
AWS
re:Invent

DAT305-PT1

Data modeling with Amazon DynamoDB – Part 1

Alex DeBrie
AWS Data Hero
Engineer, Stedi

AWS re:Invent

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

Overview

- Amazon DynamoDB basics
- SQL vs. NoSQL
- One-to-many relationships
- **DAT305-PT2**
  Data modeling with Amazon DynamoDB – Part 2
  Data modeling strategies



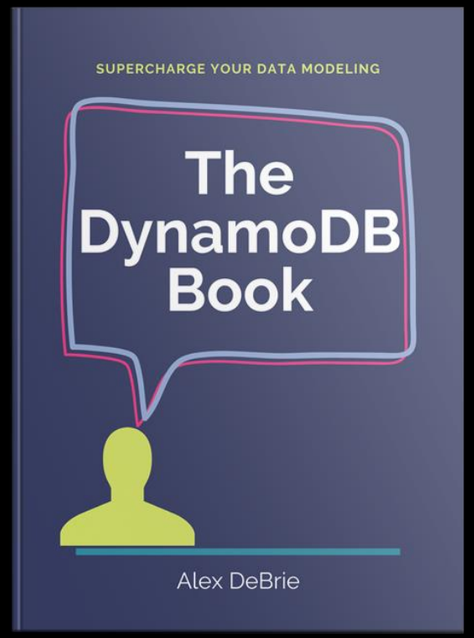
AWS re:Invent

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

Alex DeBrie

- Author of *The DynamoDB Book*
  - Comprehensive guide to data modeling with DynamoDB
- Creator of DynamoDBGuide.com
- AWS Data Hero
- Engineer at Stedi

dynamodbbook.com





DynamoDB basics



AWS re:Invent

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

DynamoDB basics

- Table
- Item
- Primary key
- Attributes

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Table

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Item

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Primary key

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Attributes

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Attributes

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Attributes

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Primary key

- Simple primary key *(partition key)*
- Composite primary key *(partition key + sort key)*

Simple primary key

Primary key	Attributes		
Partition key: Username			
alexdebrie	FirstName	LastName	Interests
	Alex	DeBrie	[DynamoDB, NBA, Nebraska]
thebezos	FirstName	LastName	Interests
	Jeff	Bezos	[Amazon, Business, Monday]
raisingthebarr	FirstName	LastName	Interests
	Jeff	Barr	[Writing, Tech, Purple]
wernerbrothers	FirstName	LastName	Interests
	Werner	Vogels	[Distributed Systems, Music]

Composite primary key

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Composite primary key

Primary key		Attributes		
Partition key: Actor	Sort key: Movie			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Composite primary key

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Composite primary key

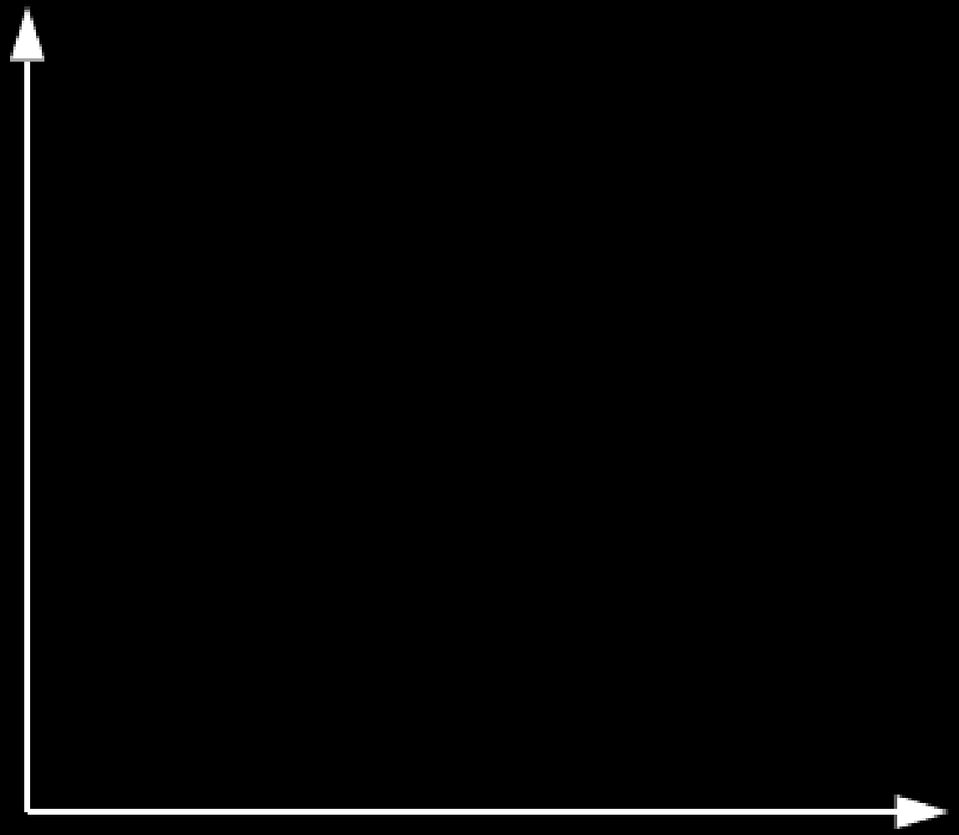
Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

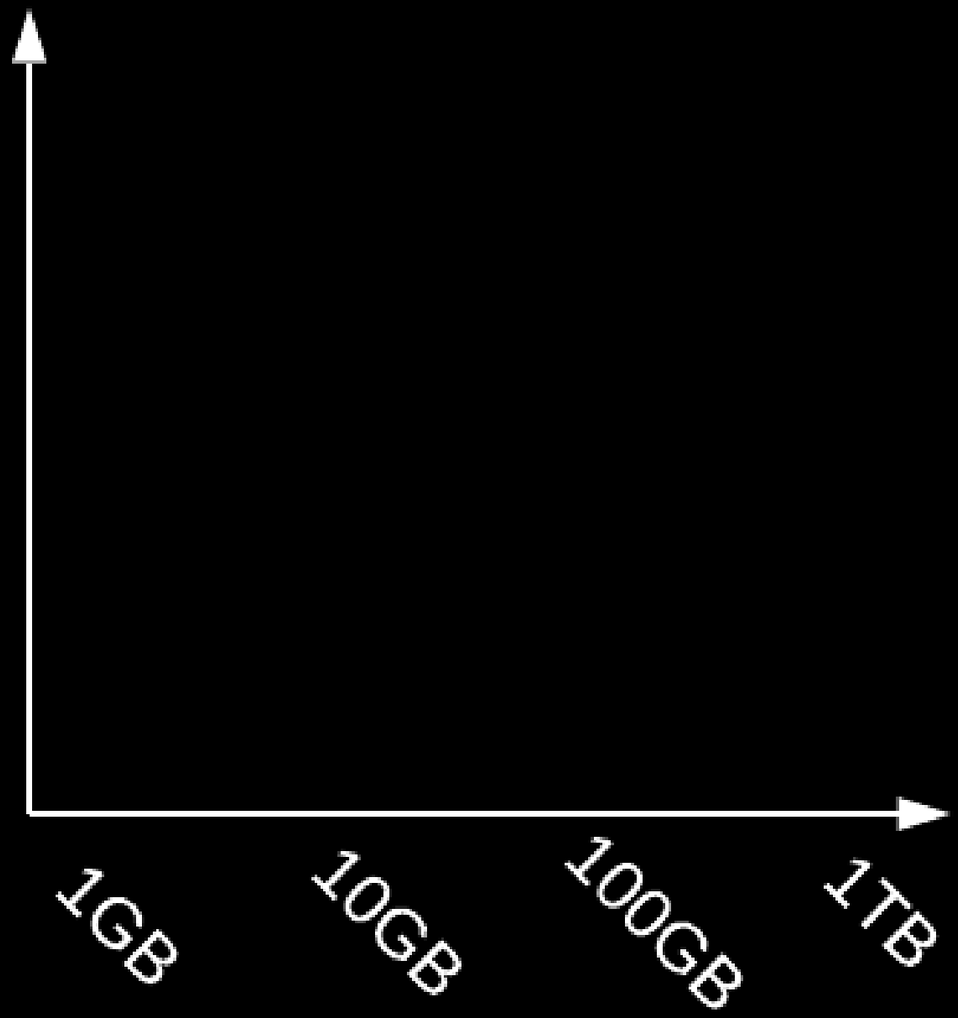
SQL vs. NoSQL



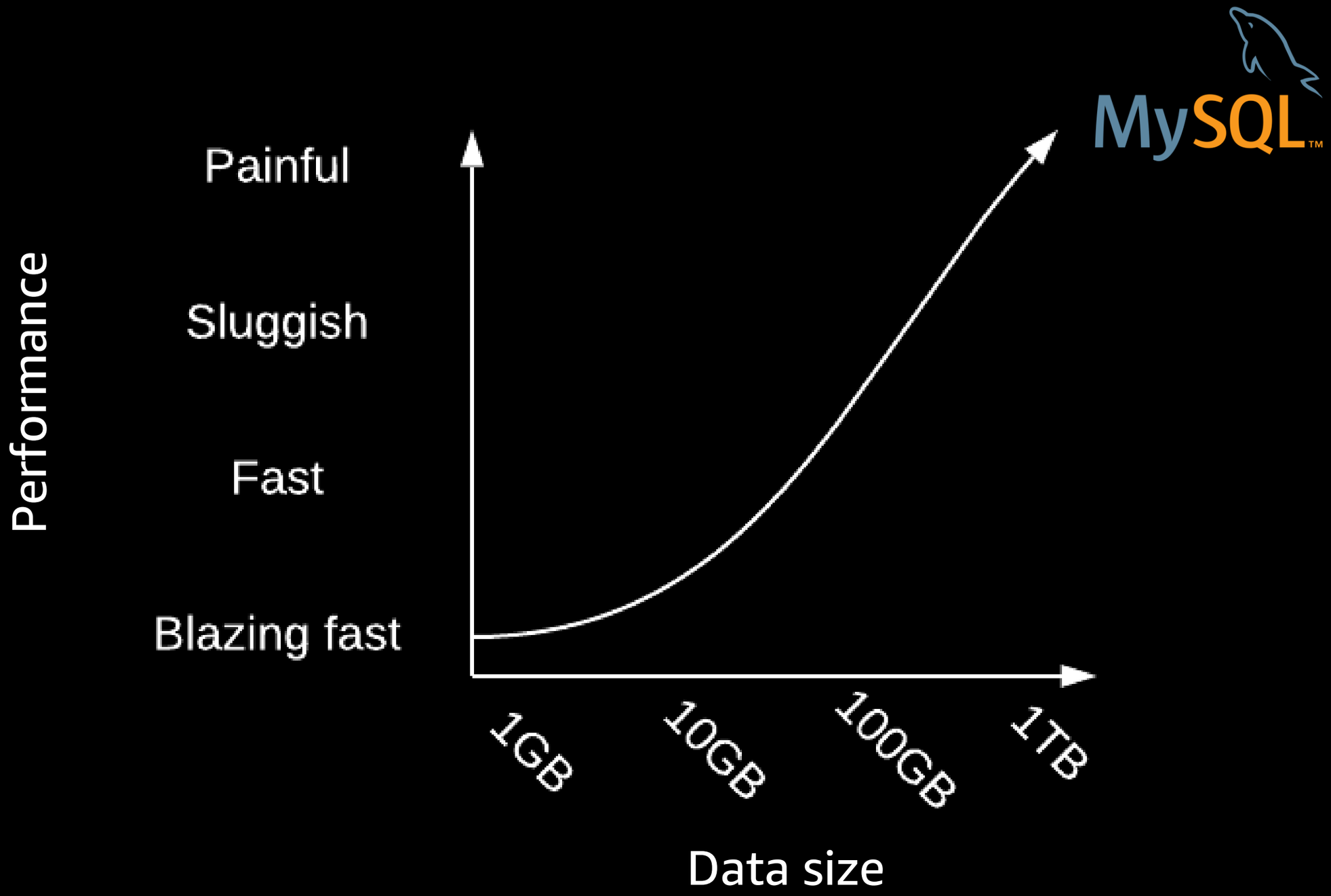
AWS re:Invent

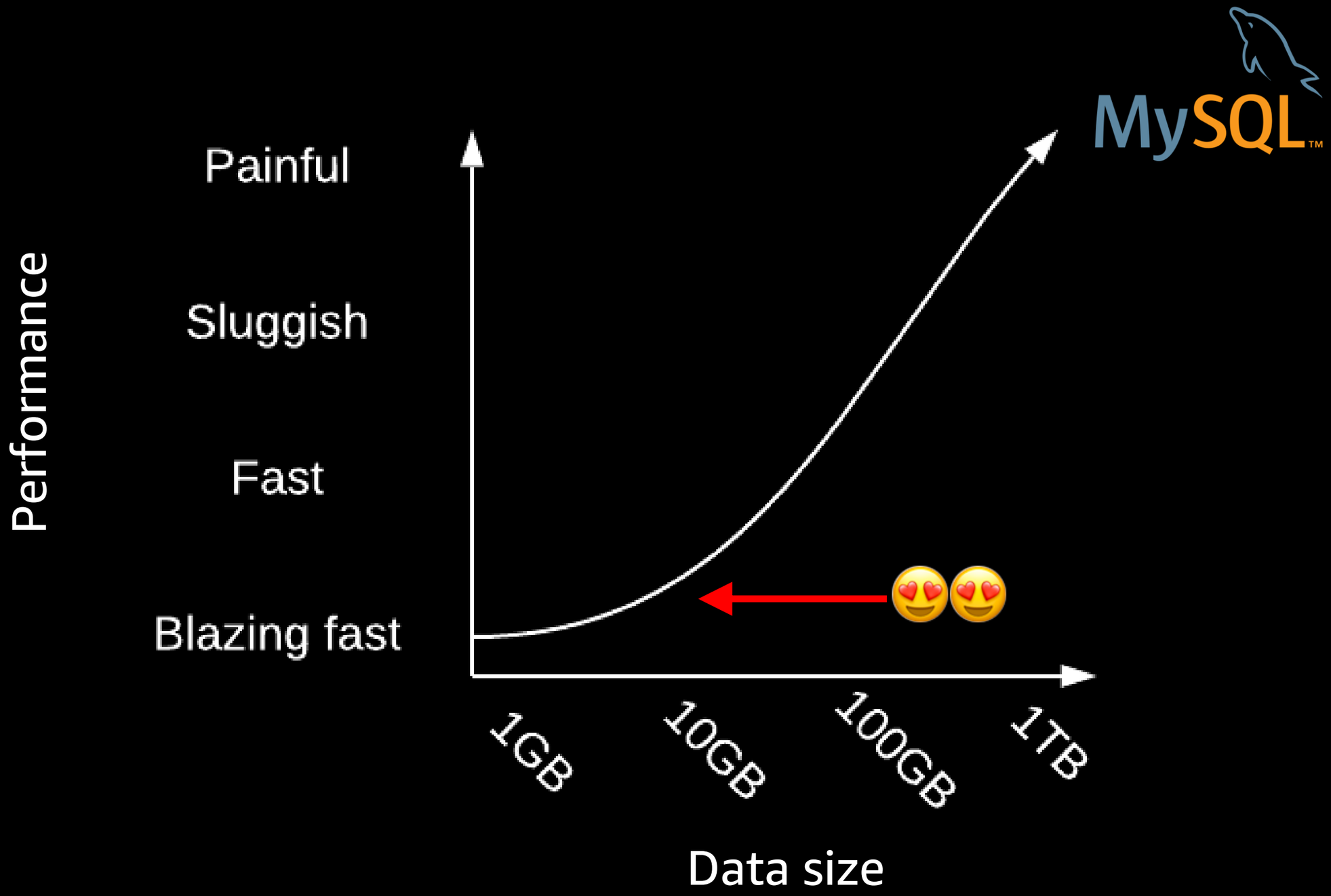
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

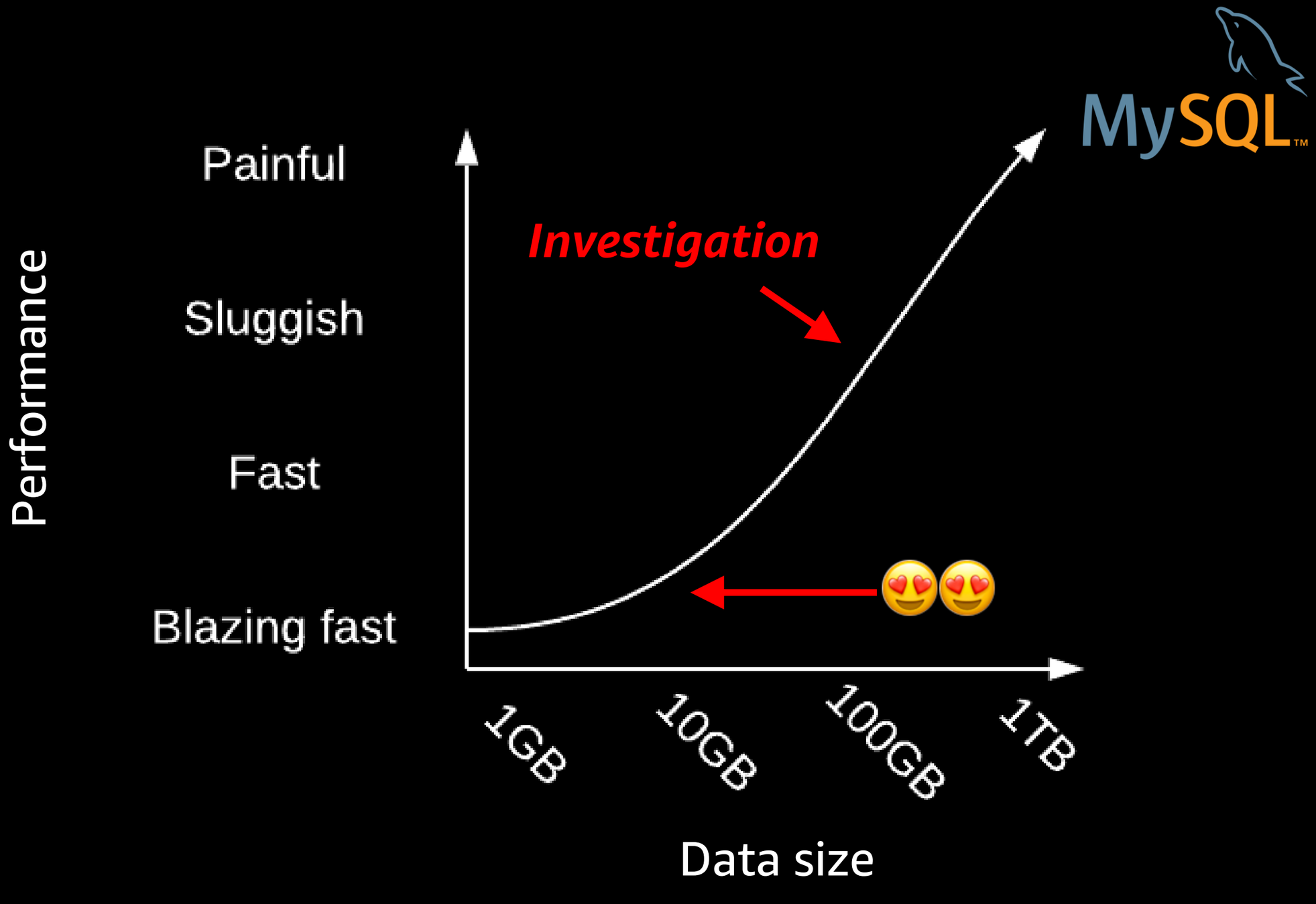


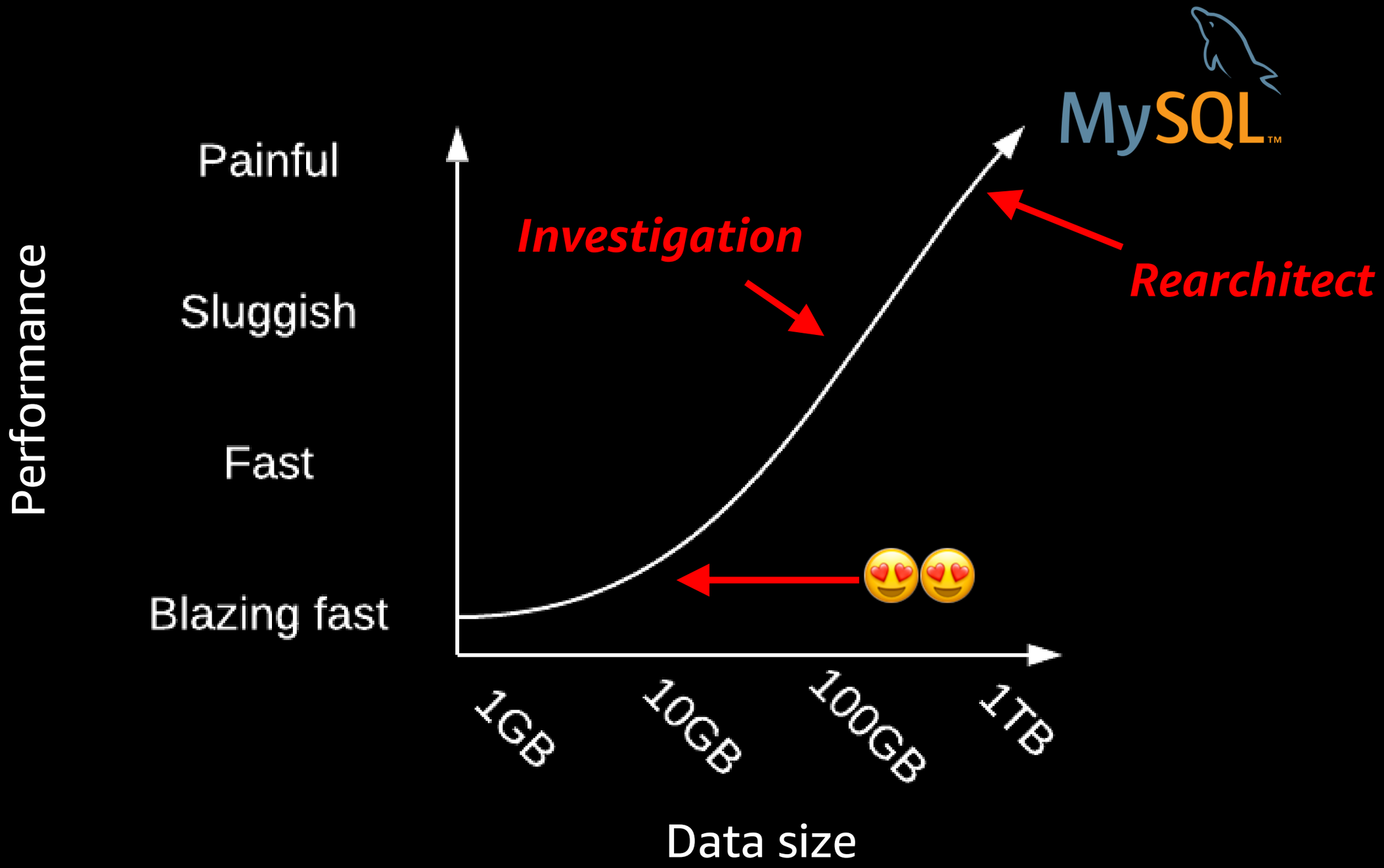

1GB
10GB
100GB
1TB
Data size

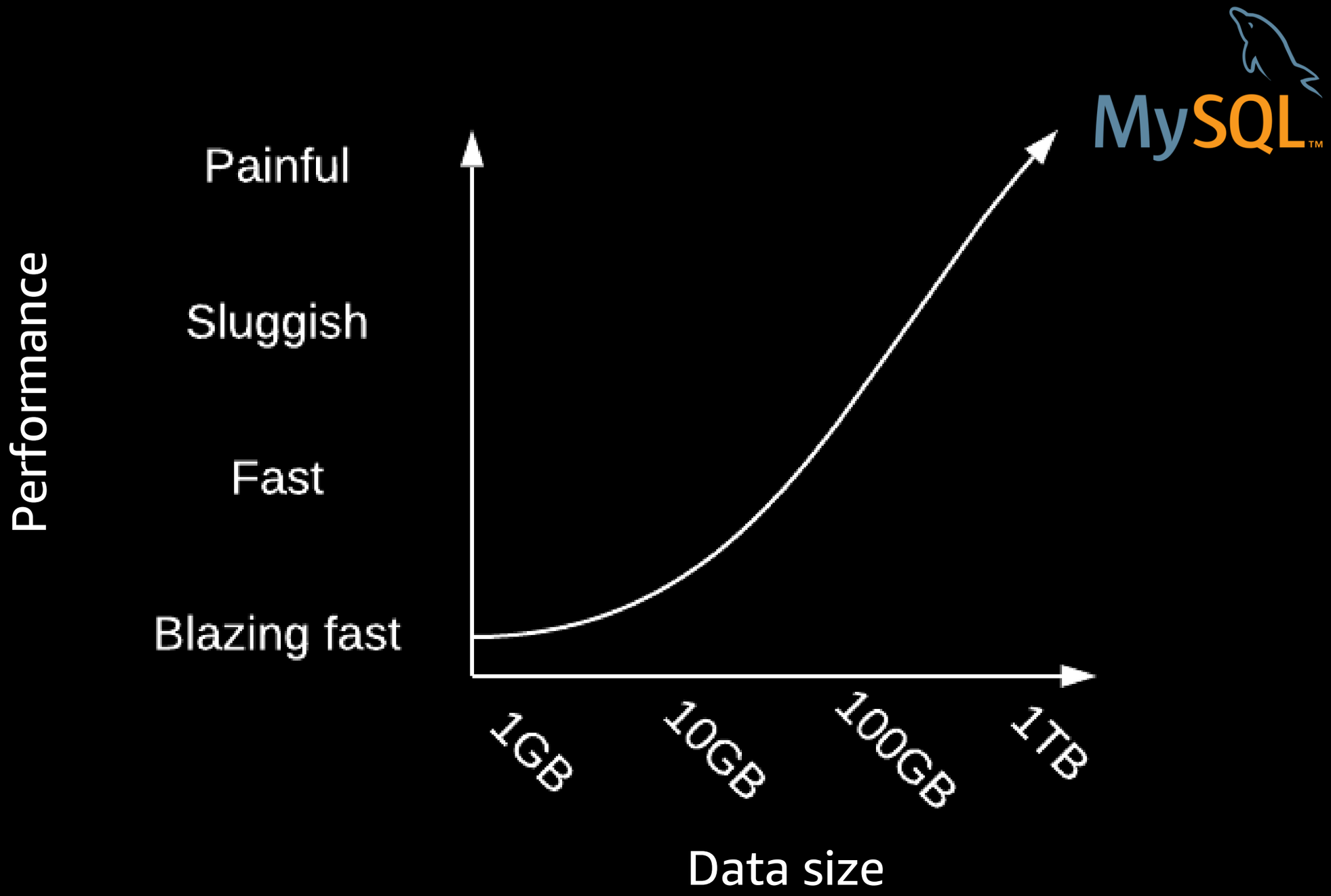

Performance
Painful
Sluggish
Fast
Blazing fast
1GB
10GB
100GB
1TB
Data size

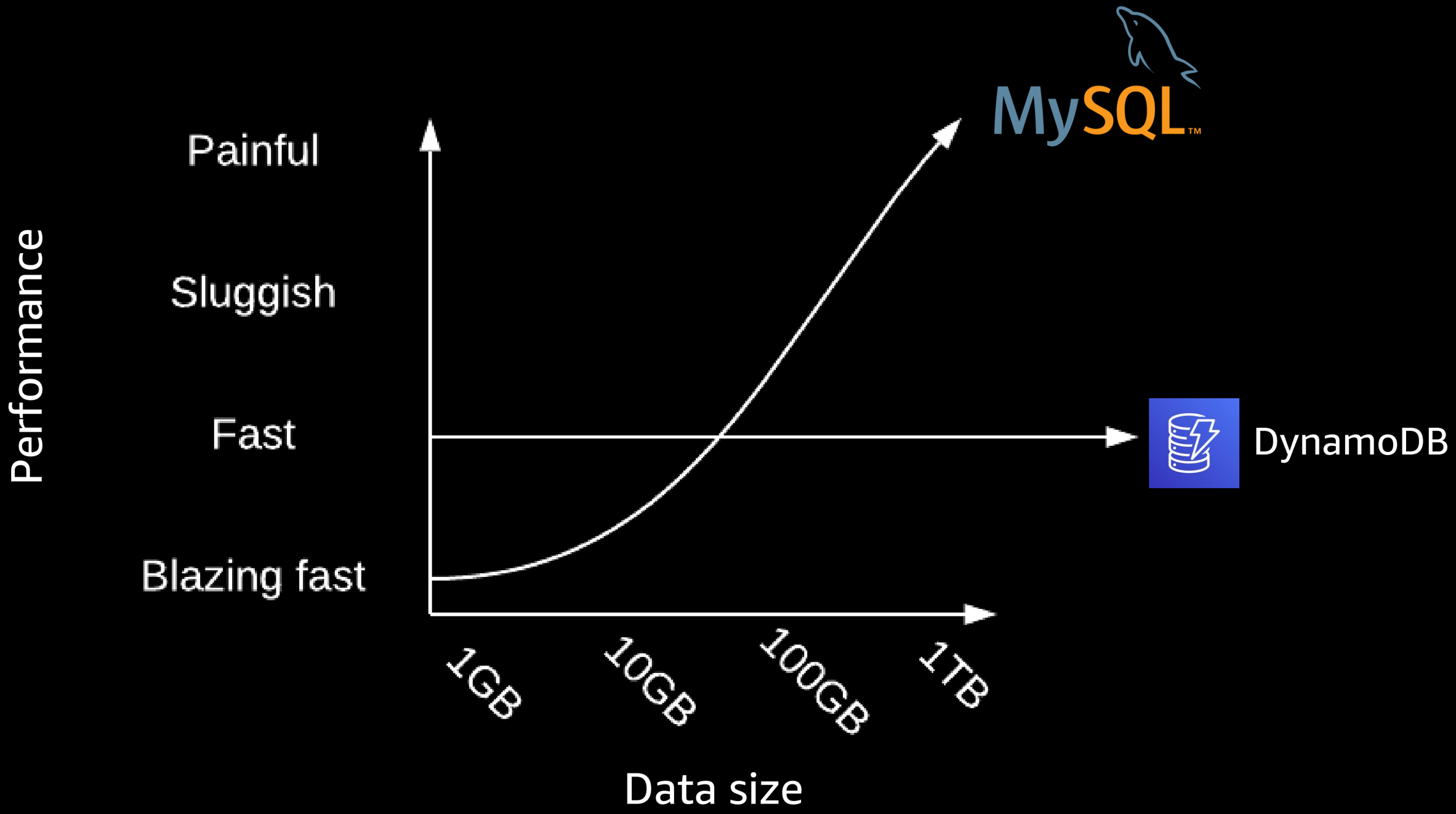

MySQL
Performance
Painful
Sluggish
Fast
Blazing fast
1GB
10GB
100GB
1TB
Data size


MySQL
Painful
Sluggish
Fast
Blazing fast
Performance
1GB
10GB
100GB
1TB
Data size


MySQL
Performance
Painful
Sluggish
Fast
Blazing fast
Investigation
1GB
10GB
100GB
1TB
Data size


MySQL
Painful
Sluggish
Fast
Blazing fast
Performance
Investigation
Rearchitect
1GB
10GB
100GB
1TB
Data size


MySQL
Performance
Painful
Sluggish
Fast
Blazing fast
1GB
10GB
100GB
1TB
Data size


MySQL
Painful
Sluggish
Fast
Blazing fast
Performance
DynamoDB
1GB
10GB
100GB
1TB
Data size

DynamoDB’s guiding principle

DynamoDB's guiding principle

Don't allow operations that won't scale

From SQL to NoSQL

From SQL to NoSQL

- Importance of primary keys

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Query with these

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Query with these

Not with these

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

From SQL to NoSQL

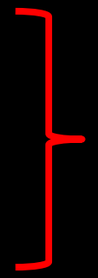
- Importance of primary keys

From SQL to NoSQL

- Importance of primary keys
- No joins

From SQL to NoSQL

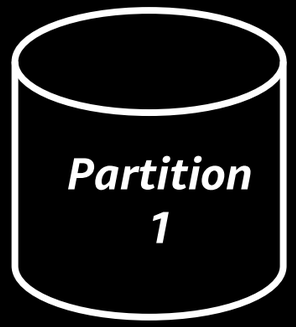
- Importance of primary keys
- No joins

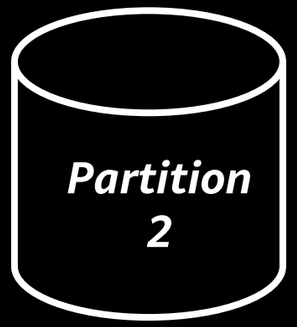


Why?



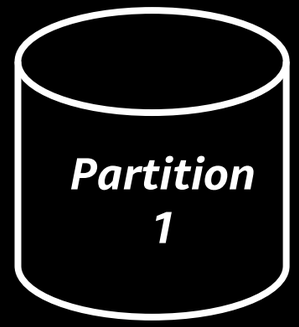


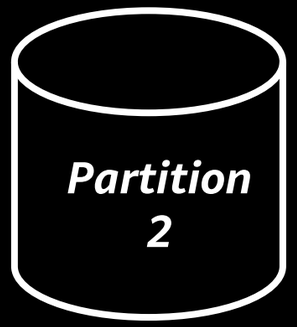

Partition
1


Partition
2

PutItem:
Actor: “Tom Hanks”
Movie: “Big”



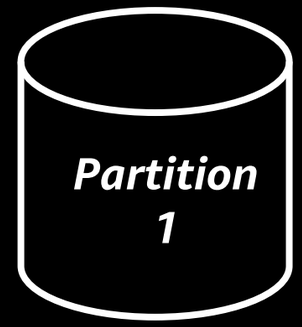


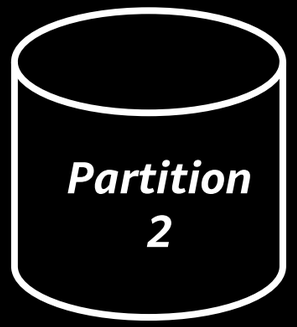


PutItem:
Actor: “Tom Hanks”
Movie: “Big”



fx(Actor):
This item belongs to Partition 1






PutItem:
Actor: “Tom Hanks”
Movie: “Big”
fx(Actor):
This item belongs to
Partition 1
Partition 1
Partition 2

PutItem:
Actor: "Tom Hanks"
Movie: "Big"



fx(Actor):
This item belongs to Partition 1

Partition 1

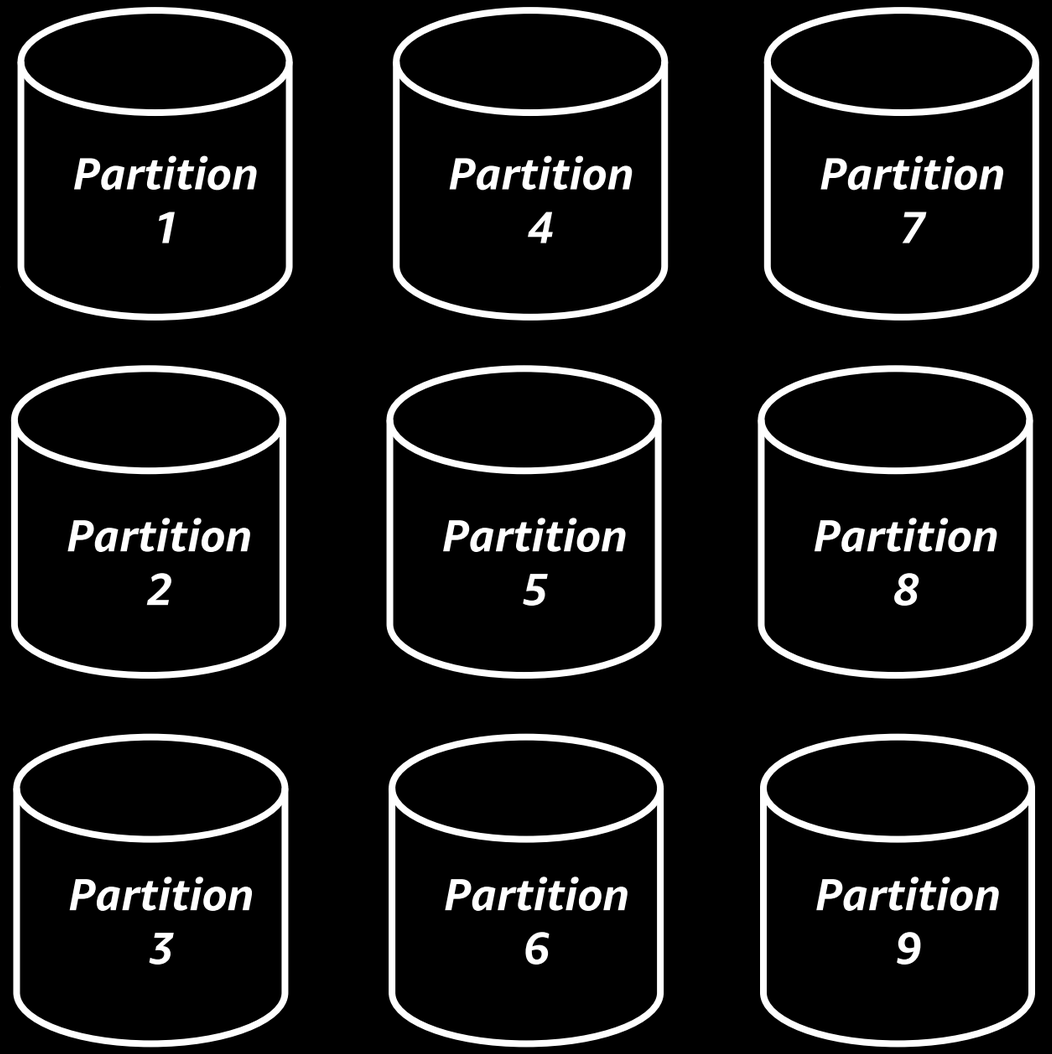
Partition 2

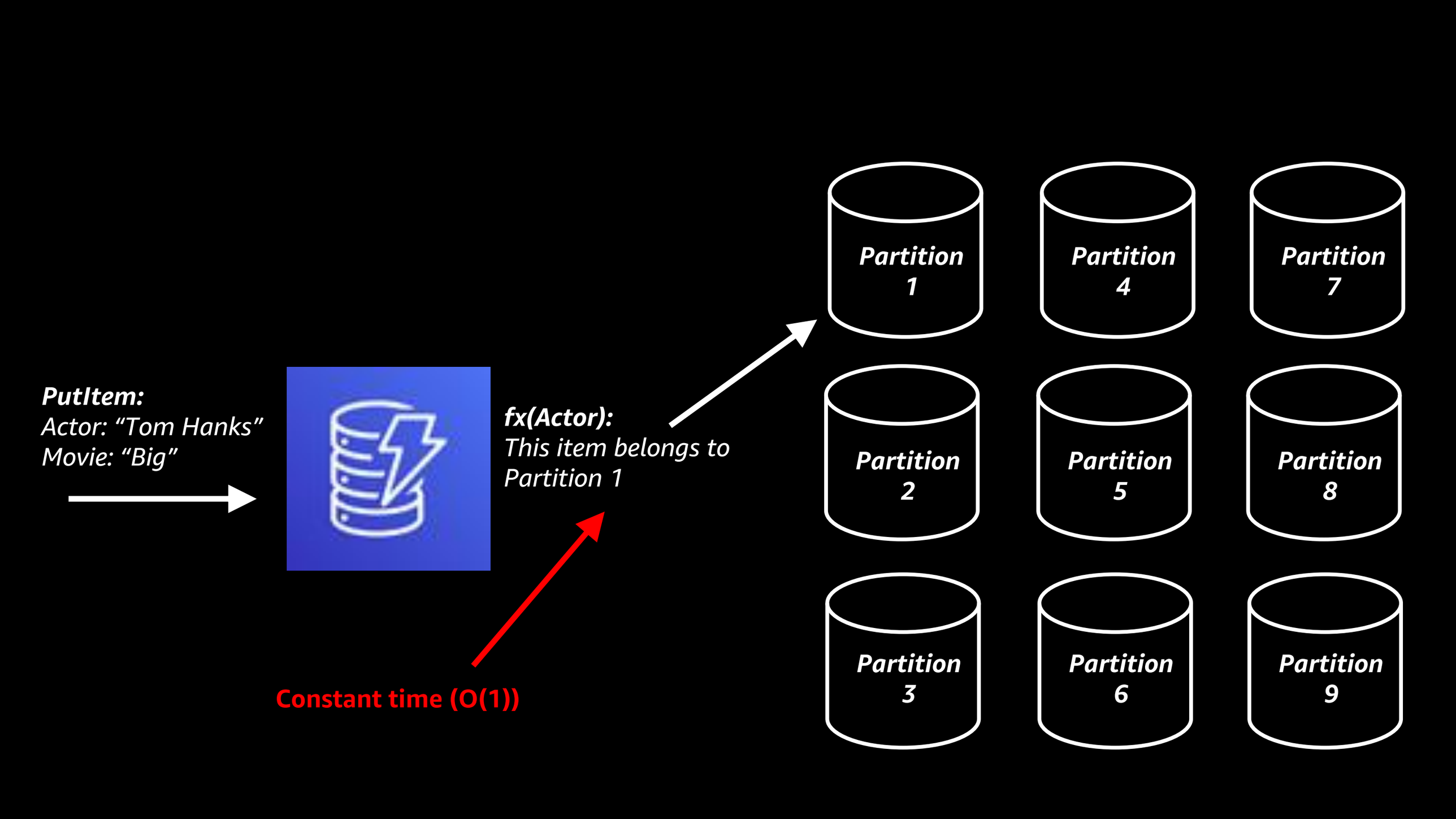
Partition 3

PutItem:
Actor: "Tom Hanks"
Movie: "Big"



fx(Actor):
This item belongs to Partition 1




PutItem:
Actor: “Tom Hanks”
Movie: “Big”
fx(Actor):
This item belongs to
Partition 1
Constant time (O(1))
Partition 1
Partition 4
Partition 7
Partition 2
Partition 5
Partition 8
Partition 3
Partition 6
Partition 9

From SQL to NoSQL

- Importance of primary keys
- No joins

From SQL to NoSQL

- Importance of primary keys
- No joins

Implications:

From SQL to NoSQL

- Importance of primary keys
- No joins

Implications:

- Know your access patterns

From SQL to NoSQL

- Importance of primary keys
- No joins

Implications:

- Know your access patterns
- Use secondary indexes

Query with these

Not with these

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Query with these

Not with these

Primary key		Attributes		
Partition key: Actor	**Sort key: Movie**			
Tom Hanks	Cast Away	Role	Year	Genre
		Chuck Noland	2000	Drama
	Toy Story	Role	Year	Genre
		Woody	1995	Children's
Tim Allen	Toy Story	Role	Year	Genre
		Buzz Lightyear	1995	Children's
Natalie Portman	Black Swan	Role	Year	Genre
		Nina Sayers	2010	Drama

Primary key		Attributes		
Partition key: Actor	**Sort key: Year**			
Tom Hanks	1995	Movie	Role	Genre
		Toy Story	Woody	Children's
	2000	Movie	Role	Genre
		Cast Away	Chuck Noland	Drama
Tim Allen	1995	Movie	Role	Genre
		Toy Story	Buzz Lightyear	Children's
Natalie Portman	2010	Movie	Role	Genre
		Black Swan	Nina Sayers	Drama

Secondary indexes

Secondary indexes

- Data from base table is copied into secondary index with new primary key

Secondary indexes

- Data from base table is copied into secondary index with new primary key

- Can use read-based operations on secondary indexes – no writes

From SQL to NoSQL

- Importance of primary keys
- No joins

Implications:

- Know your access patterns
- Use secondary indexes

From SQL to NoSQL

- Importance of primary keys
- No joins

Implications:

- Know your access patterns
- Use secondary indexes
- All entities in one table + generic primary keys

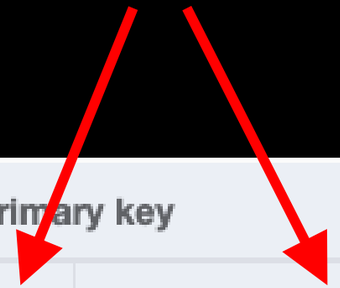
Example

- SaaS application
- Organizations and users

Example

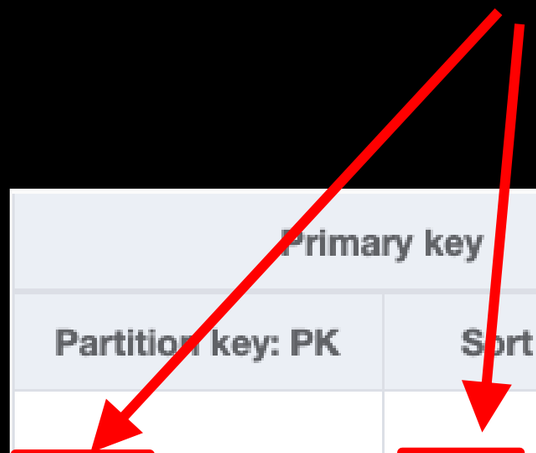
Primary key		Attributes			
Partition key: PK	**Sort key: SK**				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook

Example



Primary key		Attributes			
Partition key: PK	Sort key: SK				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook

Example



Primary key		Attributes			
Partition key: PK	Sort key: SK				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook

Example

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Example

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Example

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Example

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Example

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

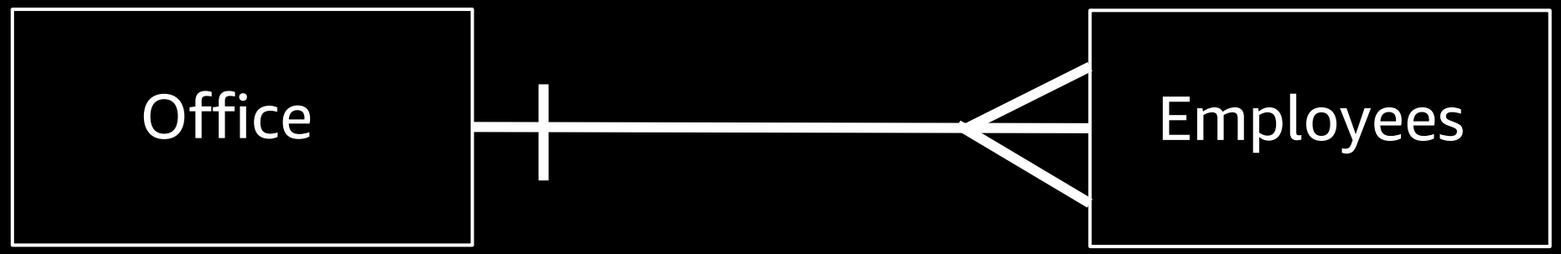
One-to-many relationships



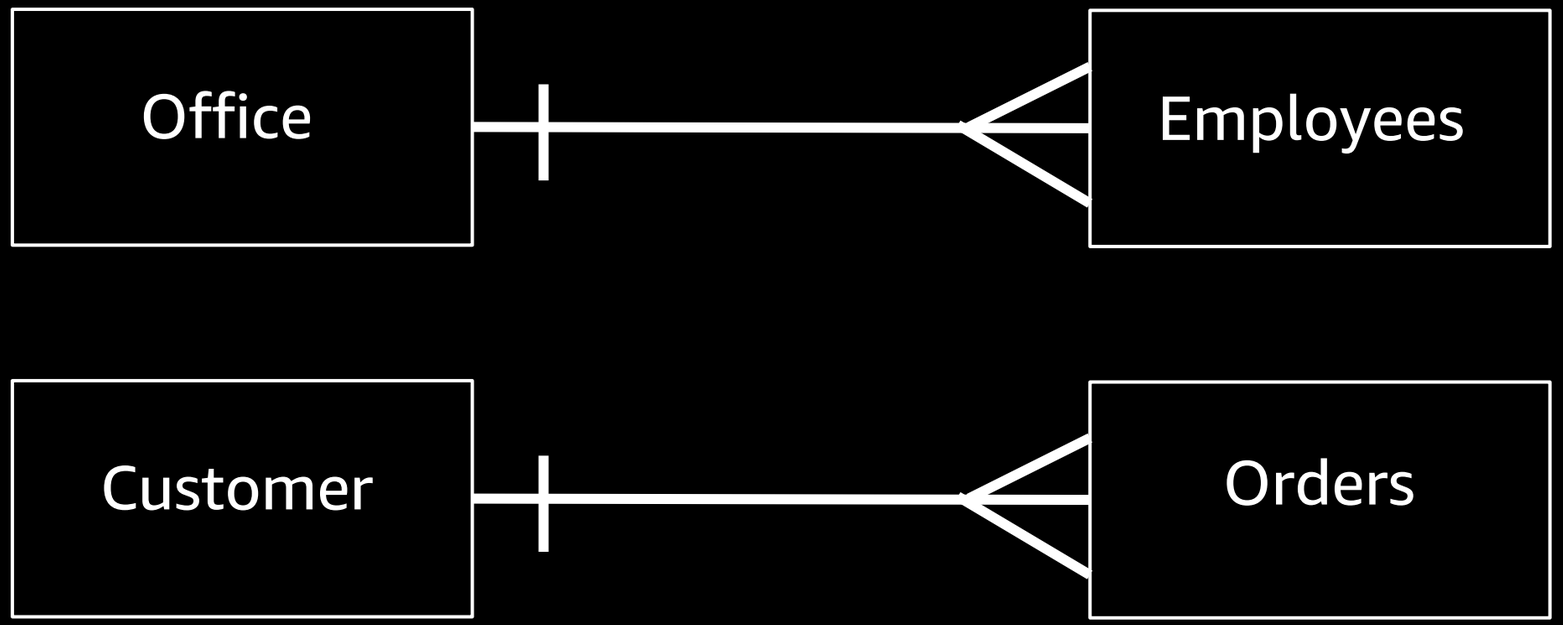
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

One-to-many relationships

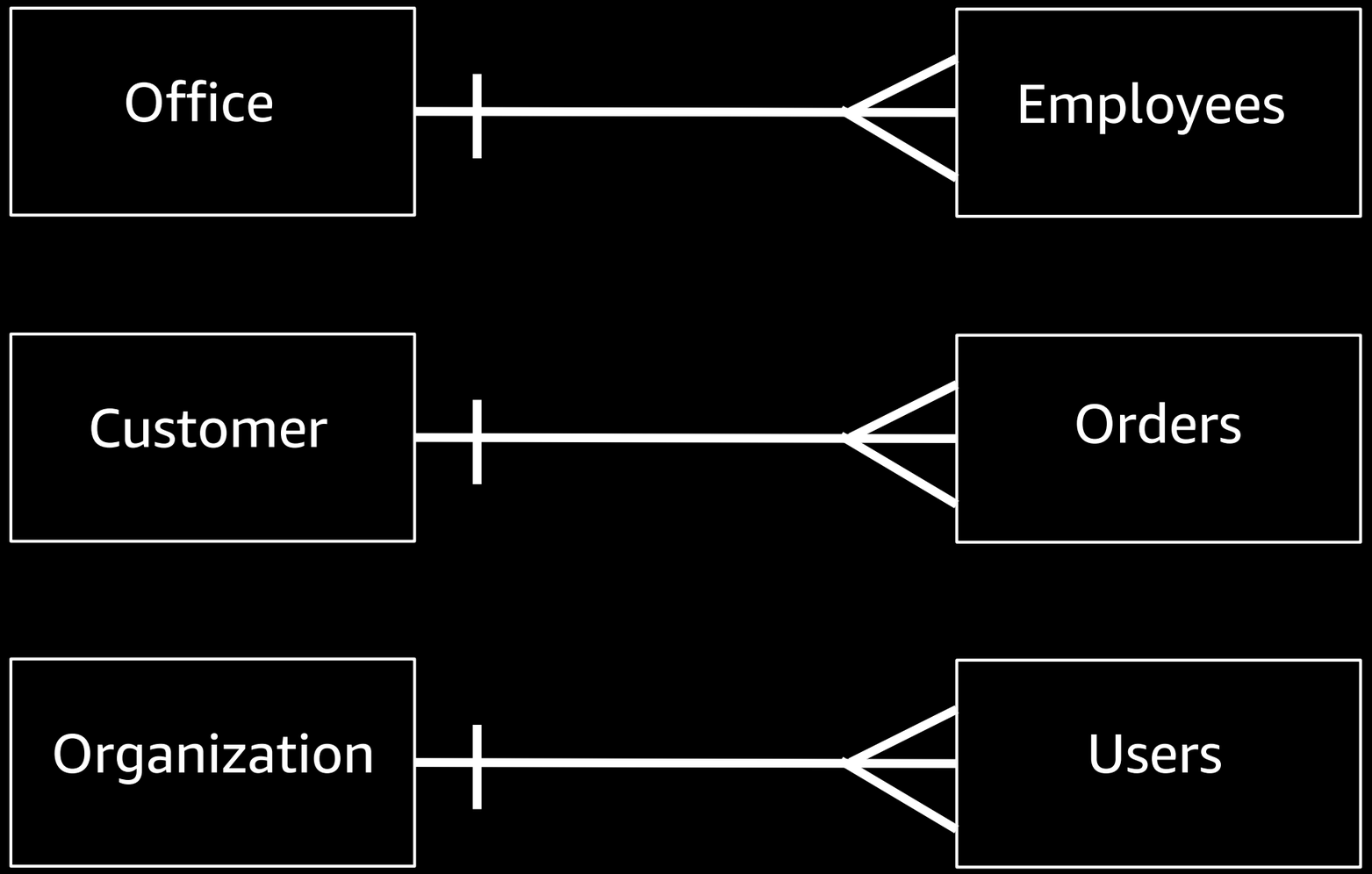
One-to-many relationships



One-to-many relationships



One-to-many relationships



One-to-many relationships

One-to-many relationships

Key problem – How to get information about parent item when fetching related item(s)

One-to-many relationships

Key problem – How to get information about parent item when fetching related item(s)

SQL – Normalization + joins

One-to-many relationships

Key problem – How to get information about parent item when fetching related item(s)

SQL – Normalization + joins

DynamoDB – Multiple strategies

One-to-many relationships

Key problem – How to get information about parent item when fetching related item(s)

SQL – Normalization + joins

DynamoDB – Multiple strategies

- Denormalization

One-to-many relationships

Key problem – How to get information about parent item when fetching related item(s)

SQL – Normalization + joins

DynamoDB – Multiple strategies

- Denormalization
- Pre-join data

One-to-many relationship strategies

One-to-many relationship strategies

1. Denormalization + complex attribute

Denormalization + complex attribute

Primary key		Attributes	
Partition key: PK	Sort key: SK		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Denormalization + complex attribute

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	PaymentMethod
		Berkshire Hathaway	Enterprise	{"Default":{"Type":"Credit Card","Number":"4465...."},"Backup":{"Type":"Bank Account","Routing Number":"104..."}}
	USER#CHARLIEMUNGER	UserName	Role	
		Charlie Munger	Member	
	USER#WARRENBUFFETT	UserName	Role	
		Warren Buffett	Admin	
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	PaymentMethod
		Facebook	Pro	{"Default":{"Type":"Credit Card","Number":"4462...."},"Backup":{"Type":"Bank Account","Routing Number":"103..."}}
	USER#SHERYLSANDBERG	UserName	Role	
		Sheryl Sandberg	Admin	

Denormalization + complex attribute

Primary key		Attributes		
Partition key: PK	Sort key: SK			
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	PaymentMethod
		Berkshire Hathaway	Enterprise	{"Default":{"Type":"Credit Card","Number":"4465...."},"Backup":{"Type":"Bank Account","Routing Number":"104..."}}
	USER#CHARLIEMUNGER	UserName	Role	
		Charlie Munger	Member	
	USER#WARRENBUFFETT	UserName	Role	
		Warren Buffett	Admin	
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	PaymentMethod
		Facebook	Pro	{"Default":{"Type":"Credit Card","Number":"4462...."},"Backup":{"Type":"Bank Account","Routing Number":"103..."}}
	USER#SHERYLSANDBERG	UserName	Role	
		Sheryl Sandberg	Admin	

One-to-many relationship strategies

1. Denormalization + complex attribute

One-to-many relationship strategies

1. Denormalization + complex attribute

   Good when:

One-to-many relationship strategies

1. Denormalization + complex attribute

   Good when:

   - No access pattern on related items directly

One-to-many relationship strategies

1. Denormalization + complex attribute

   Good when:

   - No access pattern on related items directly
   - Limited number of related items (400KB)

One-to-many relationship strategies

1. Denormalization + complex attribute

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

Denormalization + duplication

Primary key		Attributes	
Partition key: AuthorName	**Sort key: BookName**		
Stephen King	It	AuthorBirthdate	ReleaseYear
		September 21, 1947	1986
	The Shining	AuthorBirthdate	ReleaseYear
		September 21, 1947	1977
J.K. Rowling	Harry Potter and the Sorcerer's Stone	AuthorBirthdate	ReleaseYear
		July 31, 1965	1997

Denormalization + duplication

Primary key		Attributes	
Partition key: AuthorName	**Sort key: BookName**		
Stephen King	It	AuthorBirthdate	ReleaseYear
		September 21, 1947	1986
	The Shining	AuthorBirthdate	ReleaseYear
		September 21, 1947	1977
J.K. Rowling	Harry Potter and the Sorcerer's Stone	AuthorBirthdate	ReleaseYear
		July 31, 1965	1997

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

   Good when:

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

   Good when:

   - Duplicated data is immutable, OR

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

   Good when:

   - Duplicated data is immutable, OR
   - Duplicated data doesn’t change often or is not replicated much

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication
3. Composite primary key + query

Composite primary key + query

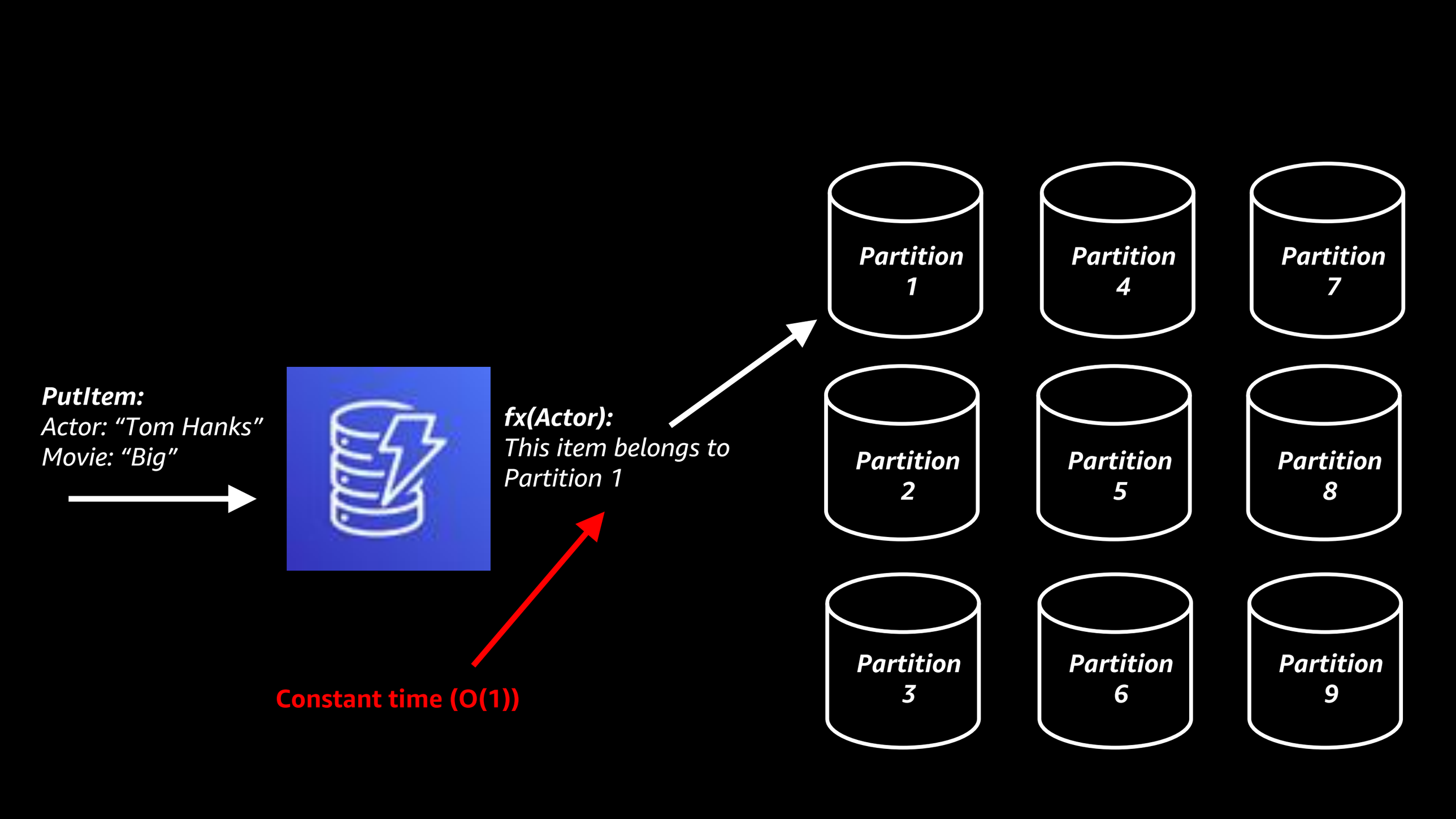
Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Composite primary key + query

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Composite primary key + query

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin


PutItem:
Actor: “Tom Hanks”
Movie: “Big”
fx(Actor):
This item belongs to
Partition 1
Constant time (O(1))
Partition 1
Partition 4
Partition 7
Partition 2
Partition 5
Partition 8
Partition 3
Partition 6
Partition 9

Composite primary key + query

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
Item collection	ORG#BERKSHIRE	OrgName	SubscriptionLevel
		Berkshire Hathaway	Enterprise
ORG#BERKSHIRE	USER#CHARLIEMUNGER	UserName	Role
		Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

Composite primary key + query

Primary key		Attributes	
Partition key: PK	**Sort key: SK**		
Item collection	ORG#BERKSHIRE	OrgName	SubscriptionLevel
	Organization item	Berkshire Hathaway	Enterprise
ORG#BERKSHIRE	USER#CHARLIEMUNGER	UserName	Role
	User items	Charlie Munger	Member
	USER#WARRENBUFFETT	UserName	Role
		Warren Buffett	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel
		Facebook	Pro
	USER#SHERYLSANDBERG	UserName	Role
		Sheryl Sandberg	Admin

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication
3. Composite primary key + query

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication
3. Composite primary key + query

   “Pre-join” your data in an item collection

One-to-many relationship strategies

1. Denormalization + complex attribute
2. Denormalization + duplication
3. Composite primary key + query

Summary



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

Summary

- DynamoDB basics
- SQL vs. NoSQL
- One-to-many relationships

Keep watching…

DAT305-PT2

Data modeling with Amazon DynamoDB – Part 2

AWS re:Invent

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

Related sessions

DAT402-PT1

Amazon DynamoDB Advanced Design Patterns - Part 1

DAT402-PT2

Amazon DynamoDB Advanced Design Patterns – Part 2

AWS re:Invent

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

Thank you!

Alex DeBrie

@alexbdebrie

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.