate
		album	17.49	Muff Winwood	10/7/78
	Down to the Waterline	Duration	TrackNo		
		3:55	1		
	Water of Love	Duration	TrackNo		
		5:23	2		
	Setting Me Up	Duration	TrackNo		
		3:18	3		
	Six Blade Knife	Duration	TrackNo		
		4:10	4		
	Southbound Again	Duration	TrackNo		
		2:58	5		
	Sultans of Swing	Duration	TrackNo		
		5:47	6		
	In the Gallery	Duration	TrackNo		
		6:16	7		
	Wild West End	Duration	TrackNo		
		4:42	8		
	Lions	Duration	TrackNo		
		5:05	9		
Big	Penny Marshall	Type	Price	Writer	ReleaseDate
		video	14.99	Ann Spielberg	6/5/88
	Tom Hanks	Character	Gender	BirthDate	
		Josh	Male	7/9/56	
	Elizabeth Perkins	Character	Gender	BirthDate	
		Susan	Female	11/18/60	
	Robert Loggia	Character	Gender	BirthDate	
		MacMillan	Male	1/3/30	
Tom Hanks	Tom Hanks	Gender	BirthDate	Bio	
		Male	7/9/56	{…}	

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modeled “joins” in NoSQL

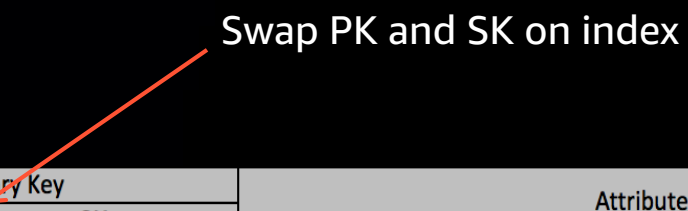
SELECT * WHERE SK=“Author Name”

SELECT * WHERE SK=“Song Title”

SELECT * WHERE SK=“Actor Name”

SELECT * WHERE SK=“Director Name”

SELECT * WHERE SK=“Musician”



Primary Key		Attributes			
PK	SK				
Mary Shelley	Frankenstein	Type	Price	Publisher	ISBN-10
		book	11.99	Bantam	553212478
Sultans of Swing	Dire Straits	Duration	TrackNo		
		5:47	6		
	Sultans of Swing: The Very Best of Dire Straits	Duration	TrackNo		
		5:50	1		
Tom Hanks	Big	Type	Gender	BirthDate	
		Josh	Male	7/9/56	
	Saving Private Ryan	Character	Gender	BirthDate	
		Captain Miller	Male	7/9/56	
	Tom Hanks	Gender	BirthDate	Bio	
		Male	7/9/56	{…}	
Penny Marshall	Big	Type	Price	Writer	ReleaseDate
		video	14.99	Ann Spielberg	6/5/88
Dire Straits	Dire Straits	Type	Price	Producer	ReleaseDate
		album	17.49	Muff Winwood	10/7/78
	Sultans of Swing: The Very Best of Dire Straits	Type	Price	Producer	ReleaseDate
		album	25.99	Various	10/19/98

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

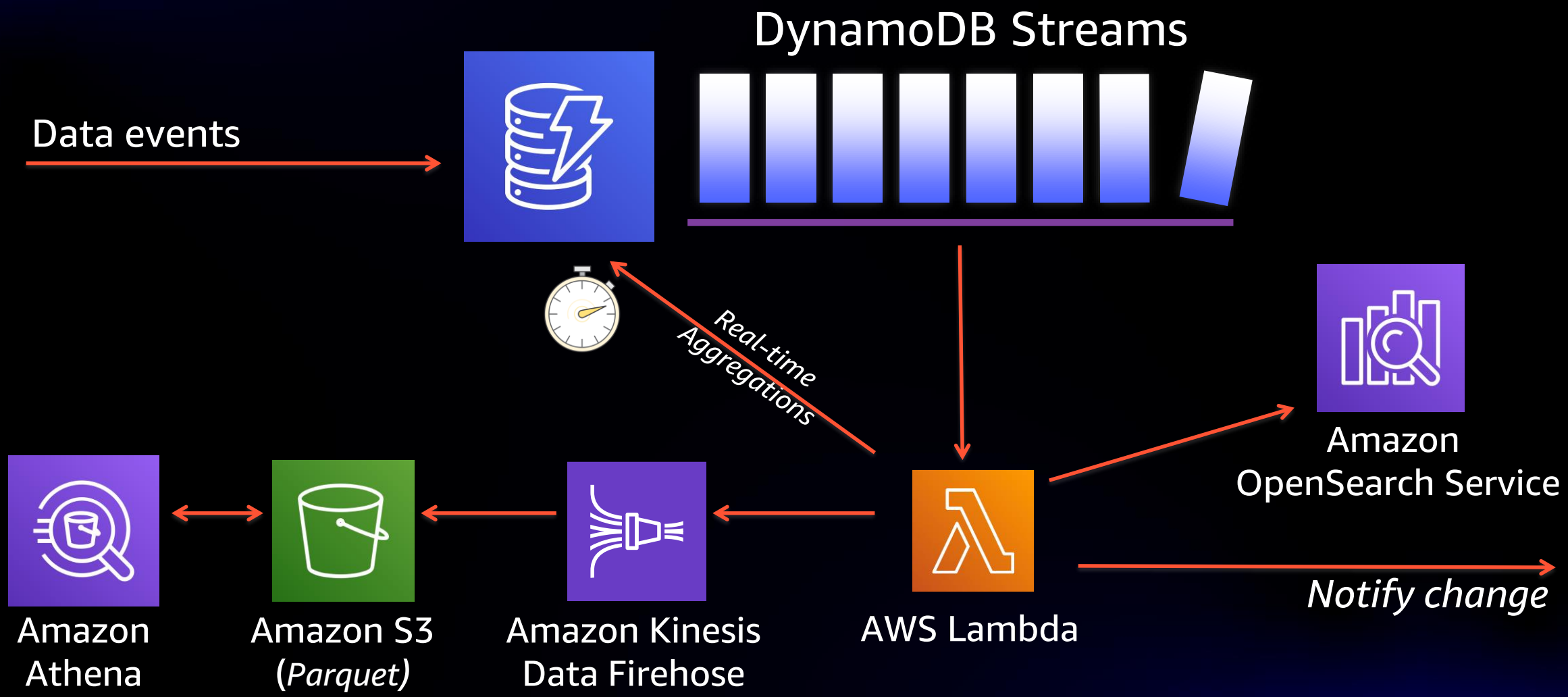
Complex queries

“Computers are useless. They can only give you answers.”

Pablo Picasso

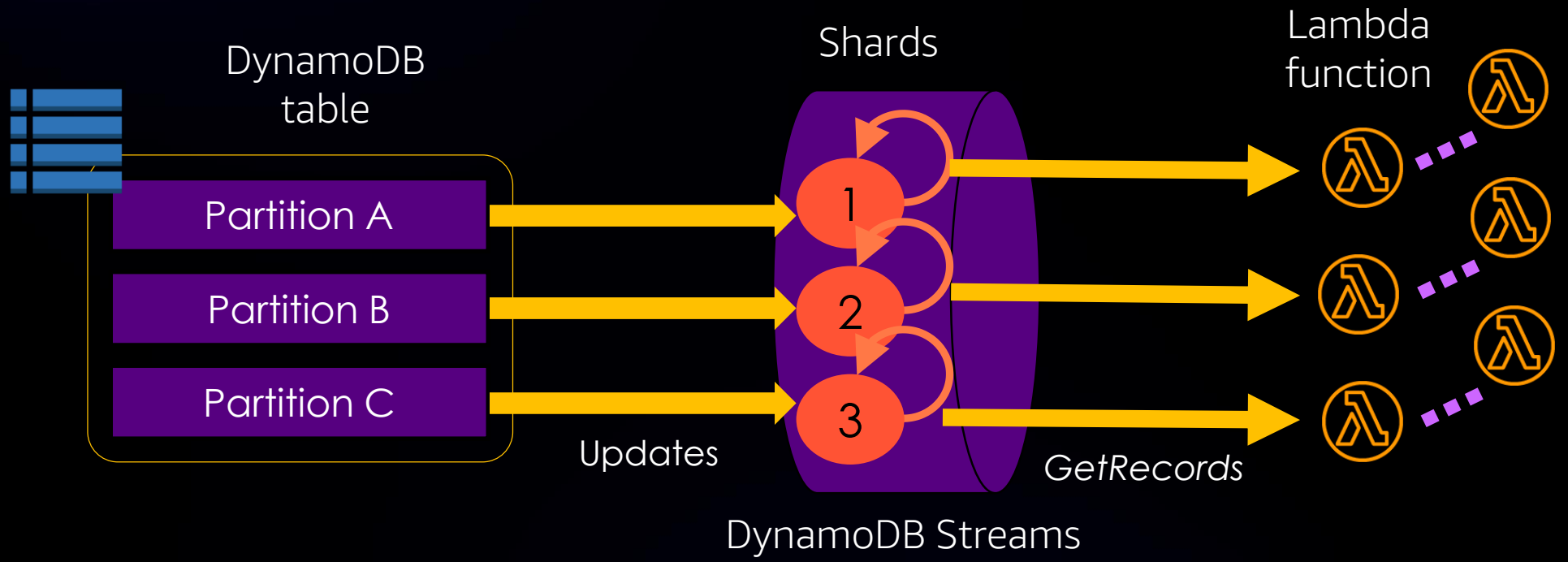
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless & event-driven architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

DynamoDB Streams and Lambda

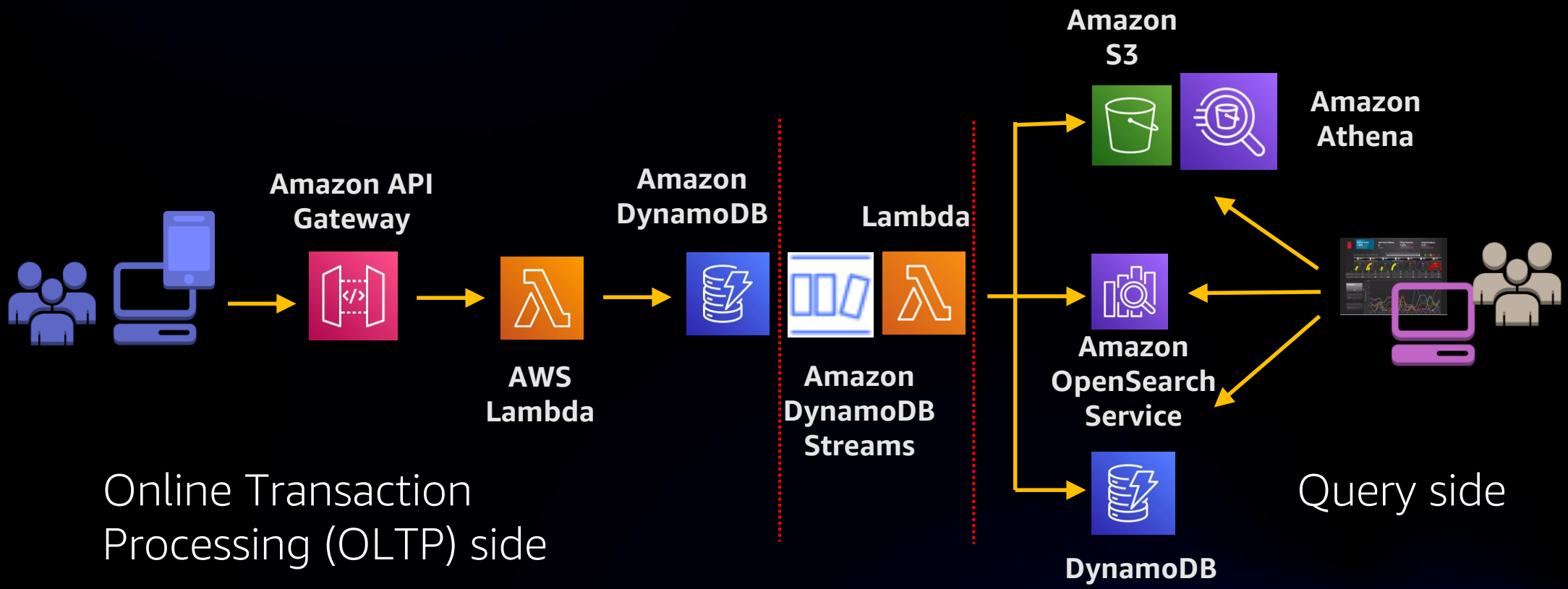


DynamoDB Streams + AWS Lambda = reliable “at least once” event delivery

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example

A serverless app



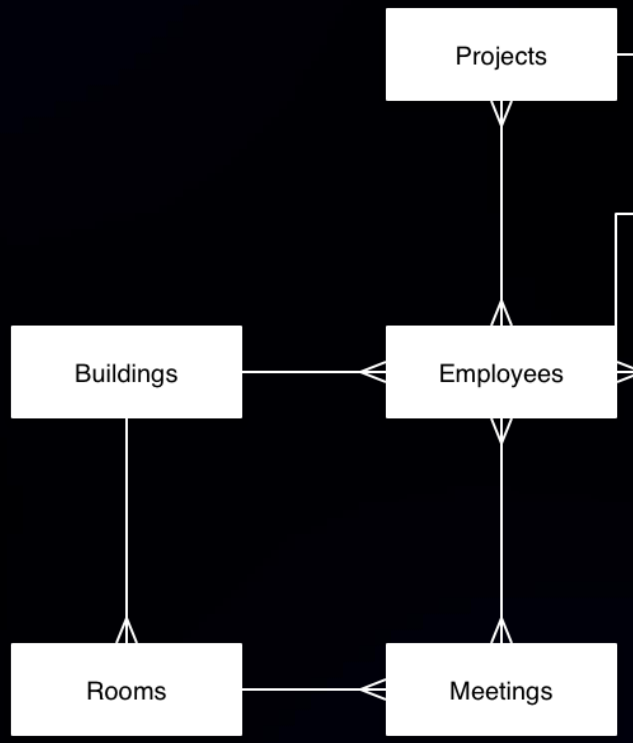
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modeling relational data

“Dude, where’s my lookup table?”

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Modeling complex relationships



Access Patterns		
	Get meetings	
1		by date and email
2		by date and employeeId
3		by date and building/floor/room
	Load employee dashboard by email	
4		Get employee data
5		Get meetings
6		Get tickets
7		Get reservations
8		Get time cards
	Get employee info	
9		by employeeID
10		by email
	Get Ticket history	
11		by Ticket ID
12		by employee email
13		by assignee email
	Get employees	
14		by city, building, floor, aisle, desk
15		by manager
	Get assigned tickets	
16		by email
	Get Tickets	
17		by last touched > 24 hours
	Get project(s)	
18		by status, start and target date
19		by name
	Get project history	
20		by date range
21		by role
	Get Rooms	
22		by buildingID
23		by availability and time range

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The table

	Access Patterns		Key Condition	Filter Condition
2	Get meetings	by date and employeeId	PK = employeeId, SK between(date1, date2)	duration > 0
3	Get meetings	by date and building/floor/room	PK = buildingId, SK between(date1, date2)	SK contains(building/floor/room)
9	Get employee info	by employeeID	PK = employeeId, SK startsWith("E")	
11	Get Ticket History	by Ticket ID	PK = ticketId	
19	Get project	by name	PK = projectName, SK = projectName	
20	Get project history	by date range	PK = projectName, SK between(date1, date2)	
21	Get project history	by role	PK = projectName	role = roleName
22	Get rooms	by buildingId	PK = buildingID	
23	Get rooms	by Availability and Time Range	PK = buildingId, SK between(date1, date2)	

Primary key		Attributes						
Partition key: pk	Sort key: sk							
SEA58	2019-08-20T10:00:00Z|07.106	GSI1pk	GSIsk	Duration	Attendees	Subject		
		john@example.com	2019-08-20T10:00:00Z|07.106	30	[...]	Discuss ProjectX		
	2019-08-20T10:15:00Z|07.106	Attendees	Subject	Organizer				
		[...]	Discuss ProjectX	john@example.com				
	Rooms	RoomSpecs						
		[...]						
EMPLOYEE_1	2019-08-20T10:00:00Z|07.106	GSI1pk	GSIsk	Duration	Attendees	Subject		
		richard@example.com	2019-08-20T10:00:00Z|07.106	30	[...]	Discuss ProjectX		
	E#999	GSI1pk	GSIsk	GSI3pk	GSI3sk	Name	Title	GSI2pk
		richard@example.com	E#999	SEA	58.07.105.B2	Richard Roe	IT Support	john@example.com
EMPLOYEE_2	E#777	GSI1pk	GSIsk	GSI3pk	GSI3sk	Name	Title	GSI2pk
		john@example.com	E#777	SEA	58.09.203.A1	John Doe	CEO	john@example.com
ProjectX	2019-09-06|john@example.com	GSI1pk	GSIsk	Hours	Role			
		john@example.com	2019-09-06	12	TPM			
	2019-09-06|richard@example.com	GSI1pk	GSIsk	Hours	Role			
		richard@example.com	2019-09-06	24	SDE2			
	ProjectX	GSI1pk	GSIsk	Description	TargetDelivery			
		Active	2019-08-30	Some project	2020-01-30			
Ticket_1	2019-08-15T12:35:00Z	GSI1pk	GSIsk	Subject	GSI3pk	GSI3sk	GSI2pk	Message
		john@example.com	2019-08-15T12:35:00Z	Badge replacement	7	2019-08-16T12:35:00Z	richard@example.com	Dog ate my badge.
	2019-08-15T12:35:05Z	GSI1pk	GSIsk	GSI2pk	Message			
		john@example.com	2019-08-15T12:35:05Z	richard@example.com	Request received.			

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The index schema (GSI1)

Access Patterns			Key Condition	Filter Condition
1	Get Meetings	by date and email	GSI1PK = email, GSISK between(date1, date2)	duration > 0
4	Load dashboard by email	Get employee data	GSI1PK = email, GSISK > 30 days ago	None
5		Get meetings		
6		Get tickets		
7		Get reservations		
8		Get time cards		
10	Get Employee info	by email	GSI1PK = email, GSISK startsWith("E")	
12	Get Ticket History	by employee email	GSI1PK = email	PK = ticketId
18	Get Projects	by status, start and target date	GSI2PK = status, GSISK > startDate	targetDelivery < targetDate

Primary key		Attributes						
Partition key: GSI1pk	Sort key: GSIsk							
staylor@abc.com	2019-08-15T12:35:00Z	pk	sk	Subject	GSI3pk	GSI3sk	GSI2pk	Message
		Ticket_1	2019-08-15T12:35:00Z	Badge replacement	7	2019-08-16T12:35:00Z	bhana@abc.com	Dog ate my badge.
	2019-08-15T12:35:05Z	pk	sk	GSI2pk	Message			
		Ticket_1	2019-08-15T12:35:05Z	bhana@abc.com	Request received.			
	2019-08-20T10:00:00Z|07.106	pk	sk	Duration	Attendees	Subject		
		SEA58	2019-08-20T10:00:00Z|07.106	30	[...]	Discuss ProjectX		
	2019-09-06	pk	sk	Hours	Role			
		ProjectX	2019-09-06|staylor@abc.com	12	TPM			
	E#777	pk	sk	GSI3pk	GSI3sk	Name	Title	GSI2pk
		EMPLOYEE_2	E#777	SEA	58.09.203.A1	Steven Taylor	CEO	staylor@abc.com
bhana@abc.com	2019-08-20T10:00:00Z|07.106	pk	sk	Duration	Attendees	Subject		
		EMPLOYEE_1	2019-08-20T10:00:00Z|07.106	30	[...]	Discuss ProjectX		
	2019-09-06	pk	sk	Hours	Role			
		ProjectX	2019-09-06|bhana@abc.com	24	SDE2			
	E#999	pk	sk	GSI3pk	GSI3sk	Name	Title	GSI2pk
		EMPLOYEE_1	E#999	SEA	58.07.105.B2	Benny Hana	IT Support	staylor@abc.com
Active	2019-08-30	pk	sk	Description	TargetDelivery			
		ProjectX	ProjectX	Some project	2020-01-30			

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The index schema (GSI2)

Access Patterns			Key Condition	Filter Condition
13	Get Ticket History	by assignee email	GSI2PK = email	PK = ticketId
15	Get employees	by manager	GSI2PK = email, SK > 3	

Primary key		Attributes						
Partition key: GSI2pk	Sort key: GSIsk							
john@example.com	E#777	pk	sk	GSI1pk	GSI3pk	GSI3sk	Name	Title
		EMPLOYEE_2	E#777	john@example.com	SEA	58.09.203.A1	John Doe	CEO
	E#999	pk	sk	GSI1pk	GSI3pk	GSI3sk	Name	Title
		EMPLOYEE_1	E#999	richard@example.com	SEA	58.07.105.B2	Richard Roe	IT Support
richard@example.com	2019-08-15T12:35:00Z	pk	sk	GSI1pk	Subject	GSI3pk	GSI3sk	Message
		Ticket_1	2019-08-15T12:35:00Z	john@example.com	Badge replacement	7	2019-08-16T12:35:00Z	Dog ate my badge.
	2019-08-15T12:35:05Z	pk	sk	GSI1pk	Message			
		Ticket_1	2019-08-15T12:35:05Z	john@example.com	Request received.			

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The index schema (GSI3)

Access Patterns			Key Condition	Filter Condition
14	Get employees	by city, building, floor, aisle, desk	GSI3PK = city, GSI3SK startsWith(building/floor/aisle/desk)	
17	Get tickets	by last touched > 24 hours	GSI3PK = (0-N), GSI3SK < yesterday	

Primary key		Attributes						
Partition key: GSI3pk	Sort key: GSI3sk							
SEA	58.07.105.B2	pk	sk	GSI1pk	GSIsk	Name	Title	GSI2pk
		EMPLOYEE_1	E#999	richard@example.com	E#999	Richard Roe	IT Support	john@example.com
	58.09.203.A1	pk	sk	GSI1pk	GSIsk	Name	Title	GSI2pk
		EMPLOYEE_2	E#777	john@example.com	E#777	John Doe	CEO	john@example.com
7	2019-08-16T12:35:00Z	pk	sk	GSI1pk	GSIsk	Subject	GSI2pk	Message
		Ticket_1	2019-08-15T12:35:00Z	john@example.com	2019-08-15T12:35:00Z	Badge replacement	richard@example.com	Dog ate my badge.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The final result

Access Patterns			Table/Index	Key Condition	Filter Condition
	Get meetings				
1		by date and email	GSI1	GSI1PK = email, GSISK between(date1, date2)	duration > 0
2		by date and employeeId	Table	PK = employeeId, SK between(date1, date2)	duration > 0
3		by date and building/floor/room	Table	PK = buildingId, SK between(date1, date2)	SK contains(building/floor/room)
	Load employee dashboard by email				
4		Get employee data	GSI1	GSI1PK = email, GSISK > 30 days ago	None
5		Get meetings			
6		Get tickets			
7		Get reservations			
8		Get time cards			
	Get employee info				
9		by employeeID	Table	PK = employeeId, SK startsWith("E")	
10		by email	GSI1	GSI1PK = email, GSISK startsWith("E")	
	Get Ticket history				
11		by Ticket ID	Table	PK = ticketId	
12		by employee email	GSI1	GSI1PK = email	PK = ticketId
13		by assignee email	GSI2	GSI2PK = email	PK = ticketId
	Get employees				
14		by city, building, floor, aisle, desk	GSI3	GSI3PK = city, GSI3SK startsWith(building/floor/aisle/desk)	
15		by manager	GSI2	GSI2PK = email, SK > 3	
	Get assigned tickets				
16		by email	GSI2	GSI1PK = email	PK = ticketId
	Get Tickets				
17		by last touched > 24 hours	GSI3	GSI3PK = (0-N), GSI3SK < yesterday	
	Get project(s)				
18		by status, start and target date	GSI1	GSI2PK = status, GSISK > startDate	targetDelivery < targetDate
19		by name	Table	PK = projectName, SK = projectName	
	Get project history				
20		by date range	Table	PK = projectName, SK between(date1, date2)	
21		by role	Table	PK = projectName	role = roleName
	Get Rooms				
22		by buildingId	Table	PK = buildingID	
23		by Availability and Time Range	Table	PK = buildingId, SK between(date1, date2)	

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Conclusions

- NoSQL does not mean nonrelational
- The ERD still matters
- RDBMS is not deprecated by NoSQL
- NoSQL can be a great fit for OLTP
- Use RDBMS for high-velocity OLAP

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Rick Houlihan

Twitter – @houlihan_rick

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.