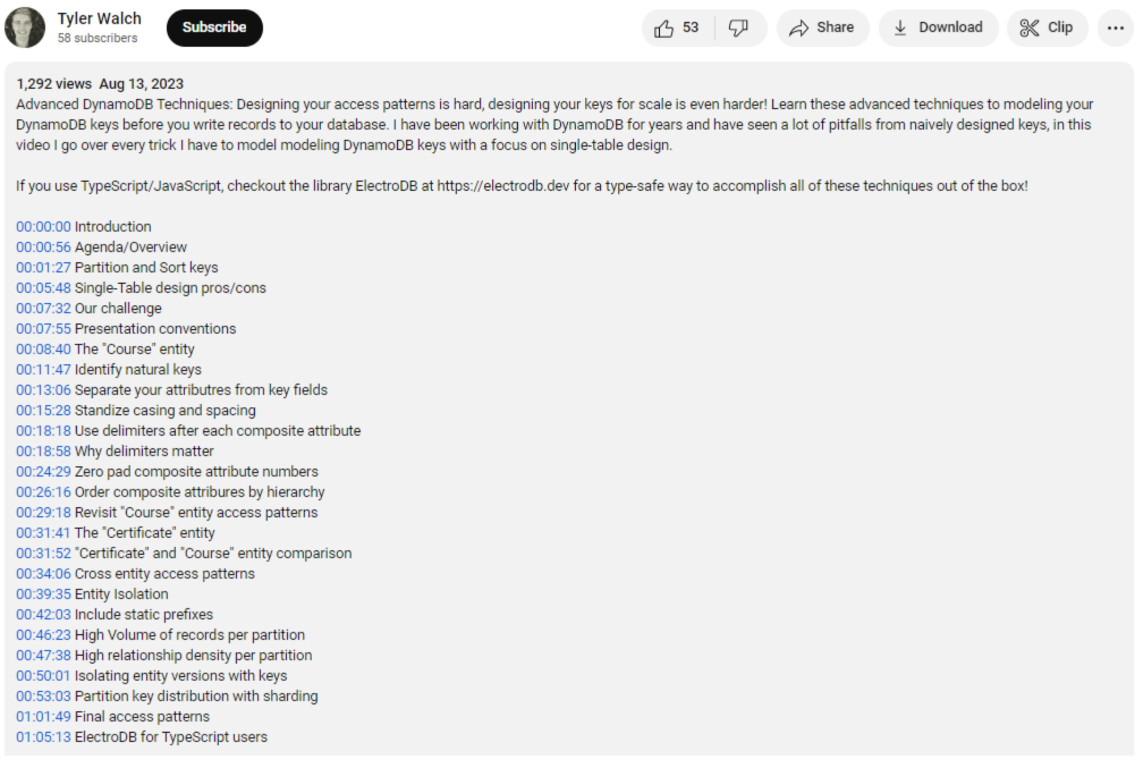


Common Single-Table design modeling mistakes with DynamoDB

Tyler Walch
58 subscribers

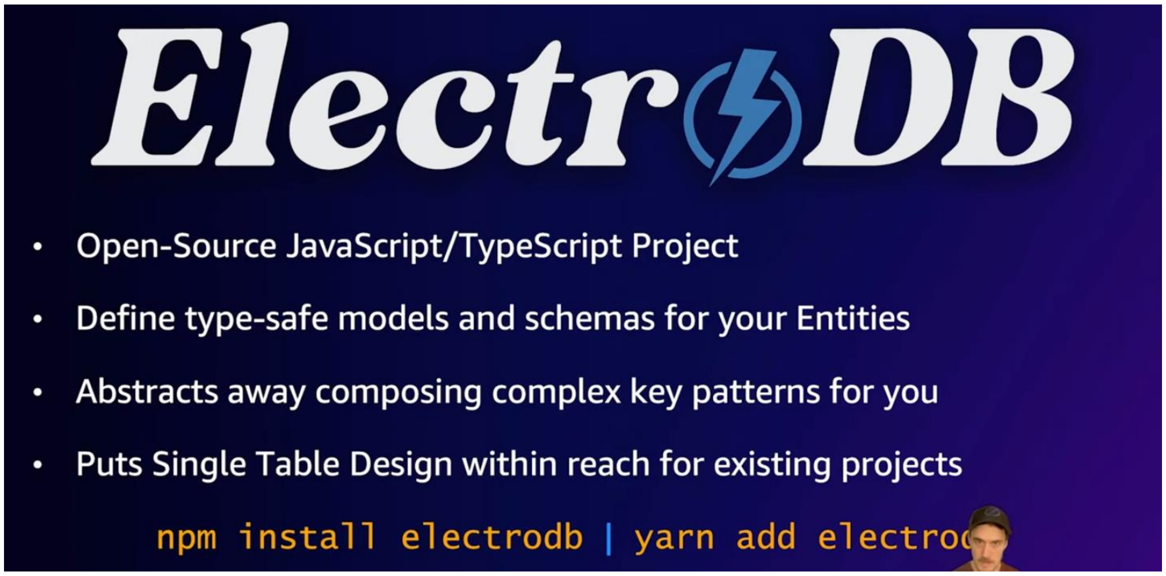
1,292 views Aug 13, 2023

Advanced DynamoDB Techniques: Designing your access patterns is hard, designing your keys for scale is even harder! Learn these advanced techniques to modeling your DynamoDB keys before you write records to your database. I have been working with DynamoDB for years and have seen a lot of pitfalls from naively designed keys, in this video I go over every trick I have to model modeling DynamoDB keys with a focus on single-table design.

If you use TypeScript/JavaScript, checkout the library ElectroDB at https://electrodb.dev for a type-safe way to accomplish all of these techniques out of the box!

Success with DynamoDB through sensible key design

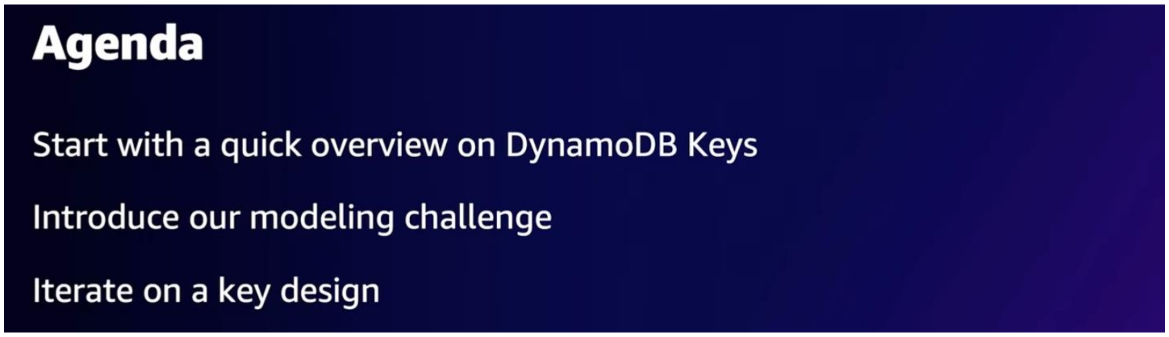
Tyler W. Walch
Staff Engineer
@TINKERTAMPER



ElectroDB

- Open-Source JavaScript/TypeScript Project
- Define type-safe models and schemas for your Entities
- Abstracts away composing complex key patterns for you
- Puts Single Table Design within reach for existing projects

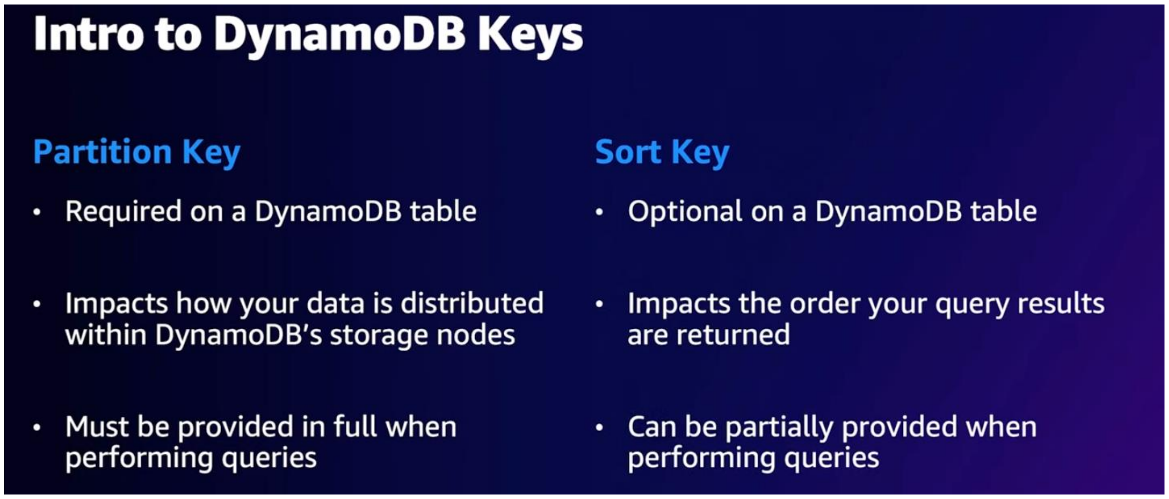
```
npm install electrodb | yarn add electro
```


Agenda

Start with a quick overview on DynamoDB Keys

Introduce our modeling challenge

Iterate on a key design

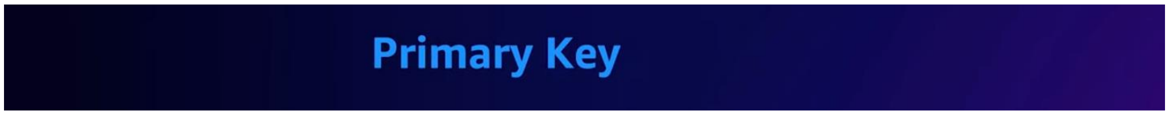

Intro to DynamoDB Keys

Partition Key

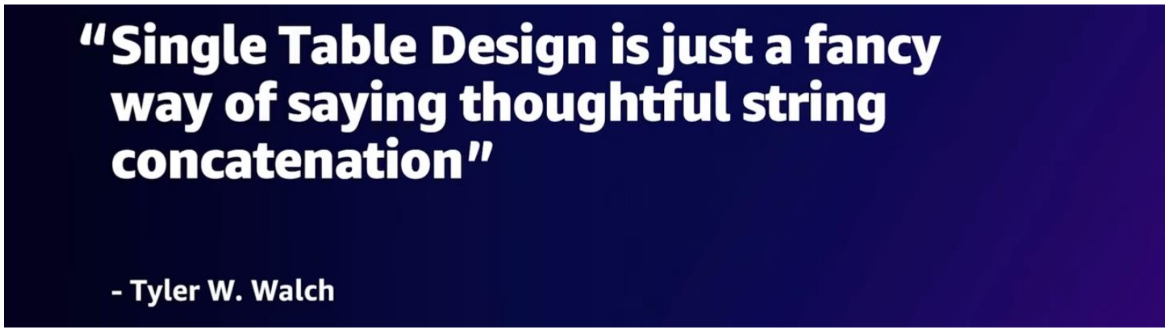
- Required on a DynamoDB table
- Impacts how your data is distributed within DynamoDB's storage nodes
- Must be provided in full when performing queries

Sort Key

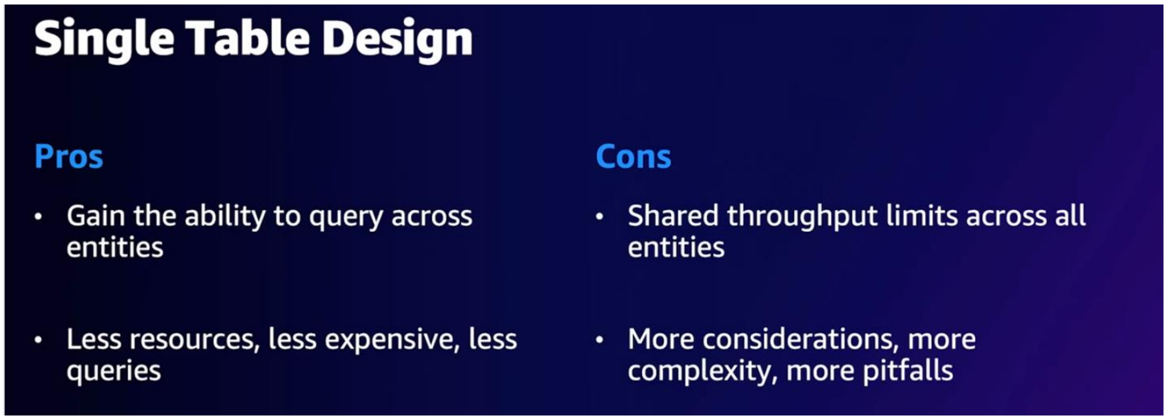
- Optional on a DynamoDB table
- Impacts the order your query results are returned
- Can be partially provided when performing queries


Primary Key

The combination of the **partitionKey** and the **sortKey** can give you a unique Primary Key


"Single Table Design is just a fancy way of saying thoughtful string concatenation"

- Tyler W. Walch

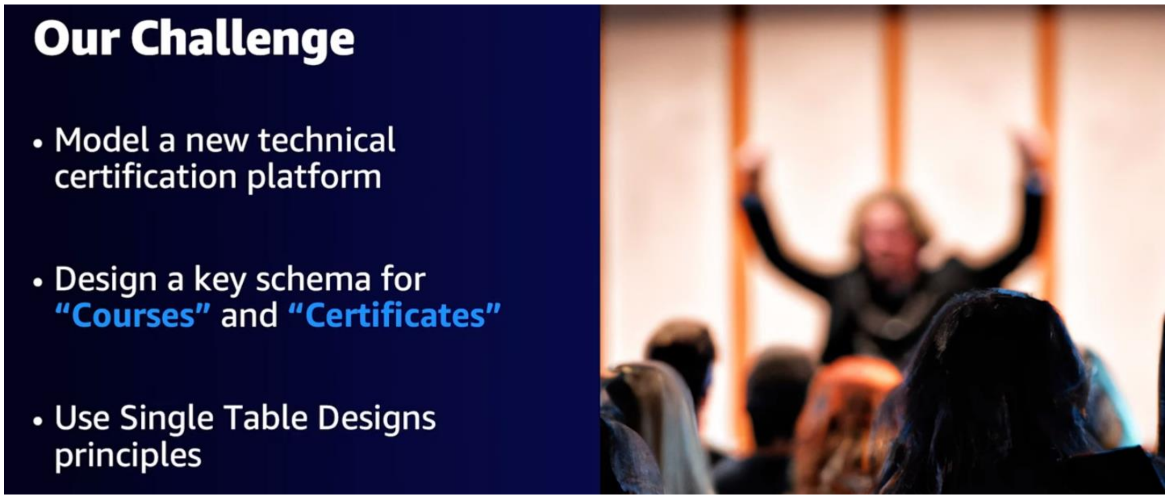

Single Table Design

Pros

- Gain the ability to query across entities
- Less resources, less expensive, less queries

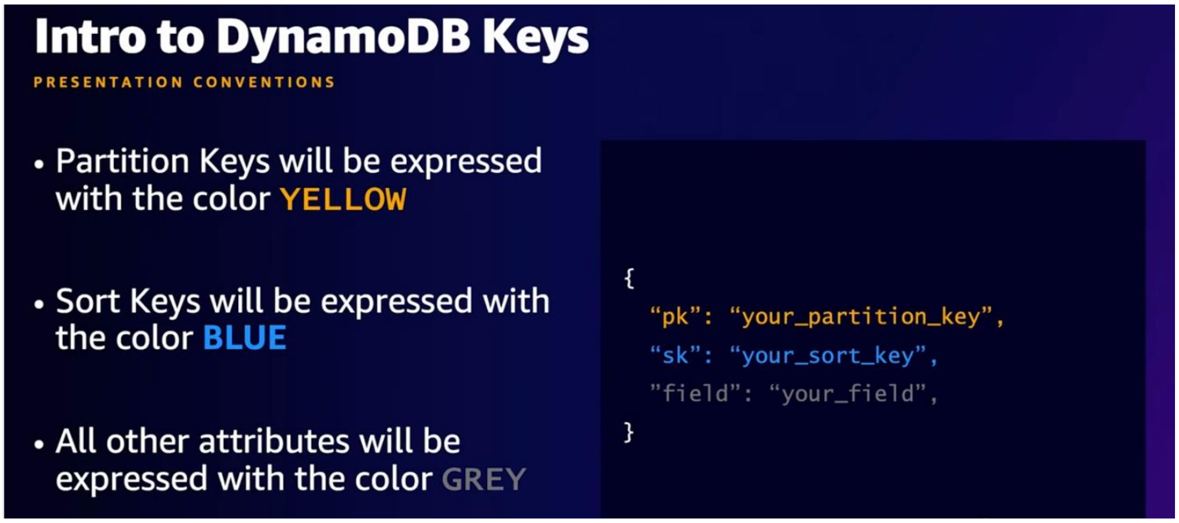
Cons

- Shared throughput limits across all entities
- More considerations, more complexity, more pitfalls



Our Challenge

- Model a new technical certification platform
- Design a key schema for **"Courses"** and **"Certificates"**
- Use Single Table Designs principles



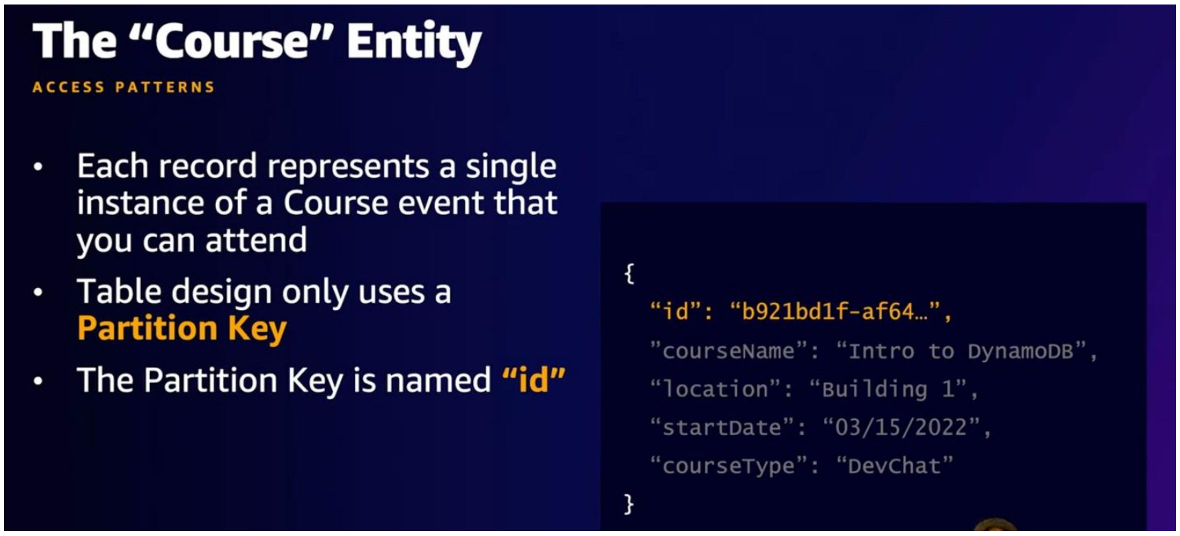
Intro to DynamoDB Keys

PRESENTATION CONVENTIONS

- Partition Keys will be expressed with the color **YELLOW**
- Sort Keys will be expressed with the color **BLUE**
- All other attributes will be expressed with the color GREY

```
{
  "pk": "your_partition_key",
  "sk": "your_sort_key",
  "field": "your_field",
}
```

The "Course" Entity

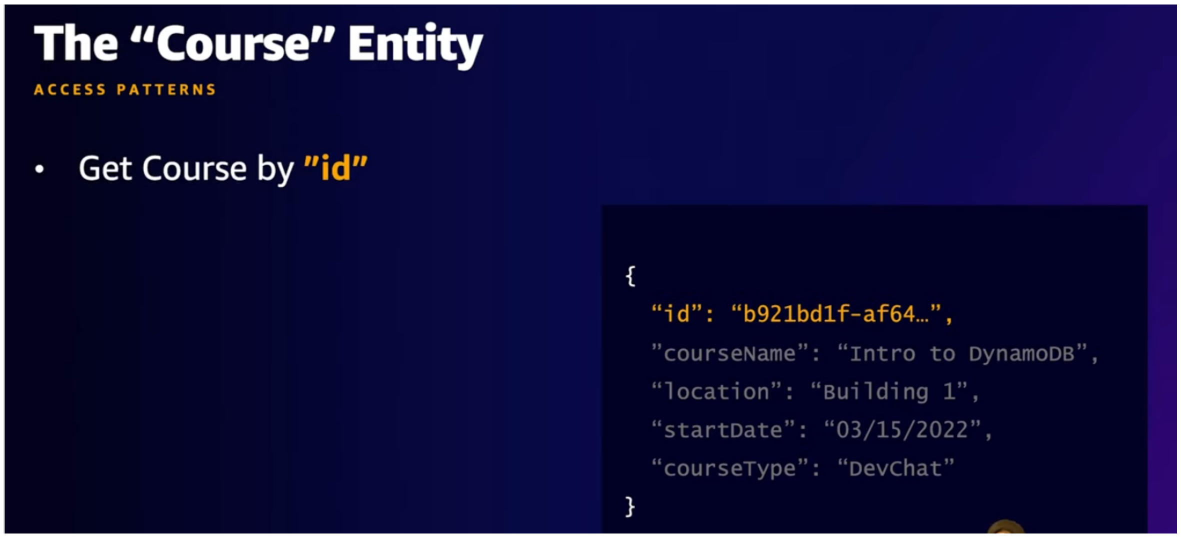


The "Course" Entity

ACCESS PATTERNS

- Each record represents a single instance of a Course event that you can attend
- Table design only uses a **Partition Key**
- The Partition Key is named **"id"**

```
{
  "id": "b921bd1f-af64…",
  "courseName": "Intro to DynamoDB",
  "location": "Building 1",
  "startDate": "03/15/2022",
  "courseType": "DevChat"
}
```

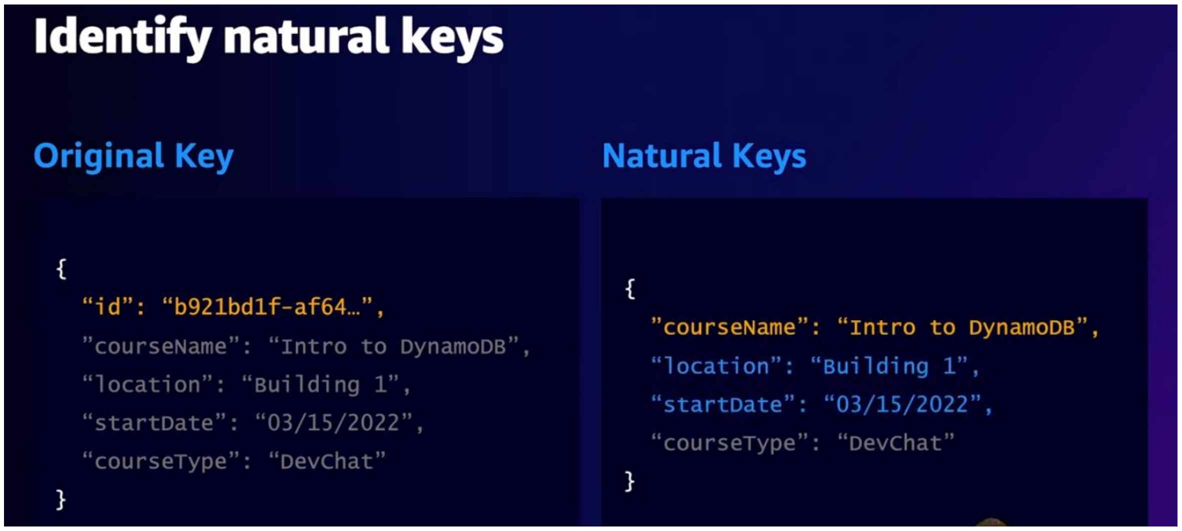


The “Course” Entity

ACCESS PATTERNS

- Get Course by **”id”**

```
{
  “id”: “b921bd1f-af64…”,
  ”courseName”: “Intro to DynamoDB”,
  “location”: “Building 1”,
  “startDate”: “03/15/2022”,
  “courseType”: “DevChat”
}
```



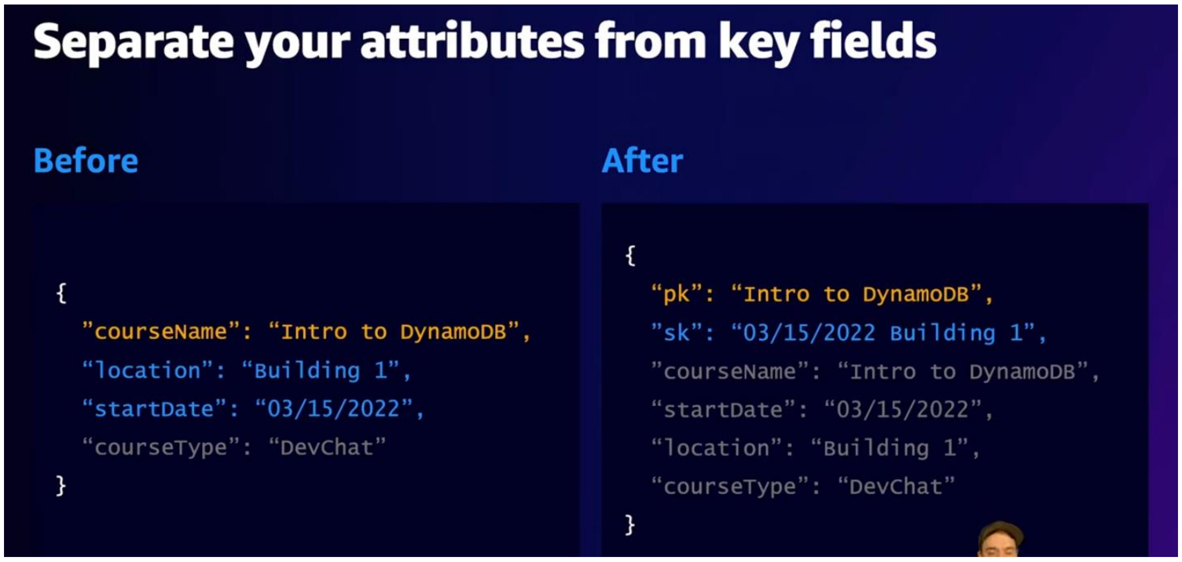
Identify natural keys

Original Key

```
{
  “id”: “b921bd1f-af64…”,
  ”courseName”: “Intro to DynamoDB”,
  “location”: “Building 1”,
  “startDate”: “03/15/2022”,
  “courseType”: “DevChat”
}
```

Natural Keys

```
{
  ”courseName”: “Intro to DynamoDB”,
  “location”: “Building 1”,
  “startDate”: “03/15/2022”,
  “courseType”: “DevChat”
}
```



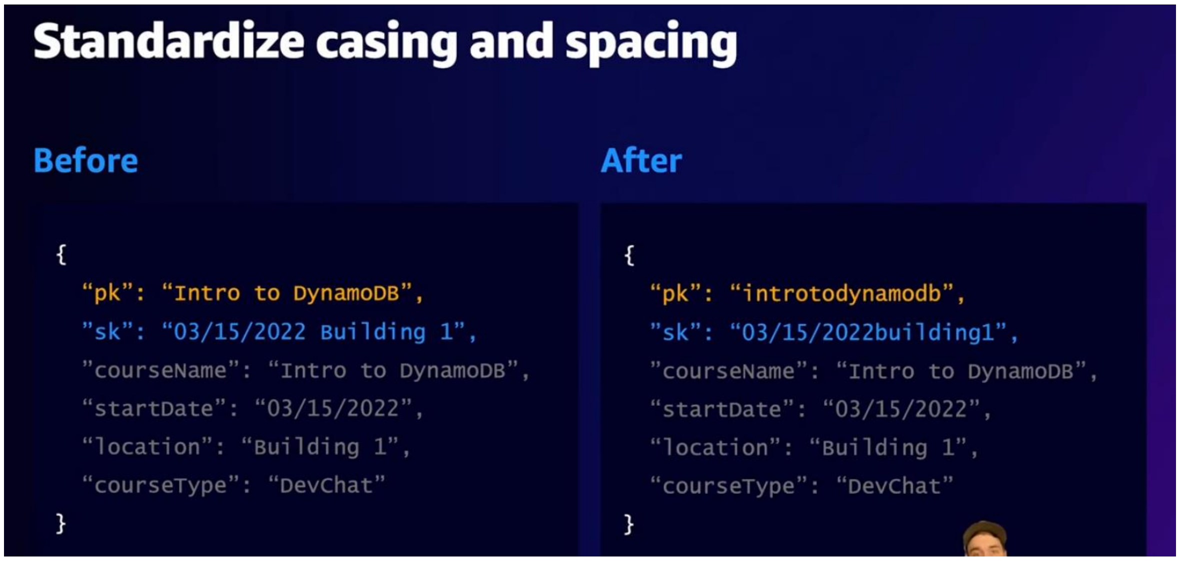
Separate your attributes from key fields

Before

```
{
  ”courseName”: “Intro to DynamoDB”,
  “location”: “Building 1”,
  “startDate”: “03/15/2022”,
  “courseType”: “DevChat”
}
```

After

```
{
  “pk”: “Intro to DynamoDB”,
  ”sk”: “03/15/2022 Building 1”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```



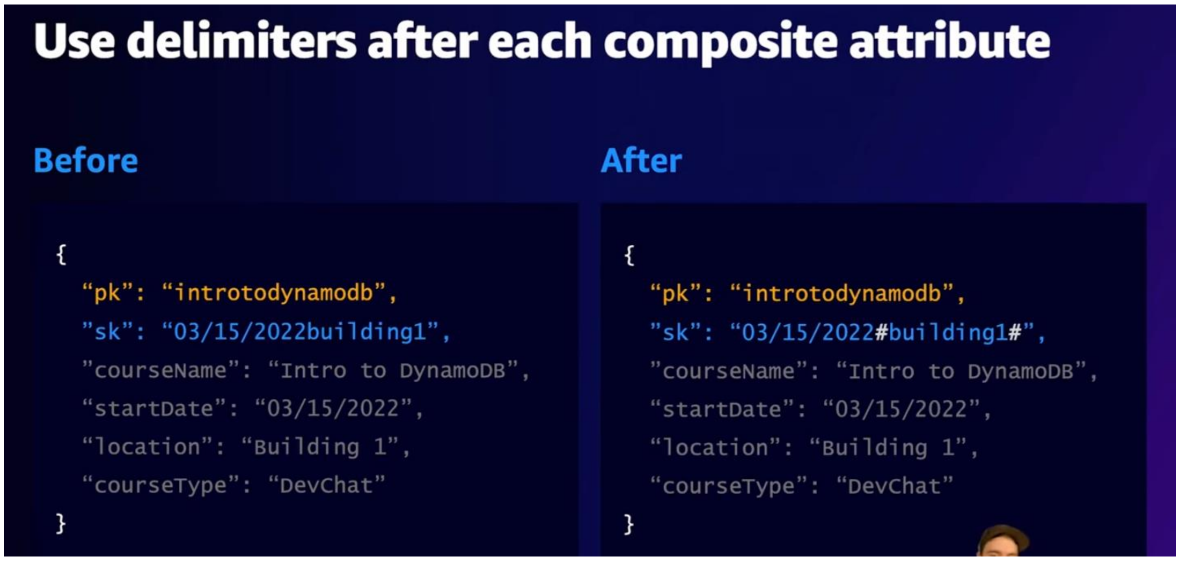
Standardize casing and spacing

Before

```
{
  “pk”: “Intro to DynamoDB”,
  ”sk”: “03/15/2022 Building 1”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

After

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “03/15/2022building1”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```



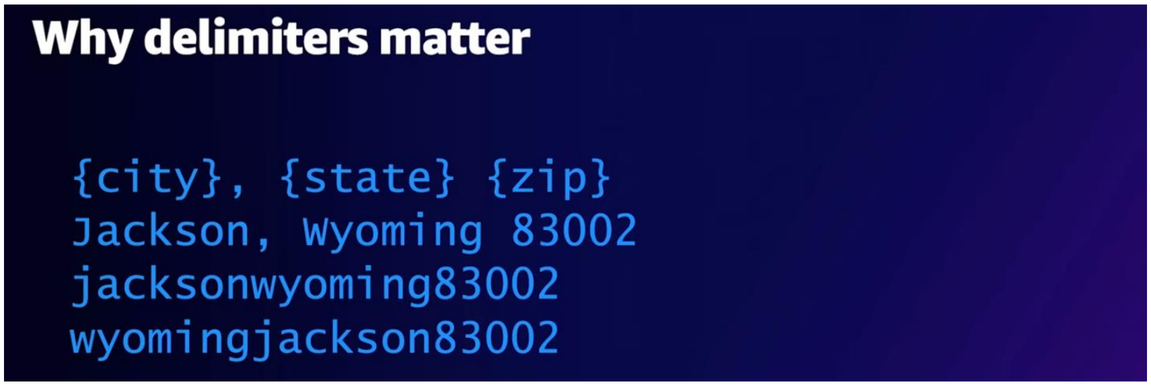
Use delimiters after each composite attribute

Before

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “03/15/2022building1”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

After

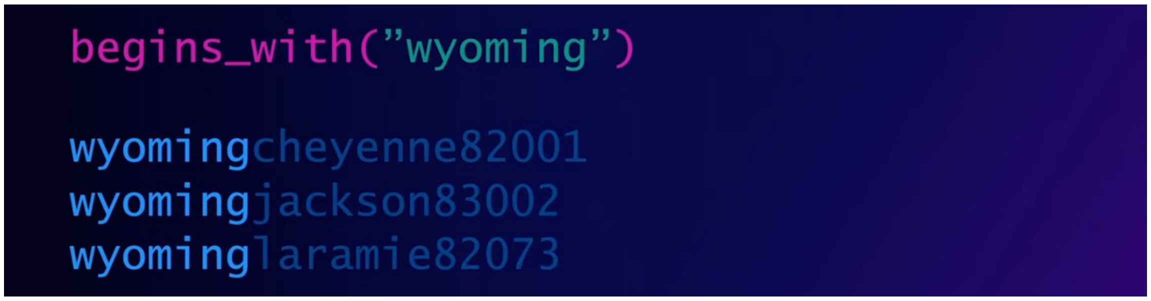
```
{
  “pk”: “introtodynamodb”,
  ”sk”: “03/15/2022#building1#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```



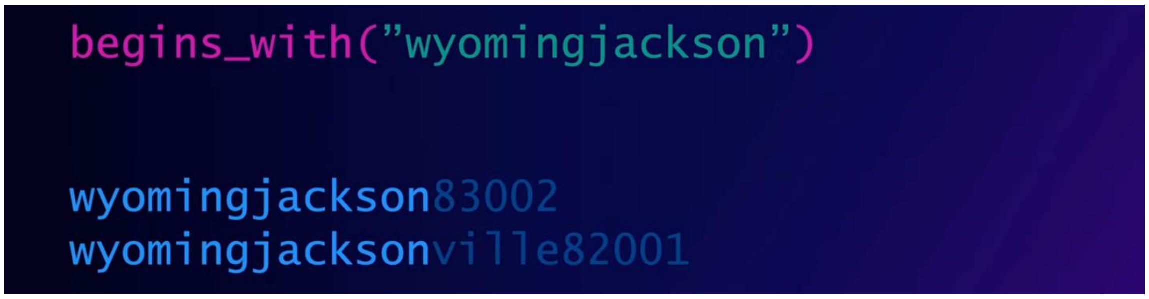
Why delimiters matter

```
{city}, {state} {zip}
Jackson, Wyoming 83002
jacksonwyoming83002
wyomingjackson83002
```

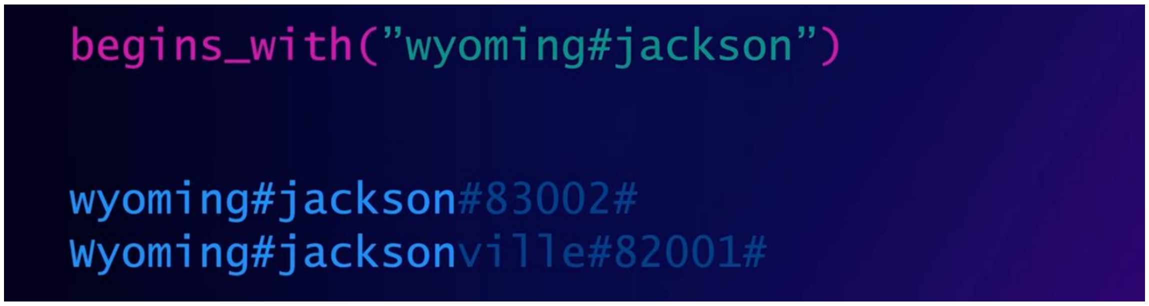
Rearrange to start broad and then make things more specific



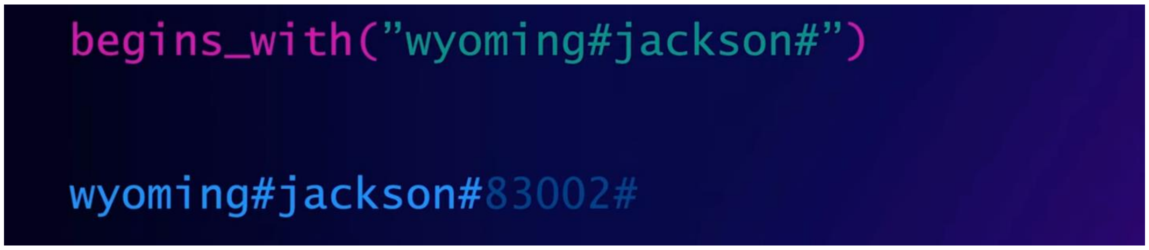
```
begins_with(”wyoming”)

wyomingcheyenne82001
wyomingjackson83002
wyominglaramie82073
```



```
begins_with("wyomingjackson")

wyomingjackson83002
wyomingjacksonville82001
```

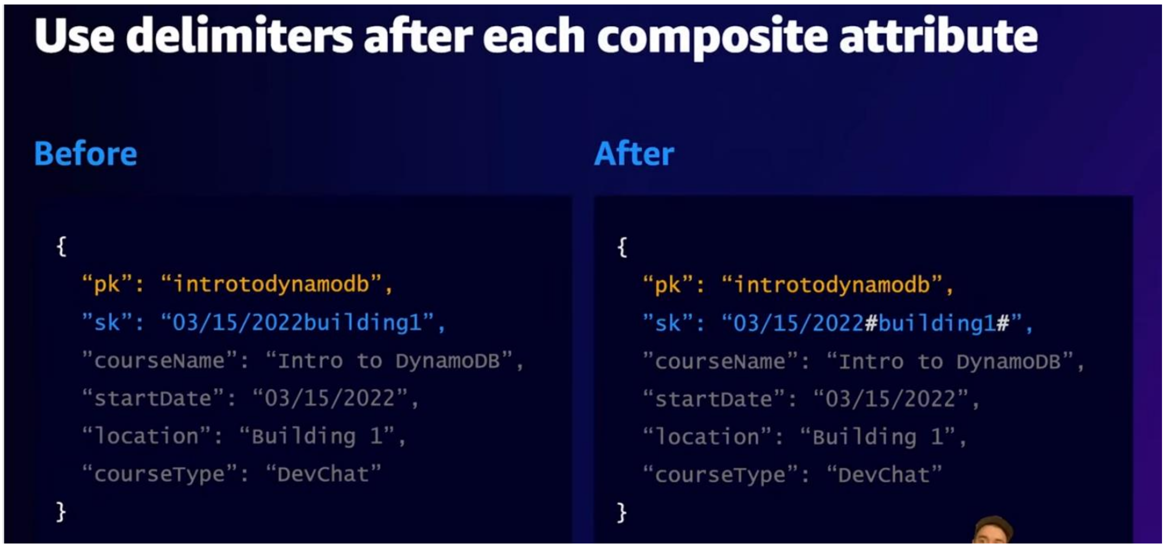


```
begins_with("wyoming#jackson")

wyoming#jackson#83002#
Wyoming#jacksonville#82001#
```



```
begins_with("wyoming#jackson#")

wyoming#jackson#83002#
```

Using delimitiers will get you all the zip codes in Wyoming and Jackson city and prevent over-querying DynamoDB tables



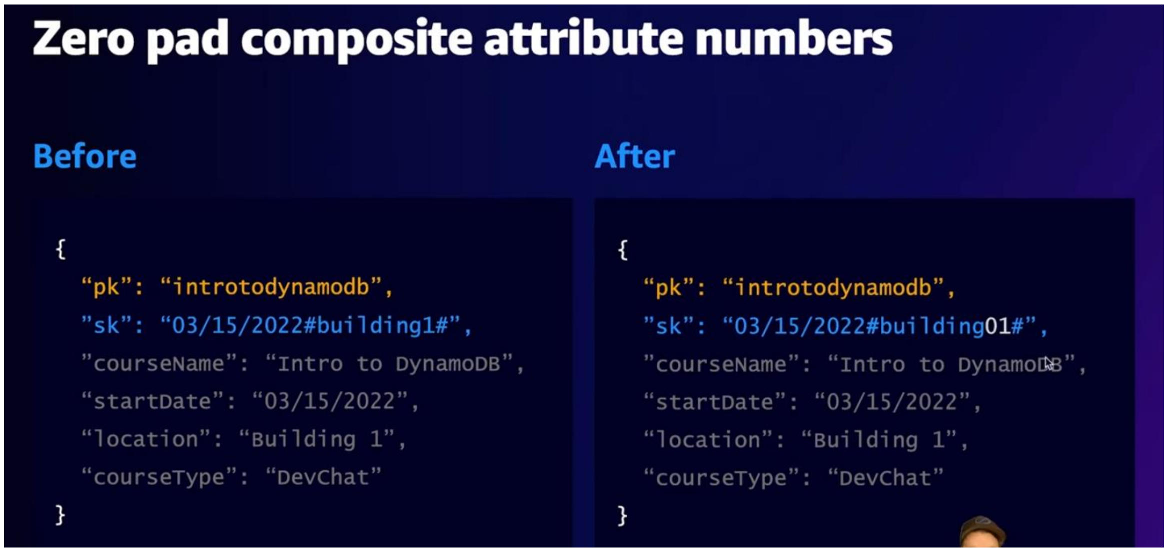
Use delimiters after each composite attribute

Before

```
{
  "pk": "introtodynamodb",
  "sk": "03/15/2022building1",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

After

```
{
  "pk": "introtodynamodb",
  "sk": "03/15/2022#building1#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```



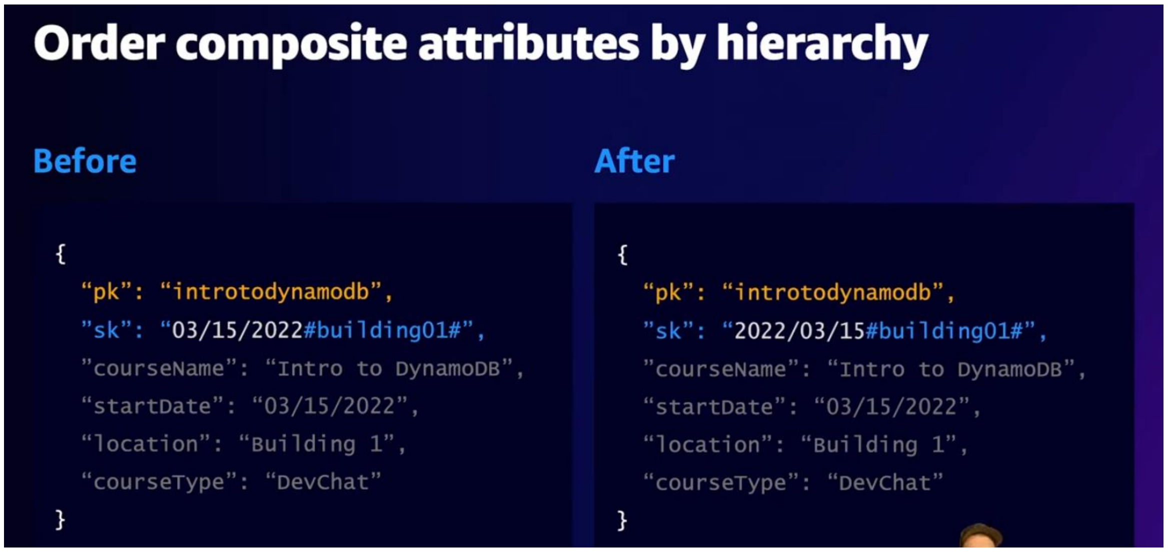
Zero pad composite attribute numbers

Before

```
{
  "pk": "introtodynamodb",
  "sk": "03/15/2022#building1#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

After

```
{
  "pk": "introtodynamodb",
  "sk": "03/15/2022#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```



Order composite attributes by hierarchy

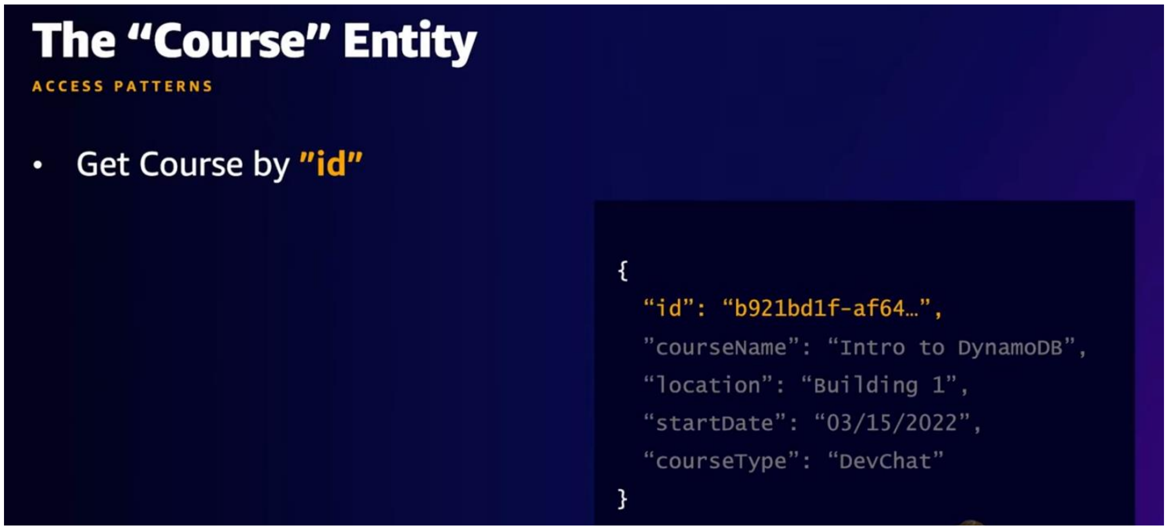
Before

```
{
  "pk": "introtodynamodb",
  "sk": "03/15/2022#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

After

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

How do you access your courses data, by date or by building?

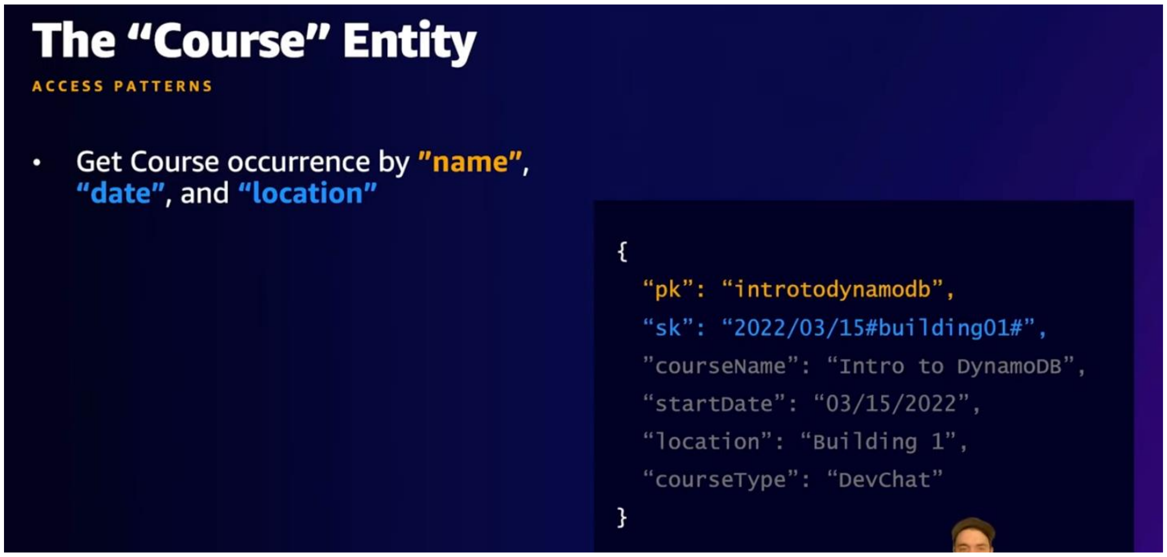


The "Course" Entity

ACCESS PATTERNS

- Get Course by "id"

```
{
  "id": "b921bd1f-af64…",
  "courseName": "Intro to DynamoDB",
  "location": "Building 1",
  "startDate": "03/15/2022",
  "courseType": "DevChat"
}
```

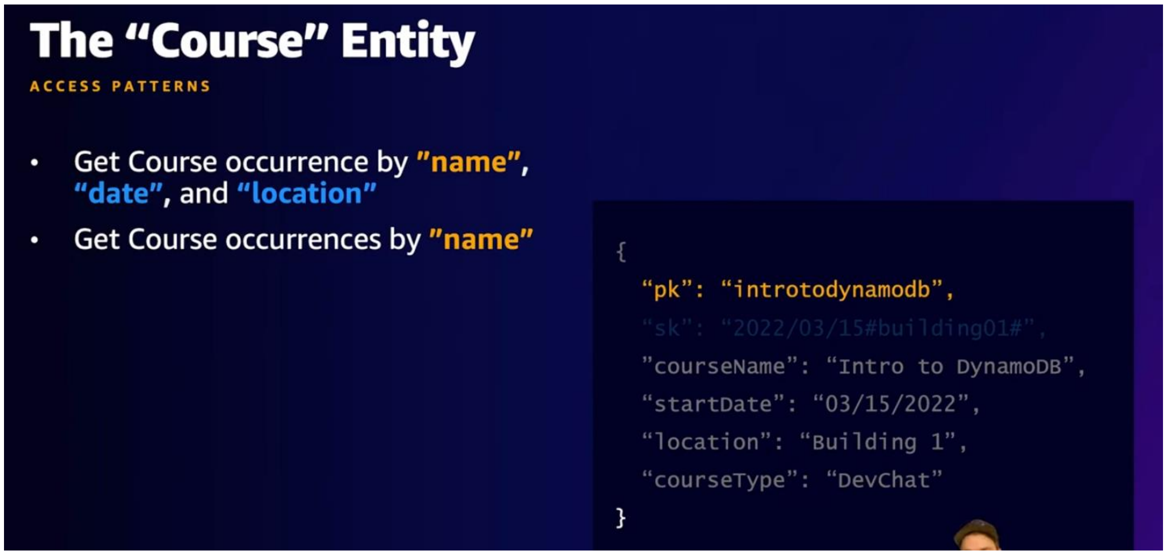


The “Course” Entity

ACCESS PATTERNS

- Get Course occurrence by **”name”**, **“date”**, and **“location”**

```
{
  “pk”: “introtodynamodb”,
  “sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

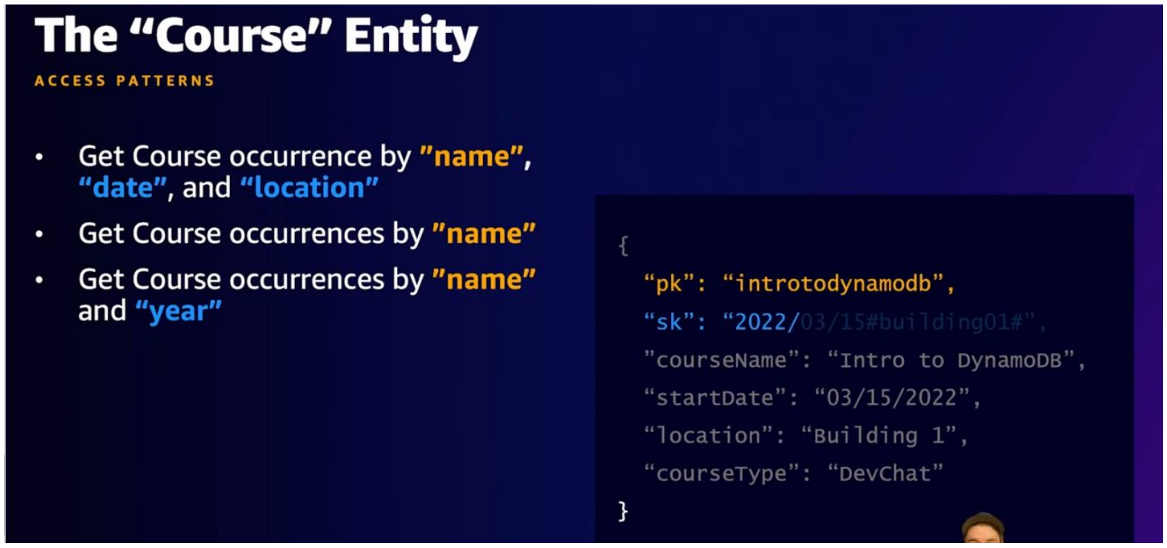


The “Course” Entity

ACCESS PATTERNS

- Get Course occurrence by **”name”**, **“date”**, and **“location”**
- Get Course occurrences by **”name”**

```
{
  “pk”: “introtodynamodb”,
  “sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

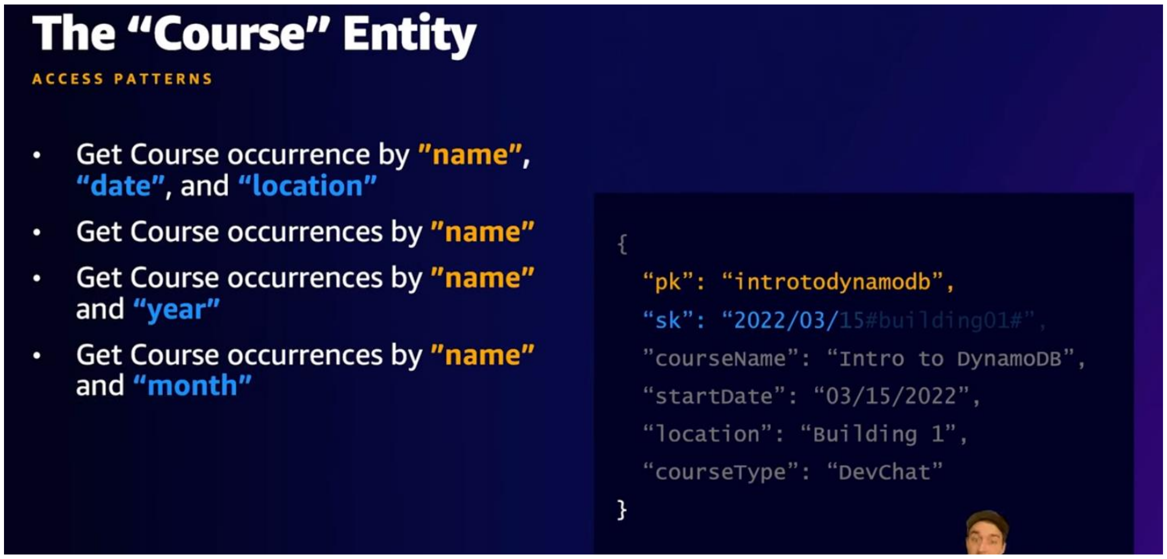


The “Course” Entity

ACCESS PATTERNS

- Get Course occurrence by **”name”**, **“date”**, and **“location”**
- Get Course occurrences by **”name”**
- Get Course occurrences by **”name”** and **“year”**

```
{
  “pk”: “introtodynamodb”,
  “sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

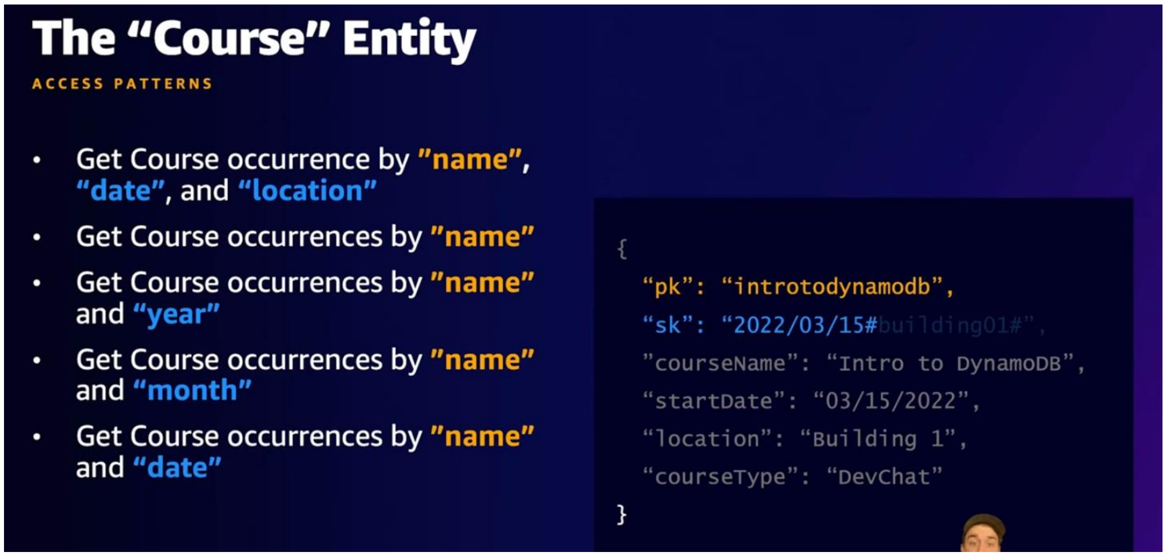


The "Course" Entity

ACCESS PATTERNS

- Get Course occurrence by "name", "date", and "location"
- Get Course occurrences by "name"
- Get Course occurrences by "name" and "year"
- Get Course occurrences by "name" and "month"

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

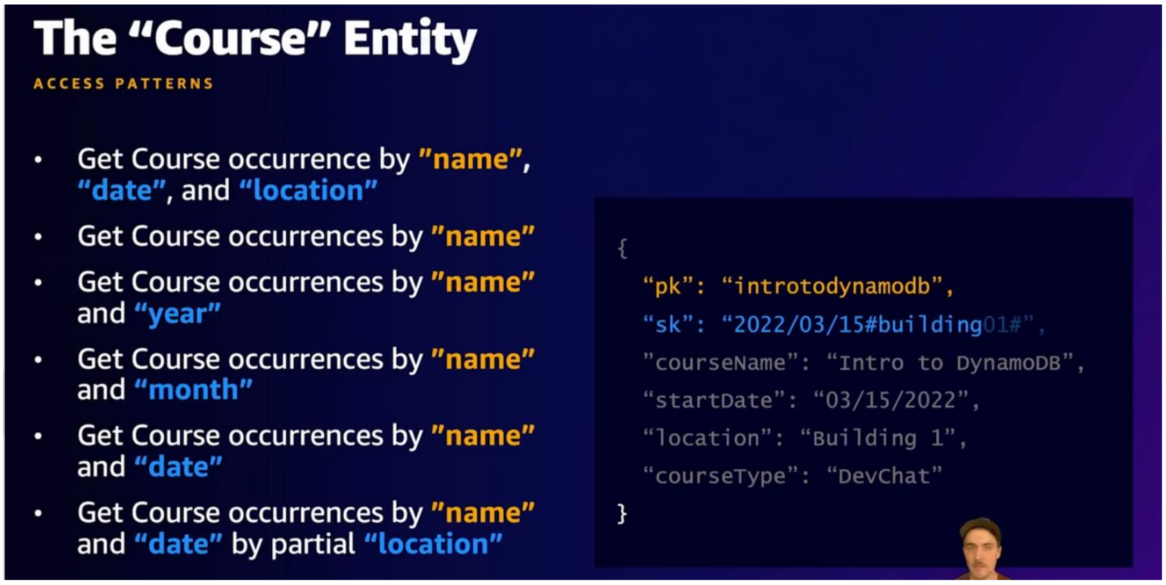


The "Course" Entity

ACCESS PATTERNS

- Get Course occurrence by "name", "date", and "location"
- Get Course occurrences by "name"
- Get Course occurrences by "name" and "year"
- Get Course occurrences by "name" and "month"
- Get Course occurrences by "name" and "date"

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```



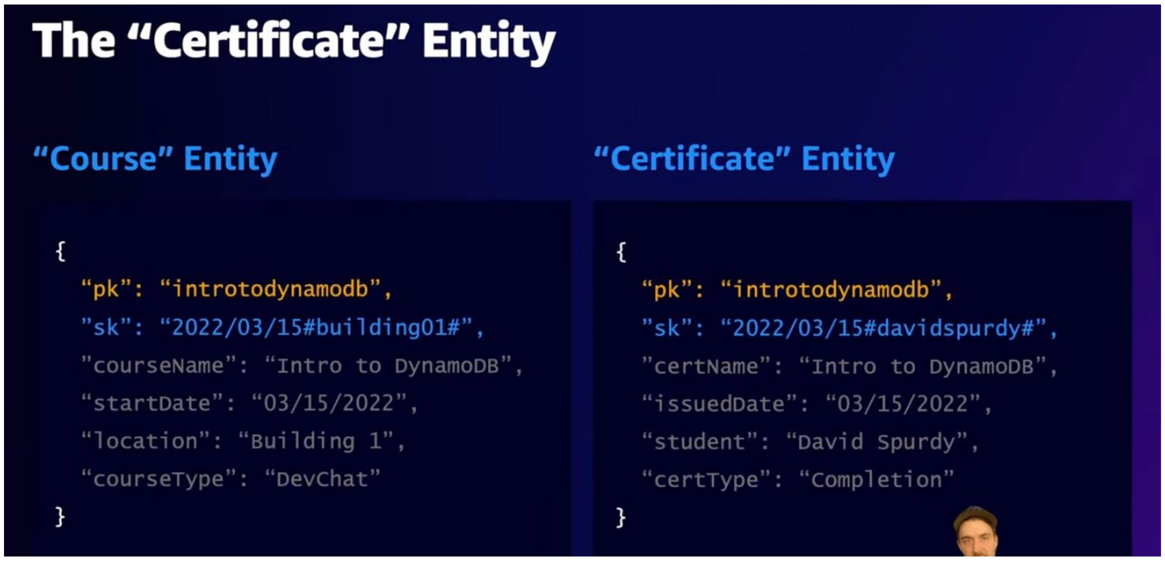
The "Course" Entity

ACCESS PATTERNS

- Get Course occurrence by "name", "date", and "location"
- Get Course occurrences by "name"
- Get Course occurrences by "name" and "year"
- Get Course occurrences by "name" and "month"
- Get Course occurrences by "name" and "date"
- Get Course occurrences by "name" and "date" by partial "location"

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#building01#",
  "courseName": "Intro to DynamoDB",
  "startDate": "03/15/2022",
  "location": "Building 1",
  "courseType": "DevChat"
}
```

The "Certificate" Entity



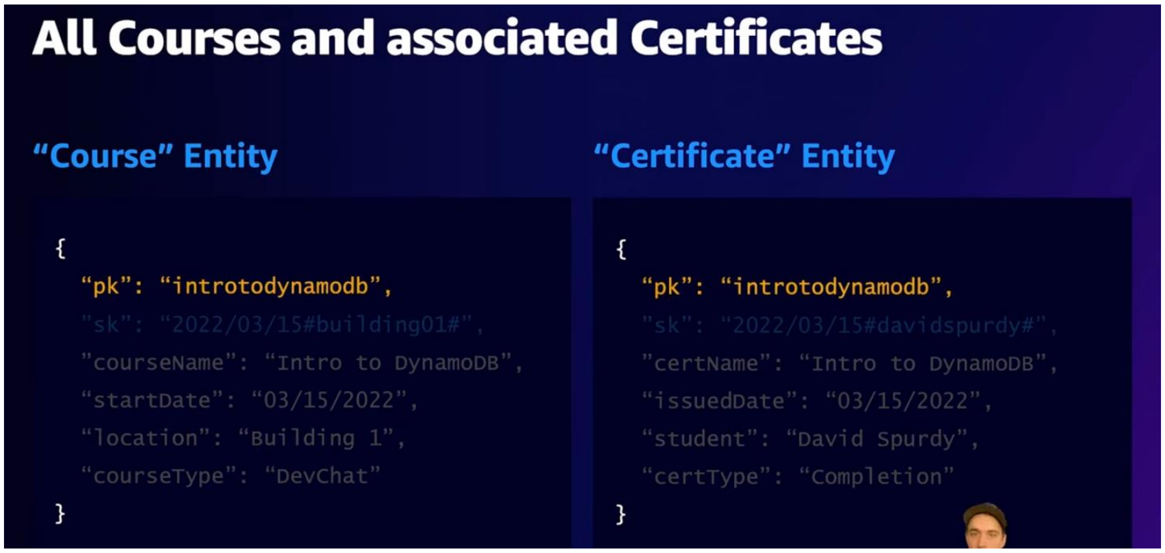
The “Certificate” Entity

“Course” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

“Certificate” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”,
  ”certName”: “Intro to DynamoDB”,
  “issuedDate”: “03/15/2022”,
  “student”: “David Spurdy”,
  “certType”: “Completion”
}
```



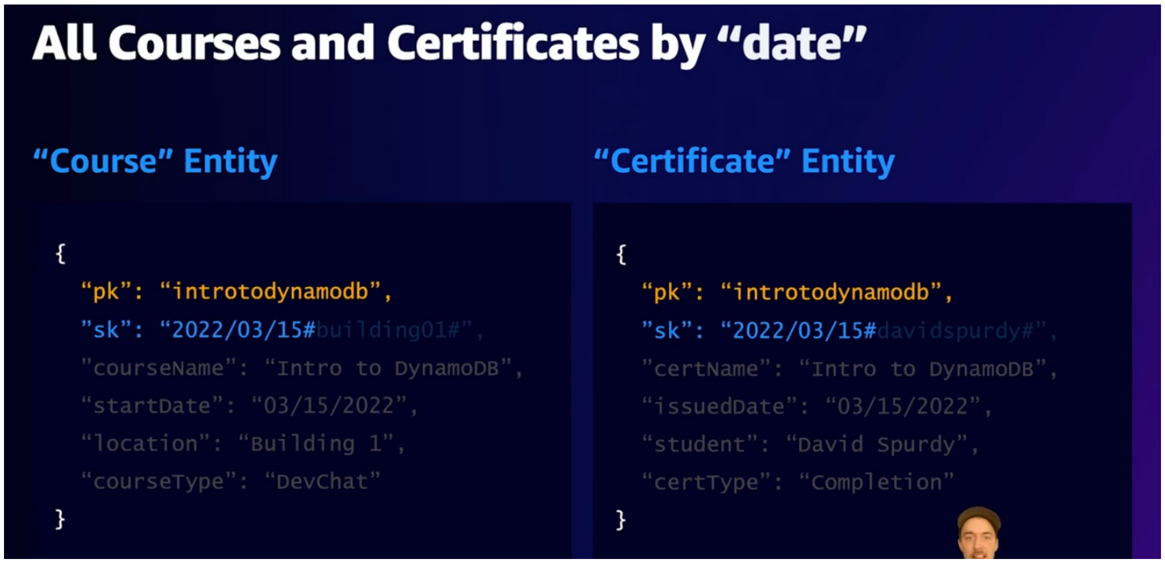
All Courses and associated Certificates

“Course” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

“Certificate” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”,
  ”certName”: “Intro to DynamoDB”,
  “issuedDate”: “03/15/2022”,
  “student”: “David Spurdy”,
  “certType”: “Completion”
}
```



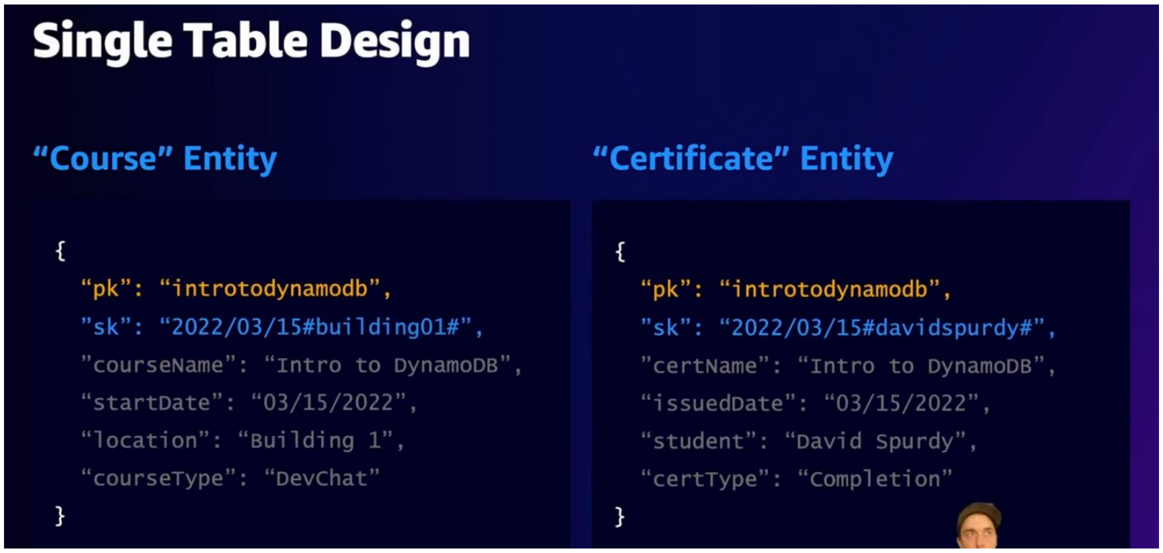
All Courses and Certificates by “date”

“Course” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

“Certificate” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”,
  ”certName”: “Intro to DynamoDB”,
  “issuedDate”: “03/15/2022”,
  “student”: “David Spurdy”,
  “certType”: “Completion”
}
```



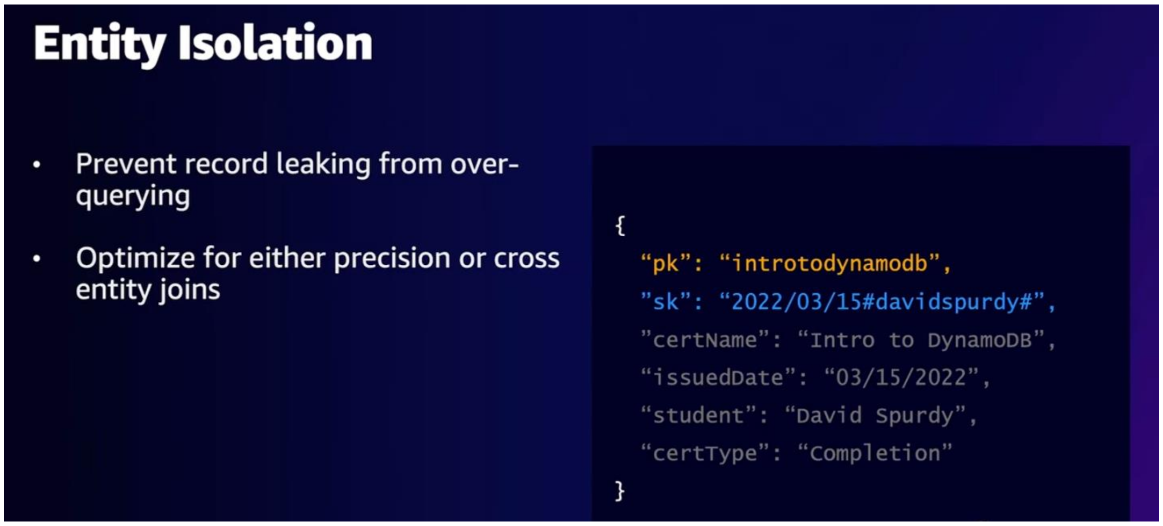
Single Table Design

“Course” Entity

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#building01#”,
  ”courseName”: “Intro to DynamoDB”,
  “startDate”: “03/15/2022”,
  “location”: “Building 1”,
  “courseType”: “DevChat”
}
```

“Certificate” Entity

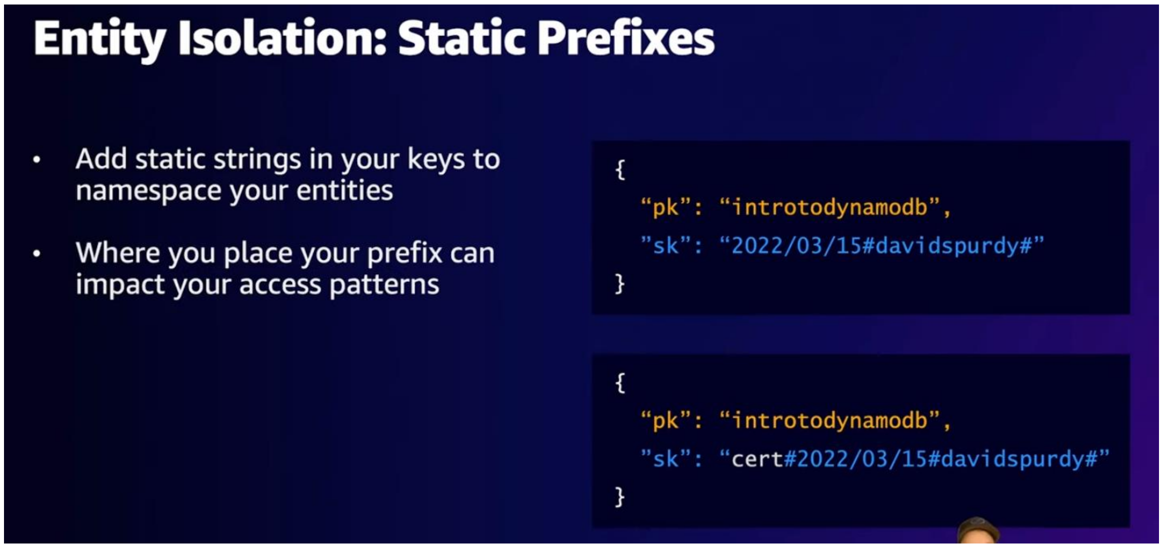
```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”,
  ”certName”: “Intro to DynamoDB”,
  “issuedDate”: “03/15/2022”,
  “student”: “David Spurdy”,
  “certType”: “Completion”
}
```



Entity Isolation

- Prevent record leaking from over-querying
- Optimize for either precision or cross entity joins

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”,
  ”certName”: “Intro to DynamoDB”,
  “issuedDate”: “03/15/2022”,
  “student”: “David Spurdy”,
  “certType”: “Completion”
}
```

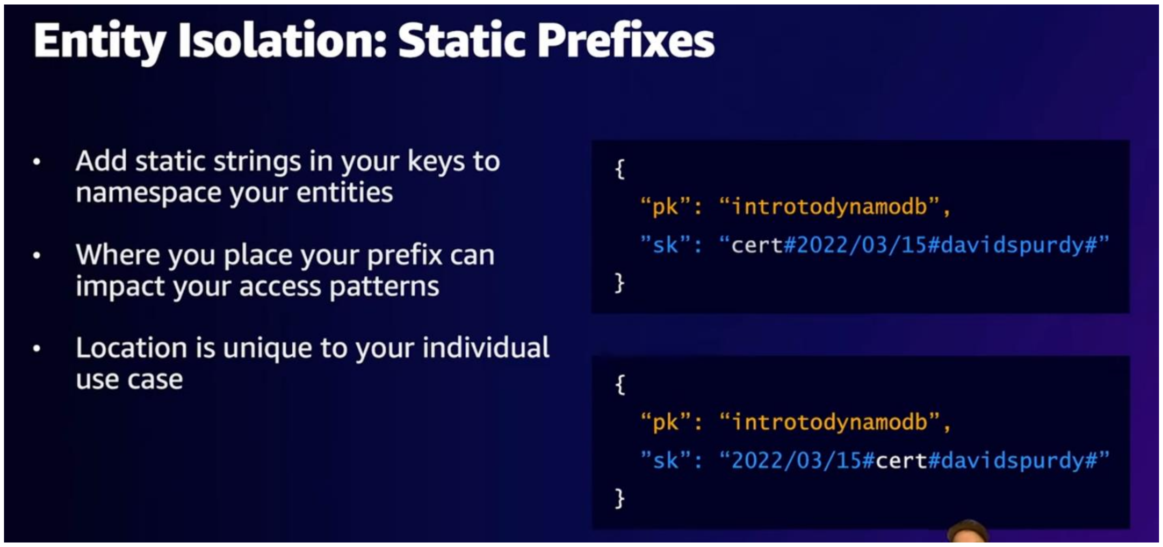


Entity Isolation: Static Prefixes

- Add static strings in your keys to namespace your entities
- Where you place your prefix can impact your access patterns

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#davidspurdy#”
}
```

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “cert#2022/03/15#davidspurdy#”
}
```

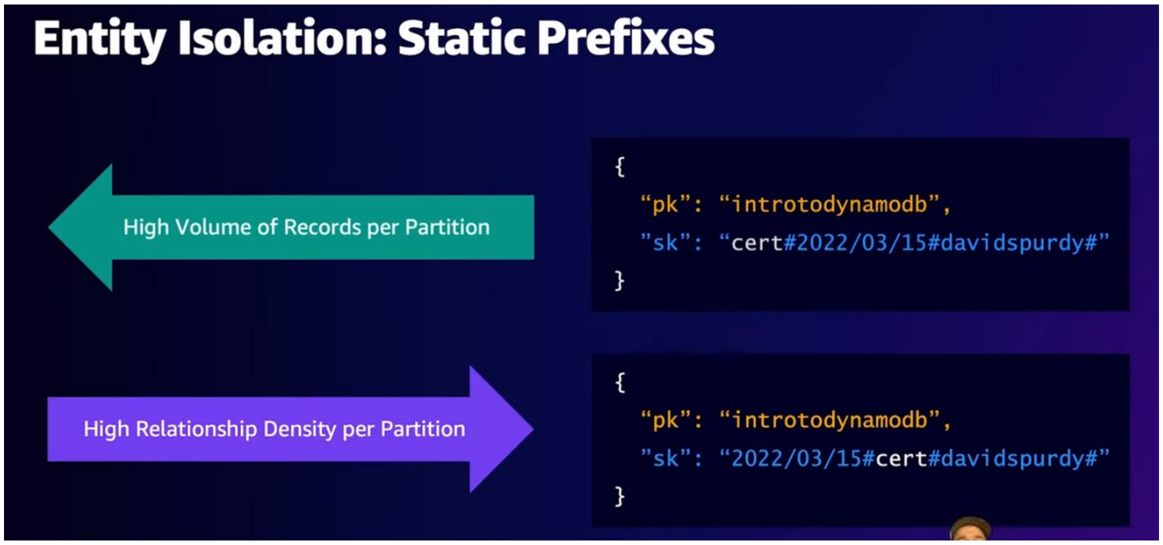


Entity Isolation: Static Prefixes

- Add static strings in your keys to namespace your entities
- Where you place your prefix can impact your access patterns
- Location is unique to your individual use case

```
{
  "pk": "introtodynamodb",
  "sk": "cert#2022/03/15#davidspurdy#"
}
```

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#davidspurdy#"
}
```

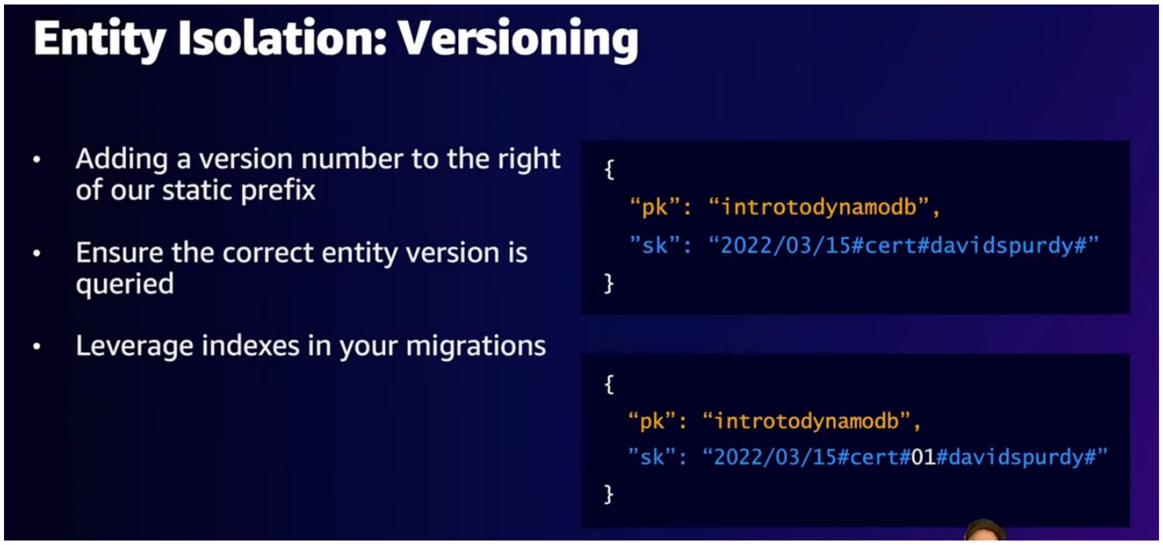


Entity Isolation: Static Prefixes

High Volume of Records per Partition
High Relationship Density per Partition

```
{
  "pk": "introtodynamodb",
  "sk": "cert#2022/03/15#davidspurdy#"
}
```

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#davidspurdy#"
}
```

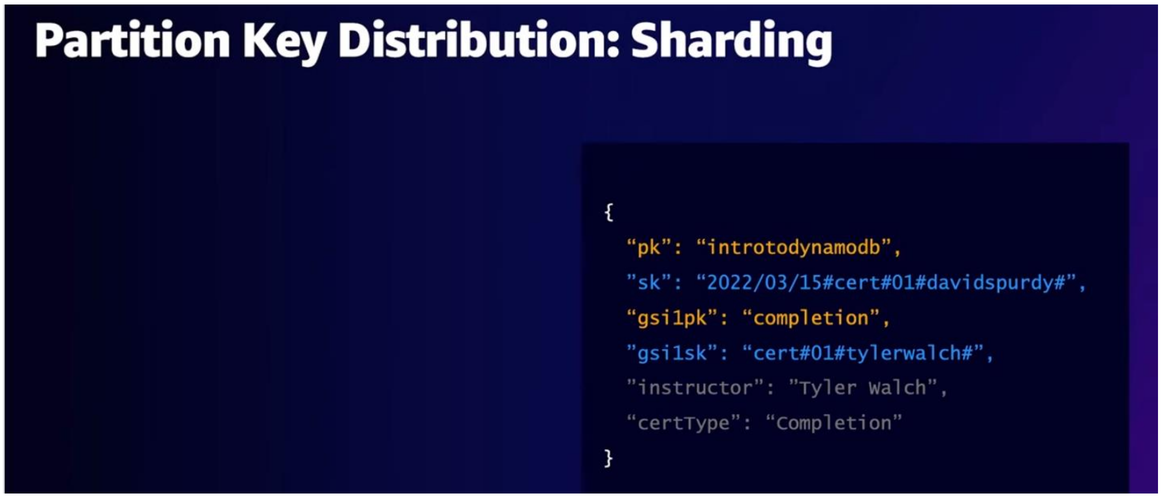


Entity Isolation: Versioning

- Adding a version number to the right of our static prefix
- Ensure the correct entity version is queried
- Leverage indexes in your migrations

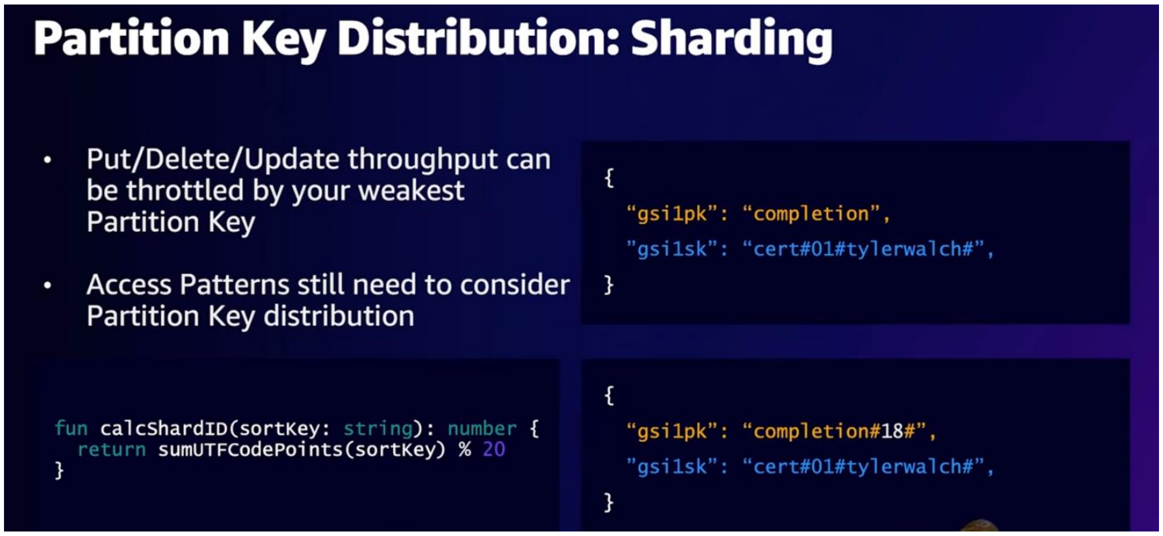
```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#davidspurdy#"
}
```

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#01#davidspurdy#"
}
```



Partition Key Distribution: Sharding

```
{
  “pk”: “introtodynamodb”,
  "sk”: “2022/03/15#cert#01#davidspurdy#”,
  “gsi1pk”: “completion”,
  "gsi1sk”: “cert#01#tylerwalch#”,
  "instructor": "Tyler Walch",
  “certType”: “Completion”
}
```



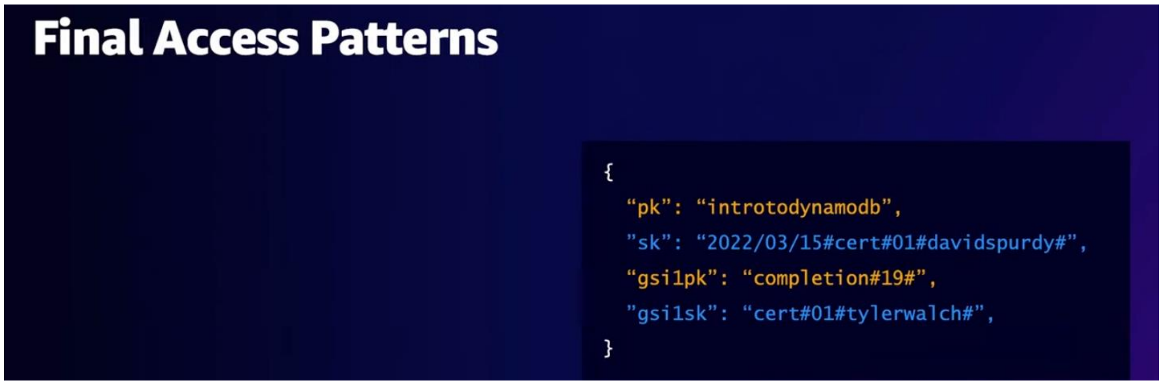
Partition Key Distribution: Sharding

- Put/Delete/Update throughput can be throttled by your weakest Partition Key
- Access Patterns still need to consider Partition Key distribution

```
{
  “gsi1pk”: “completion”,
  "gsi1sk”: “cert#01#tylerwalch#”,
}
```

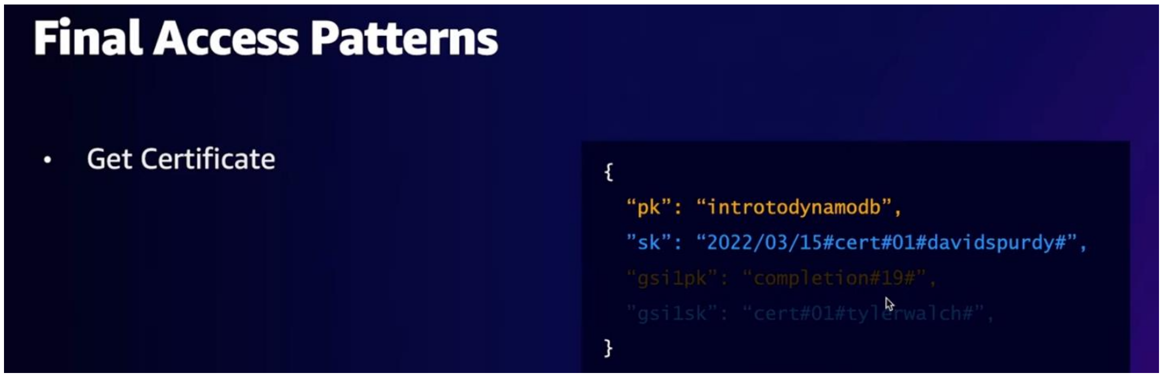
```
fun calcShardID(sortKey: string): number {
  return sumUTFCodePoints(sortKey) % 20
}
```

```
{
  “gsi1pk”: “completion#18#”,
  "gsi1sk”: “cert#01#tylerwalch#”,
}
```



Final Access Patterns

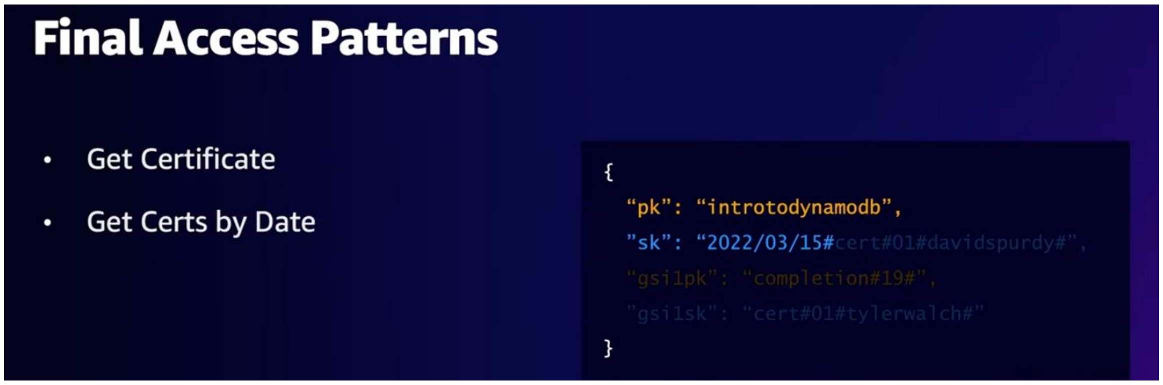
```
{
  “pk”: “introtodynamodb”,
  "sk”: “2022/03/15#cert#01#davidspurdy#”,
  “gsi1pk”: “completion#19#”,
  "gsi1sk”: “cert#01#tylerwalch#”,
}
```



Final Access Patterns

- Get Certificate

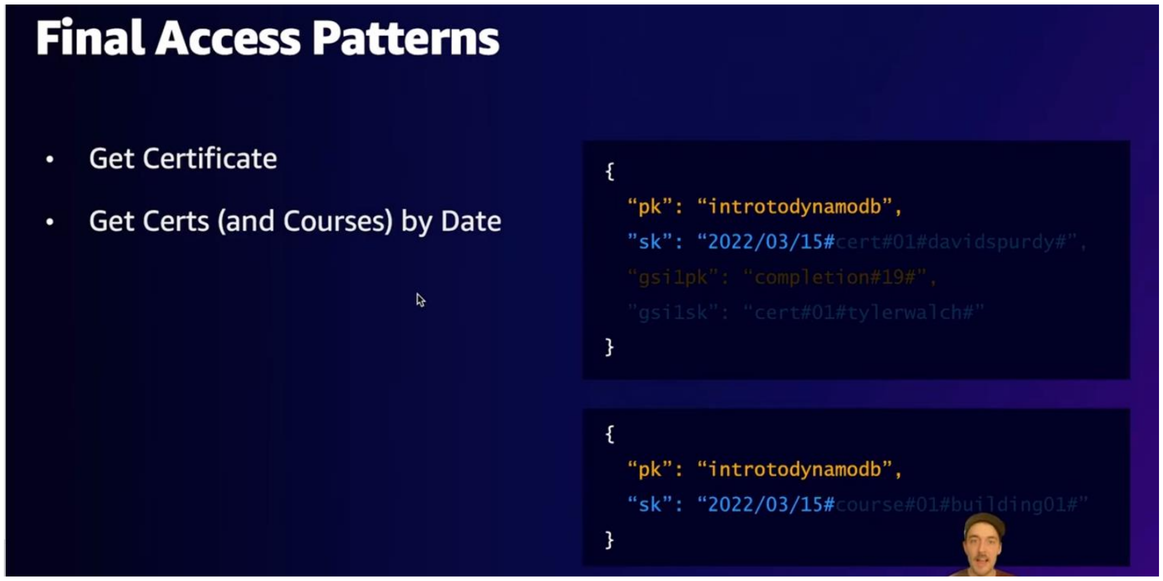
```
{
  “pk”: “introtodynamodb”,
  "sk”: “2022/03/15#cert#01#davidspurdy#”,
  “gsi1pk”: “completion#19#”,
  "gsi1sk”: “cert#01#tylerwalch#”,
}
```



Final Access Patterns

- Get Certificate
- Get Certs by Date

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#cert#01#davidspurdy#”,
  ”gsi1pk”: ”completion#19#”,
  ”gsi1sk”: ”cert#01#tylerwalch#”
}
```

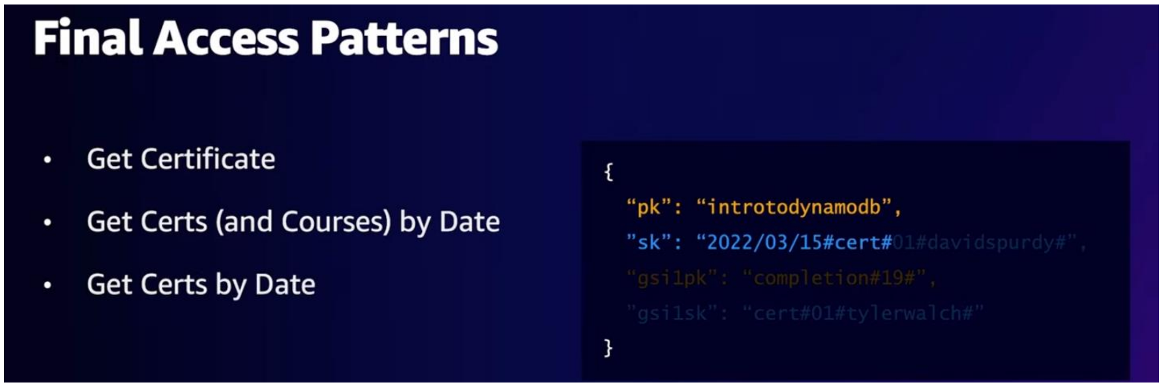


Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date

```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#cert#01#davidspurdy#”,
  ”gsi1pk”: ”completion#19#”,
  ”gsi1sk”: ”cert#01#tylerwalch#”
}
```

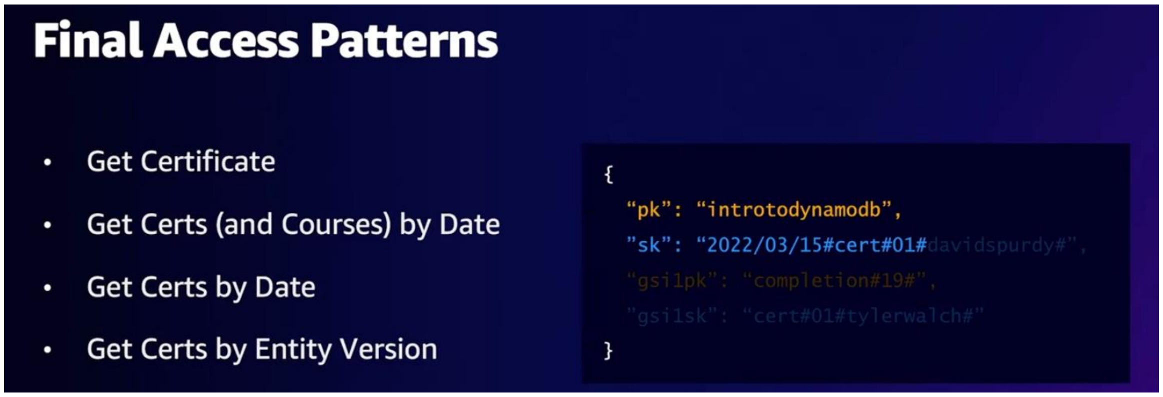
```
{
  “pk”: “introtodynamodb”,
  “sk”: “2022/03/15#course#01#building01#”
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date

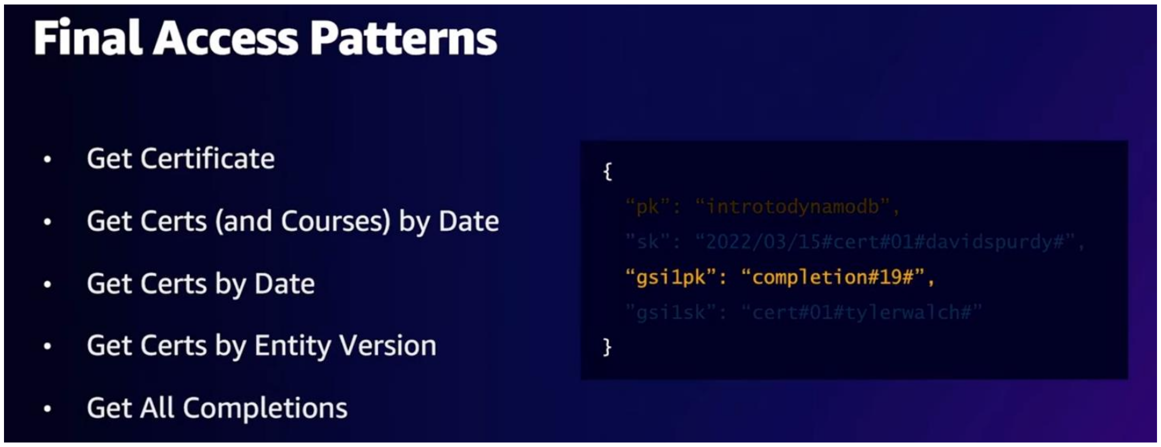
```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#cert#01#davidspurdy#”,
  ”gsi1pk”: ”completion#19#”,
  ”gsi1sk”: ”cert#01#tylerwalch#”
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date
- Get Certs by Entity Version

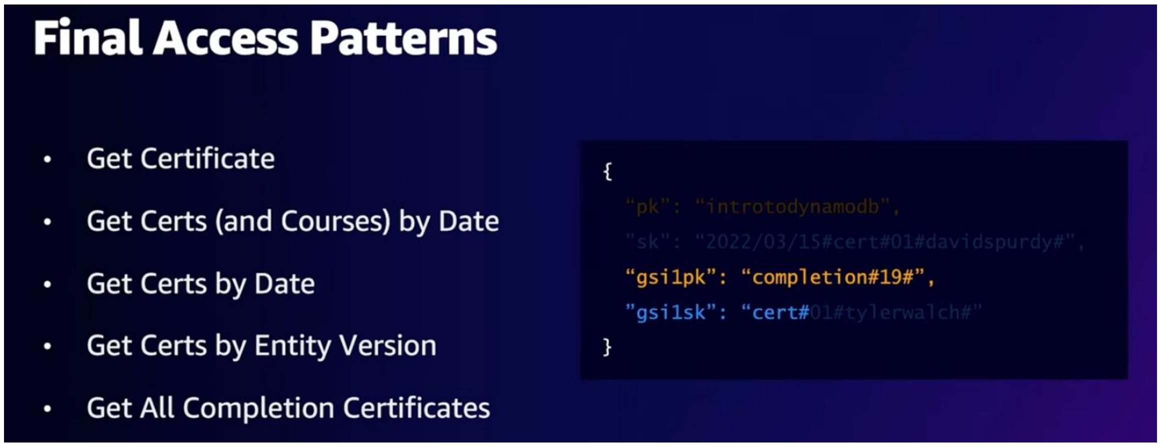
```
{
  “pk”: “introtodynamodb”,
  ”sk”: “2022/03/15#cert#01#davidspurdy#”,
  ”gsi1pk”: ”completion#19#”,
  ”gsi1sk”: ”cert#01#tylerwalch#”
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date
- Get Certs by Entity Version
- Get All Completions

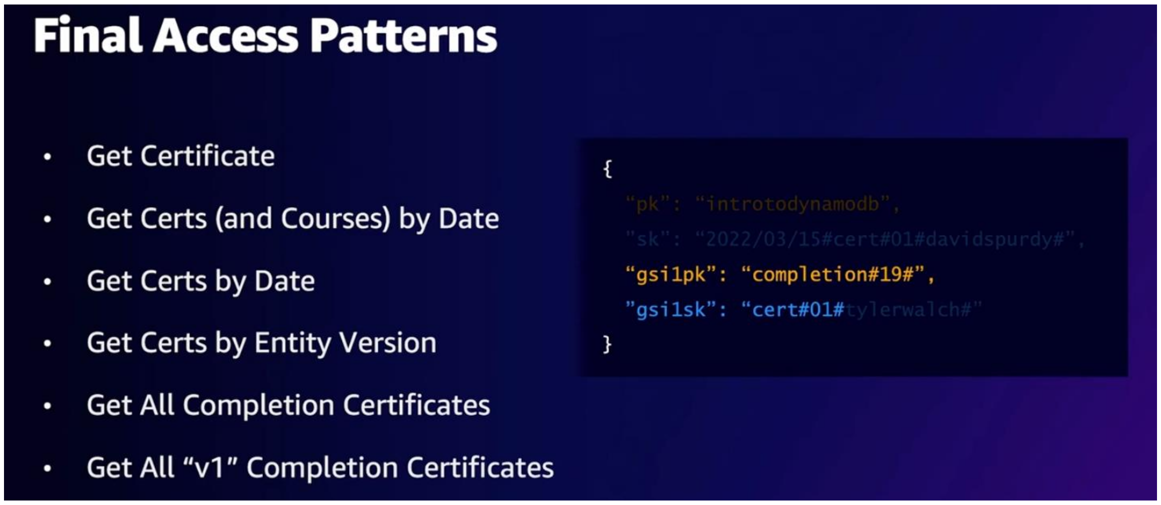
```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#01#davidspurdy#",
  "gsi1pk": "completion#19#",
  "gsi1sk": "cert#01#tylerwalch#"
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date
- Get Certs by Entity Version
- Get All Completion Certificates

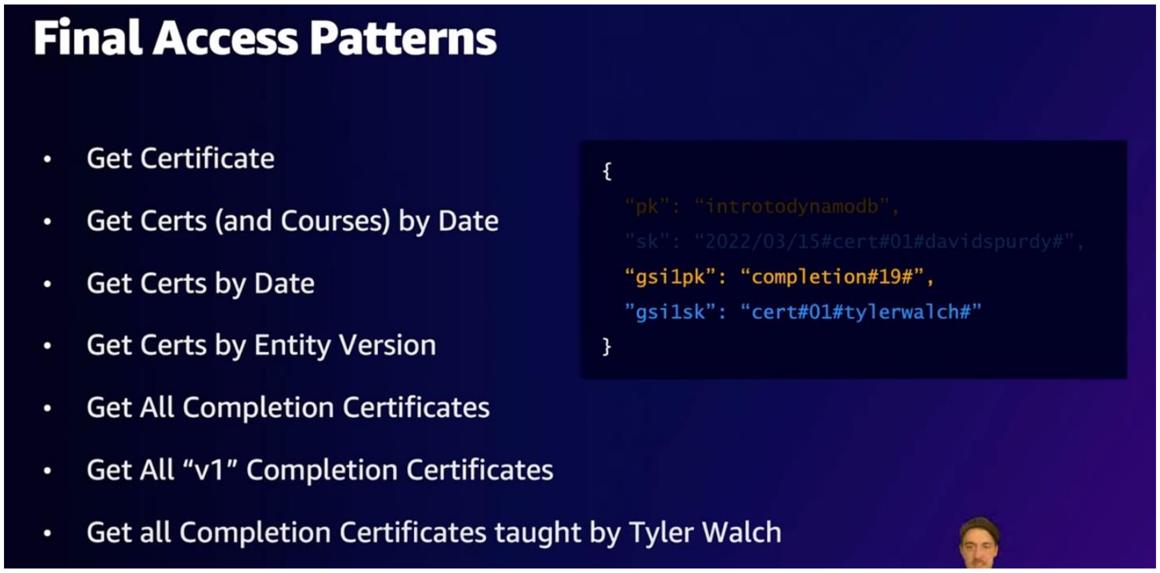
```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#01#davidspurdy#",
  "gsi1pk": "completion#19#",
  "gsi1sk": "cert#01#tylerwalch#"
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date
- Get Certs by Entity Version
- Get All Completion Certificates
- Get All “v1” Completion Certificates

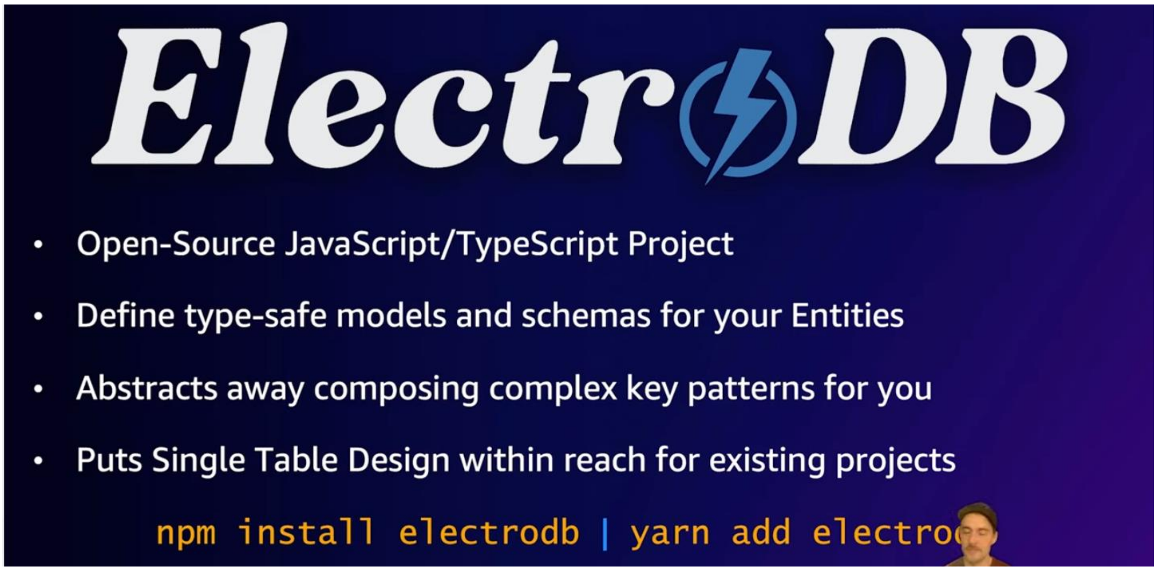
```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#01#davidspurdy#",
  "gsi1pk": "completion#19#",
  "gsi1sk": "cert#01#tylerwalch#"
}
```



Final Access Patterns

- Get Certificate
- Get Certs (and Courses) by Date
- Get Certs by Date
- Get Certs by Entity Version
- Get All Completion Certificates
- Get All "v1" Completion Certificates
- Get all Completion Certificates taught by Tyler Walch

```
{
  "pk": "introtodynamodb",
  "sk": "2022/03/15#cert#01#davidspurdy#",
  "gsi1pk": "completion#19#",
  "gsi1sk": "cert#01#tylerwalch#"
}
```



ElectroDB

- Open-Source JavaScript/TypeScript Project
- Define type-safe models and schemas for your Entities
- Abstracts away composing complex key patterns for you
- Puts Single Table Design within reach for existing projects

```
npm install electrodb | yarn add electro
```