aws modern apps

# Padrões Avançados de Design e Consultas para Amazon DynamoDB

Peterson Larentis
Enterprise Solutions Architect, AWS

THE DEVELOPER'S CONFERENCE

# Apresentação



/peterson-larentis

Peterson Larentis é **Arquiteto de Soluções na AWS** e **professor de MBA na FIAP**, **pragmático e entusiasta por Cloud Computing, Serverless, DevOps e desenvolvimento Ágil.**

**Head Trainer da IWorks Education**

Na AWS auxilia os maiores e-commerces da américa latina a construírem arquiteturas seguras, confiáveis, elásticas e escaláveis com excelência operacional utilizando as melhores práticas ágeis de engenharia de software e é **responsável pela Track de DevOps no AWS Summit Brasil.**

Na FIAP ministra disciplinas de Serverless, Microservices, DevOps e Engenharia de Software ágil nos MBA em Cloud Computing, MBA em Full StacK Development e MBA em Business Agility.

Formado em Análise e Desenvolvimento de Sistemas possui MBA em Marketing Digital pela Fundação Getúlio Vargas e 6 AWS Certifications.



aws modern apps

# Characteristics of internet-scale apps

E-commerce | Media streaming | Social media | Online gaming | Shared economy

| | |
|---|---|
| Users | 1 million+ |
| Data volume | TB, PB, EB |
| Locality | Global |
| Performance | Microsecond latency |
| Request rate | Millions per second |
| Access | Mobile, IoT, devices |
| Scale | Up and down |
| Economics | Pay as you go |
| Developer access | Instant API access |



aws

# DynamoDB use cases by industry

## Customers rely on DynamoDB to support their mission-critical workloads

### Banking and finance

Fraud detection
User transactions
Mainframe offloading
(Capital One, Vanguard, Fannie Mae)

### Gaming

Game states
Leaderboards
Player data stores
(Riot Games, Electronic Arts, PennyPop)

### Software and internet

Metadata caches
Ride-tracking data stores
Relationship graph data stores
(Uber, Lyft, Swiggy, Snap, Duolingo)

### Ad tech

User profile stores
Metadata stores for assets
Popular-item cache
(AdRoll, GumGum, Branch, DataXu)

### Retail

Shopping carts
Workflow engines
Customer profiles
(Nordstrom, Nike, Zalando, Mercado Libre)

### Media & Entertainment

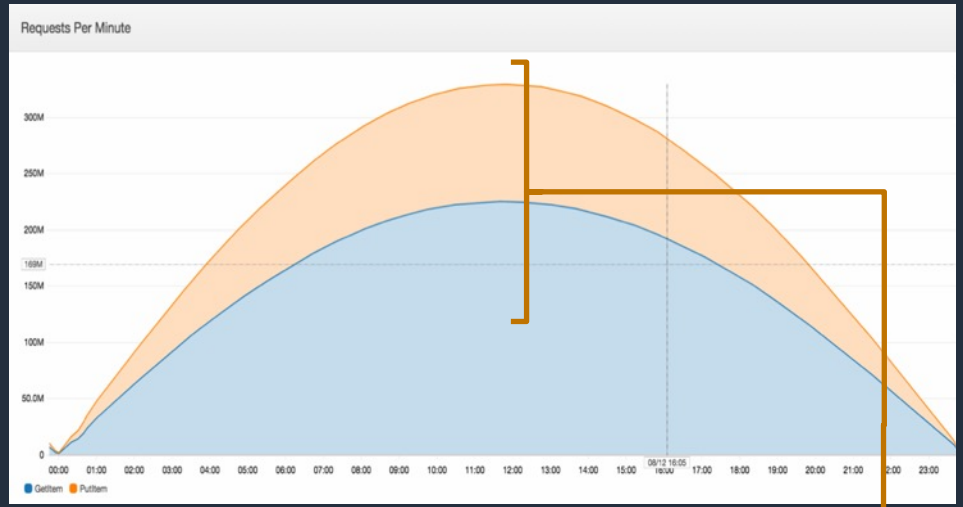
User data stores
Media metadata stores
Digital rights management stores
(Airtel Wynk, Amazon Prime, Netflix)



aws modern apps

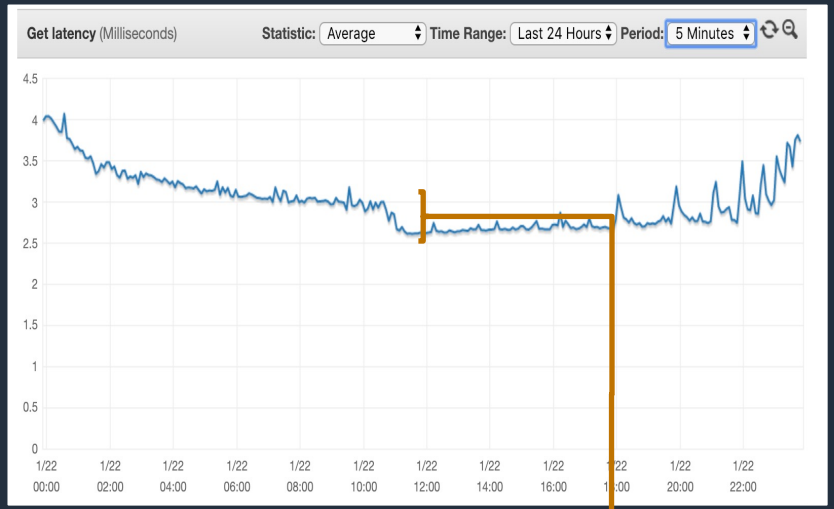
# Performance at any scale

## High request volume




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

## Consistent low latency



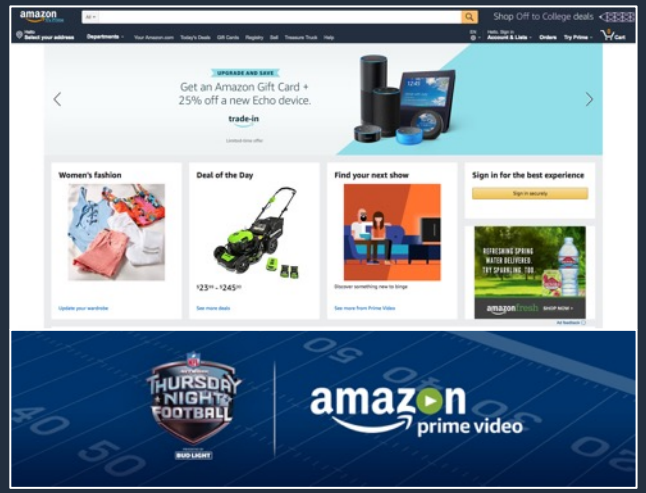

**Millisecond** variance



aws modern apps

amazon

Retail




Amazon DynamoDB supports multiple high-traffic sites and systems including Alexa, the Amazon.com sites, and 442 Amazon fulfillment centers. **Across the 66-hour 2020 Prime Day, these sources made 16.4 trillion calls to the DynamoDB API, peaking at 80.1 million requests per second.**

The internal Amazon.com Herd system supports 100s of millions of active workflows.

## Migrated from Oracle to DynamoDB

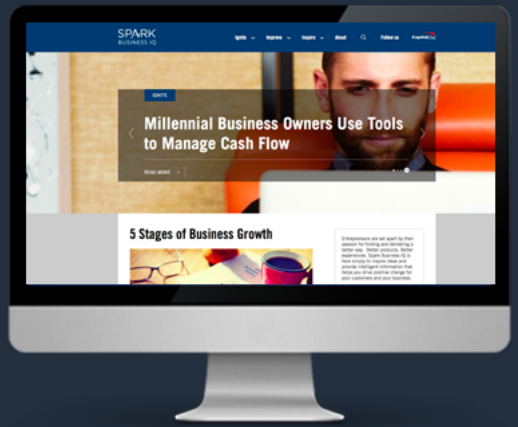
- **Improved customer experience:** Workflow processing delays dropped from 1 second to 100 milliseconds.
- **Reduced cost:** Scaling and maintenance effort dropped 10 times.
- **Reduced complexity and risk:** Retired more than 300 Oracle hosts.



aws modern apps

Capital One

Banking and finance





"The new solution is so much faster...with an average response time of 55 ms."

**—Srini Uppalapati**
Capital One

Capital One completes migration in 2020 from data centers to AWS, becomes first US bank to go all in on the cloud

## Migrated from mainframe to DynamoDB:

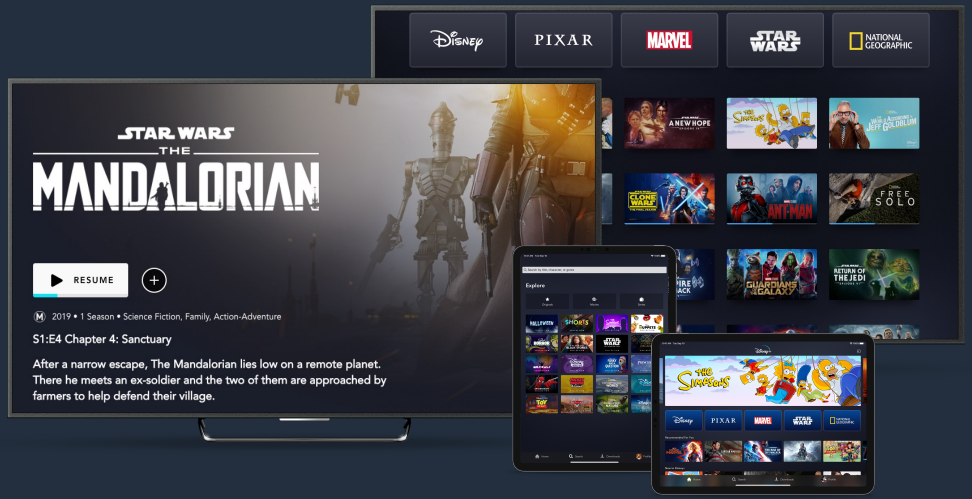
- Previously all apps were served by a single mainframe sitting in the middle of their physical business
- Product teams busy coming up with new mobile products for customers were often blocked by the mainframe
- DynamoDB and microservices give app developers unbounded scale, nimbleness, and the ability to roll out all new services



aws modern apps

Disney+

Media and entertainment




"Billions of bookmarks ingested a day over Amazon Kinesis and into Amazon DynamoDB"

**—Attilio Giue**
Director of Content Discovery, Disney+

Disney+ launched in November 2019 and delivers its extensive library of digital content directly to the homes of over 60.5 million subscribers, and DynamoDB is one of the technologies that supports this global footprint.

## Disney+ chose DynamoDB to help with:

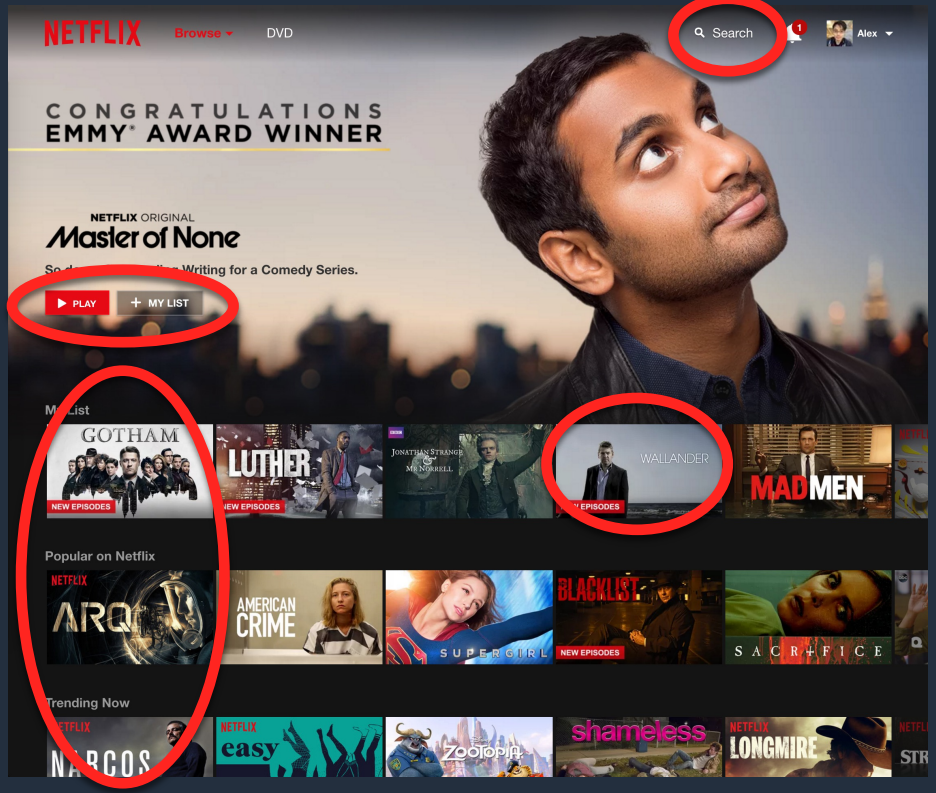
- Utilizing multi-Region replication with single-digit latency to shift traffic without experiencing data issues
- Adding another AWS Region in global tables to launch into new countries, providing low latency
- Scaling Recommendations and Bookmarks with little to no operational overhead
- Having the ability to switch back and forth between on-demand and provisioned capacity modes when entering new Regions



aws modern apps

# NETFLIX

Media and entertainment




## Netflix uses DynamoDB as a massive-scale metadata store for A/B testing

Netflix selected DynamoDB for operational resiliency:

- They operate their DynamoDB metadata store across multiple AWS Regions

Chose DynamoDB to handle Netflix scale:

- Netflix runs hundreds of A/B tests at any given time and DynamoDB can handle that scaling up and down
- Across millions of user accounts

Selected DynamoDB for performance:

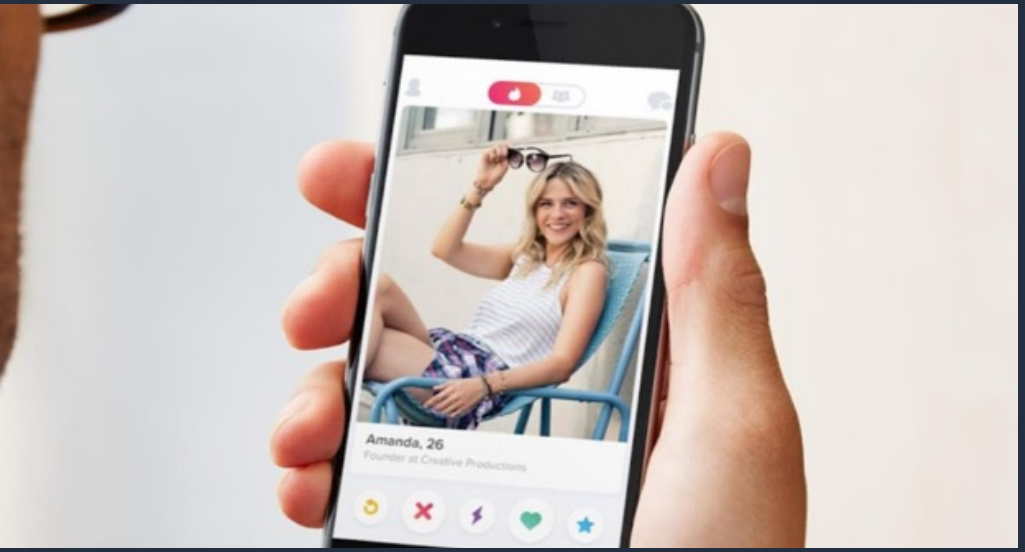
- Netflix requires low latency across millions of JSON documents



aws modern apps

tinder™

Software and internet




"DynamoDB helps us achieve greater developer efficiency…at a lower cost."

**Jun-young Kwak**
Tinder

Profile, match, and swipe data

25 TB data, 20 billion matches, 190 countries

Migrated from MongoDB to DynamoDB

- 60 percent cost savings



aws modern apps

The Pokémon Company INTERNATIONAL

Gaming

"Using the built-in TTL settings in DynamoDB, we can track when a user exceeds the maximum login attempt threshold and deny entry. The result has been a 90 percent reduction in bot login attempts, which frees up system resources for legitimate users and reduces our need to overscale"

**– Jeff Webb**
The Pokémon Company

The Pokémon Company migrated to AWS purpose-built databases.

In the process, they migrated global configuration and Time to Live (TTL) data to DynamoDB.

Their monthly database cost has dropped by tens of thousands of dollars.

They had 68 hours of downtime or performance degradation in the six months before migration. After? Zero.



aws modern apps

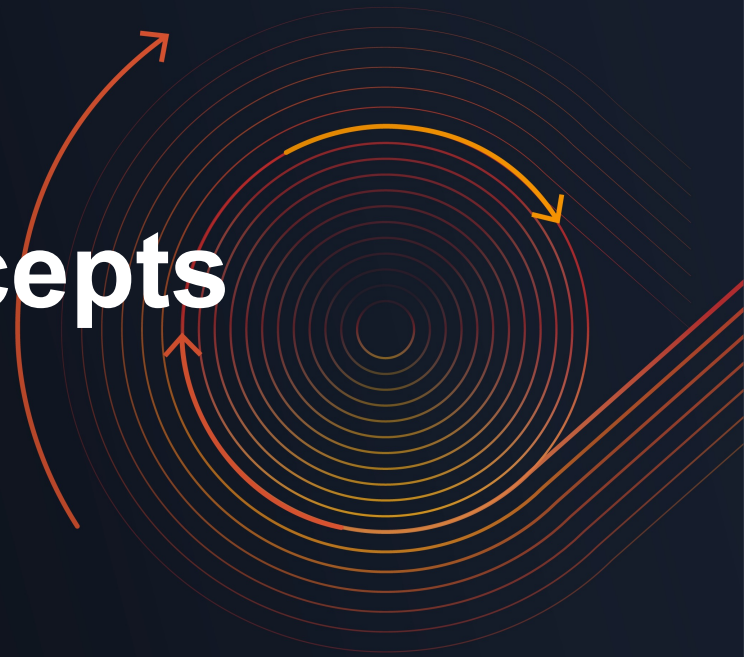
# Hundreds of thousands of customers have chosen DynamoDB




aws modern apps

# DynamoDB Key Concepts

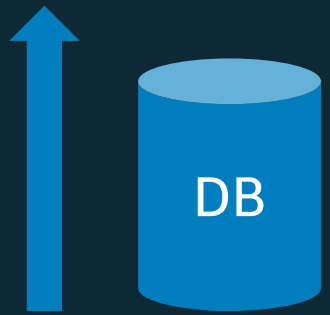
Quick Review



aws modern apps

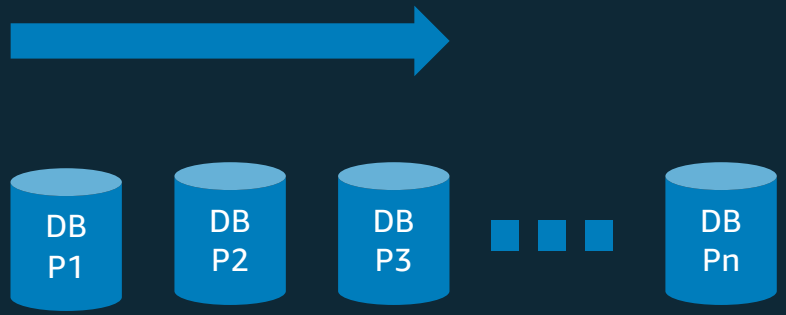
# Scaling databases

## Traditional SQL




Scale up

## NoSQL




Scale out to many shards

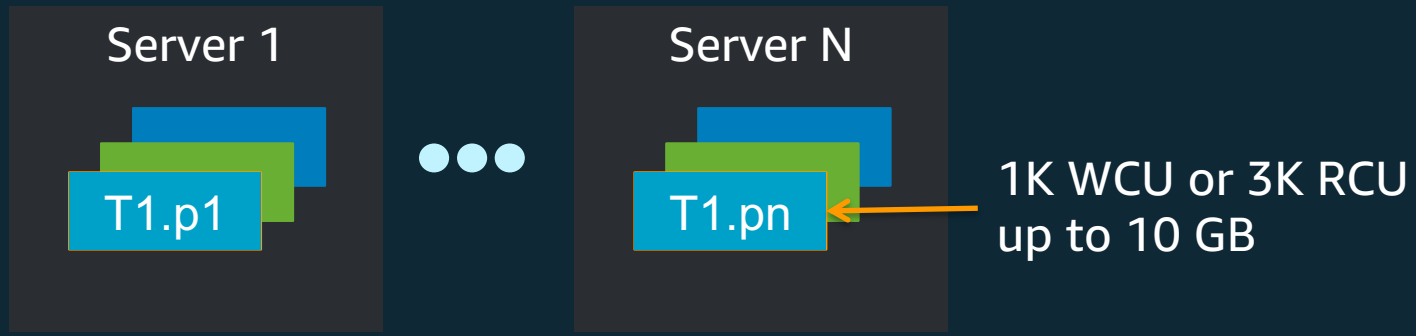
Basic premise: There is a way to design data that's horizontally scalable.



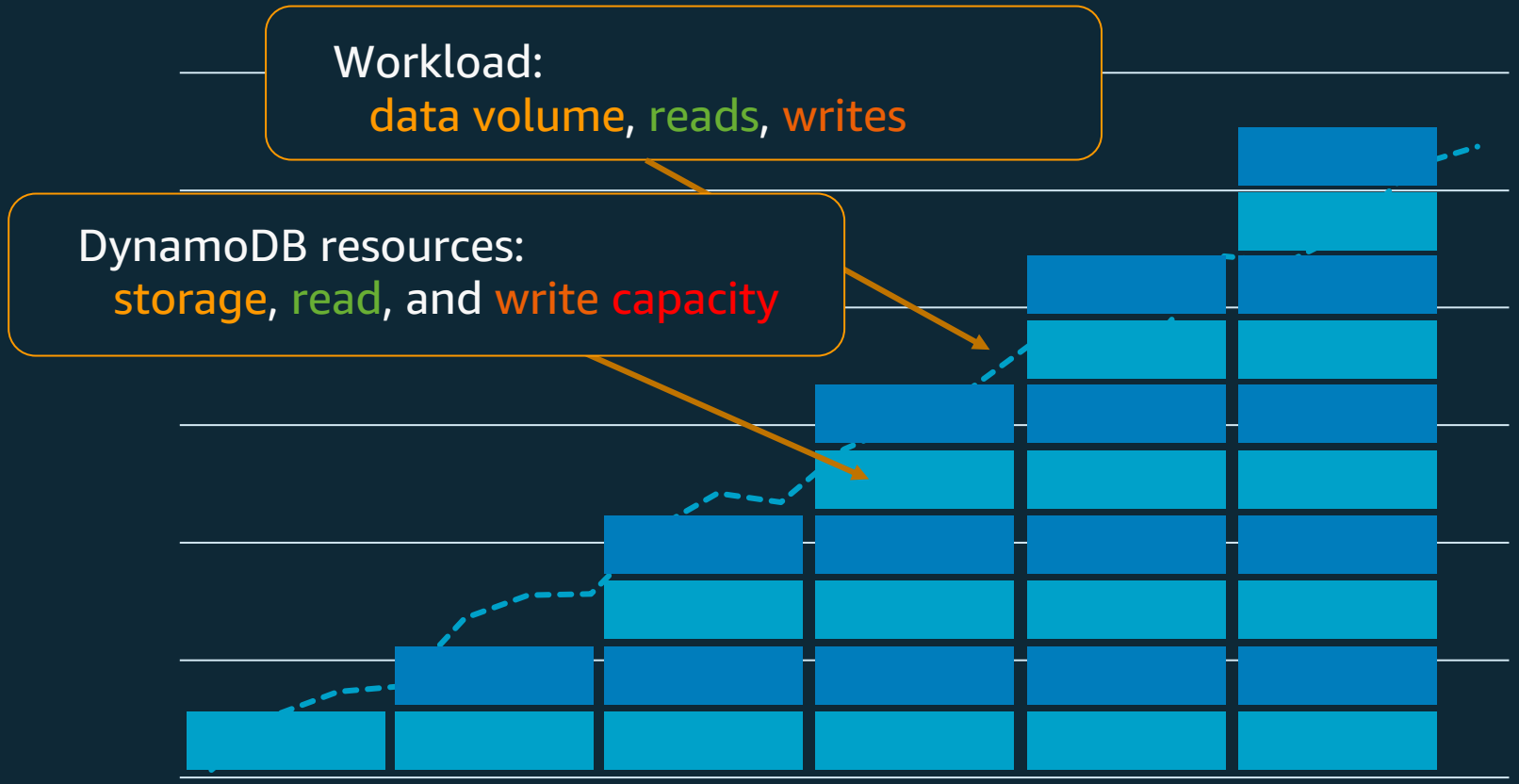
aws

# You work with tables…

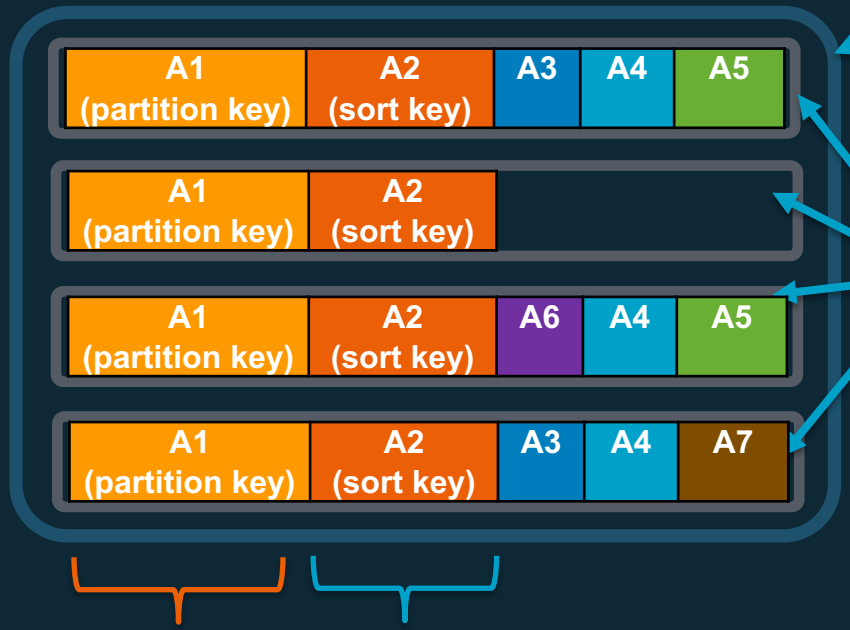

# DynamoDB does the rest under the hood…



# Horizontal scaling with DynamoDB



# DynamoDB Table




Table

Items

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

**Partition Key**

Mandatory
Key-value access pattern
Determines data distribution

**SortKey**

Optional
Model 1:N relationships
Enables rich query capabilities



aws

# Data types

| Data Type | DynamoDB Type |
|---|---|
| String | String |
| Integer, Float | Number |
| Timestamp | Number or String |
| Blob | Binary |
| Boolean | Bool |
| Null | Null |
| List | List |
| Set | Set of String, Number, or Binary |
| Map | Map |

# Operation types

| Data Operations |
|---|
| Query |
| Scan |
| GetItem |
| BatchGetItem |
| PutItem |
| BatchWriteItem |
| UpdateItem |
| DeleteItem |
| TransactWriteItems |
| TransactReadItems |



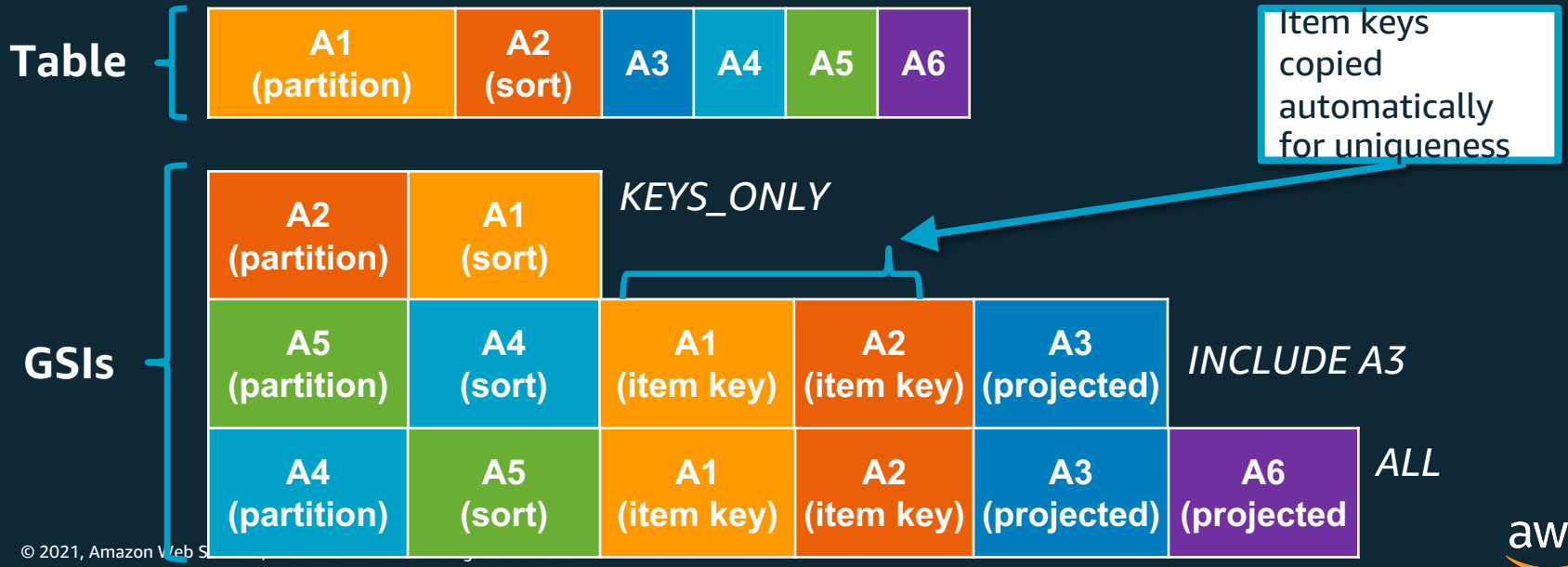
aws

# Global secondary index (GSI)

**Up to 20 GSIs per table**

**RCUs/WCUs provisioned separately for GSIs**

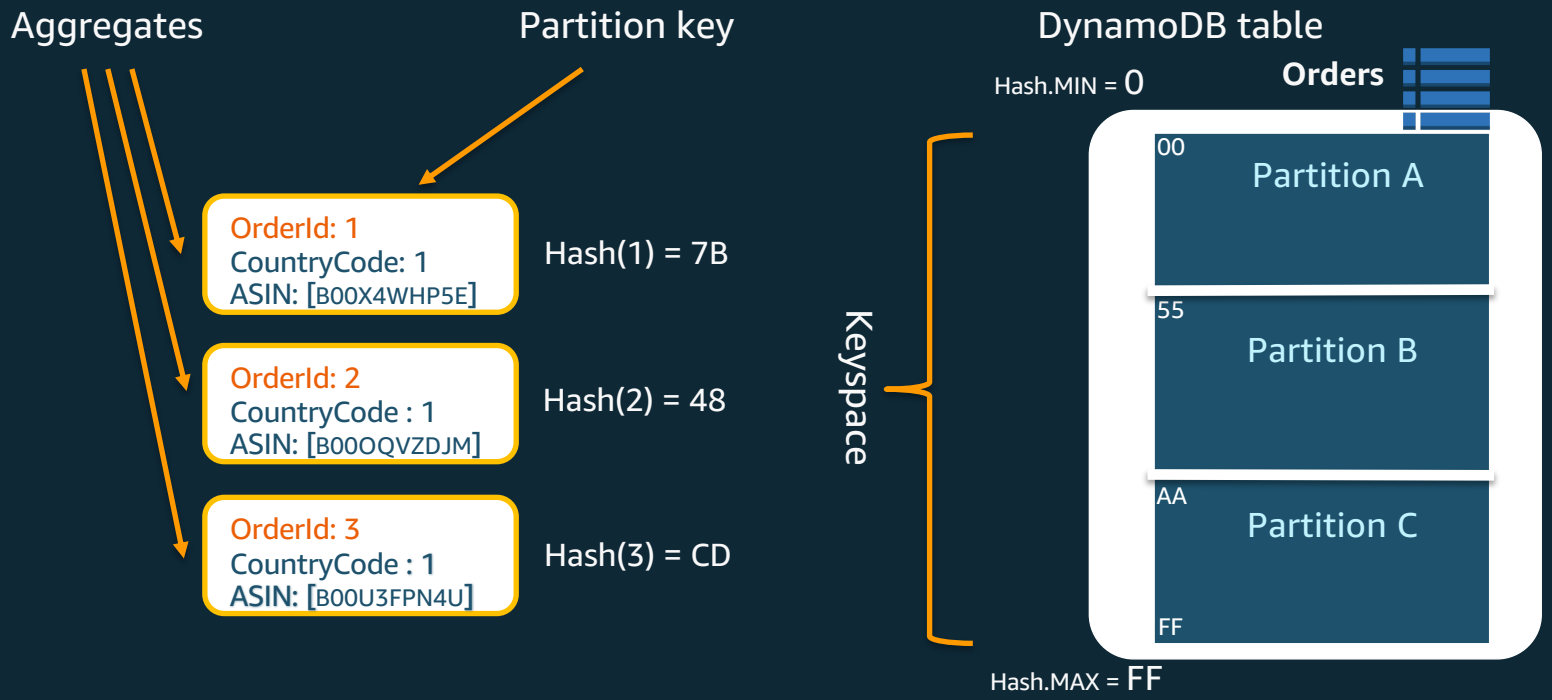
Alternate partition and/or sort key
Index is across all partition keys
Use composite sort keys for compound indexes






aws

# Item Distribution (*happy path*)

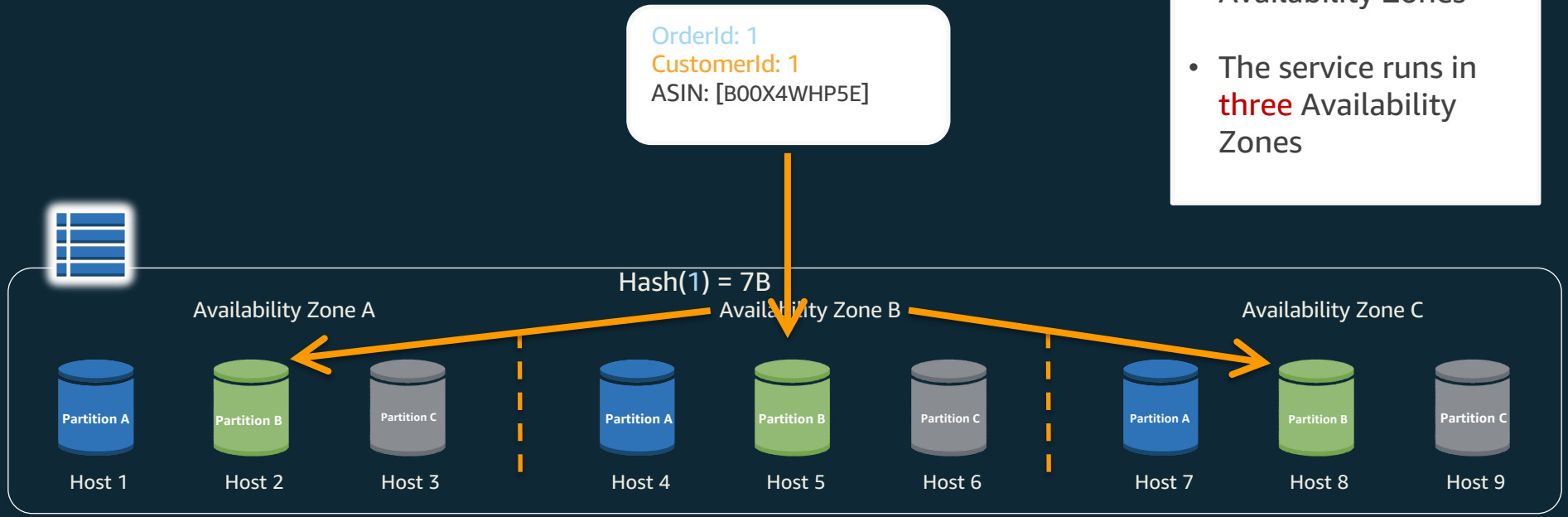



Related data (aggregate) is stored together for efficient access

# A view “from a different angle”

## Three-way replication

- Data is always replicated to three Availability Zones
- The service runs in three Availability Zones




**CustomerOrdersTable**

# Service at Scale


AVAILABILITY ZONE 1

AVAILABILITY ZONE 2

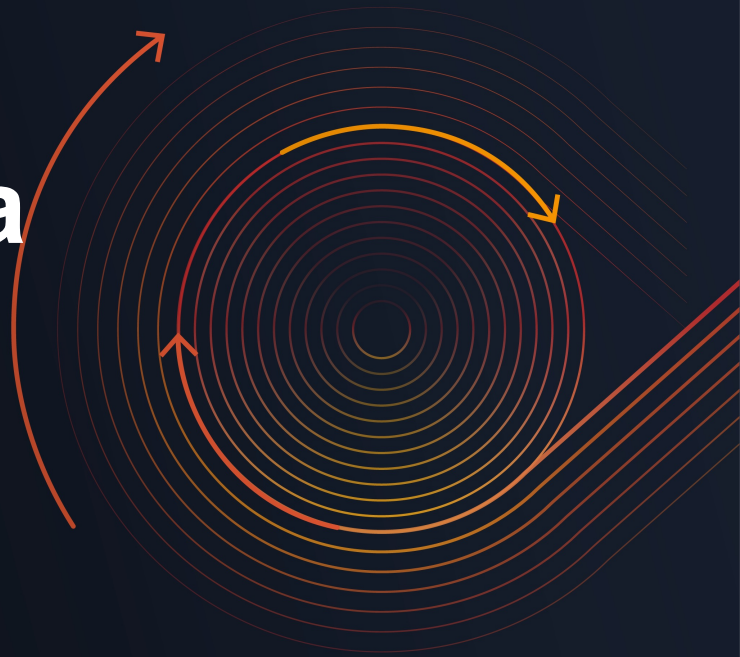
AVAILABILITY ZONE 3

RR



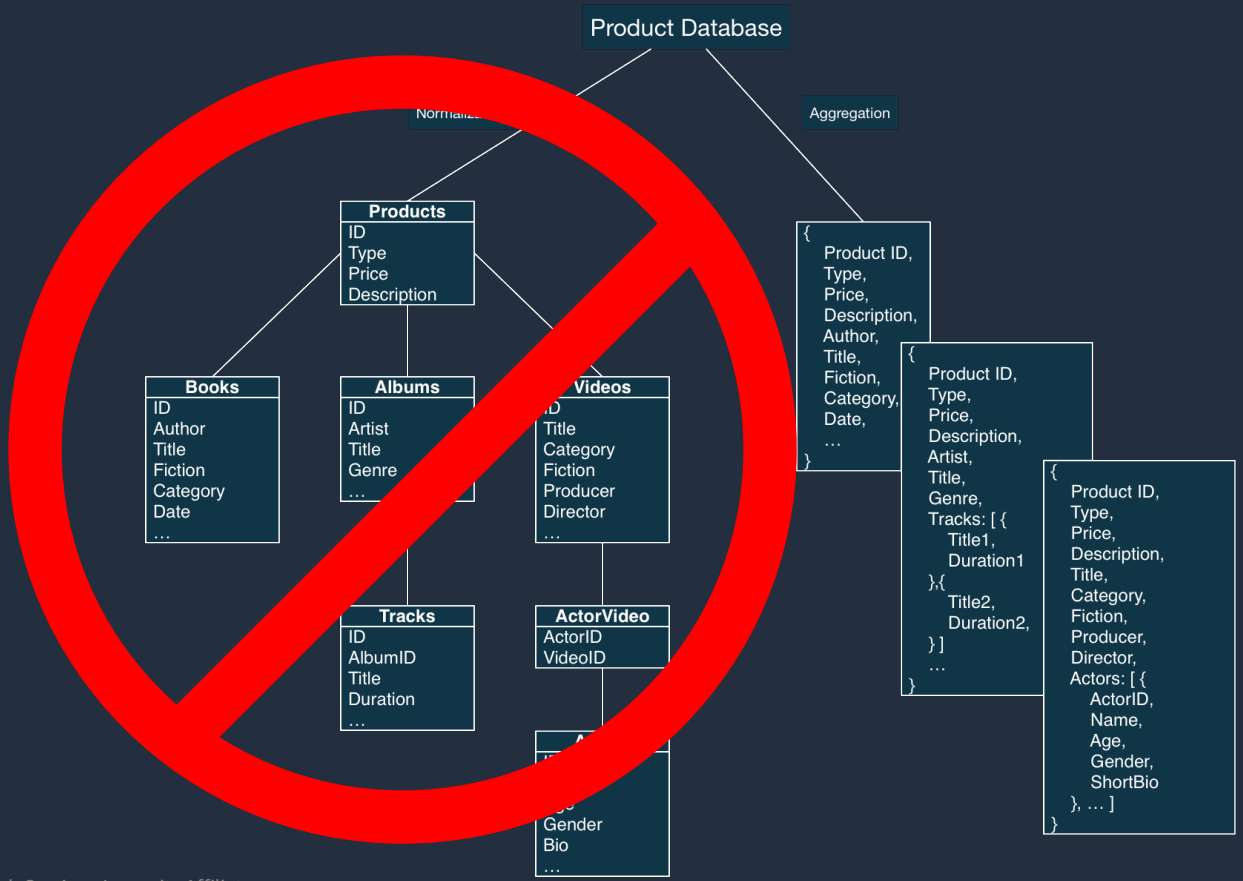
aws

# Tenets of NoSQL Data Modeling



aws modern apps

# SQL vs. NoSQL design pattern





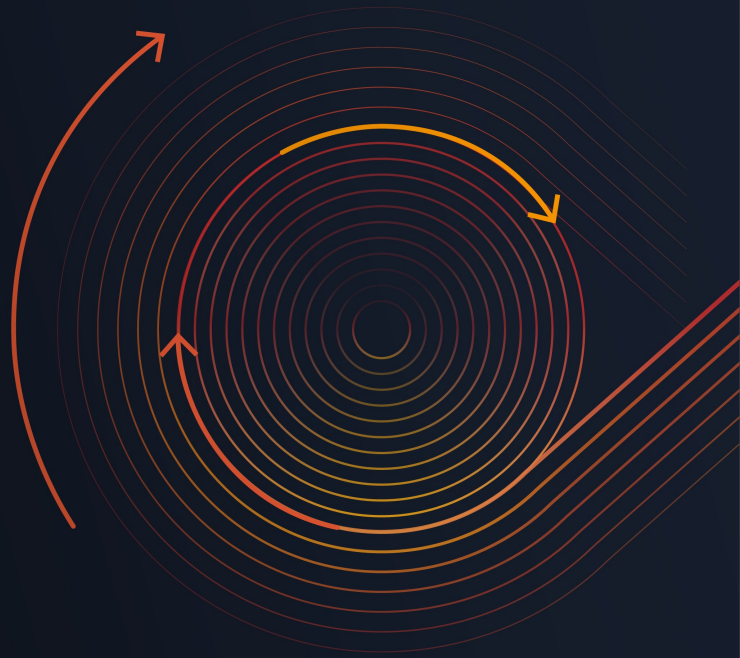

aws modern apps

# TENETS OF NoSQL DATA MODELING

- ***Define the use case***
- Identify the access patterns
  - Read/Write workloads
  - Query dimensions
  - Aggregations
- Data-modeling
  - Avoid relational design patterns
  - Start with one table
- Application type: OLTP vs OLAP
- Data Life Cycle:
  - *TTL, Backup/Archival, etc.*
- Identify Primary Keys
  - How are inserts and reads?
  - Overload items into partitions

# Building Queries

Queries: Sort Key Condition vs Filter Expressions
Composite Keys: Data Model Fundamentals



aws modern apps

# A word on queries…

```
SELECT *  FROM Table
WHERE pk = 'CartUUID'
AND sk BETWEEN '2019-08-01' AND '2019-08-07'
AND a1 < 100
ORDER BY sk DESC;
```

```
// Create the DynamoDB Client
var ddb = new AWS.DynamoDB();
var queryInput =
{
  "TableName": "queryblobs",
  "ConsistentRead": false,
  "ScanIndexForward": false,
  "KeyConditionExpression": "pk = :cTokenpk
    And sk BETWEEN :cTokensk1 AND :cTokensk2",
  "FilterExpression": "a1 < :fTokena1",
  "ExpressionAttributeValues": {
    ":cTokenpk": {"S": "CartUUID"},
    ":cTokensk1": {"S": "2019-08-01"},
    ":cTokensk2": {"S": "2019-08-07"},
    ":fTokena1": { "N": "100"}
  }
}
// Call DynamoDB's query API
ddb.query(queryInput, function(error, queryInput))
```

- Queries return all items as a single response. *Supports pagination.*
- Query require equals on partition key.
- Single key condition expression. *begins_with, between, =, <, >, <=, and >=*
- Filter Expression on attribute values. *Supports nested values. (*Applied AFTER the query runs – so it will not change the consumed RCUs. Support filtering on any non-key attribute (and nested attributes). Supports above conditions as well as IN & CONTAIN type conditions)
- REST API syntax – not SQL.



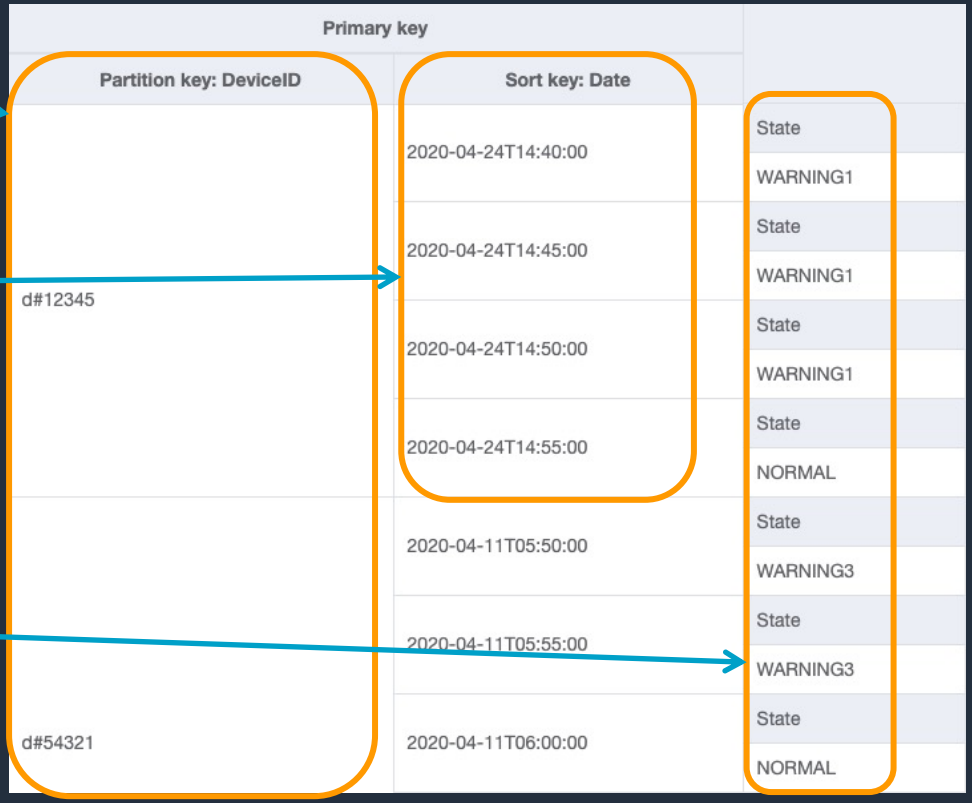
aws modern apps

# Device Logs: Filter Expressions vs Sort Key Conditions

High cardinality

Sorted by date time

Filter by Status






aws modern apps

# Access Pattern:

## Fetch all warning logs for a device that are sorted in descending order

```
SELECT * FROM DeviceLog
WHERE DeviceID = ‘d#12345'
ORDER BY Date DESC
FILTER ON State=‘WARNING1'
```

*Returned*

*Filtered*

| Primary key | | |
|---|---|---|
| Partition key: DeviceID | Sort key: Date | |
| d#12345 | 2020-04-24T14:40:00 | State |
| | | WARNING1 |
| | 2020-04-24T14:45:00 | State |
| | | WARNING1 |
| | 2020-04-24T14:50:00 | State |
| | | WARNING1 |
| | 2020-04-24T14:55:00 | State |
| | | NORMAL |
| d#54321 | 2020-04-11T05:50:00 | State |
| | | WARNING3 |
| | 2020-04-11T05:55:00 | State |
| | | WARNING3 |
| | 2020-04-11T06:00:00 | State |
| | | NORMAL |

```
aws dynamodb query
--table-name DeviceLog
--key-condition-expression "#dID = :dID"
--no-scan-index-forward
--filter-expression "#s = :s"
--expression-attribute-names ‘{“#dID": "DeviceID", “#s": “State}’
--expression-attribute-values ‘{“:dID": {“S":"d#12345"}, ":s": {“S":"WARNING1"}}’
```

# Build *Composite Sort Keys* to use *Sort Key conditions*

```
SELECT * FROM DeviceLog
WHERE DeviceID = 'd#12345'
ORDER BY Date DESC
BEGINS_WITH
    State#Date='WARNING1'
```

| Primary key | |
|---|---|
| **Partition key: DeviceID** | **Sort key: State#Date** |
| d#12345 | NORMAL#2020-04-24T14:55:00 |
| | WARNING1#2020-04-24T14:40:00 |
| | WARNING1#2020-04-24T14:45:00 |
| | WARNING1#2020-04-24T14:50:00 |
| d#54321 | NORMAL#2020-04-11T06:00:00 |
| | NORMAL#2020-04-11T09:30:00 |
| | WARNING2#2020-04-11T09:25:00 |
| | WARNING3#2020-04-11T05:50:00 |
| | WARNING3#2020-04-11T05:55:00 |

```
aws dynamodb query
--table-name DeviceLog
--no-scan-index-forward
--key-condition-expression "#dID = :dID  AND begins_with(#s, :sd)"
--expression-attribute-names '{"#cId": "DeviceID", "#s": "State#Date"}'
--expression-attribute-values '{":cId": {"S":"d#12345"}, ":sd": {"S":"WARNING1#"}}'
```



aws modern apps

# New Access Pattern:

## Fetch all device logs for a given operator between two dates

Base Table

| Primary key | | Attributes | | |
|---|---|---|---|---|
| Partition key: DeviceID | Sort key: State#Date | | | |
| d#12345 | NORMAL#2020-04-24T14:55:00 | Operator | Date | |
| | | Liz | 2020-04-24 | |
| | WARNING1#2020-04-24T14:45:00 | Operator | Date | |
| | | Liz | 2020-04-24 | |
| | WARNING1#2020-04-24T14:50:00 | Operator | Date | |
| | | Liz | 2020-04-24 | |
| d#54321 | NORMAL#2020-04-11T06:00:00 | Operator | Date | |
| | | Liz | 2020-04-11 | |
| | NORMAL#2020-04-11T09:30:00 | Operator | Date | |
| | | Sue | 2020-04-11 | |
| | WARNING2#2020-04-11T09:25:00 | Operator | Date | |
| | | Sue | 2020-04-11 | |
| | WARNING3#2020-04-11T05:55:00 | Operator | Date | |
| | | Liz | 2020-04-11 | |
| d#11223 | WARNING4#2020-04-27T16:10:00 | Operator | Date | |
| | | Sue | 2020-04-27 | |
| | WARNING4#2020-04-27T16:15:00 | Operator | Date | EscalatedTo |
| | | Sue | 2020-04-27 | Sara |

GSI-Operator

| Primary key | | Attributes | |
|---|---|---|---|
| Partition key: Operator | Sort key: Date | | |
| Liz | 2020-04-11 | State#Date | DeviceID |
| | | WARNING3#2020-04-11T05:55:00 | d#54321 |
| | 2020-04-11 | State#Date | DeviceID |
| | | NORMAL#2020-04-11T06:00:00 | d#54321 |
| | 2020-04-24 | State#Date | DeviceID |
| | | WARNING1#2020-04-24T14:45:00 | d#12345 |
| | 2020-04-24 | State#Date | DeviceID |
| | | WARNING1#2020-04-24T14:50:00 | d#12345 |
| | 2020-04-24 | State#Date | DeviceID |
| | | NORMAL#2020-04-24T14:55:00 | d#12345 |
| Sue | 2020-04-11 | State#Date | DeviceID |
| | | WARNING2#2020-04-11T09:25:00 | d#54321 |
| | 2020-04-11 | State#Date | DeviceID |
| | | NORMAL#2020-04-11T09:30:00 | d#54321 |
| | 2020-04-27 | State#Date | DeviceID |
| | | WARNING4#2020-04-27T16:10:00 | d#11223 |
| | 2020-04-27 | State#Date | DeviceID |
| | | WARNING4#2020-04-27T16:15:00 | d#11223 |



aws modern apps

# Access Pattern: logs for a given operator between two dates

```
SELECT * FROM GSI–Operator
WHERE Operator = 'Liz'
ORDER BY Date DESC
BETWEEN
    Date='2020-04-20' AND
          '2020-04-25'
```

| Primary key | | Attributes | |
|---|---|---|---|
| Partition key: Operator | Sort key: Date | | |
| | 2020-04-11 | State#Date | DeviceID |
| | | WARNING3#2020-04-11T05:55:00 | d#54321 |
| | 2020-04-11 | State#Date | DeviceID |
| | | NORMAL#2020-04-11T06:00:00 | d#54321 |
| Liz | 2020-04-24 | State#Date | DeviceID |
| | | WARNING1#2020-04-24T14:45:00 | d#12345 |
| | 2020-04-24 | State#Date | DeviceID |
| | | WARNING1#2020-04-24T14:50:00 | d#12345 |
| | 2020-04-24 | State#Date | DeviceID |
| | | NORMAL#2020-04-24T14:55:00 | d#12345 |
| | 2020-04-11 | State#Date | DeviceID |
| | | WARNING2#2020-04-11T09:25:00 | d#54321 |
| | 2020-04-11 | State#Date | DeviceID |
| | | NORMAL#2020-04-11T09:30:00 | d#54321 |
| Sue | 2020-04-27 | State#Date | DeviceID |
| | | WARNING4#2020-04-27T16:10:00 | d#11223 |
| | 2020-04-27 | State#Date | DeviceID |
| | | WARNING4#2020-04-27T16:15:00 | d#11223 |

GSI – Operator

```
aws dynamodb query
--table-name DeviceLog
--index-name GSI–Operator
--key-condition-expression "#op = :op AND #d  between :d1 AND :d2"
--expression-attribute-names '{"#op": "Operator" , "#d": "Date"}'
--expression-attribute-values '{":op": {"S":"Liz"} , ":d1": {"S":"2020-04-20"}, ":d2":{"S":"2020-04-25"}}'
```

# Access Pattern:

## Fetch all escalated device logs for a given *supervisor*

Base Table

| Primary key | | Attributes | |
|---|---|---|---|
| Partition key: DeviceID | Sort key: State#Date | | |
| d#12345 | NORMAL#2020-04-24T14:55:00 | Operator | Date |
| | | Liz | 2020-04-24 |
| | WARNING1#2020-04-24T14:45:00 | Operator | Date |
| | | Liz | 2020-04-24 |
| | WARNING1#2020-04-24T14:50:00 | Operator | Date |
| | | Liz | 2020-04-24 |
| d#54321 | NORMAL#2020-04-11T06:00:00 | Operator | Date |
| | | Liz | 2020-04-11 |
| | NORMAL#2020-04-11T09:30:00 | Operator | Date |
| | | Sue | 2020-04-11 |
| | WARNING2#2020-04-11T09:25:00 | Operator | Date |
| | | Sue | 2020-04-11 |
| | WARNING3#2020-04-11T05:55:00 | Operator | Date |
| | | Liz | 2020-04-11 |
| d#11223 | WARNING4#2020-04-27T16:10:00 | Operator | Date |
| | | Sue | 2020-04-27 |
| | WARNING4#2020-04-27T16:15:00 | Operator | Date |
| | | Sue | 2020-04-27 |

GSI-Supervisor

| Primary key | | Attributes | |
|---|---|---|---|
| Partition key: EscalatedTo | Sort key: State#Date | | |
| Sara | WARNING4#2020-04-27T16:15:00 | DeviceID | Operator |
| | | d#11223 | Sue |

| EscalatedTo |
|---|
| Sara |

Sparse GSI: Only items that match the GSI index are projected.

Good for:

- ‘Needle in the haystack’
- Cost effective ‘scans’
- Item management



aws modern apps

# How to query nested JSON

Vertical Partitioning
Overloaded GSIs
Future resilient data models

aws modern apps

```
{
        ”UserProfile" : {
                "FirstName": "Paul",
                "LastName": "Atreides",
                "DateJoined":"1965-08-01"},
        ”Store" : {
                ”store_id": ”STOREUID",
                 "city": "Los Angeles",
                 "zip_code":"90029”}
        "ShoppingCart" : [
                {"Spice":
                        { "SKU": "SpiceSKU",
                          "CategoryID":"FictionalSpice",
                          "DateAddded " :"2019-06-11"}},
                {"EspressoBeans":
                        { "SKU": "CaffeineSKU",
                          "CategoryID":"FOODANDDRINK",
                          "DateAddded " :"2019-06-10"}}],
        "ShippingAddress" : {
                 "street_address": "1234 Arrakis Dr",
                 "city": "Los Angeles",
                 "zip_code":"90029",
                 "status": "default"}
        ”OrderHistory#OrderUID" : {
                ”ProductA": ”SKU_A",
                ”ProductB": ”SKU_B",
                "DateOrdered":”2018-09-28"}
}
```

# Document Indexing

*Shopping Cart Document Example*

aws modern apps

# Vertical Partitioning

Split large items into logical blocks. Build a hierarchy on the SortKey by using the nested values.

Benefits:

- Enables direct queries on values with hierarchical complexity.
- Optimize reads & writes by breaking up large items
- Smaller items improve overall application performance
- Open-ended index by overloading a GSI = Future Resilient!

# Vertical Partitioning

```
SELECT * FROM CartTable
WHERE PK ='UserID'
  AND SK BEGINS_WITH 'Cart#ACTIVE'
```

Selectively query 'nested' attributes

| Primary key | | | |
|---|---|---|---|
| Partition key: PK | Sort key: SK | | |
| UserID | Address#USA#CA#LA#90029 | | GSI-SK |
| | | "Street Address" | Default |
| | Cart#ACTIVE#Coffee | data | GSI-SK |
| | | CoffeeSKU | 2019-11-27T103324 |
| | Cart#ACTIVE#Spice | data | GSI-SK |
| | | SpiceSKU | 2019-11-28T091245 |
| | Cart#SAVED#Cocoa | data | GSI-SK |
| | | CocoaSKU | 2019-11-28T125642 |
| | OrderHistory#OrderUID | data | GSI-SK |
| | | {Order:DataMap} | 2019-10-08T132612 |
| | ProfileName | data | |
| | | "Paul Atreides" | |
| | Store#StoreUID | data | GSI-SK |
| | | Los Angeles | Active |

- Optimize object size
- Selective Queries
- Reduce capacity and cost
- Improve App performance
- Overload GSI on SK…



aws modern apps

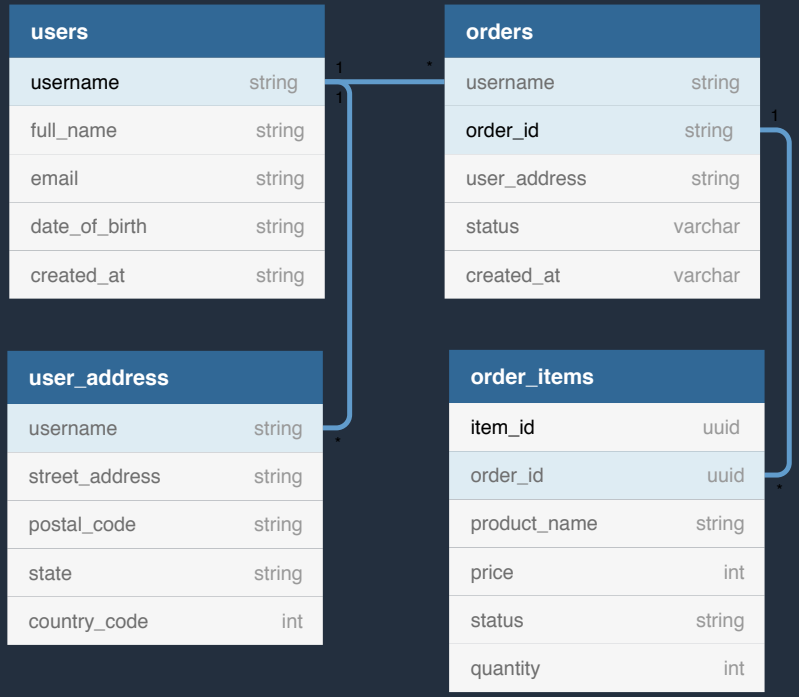
# Overloaded GSI

| Primary key | | | |
|---|---|---|---|
| Partition key: PK | Sort key: SK | | |
| UserID | Address#USA#CA#LA#90029 | data | |
| | | "Street Addr | |
| | Cart#ACTIVE#Coffee | data | GSI-SK |
| | Cart#ACTIVE#Spice | | |
| | Cart#SAVED#Cocoa | | |
| | OrderHistory#OrderUID | | |
| | ProfileName | | |
| | Store#StoreUID | | |

- Index unbounded data: Arrays or Maps
- Future Resilient: add new types!!

| Primary key | | Attributes | |
|---|---|---|---|
| Partition key: SK | Sort key: GSI-SK | | |
| Cart#ACTIVE#Spice | 2019-11-28T091245 | PK | data |
| | | UserID | SpiceSKU |
| Cart#ACTIVE#Coffee | 2019-11-27T103324 | PK | data |
| | | UserID | CoffeeSKU |
| Cart#SAVED#Cocoa | 2019-11-28T125642 | PK | data |
| | | UserID | CocoaSKU |
| Address#USA#CA#LA#90029 | Default | PK | data |
| | | UserID | "Street Address" |
| Store#StoreUID | Active | PK | data |
| | | UserID | Los Angeles |
| OrderHistory#OrderUID | 2019-10-08T132612 | PK | data |
| | | UserID | {Order:DataMap} |

# Single Table Design: Using a E-commerce Model as example






aws modern apps

# Identify data access patterns

1. Get user profile
2. Get orders for user
3. Get single order and order items
4. Get orders for user by status
5. Get open orders

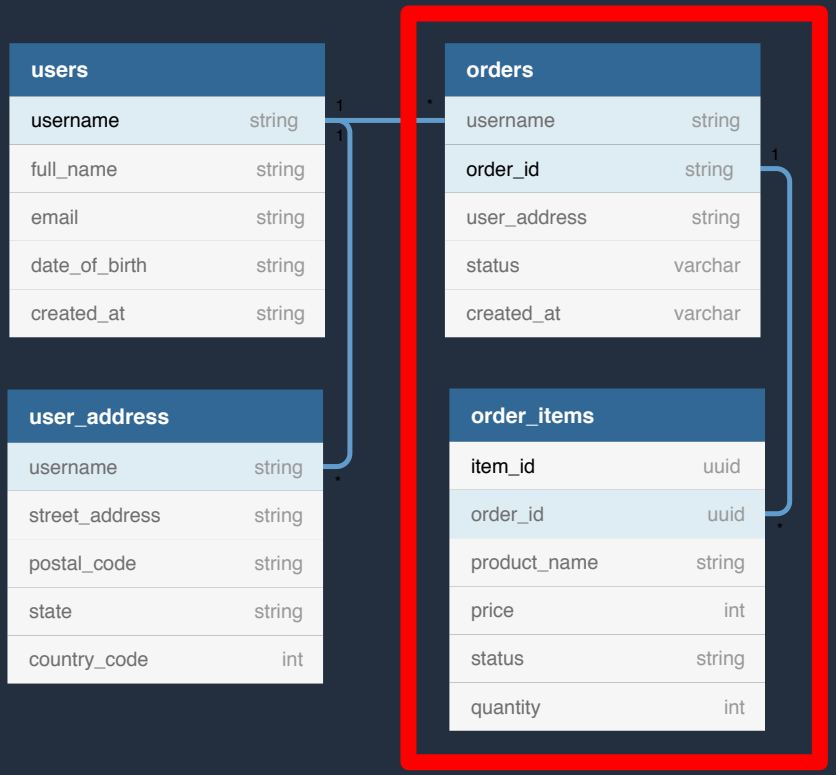
| Primary Key | | Attributes | | | |
|---|---|---|---|---|---|
| PK | SK | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 |

| Primary Key | | Attributes | | | |
|---|---|---|---|---|---|
| PK | SK | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 |

| Primary Key | | Attributes | | | |
|---|---|---|---|---|---|
| PK | SK | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | | |
| Order | | |
| Order Item | | |





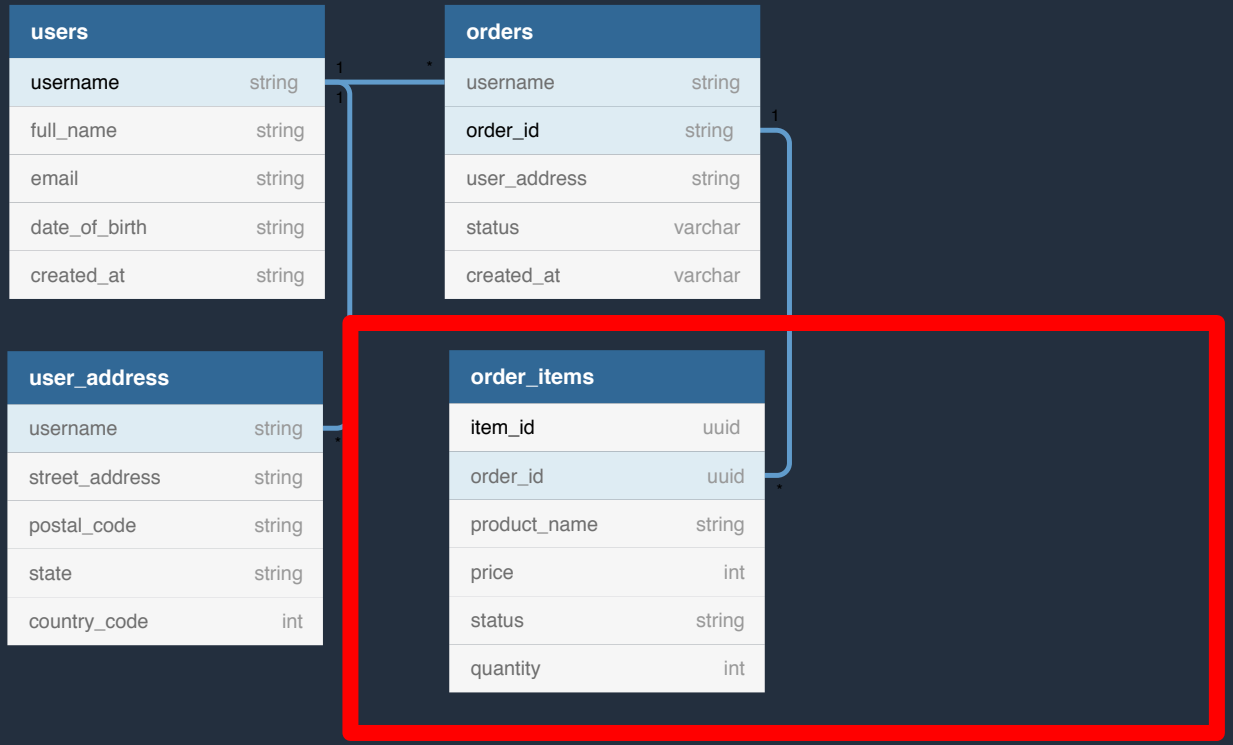

aws modern apps

# Denormalization + document types

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| **PK** | **SK** | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| USER#alexdebrie | #PROFILE#alexdebrie | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | { "Home": { "StreetAddress": "1111 1st St", "State": "Nebraska", "Country": "USA" } } |
| USER#nedstark | #PROFILE#nedstark | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | { "Home": { "StreetAddress": "1234 2nd Ave", "City": "Winterfell", "Country": "Westeros" }, "Business": { "StreetAddress": "Suite 200, Red Keep", "City": "King's Landing", "Country": "Westeros" } } |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | N/A | N/A |
| Order | | |
| Order Item | | |






aws modern apps

# One-to-many: Sort keys

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| PK | SK | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt | Addresses |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#42ef295e | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#2e7abecc | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt | Addresses |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| | ORDER#f4f80a91 | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | f4f80a91 | PLACED | 2019-05-12 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | N/A | N/A |
| Order | USER#<username> | ORDER#<orderId> |
| Order Item | | |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | N/A | N/A |
| Order | USER#<username> | ORDER#<orderId> |
| Order Item | | |

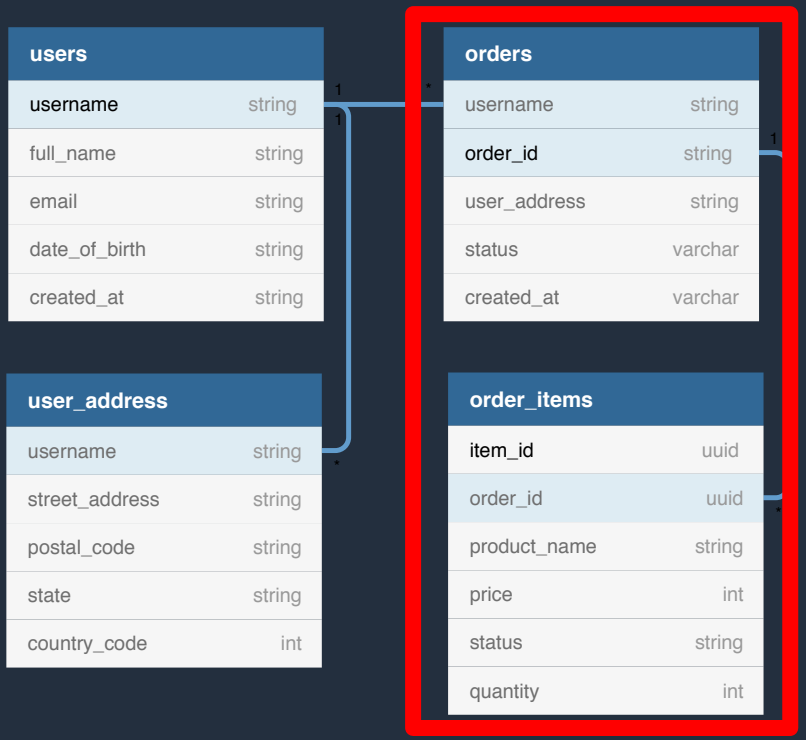


aws modern apps

# One-to-many: Sort keys

“PK = USER#alexdebrie AND BEGINS_WITH(SK, ‘ORDER#’)”

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| PK | SK | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt | Addresses |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#42ef295e | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#2e7abecc | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt | Addresses |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| | ORDER#f4f80a91 | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | f4f80a91 | PLACED | 2019-05-12 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |






aws modern apps

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| PK | SK | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt | Addresses |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#42ef295e | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#2e7abecc | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt | Addresses |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| ITEM#ab070628 | ORDER#5e7272b7 | ItemId | OrderId | ProductName | Price | Status |
| | | ab070628 | 5e7272b7 | Amazon Echo | 59.99 | FILLED |
| ITEM#4cc734ec | ORDER#5e7272b7 | ItemId | OrderId | ProductName | Price | Status |
| | | 4cc734ec | 5e7272b7 | Macbook Pro | 1399.99 | FILLED |
| ITEM#e16f4906 | ORDER#2eae1dee | ItemId | OrderId | ProductName | Price | Status |
| | | e16f4906 | 2eae1dee | Duct tape | 5.99 | FILLED |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | N/A | N/A |
| Order | USER#<username> | ORDER#<orderId> |
| Order Item | ITEM#<itemId> | ORDER#<orderId> |

# Entities

| | PK | SK |
|---|---|---|
| User | USER#<username> | #PROFILE#<username> |
| User Address | N/A | N/A |
| Order | USER#<username> | ORDER#<orderId> |
| Order Item | ITEM#<itemId> | ORDER#<orderId> |

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| **PK** | **SK** | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#42ef295e | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#2e7abecc | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| USER#nedstark | #PROFILE#nedstark | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| ITEM#ab070628 | ORDER#5e7272b7 | **ItemId** | **OrderId** | **ProductName** | **Price** | **Status** |
| | | ab070628 | 5e7272b7 | Amazon Echo | 59.99 | FILLED |
| ITEM#4cc734ec | ORDER#5e7272b7 | **ItemId** | **OrderId** | **ProductName** | **Price** | **Status** |
| | | 4cc734ec | 5e7272b7 | Macbook Pro | 1399.99 | FILLED |
| ITEM#e16f4906 | ORDER#2eae1dee | **ItemId** | **OrderId** | **ProductName** | **Price** | **Status** |
| | | e16f4906 | 2eae1dee | Duct tape | 5.99 | FILLED |

# Inverted index

| GSI1 (Inverted Index) | |
|---|---|
| SK | PK |
| #PROFILE#alexdebrie | USER#alexdebrie |
| #PROFILE#nedstark | USER#nedstark |
| ORDER#2e7abecc | USER#alexdebrie |
| ORDER#2eae1dee | USER#nedstark |
| | ITEM#e16f4906 |
| ORDER#42ef295e | USER#alexdebrie |
| ORDER#5e7272b7 | USER#alexdebrie |
| | ITEM#ab070628 |
| | ITEM#4cc734ec |

| Attributes | | | | |
|---|---|---|---|---|
| Username | FullName | Email | CreatedAt | Addresses |
| alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| Username | FullName | Email | CreatedAt | Addresses |
| nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| Username | OrderId | Status | CreatedAt | Address |
| alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| Username | OrderId | Status | CreatedAt | Address |
| nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| ItemId | OrderId | ProductName | Price | Status |
| e16f4906 | 2eae1dee | Duct tape | 5.99 | FILLED |
| Username | OrderId | Status | CreatedAt | Address |
| alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| Username | OrderId | Status | CreatedAt | Address |
| alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| ItemId | OrderId | ProductName | Price | Status |
| ab070628 | 5e7272b7 | Amazon Echo | 59.99 | FILLED |
| ItemId | OrderId | ProductName | Price | Status |
| 4cc734ec | 5e7272b7 | Macbook Pro | 1399.99 | FILLED |

# Inverted index

| GSI1 (Inverted Index) | | Attributes | | | | |
|---|---|---|---|---|---|---|
| SK | PK | | | | | |
| #PROFILE#alexdebrie | USER#alexdebrie | Username | FullName | Email | CreatedAt | Addresses |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| #PROFILE#nedstark | USER#nedstark | Username | FullName | Email | CreatedAt | Addresses |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| ORDER#2e7abecc | USER#alexdebrie | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| ORDER#2eae1dee | USER#nedstark | Username | OrderId | Status | CreatedAt | Address |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| | ITEM#e16f4906 | ItemId | OrderId | ProductName | Price | Status |
| | | e16f4906 | 2eae1dee | Duct tape | 5.99 | FILLED |
| ORDER#42ef295e | USER#alexdebrie | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| ORDER#5e7272b7 | USER#alexdebrie | Username | OrderId | Status | CreatedAt | Address |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | {"StreetAddress":"1111 1st St","State":"Nebraska","Co |
| | ITEM#ab070628 | ItemId | OrderId | ProductName | Price | Status |
| | | ab070628 | 5e7272b7 | Amazon Echo | 59.99 | FILLED |
| | ITEM#4cc734ec | ItemId | OrderId | ProductName | Price | Status |
| | | 4cc734ec | 5e7272b7 | Macbook Pro | 1399.99 | FILLED |

# Filtering access pattern

| Primary Key | | Attributes | | | |
|---|---|---|---|---|---|
| **PK** | **SK** | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | **Username** | **FullName** | **Email** | **CreatedAt** |
| | | alexdebrie | Alex DeBrie | alexdebrie1@x | 2018-03-23 |
| | ORDER#5e7272b7 | **Username** | **OrderId** | **Status** | **CreatedAt** |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 |
| | ORDER#42ef295e | **Username** | **OrderId** | **Status** | **CreatedAt** |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 |
| | ORDER#2e7abecc | **Username** | **OrderId** | **Status** | **CreatedAt** |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 |
| USER#nedstark | #PROFILE#nedstark | **Username** | **FullName** | **Email** | **CreatedAt** |
| | | nedstark | Eddard Stark | lord@winterfel | 2016-02-27 |
| | ORDER#2eae1dee | **Username** | **OrderId** | **Status** | **CreatedAt** |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 |

# Filtering access pattern

| Status | CreatedAt | | OrderStatusDate |
|---|---|---|---|
| PLACED | 2019-04-21 | → | PLACED#2019-04-21 |
| **Status** | **CreatedAt** | | **OrderStatusDate** |
| PLACED | 2019-04-25 | → | PLACED#2019-04-25 |
| **Status** | **CreatedAt** | | **OrderStatusDate** |
| SHIPPED | 2018-12-25 | → | SHIPPED#2018-12-25 |

# Filtering access pattern

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| PK | SK | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | Username | FullName | Email | CreatedAt | Addresses |
| | | alexdebrie | Alex DeBrie | alexdebrie1@g | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | PLACED#2019-04-21 |
| | ORDER#42ef295e | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | PLACED#2019-04-25 |
| | ORDER#2e7abecc | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | SHIPPED#2018-12-25 |
| USER#nedstark | #PROFILE#nedstark | Username | FullName | Email | CreatedAt | Addresses |
| | | nedstark | Eddard Stark | lord@winterfel | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | SHIPPED#2019-01-15 |

# Filtering patterns: Composite sort key

| GS2 (OrderStatusDate) | |
|---|---|
| PK | OrderStatusDate |
| USER#alexdebrie | PLACED#2019-04-21 |
| | PLACED#2019-04-25 |
| | SHIPPED#2018-12-25 |
| USER#nedstark | SHIPPED#2019-01-15 |

| Attributes | | | | |
|---|---|---|---|---|
| Username | OrderId | Status | CreatedAt | OrderStatusDate |
| lexdebrie | 5e7272b7 | PLACED | 2019-04-21 | PLACED#2019-04-21 |
| Username | OrderId | Status | CreatedAt | OrderStatusDate |
| lexdebrie | 42ef295e | PLACED | 2019-04-25 | PLACED#2019-04-25 |
| Username | OrderId | Status | CreatedAt | OrderStatusDate |
| lexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | SHIPPED#2018-12-25 |
| Username | OrderId | Status | CreatedAt | OrderStatusDate |
| nedstark | 2eae1dee | SHIPPED | 2019-01-15 | SHIPPED#2019-01-15 |

# Filtering patterns: Composite sort key

“PK = USER#alexdebrie AND BEGINS_WITH(OrderStatusDate, ‘SHIPPED#’)”

| GS2 (OrderStatusDate) | | Attributes | | | | |
|---|---|---|---|---|---|---|
| PK | OrderStatusDate | | | | | |
| USER#alexdebrie | PLACED#2019-04-21 | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | PLACED#2019-04-21 |
| | PLACED#2019-04-25 | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-25 | PLACED#2019-04-25 |
| | SHIPPED#2018-12-25 | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | SHIPPED#2018-12-25 |
| USER#nedstark | SHIPPED#2019-01-15 | Username | OrderId | Status | CreatedAt | OrderStatusDate |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | SHIPPED#2019-01-15 |

# Filtering access patterns: Sparse index

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| **PK** | **SK** | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 02/22/2018 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 5e7272b7 | PLACED | 04/21/2019 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#42ef295e | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 42ef295e | PLACED | 04/25/2019 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| | ORDER#2e7abecc | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | alexdebrie | 2e7abecc | SHIPPED | 12/25/2018 | {"StreetAddress":"1111 1st St","State":"Nebraska","C |
| USER#nedstark | #PROFILE#nedstark | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | nedstark | Eddard Stark | lord@winterfell.com | 02/27/2016 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | nedstark | 2eae1dee | SHIPPED | 01/15/2019 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |
| | ORDER#f4f80a91 | **Username** | **OrderId** | **Status** | **CreatedAt** | **Address** |
| | | nedstark | f4f80a91 | PLACED | 05/12/2019 | {"StreetAddress":"Suite 200, Red Keep","City":"King's L |

# Filtering access patterns: Sparse index

| Primary Key | | Attributes | | | | |
|---|---|---|---|---|---|---|
| **PK** | **SK** | | | | | |
| USER#alexdebrie | #PROFILE#alexdebrie | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | alexdebrie | Alex DeBrie | alexdebrie1@gmail.com | 2018-03-23 | {"Home":{"StreetAddress":"1111 1st St","State":"Nebr |
| | ORDER#5e7272b7 | **Username** | **OrderId** | **Status** | **CreatedAt** | **PlacedId** |
| | | alexdebrie | 5e7272b7 | PLACED | 2019-04-2 | fb7f |
| | ORDER#42ef295e | **Username** | **OrderId** | **Status** | **CreatedAt** | **PlacedId** |
| | | alexdebrie | 42ef295e | PLACED | 2019-04-2 | c81c |
| | ORDER#2e7abecc | **Username** | **OrderId** | **Status** | **CreatedAt** | |
| | | alexdebrie | 2e7abecc | SHIPPED | 2018-12-25 | |
| USER#nedstark | #PROFILE#nedstark | **Username** | **FullName** | **Email** | **CreatedAt** | **Addresses** |
| | | nedstark | Eddard Stark | lord@winterfell.com | 2016-02-27 | {"Home":{"StreetAddress":"1234 2nd Ave","City":"Wir |
| | ORDER#2eae1dee | **Username** | **OrderId** | **Status** | **CreatedAt** | |
| | | nedstark | 2eae1dee | SHIPPED | 2019-01-15 | |
| | ORDER#f4f80a91 | **Username** | **OrderId** | **Status** | **CreatedAt** | **PlacedId** |
| | | nedstark | f4f80a91 | PLACED | 2019-05-1 | ac40 |

# Filtering access patterns: Sparse index

| GSI (PlacedId) | Attributes | | | | |
|---|---|---|---|---|---|
| PlacedId | Username | OrderId | Status | CreatedAt | PlacedId |
| fb7f | alexdebrie | 5e7272b7 | PLACED | 2019-04-21 | fb7f |
| c81c | alexdebrie | 42ef295e | PLACED | 2019-04-25 | c81c |
| ac40 | nedstark | f4f80a91 | PLACED | 2019-05-12 | ac40 |

# Recap – what did we cover?

- Tenets of NoSQL Data Modeling
    - Work backwards from workload dimensions.
- Design Patterns
    - Key Conditions & Filter Expressions
    - Artificial Sharding & Key Space Segmentation
    - Vertical Partitions & GSI Overloading
    - Distributed Concurrency and Transactions
    - Aggregate Queries and Analytics
    - Complex Relationship Modeling
    - DynamoDB in the AWS Ecosystem

aws modern apps

**Deixe seu Feedback! Sua opinião é muito Importante para nós!**

**Repo Git com Apresentação + Materiais Adicionais**

# Obrigado!

Peterson Larentis
Enterprise Solutions Architect, AWS
https://www.linkedin.com/in/peterson-larentis/