
AWS
re:Invent

DAT305-PT2

Data modeling with Amazon DynamoDB – Part 2

Alex DeBrie
AWS Data Hero
Engineer, Stedi

AWS re:Invent

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

What we covered in Part 1

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



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

Overview

- Data modeling process
- Filtering strategies
- Sorting strategies

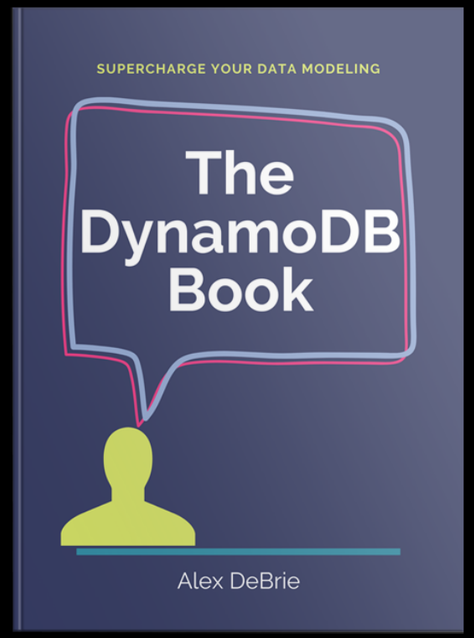


© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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





Data modeling process



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

Data modeling process

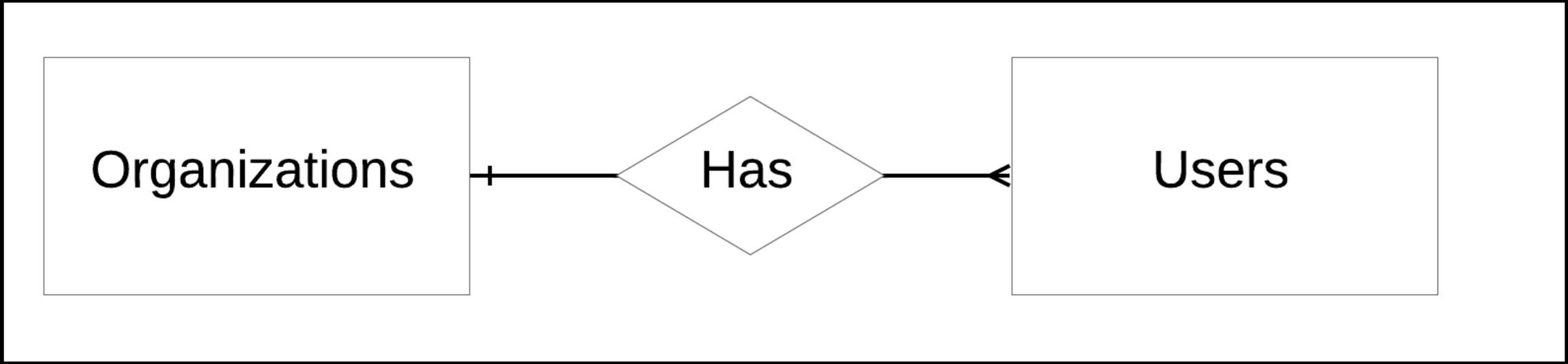
Data modeling process

- Understand your application

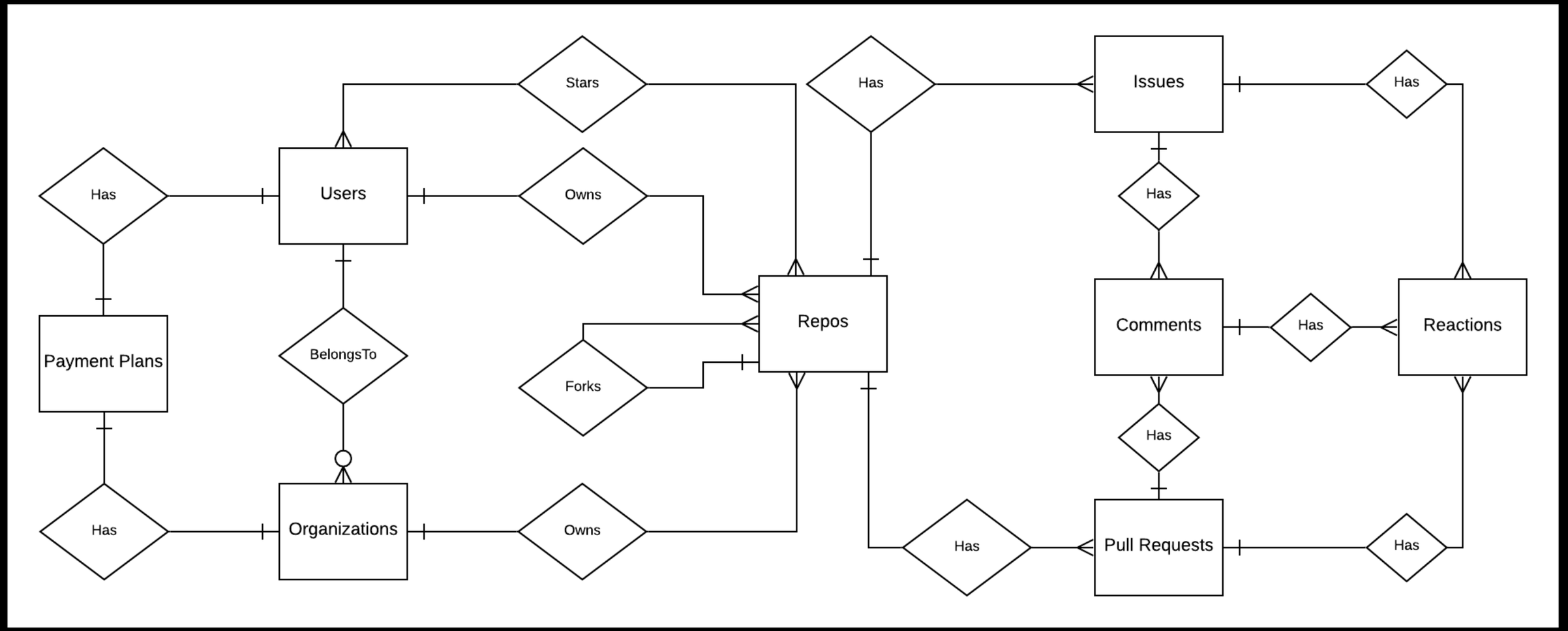
Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)

Data modeling process



Data modeling process



Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)

Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)
- List your access patterns

Data modeling process

- Messages:
  - View Messages for User
  - View Unread Messages for User
  - Send Hot New Deal Message to all Users
  - Send new Brand Deal Message to Brand Watchers
  - Send new Category Deal Message to Category Watchers

Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)
- List your access patterns

Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)
- List your access patterns
- Design your table to handle your access patterns

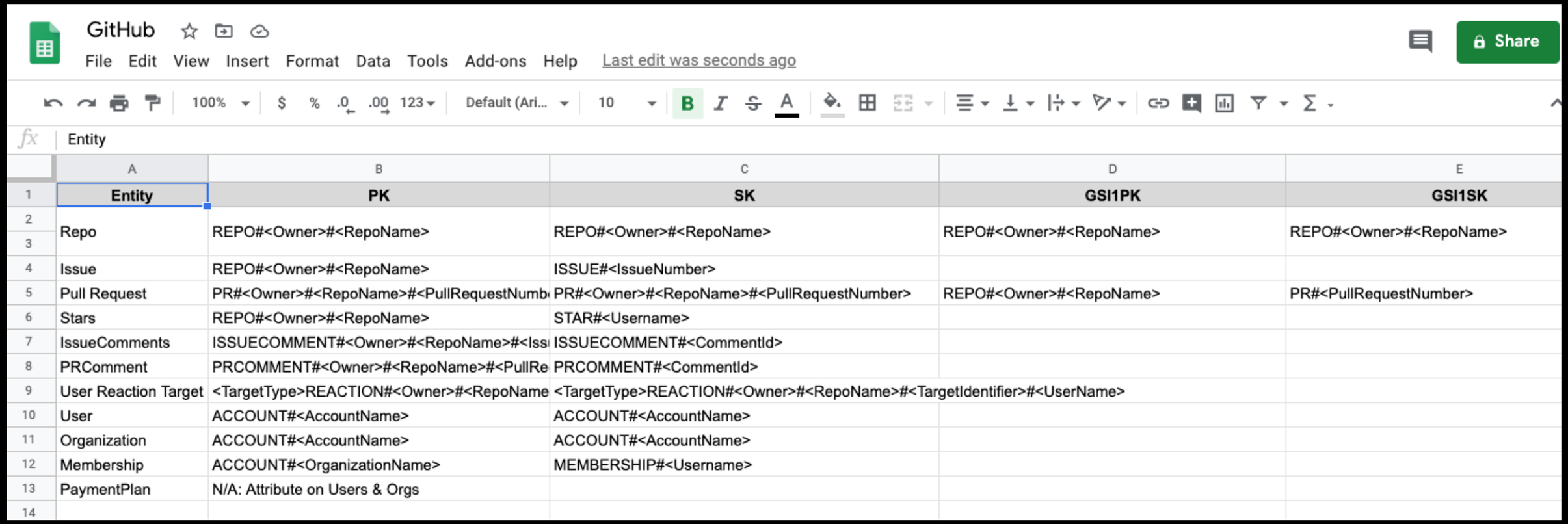
Data modeling process

Access pattern	Index	Parameters	Notes
View Messages for User	Main table	• Username	Query to find all Messages
View Unread Messages for User	GSI1	• Username	Query to find all Messages
Send Hot New Deal Message to all Users	UserIndex	N/A	1. Scan UserIndex to find all Users
	Main table	• Username	2. Create Message for each User in step 1
Send new Brand Deal Message to Brand Watchers	Main table	• Username • BrandName	1. Query BrandWatchers partition to find watchers for brand
	Main table	• Username	2. Create Message for each User in step 1
Send new Category Deal Message to Category Watchers	Main table	• Username • CategoryName	1. Query CategoryWatchers partition to find watchers for category
	Main table	• Username	2. Create Message for each User in step 1

Data modeling process

Entity	PK	SK
Users		
Organizations		
Payment Plans		
Repos		
Issues		
Pull Requests		
Comments		
Reactions		

Data modeling process



GitHub

File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

Share

100% $ % .0 .00 123 Default (Ari... 10

fx Entity

	A	B	C	D	E
1	**Entity**	**PK**	**SK**	**GSI1PK**	**GSI1SK**
2	Repo	REPO#<Owner>#<RepoName>	REPO#<Owner>#<RepoName>	REPO#<Owner>#<RepoName>	REPO#<Owner>#<RepoName>
3					
4	Issue	REPO#<Owner>#<RepoName>	ISSUE#<IssueNumber>		
5	Pull Request	PR#<Owner>#<RepoName>#<PullRequestNumb	PR#<Owner>#<RepoName>#<PullRequestNumber>	REPO#<Owner>#<RepoName>	PR#<PullRequestNumber>
6	Stars	REPO#<Owner>#<RepoName>	STAR#<Username>		
7	IssueComments	ISSUECOMMENT#<Owner>#<RepoName>#<Iss	ISSUECOMMENT#<CommentId>		
8	PRComment	PRCOMMENT#<Owner>#<RepoName>#<PullRe	PRCOMMENT#<CommentId>		
9	User Reaction Target	<TargetType>REACTION#<Owner>#<RepoName	<TargetType>REACTION#<Owner>#<RepoName>#<TargetIdentifier>#<UserName>		
10	User	ACCOUNT#<AccountName>	ACCOUNT#<AccountName>		
11	Organization	ACCOUNT#<AccountName>	ACCOUNT#<AccountName>		
12	Membership	ACCOUNT#<OrganizationName>	MEMBERSHIP#<Username>		
13	PaymentPlan	N/A: Attribute on Users & Orgs			
14					

Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)
- List your access patterns
- Design your table to handle your access patterns

Data modeling process

- Understand your application
- Create your entity-relationship diagram (ERD)
- List your access patterns
- Design your table to handle your access patterns
- Know your strategies

Filtering strategies



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

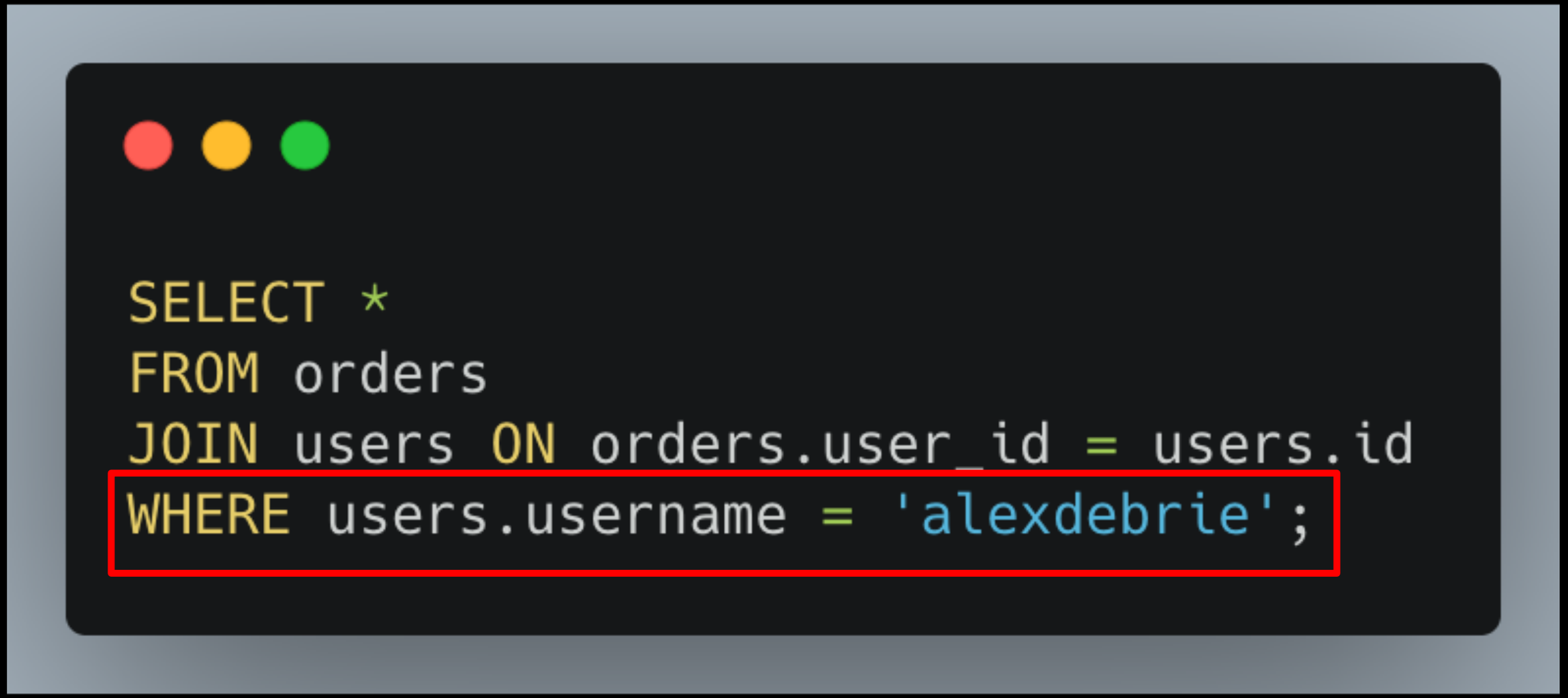
Filtering strategies

Filtering strategies



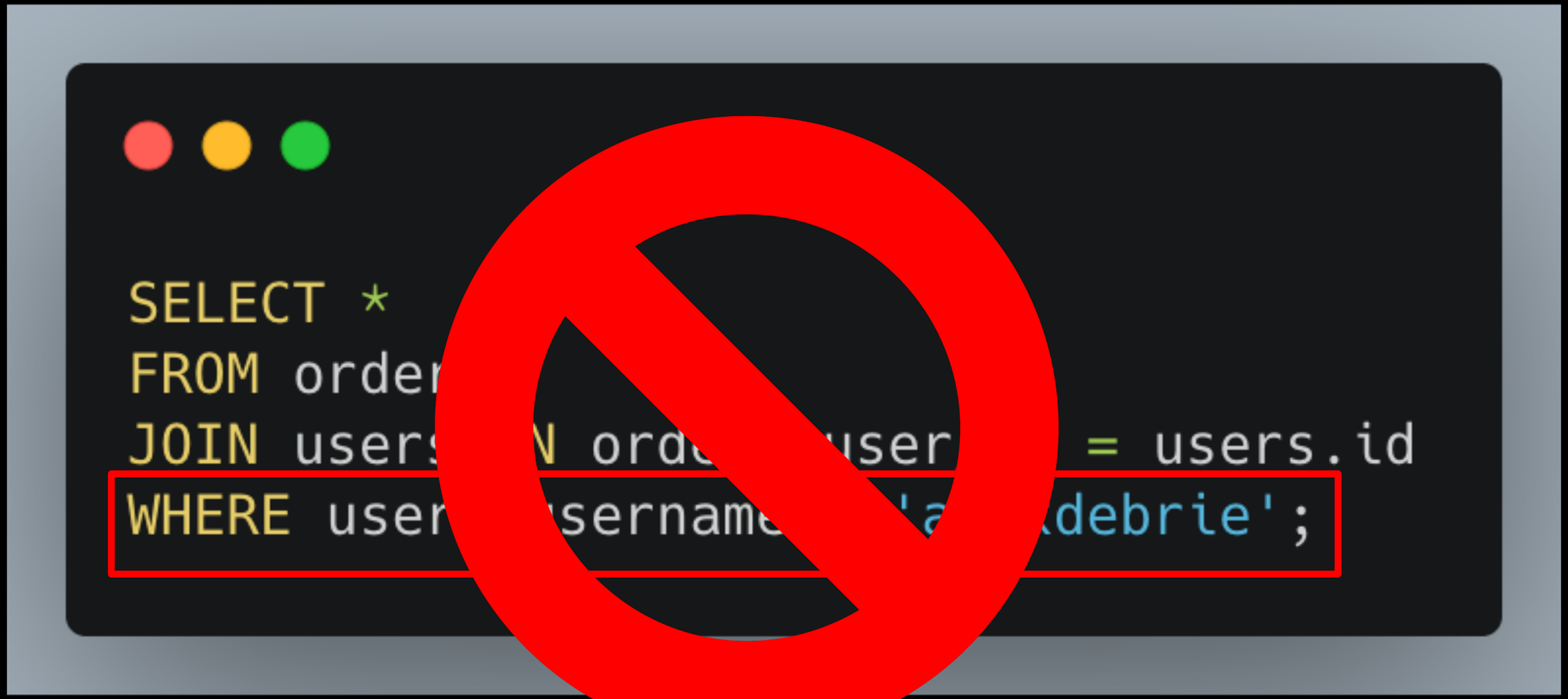
```sql
SELECT *
FROM orders
JOIN users ON orders.user_id = users.id
WHERE users.username = 'alexdebrie';
```

Filtering strategies



```sql
SELECT *
FROM orders
JOIN users ON orders.user_id = users.id
WHERE users.username = 'alexdebrie';
```

Filtering strategies



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

Filtering

Key point:

Almost all filtering happens with primary keys

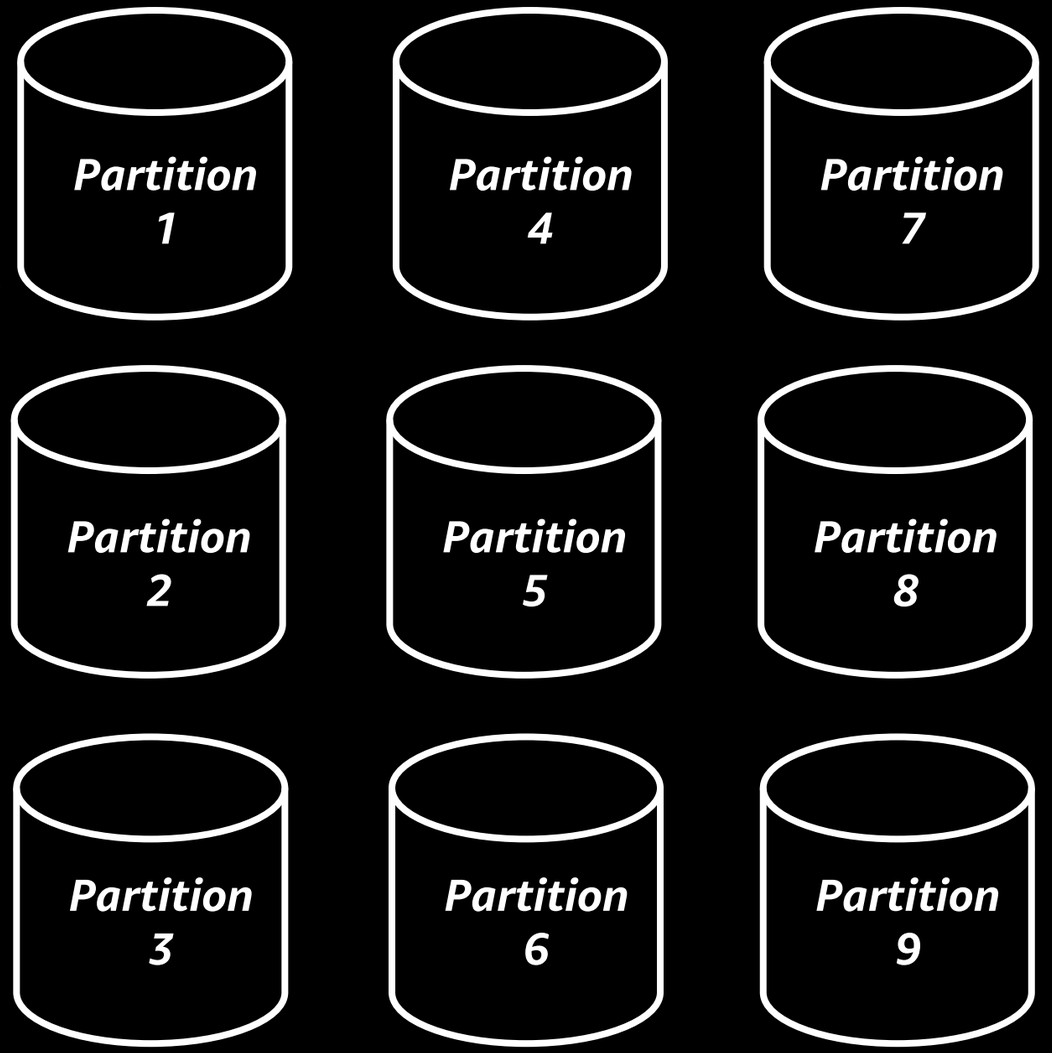
Filtering strategies

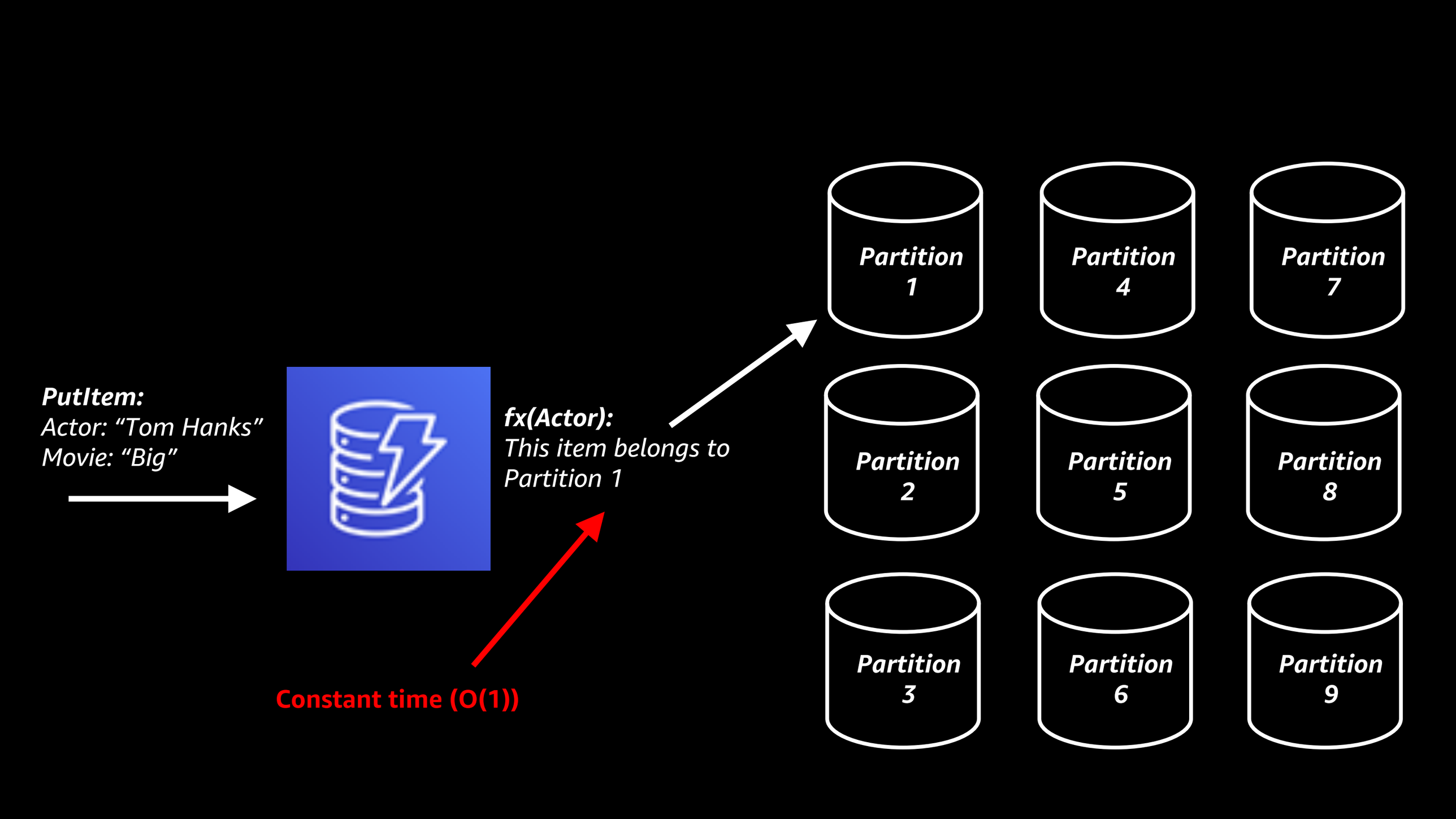
1. Filtering with the partition key

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

Filtering with the partition 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

Filtering with the partition 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

Filtering with the partition 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

Filtering strategies

1. Filtering with the partition key

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key

Filtering with the sort key

Filtering with the sort key

- Sort key is sorted

Filtering with the sort key

- Sort key is sorted
- 2 patterns:

Filtering with the sort key

- Sort key is sorted
- 2 patterns:
  1. Sort key itself has meaning

Filtering with the sort key

Primary key		Attributes
Partition key: SensorId	Sort key: Timestamp	
14891	2020-02-15 00:00:00	Temperature
		32.4
	2020-02-15 00:01:00	Temperature
		32.4
	2020-02-15 00:02:00	Temperature
		32.3
	2020-02-15 01:...	
	2020-02-15 02:...	
	2020-02-15 15:47:00	Temperature
		67.1
	2020-02-15 15:48:00	Temperature
		67.4
	2020-02-15 15:49:00	Temperature
		67.5

Filtering with the sort key

- Sort key is sorted
- 2 patterns:
  1. Sort key itself has meaning

Filtering with the sort key

- Sort key is sorted
- 2 patterns:
  1. Sort key itself has meaning
  2. Sort key has meaning based on how you've modeled it

Filtering with the sort key

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Filtering with the sort key

Primary key		Attributes						
Partition key: PK	Sort key: SK							
Item collection	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
REPO#alexdebrie #dynamodb-book		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Filtering with the sort key

Item collection

Repo item

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Filtering with the sort key

Item collection

Issue items

Repo item

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Filtering with the sort key

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Item collection

Issue items

Repo item

Star items

Filtering with the sort key

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Repo + Issues

Filtering with the sort key

Primary key		Attributes						
Partition key: PK	Sort key: SK							
REPO#alexdebrie#dynamodb-book	ISSUE#2	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-20 15:32:32	2	Open	ISSUE#alexdebrie#dynamodb-book#2	ISSUE#alexdebrie#dynamodb-book#2
	ISSUE#3	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-10-22 19:35:37	3	Closed	ISSUE#alexdebrie#dynamodb-book#3	ISSUE#alexdebrie#dynamodb-book#3
	ISSUE#4	RepoName	RepoOwner	CreatedAt	IssueNumber	Status	GSI1PK	GSI1SK
		dynamodb-book	alexdebrie	2019-11-07 03:43:42	4	Open	ISSUE#alexdebrie#dynamodb-book#4	ISSUE#alexdebrie#dynamodb-book#4
Repo + Stars	REPO#alexdebrie#dynamodb-book	RepoName	RepoOwner	CreatedAt	GSI1PK	GSI1SK		
		dynamodb-book	alexdebrie	2019-10-07 23:38:31	REPO#alexdebrie#dynamodb-book	REPO#alexdebrie#dynamodb-book		
	STAR#danny-developer	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2019-12-25 01:57:12	danny-developer			
	STAR#sally-scala	RepoName	RepoOwner	CreatedAt	StarringUser			
		dynamodb-book	alexdebrie	2020-01-07 10:35:13	sally-scala			

Filtering with the sort key

- Sort key is sorted
- 2 patterns:
  1. Sort key itself has meaning
  2. Sort key has meaning based on how you've modeled it

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key
3. Sparse index

Sparse index

Sparse index

- Only items that have all elements of a secondary index primary key will be copied

Sparse index

- Only items that have all elements of a secondary index primary key will be copied
- 2 strategies:

Sparse index

- Only items that have all elements of a secondary index primary key will be copied
- 2 strategies:

1. Filtering within entity type based on condition

Primary key			
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

Primary key		Attributes			
Partition key: PK	**Sort key: SK**				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
	USER#CHARLIEMUNGER	UserName	Role		
		Charlie Munger	Member		
	USER#WARRENBUFFETT	UserName	Role	GSI1PK	GSI1SK
		Warren Buffett	Admin	ORG#BERKSHIREHATHAWAY	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook
	USER#SHERYLSANDBERG	UserName	Role	GSI1PK	GSI1SK
		Sheryl Sandberg	Admin	ORG#FACEBOOK	Admin

Primary key		Attributes			
Partition key: PK	**Sort key: SK**				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
	USER#CHARLIEMUNGER	UserName	Role		
		Charlie Munger	Member		
	USER#WARRENBUFFETT	UserName	Role	GSI1PK	GSI1SK
		Warren Buffett	Admin	ORG#BERKSHIREHATHAWAY	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook
	USER#SHERYLSANDBERG	UserName	Role	GSI1PK	GSI1SK
		Sheryl Sandberg	Admin	ORG#FACEBOOK	Admin

Primary key		Attributes			
Partition key: PK	**Sort key: SK**				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
	USER#CHARLIEMUNGER	UserName	Role		
		Charlie Munger	Member		
	USER#WARRENBUFFETT	UserName	Role	GSI1PK	GSI1SK
		Warren Buffett	Admin	ORG#BERKSHIREHATHAWAY	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook
	USER#SHERYLSANDBERG	UserName	Role	GSI1PK	GSI1SK
		Sheryl Sandberg	Admin	ORG#FACEBOOK	Admin

Primary key		Attributes			
Partition key: PK	**Sort key: SK**				
ORG#BERKSHIRE	ORG#BERKSHIRE	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Berkshire Hathaway	Enterprise	ORGANIZATIONS	Berkshire Hathaway
	USER#CHARLIEMUNGER	UserName	Role		
		Charlie Munger	Member		
	USER#WARRENBUFFETT	UserName	Role	GSI1PK	GSI1SK
		Warren Buffett	Admin	ORG#BERKSHIREHATHAWAY	Admin
ORG#FACEBOOK	ORG#FACEBOOK	OrgName	SubscriptionLevel	GSI1PK	GSI1SK
		Facebook	Pro	ORGANIZATIONS	Facebook
	USER#SHERYLSANDBERG	UserName	Role	GSI1PK	GSI1SK
		Sheryl Sandberg	Admin	ORG#FACEBOOK	Admin

Primary key		Attributes			
Partition key: GSI1PK	**Sort key: GSI1SK**				
ORGANIZATIONS	Berkshire Hathaway	PK	SK	OrgName	SubscriptionLevel
		ORG#BERKSHIRE	ORG#BERKSHIRE	Berkshire Hathaway	Enterprise
	Facebook	PK	SK	OrgName	SubscriptionLevel
		ORG#FACEBOOK	ORG#FACEBOOK	Facebook	Pro
ORG#BERKSHIREHATHAWAY	Admin	PK	SK	UserName	Role
		ORG#BERKSHIRE	USER#WARRENBUFFETT	Warren Buffett	Admin
ORG#FACEBOOK	Admin	PK	SK	UserName	Role
		ORG#FACEBOOK	USER#SHERYLSANDBERG	Sheryl Sandberg	Admin

Primary key		Attributes			
Partition key: GSI1PK	**Sort key: GSI1SK**				
ORGANIZATIONS	Berkshire Hathaway	PK	SK	OrgName	SubscriptionLevel
		ORG#BERKSHIRE	ORG#BERKSHIRE	Berkshire Hathaway	Enterprise
	Facebook	PK	SK	OrgName	SubscriptionLevel
		ORG#FACEBOOK	ORG#FACEBOOK	Facebook	Pro
ORG#BERKSHIREHATHAWAY	Admin	PK	SK	UserName	Role
		ORG#BERKSHIRE	USER#WARRENBUFFETT	Warren Buffett	Admin
ORG#FACEBOOK	Admin	PK	SK	UserName	Role
		ORG#FACEBOOK	USER#SHERYLSANDBERG	Sheryl Sandberg	Admin

Sparse index

- Only items that have all elements of a secondary index primary key will be copied
- 2 strategies:

1. Filtering within entity type based on condition

Sparse index

- Only items that have all elements of a secondary index primary key will be copied
- 2 strategies:

1. Filtering within entity type based on condition
2. Projecting a single type of entity into a secondary index

Primary key		Attributes	
Partition key: PK	Sort key: SK		
CUSTOMER#ALEXDEBRIE	CUSTOMER#ALEXDEBRIE	CustomerName	JoinedAt
		Alex DeBrie	2020-01-12 08:02:57
CUSTOMER#BARRYBONDS	CUSTOMER#BARRYBONDS	CustomerName	JoinedAt
		Barry Bonds	2020-01-18 19:53:59
ORDER#1MLKM4	ORDER#1MLKM4	OrderDate	OrderCustomer
		2020-01-19 23:08:31	Barry Bonds
ORDER#OIQ410	ORDER#OIQ410	OrderDate	OrderCustomer
		2020-01-24 22:27:43	Alex DeBrie
INVENTORY#PQOIUA0	INVENTORY#PQOIUA0	ItemName	Price
		Spalding basketball	29.99
INVENTORY#DAJR10	INVENTORY#DAJR10	ItemName	Price
		A Tale of Two Cities	9.99

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER#ALEXDEBRIE	CUSTOMER#ALEXDEBRIE	CustomerName	JoinedAt	CustomerIndexId
		Alex DeBrie	2020-01-12 08:02:57	Alex DeBrie
CUSTOMER#BARRYBONDS	CUSTOMER#BARRYBONDS	CustomerName	JoinedAt	CustomerIndexId
		Barry Bonds	2020-01-18 19:53:59	Barry Bonds
ORDER#1MLKM4	ORDER#1MLKM4	OrderDate	OrderCustomer	
		2020-01-19 23:08:31	Barry Bonds	
ORDER#OIQ410	ORDER#OIQ410	OrderDate	OrderCustomer	
		2020-01-24 22:27:43	Alex DeBrie	
INVENTORY#PQOIUA0	INVENTORY#PQOIUA0	ItemName	Price	
		Spalding basketball	29.99	
INVENTORY#DAJR10	INVENTORY#DAJR10	ItemName	Price	
		A Tale of Two Cities	9.99	



Primary key	Attributes			
Partition key: CustomerIndexId				
Alex DeBrie	PK	SK	CustomerName	JoinedAt
	CUSTOMER#ALEXDEBRIE	CUSTOMER#ALEXDEBRIE	Alex DeBrie	2020-01-12 08:02:57
Barry Bonds	PK	SK	CustomerName	JoinedAt
	CUSTOMER#BARRYBONDS	CUSTOMER#BARRYBONDS	Barry Bonds	2020-01-18 19:53:59

Sparse index

- Only items that have all elements of a secondary index primary key will be copied
- 2 strategies:

1. Filtering within entity type based on condition
2. Projecting a single type of entity into a secondary index

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key
3. Sparse index

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key
3. Sparse index
4. Client-side filtering

Client-side filtering

Client-side filtering

- Filtering in backend server or on front end

Client-side filtering

- Filtering in backend server or on front end
- Good when:

Client-side filtering

- Filtering in backend server or on front end
- Good when:
  - Filtering is difficult to model in the database

Client-side filtering

- Filtering in backend server or on front end
- Good when:
  - Filtering is difficult to model in the database
  - Dataset isn’t that big

Client-side filtering

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

Client-side filtering

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

Client-side filtering

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

Client-side filtering

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

Filtering strategies

1. Filtering with the partition key
2. Filtering with the sort key
3. Sparse index
4. Client-side filtering

Sorting strategies



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

Sorting strategies

Sorting strategies

- Basics of sorting

Basics of sorting

Basics of sorting

- Only the sort key is sorted

Basics of sorting

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

Basics of sorting

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

Basics of sorting

- Only the sort key is sorted

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes

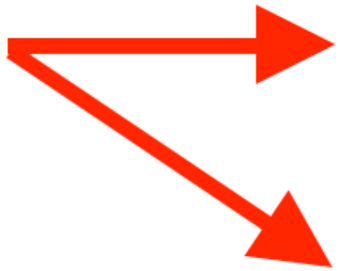
Basics of sorting

Primary key		Attributes
Partition key: Initial	Sort key: LastName	
D	DeBrie	Name
		Alex DeBrie
	Dean	Name
		Jimmy Dean
	Dern	Name
		Laura Dern

Basics of sorting

Primary key		Attributes
Partition key: Initial	**Sort key: LastName**	
D	DeBrie	Name
		Alex DeBrie
	Dean	Name
		Jimmy Dean
	Dern	Name
		Laura Dern

DeBrie before Dean??



Basics of sorting

Primary key		Attributes
Partition key: Initial	Sort key: LastName	
D	DEAN	Name
		Jimmy Dean
	DEBRIE	Name
		Alex DeBrie
	DERN	Name
		Laura Dern

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps

Basics of sorting

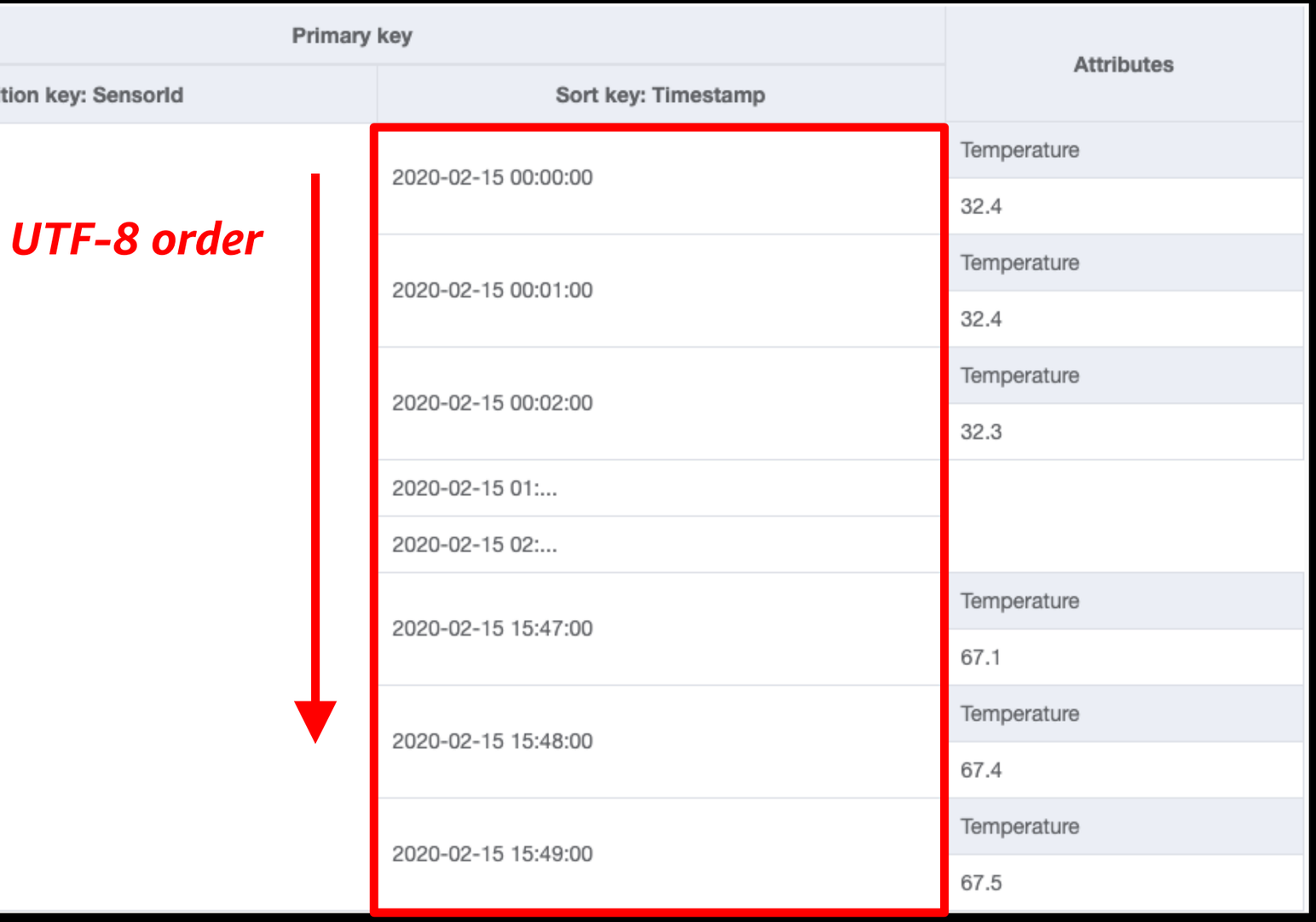
- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps
  - Epoch or ISO-8601 works; ISO-8601 is human readable

Basics of sorting

Primary key		Attributes
Partition key: SensorId	Sort key: Timestamp	
14891	2020-02-15 00:00:00	Temperature
		32.4
	2020-02-15 00:01:00	Temperature
		32.4
	2020-02-15 00:02:00	Temperature
		32.3
	2020-02-15 01:...	
	2020-02-15 02:...	
	2020-02-15 15:47:00	Temperature
		67.1
	2020-02-15 15:48:00	Temperature
		67.4
	2020-02-15 15:49:00	Temperature
		67.5

Basics of sorting

Primary key		Attributes
Partition key: SensorId	Sort key: Timestamp	
14891	2020-02-15 00:00:00	Temperature
		32.4
	2020-02-15 00:01:00	Temperature
		32.4
	2020-02-15 00:02:00	Temperature
		32.3
	2020-02-15 01:...	
	2020-02-15 02:...	
	2020-02-15 15:47:00	Temperature
		67.1
	2020-02-15 15:48:00	Temperature
		67.4
	2020-02-15 15:49:00	Temperature
		67.5



Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps
  - Epoch or ISO-8601 works; ISO-8601 is human readable

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps
  - Epoch or ISO-8601 works; ISO-8601 is human readable
- Unique, sortable IDs

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps
  - Epoch or ISO-8601 works; ISO-8601 is human readable
- Unique, sortable IDs
  - ULID or KSUID

Unique, sortable IDs

Unique, sortable IDs

- ULID



Unique, sortable IDs

- ULID
  - 128 bits



Unique, sortable IDs

- ULID
  - 128 bits
    - First 48 bits – millisecond timestamp
    - Last 80 bits – randomness



Unique, sortable IDs

- ULID
  - 128 bits
    - First 48 bits – millisecond timestamp
    - Last 80 bits – randomness
  - URL safe



Unique, sortable IDs

- ULID
  - 128 bits
    - First 48 bits – millisecond timestamp
    - Last 80 bits – randomness
  - URL safe
  - Lexicographically sortable



Unique, sortable IDs

- ULID
  - 128 bits
    - First 48 bits – millisecond timestamp
    - Last 80 bits – randomness
  - URL safe
  - Lexicographically sortable
- KSUID



Unique, sortable IDs

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	NumberItems
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	6
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	NumberItems
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	3
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	NumberItems
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	4

Unique, sortable IDs

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	NumberItems
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	6
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	NumberItems
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	3
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	NumberItems
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	4



Unique, sortable IDs

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	NumberItems
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	6
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	NumberItems
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	3
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	NumberItems
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	4

Unique, sortable IDs

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	NumberItems
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	6
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	NumberItems
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	3
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	NumberItems
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	4

Basics of sorting

- Only the sort key is sorted
- UTF-8 bytes
  - Alphabetical
  - Uppercase before lowercase
  - Numbers and symbols
- Sorting with timestamps
  - Epoch or ISO-8601 works; ISO-8601 is human readable
- Unique, sortable IDs
  - ULID or KSUID

Sorting strategies

- Basics of sorting

Sorting strategies

- Basics of sorting
- Sorting on changing attributes

Sorting on changing attributes

Sorting on changing attributes

- Changing a primary key requires delete + put

Sorting on changing attributes

- Changing a primary key requires delete + put
- Use secondary index, if possible

Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	NumberItems
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	6
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	NumberItems
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	3
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	NumberItems
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	4

Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	UpdatedAt
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	UpdatedAt
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	UpdatedAt
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	UpdatedAt
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

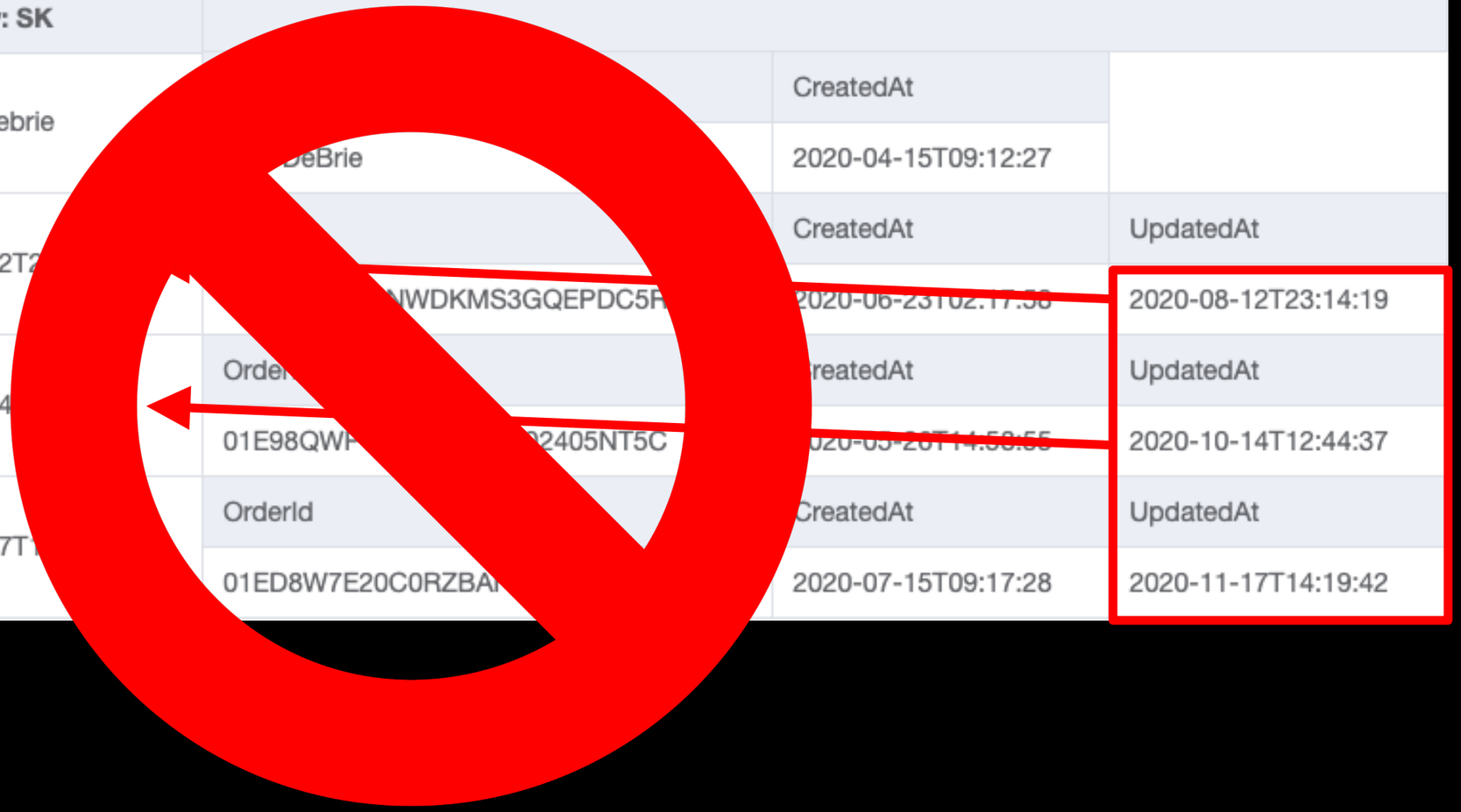
Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#2020-08-12T23:14:19	OrderId	CreatedAt	UpdatedAt
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19
	ORDER#2020-10-14T12:44:37	OrderId	CreatedAt	UpdatedAt
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37
	ORDER#2020-11-17T14:19:42	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#2020-08-12T23:14:19	OrderId	CreatedAt	UpdatedAt
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:38	2020-08-12T23:14:19
	ORDER#2020-10-14T12:44:37	OrderId	CreatedAt	UpdatedAt
		01E98QWPWRC1QSE4E02405NT5C	2020-09-26T14:58:55	2020-10-14T12:44:37
	ORDER#2020-11-17T14:19:42	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	Sort key: SK			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie		CreatedAt	
		eBrie	2020-04-15T09:12:27	
	ORDER#2020-08-12T2		CreatedAt	UpdatedAt
		WDKMS3GQEPDC5F	2020-06-23T02:17:38	2020-08-12T23:14:19
	ORDER#2020-10-14	Order	reatedAt	UpdatedAt
		01E98QWF 2405NT5C	020-05-26T14:58:55	2020-10-14T12:44:37
	ORDER#2020-11-17T1	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAF	2020-07-15T09:17:28	2020-11-17T14:19:42



Sorting on changing attributes

Primary key		Attributes		
Partition key: PK	**Sort key: SK**			
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt	
		Alex DeBrie	2020-04-15T09:12:27	
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	UpdatedAt
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	UpdatedAt
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	UpdatedAt
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes				
Partition key: PK	**Sort key: SK**					
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt			
		Alex DeBrie	2020-04-15T09:12:27			
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37	CUSTOMER#alexdebrie	UPDATED#2020-10-14T12:44:37
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19	CUSTOMER#alexdebrie	UPDATED#2020-08-12T23:14:19
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42	CUSTOMER#alexdebrie	UPDATED#2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes				
Partition key: PK	Sort key: SK					
CUSTOMER#alexdebrie	CUSTOMER#alexdebrie	CustomerName	CreatedAt			
		Alex DeBrie	2020-04-15T09:12:27			
	ORDER#01E98QWPWRC1QSE4E02405NT5C	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37	CUSTOMER#alexdebrie	UPDATED#2020-10-14T12:44:37
	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19	CUSTOMER#alexdebrie	UPDATED#2020-08-12T23:14:19
	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	OrderId	CreatedAt	UpdatedAt	GSI1PK	GSI1SK
		01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42	CUSTOMER#alexdebrie	UPDATED#2020-11-17T14:19:42

Sorting on changing attributes

Primary key		Attributes				
Partition key: GSI1PK	Sort key: GSI1SK					
CUSTOMER#alexdebrie	UPDATED#2020-08-12T23:14:19	PK	SK	OrderId	CreatedAt	UpdatedAt
		CUSTOMER#alexdebrie	ORDER#01EBFFFFZGNWDKMS3GQEPDC5RM	01EBFFFFZGNWDKMS3GQEPDC5RM	2020-06-23T02:17:58	2020-08-12T23:14:19
	UPDATED#2020-10-14T12:44:37	PK	SK	OrderId	CreatedAt	UpdatedAt
		CUSTOMER#alexdebrie	ORDER#01E98QWPWRC1QSE4E02405NT5C	01E98QWPWRC1QSE4E02405NT5C	2020-05-26T14:58:55	2020-10-14T12:44:37
	UPDATED#2020-11-17T14:19:42	PK	SK	OrderId	CreatedAt	UpdatedAt
		CUSTOMER#alexdebrie	ORDER#01ED8W7E20C0RZBAFPC6HVAJAG	01ED8W7E20C0RZBAFPC6HVAJAG	2020-07-15T09:17:28	2020-11-17T14:19:42

Sorting on changing attributes

- Changing a primary key requires delete + put
- Use secondary index, if possible

Sorting strategies

- Basics of sorting
- Sorting on changing attributes

Sorting strategies

- Basics of sorting
- Sorting on changing attributes
- Ascending vs. descending

Ascending vs. descending

Ascending vs. descending

- ScanIndexForward property

Ascending vs. descending

- ScanIndexForward property
- When modeling, important when combining multiple entity types in single item collection

Ascending vs. descending

Primary key		Attributes
Partition key: PK	**Sort key: SK**	
DEVICE#123	DEVICE#123	DeviceLocation
		Omaha, NE
	READING#2020-03-14T10:33:00	Temperature
		32.1
	READING#2020-03-14T10:34:00	Temperature
		32.2
	READING#2020-03-14T10:35:00	Temperature
		32.2
	READING#2020-03-14T10:36:00	Temperature
		32.4
	READING#2020-03-14T10:37:00	Temperature
		32.4

Ascending vs. descending

Primary key		Attributes
Partition key: PK	Sort key: SK	
Item collection	DEVICE#123 — Device item	DeviceLocation
		Omaha, NE
DEVICE#123	READING#2020-03-14T10:33:00 — Reading items	Temperature
		32.1
	READING#2020-03-14T10:34:00	Temperature
		32.2
	READING#2020-03-14T10:35:00	Temperature
		32.2
	READING#2020-03-14T10:36:00	Temperature
		32.4
	READING#2020-03-14T10:37:00	Temperature
		32.4

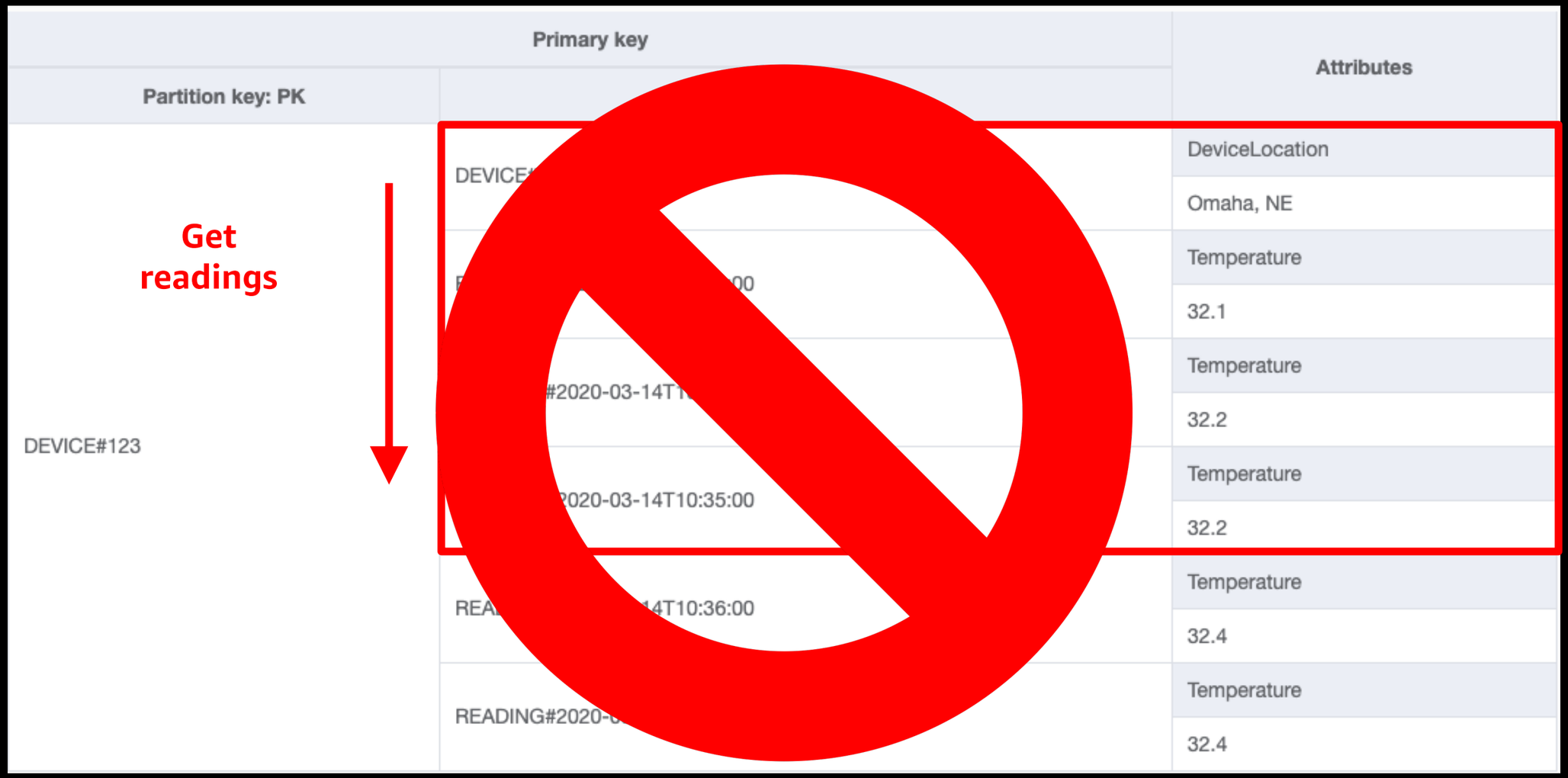
Ascending vs. descending

Primary key		Attributes
Partition key: PK	Sort key: SK	
DEVICE#123	DEVICE#123	DeviceLocation
		Omaha, NE
	READING#2020-03-14T10:33:00	Temperature
		32.1
	READING#2020-03-14T10:34:00	Temperature
		32.2
	READING#2020-03-14T10:35:00	Temperature
		32.2
	READING#2020-03-14T10:36:00	Temperature
		32.4
	READING#2020-03-14T10:37:00	Temperature
		32.4

Get readings



Ascending vs. descending



Ascending vs. descending

Primary key		Attributes
Partition key: PK	**Sort key: SK**	
DEVICE#123	DEVICE#123	DeviceLocation
		Omaha, NE
	READING#2020-03-14T10:33:00	Temperature
		32.1
	READING#2020-03-14T10:34:00	Temperature
		32.2
	READING#2020-03-14T10:35:00	Temperature
		32.2
	READING#2020-03-14T10:36:00	Temperature
		32.4
	READING#2020-03-14T10:37:00	Temperature
		32.4

Ascending vs. descending

Primary key		Attributes
Partition key: PK	**Sort key: SK**	
DEVICE#123	#READING#2020-03-14T10:33:00	Temperature
		32.1
	#READING#2020-03-14T10:34:00	Temperature
		32.2
	#READING#2020-03-14T10:35:00	Temperature
		32.2
	#READING#2020-03-14T10:36:00	Temperature
		32.4
	#READING#2020-03-14T10:37:00	Temperature
		32.4
	DEVICE#123	DeviceLocation
		Omaha, NE

Ascending vs. descending

Primary key		Attributes
Partition key: PK	Sort key: SK	
DEVICE#123	#READING#2020-03-14T10:33:00	Temperature
		32.1
	#READING#2020-03-14T10:34:00	Temperature
		32.2
	#READING#2020-03-14T10:35:00	Temperature
		32.2
	#READING#2020-03-14T10:36:00	Temperature
		32.4
	#READING#2020-03-14T10:37:00	Temperature
		32.4
	DEVICE#123	DeviceLocation
		Omaha, NE

Ascending vs. descending

Primary key		Attributes
Partition key: PK	Sort key: SK	
Item collection		
DEVICE#123	#READING#2020-03-14T10:33:00	Temperature
		32.1
	#READING#2020-03-14T10:34:00	Temperature
		32.2
	#READING#2020-03-14T10:35:00	Temperature
		32.2
	#READING#2020-03-14T10:36:00	Temperature
		32.4
	#READING#2020-03-14T10:37:00	Temperature
		32.4
	DEVICE#123	DeviceLocation
		Omaha, NE

Reading items

Device item

Ascending vs. descending

Primary key		Attributes
Partition key: PK	Sort key: SK	
DEVICE#123	#READING#2020-03-14T10:33:00	Temperature
		32.1
	#READING#2020-03-14T10:34:00	Temperature
		32.2
	#READING#2020-03-14T10:35:00	Temperature
		32.2
	#READING#2020-03-14T10:36:00	Temperature
		32.4
	#READING#2020-03-14T10:37:00	Temperature
		32.4
	DEVICE#123	DeviceLocation
		Omaha, NE



Get readings

Ascending vs. descending

- ScanIndexForward property
- When modeling, important when combining multiple entity types in single item collection

Sorting strategies

- Basics of sorting
- Sorting on changing attributes
- Ascending vs. descending

Summary



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

Summary

- Data modeling process
- Filtering strategies
- Sorting strategies

What we covered in Part 1

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

In case you missed it…

DAT305-PT1

Data modeling with Amazon DynamoDB – Part 1

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.