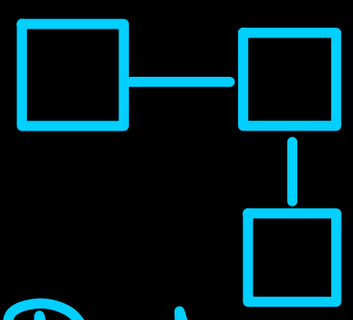
@asksrihari

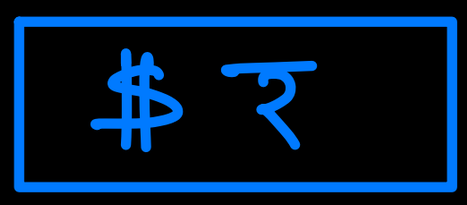
DynamoDB Best Practices

18-07-21

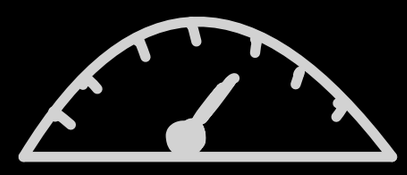
a. Overview



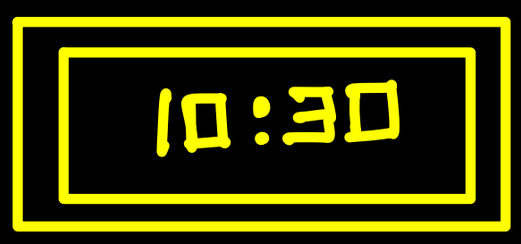
Data Modeling



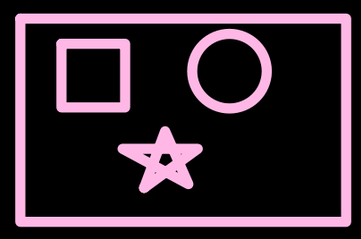
Cost Reduction



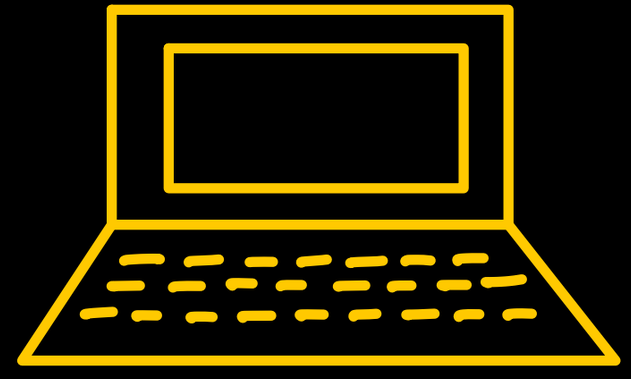
Performance



Time to live



Caching



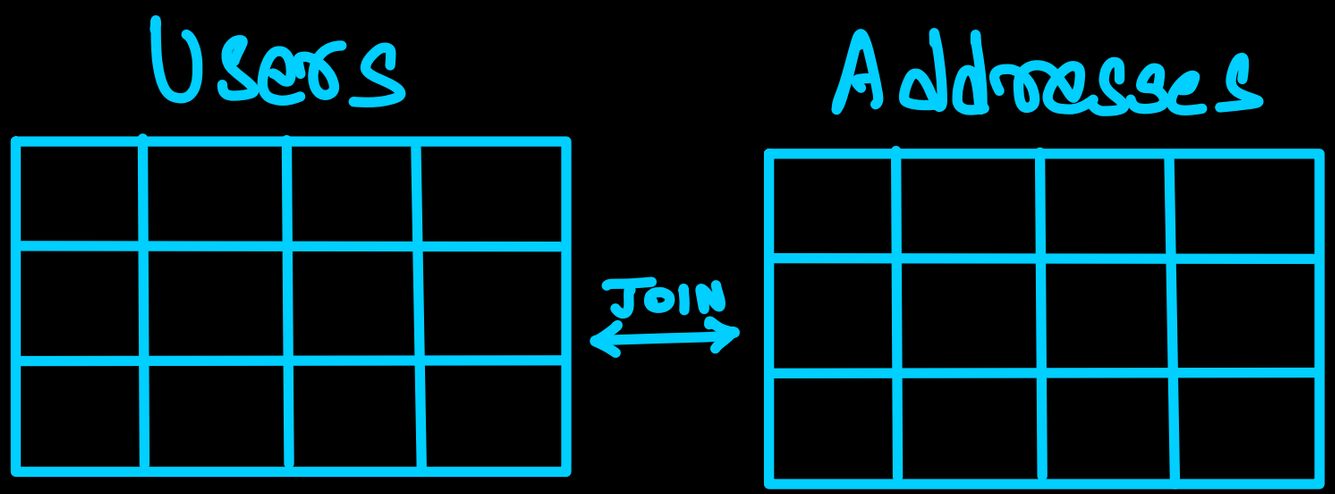
Local Development Environment

b. Relational vs. NoSQL Data Modeling

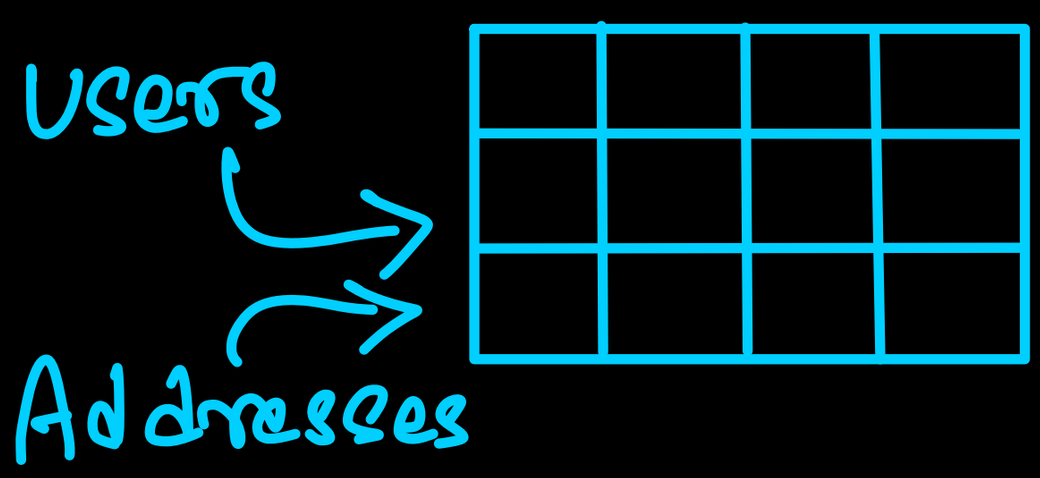
Normalized (3NF)	Data can be denormalized
No data duplication	May have duplication
Less disk usage	More disk space
More CPU usage	Lower CPU usage
Schema and queries are separated	Schema design based on queries
Allows flexible queries	Not flexible
Lower performance and non-scalable	Higher performance

SRIHARI SRIDHARAN

RDBMS



No SQL → Does not stop from using multiple tables



Single table design is possible

Trade Offs of NoSQL Data Modeling

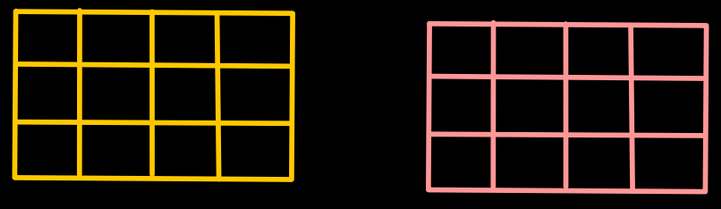
- Get all relate data with one request
  - Low latency
- Requires thinking about queries upfront
  - understand access patterns
  - invest time in design

SRIHARI SRIDHARAN

C. 1:1 and 1:M relationships

Modeling
1:1 relationships

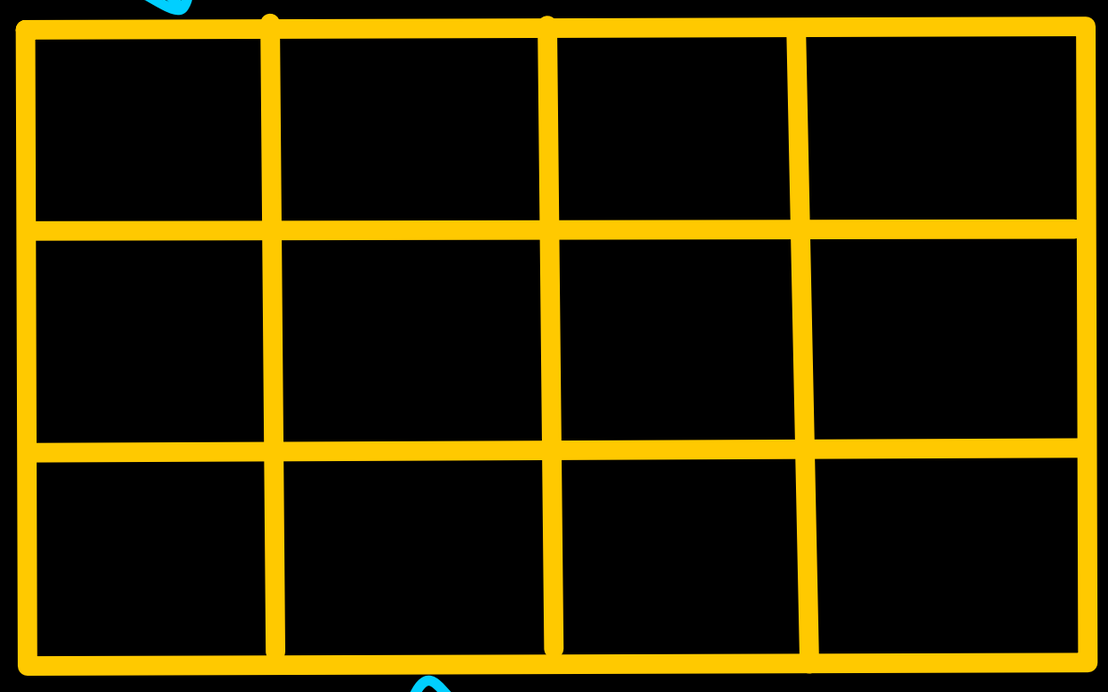
2 tables



Advantages (or) Why do this?

- DynamoDB item size limitation < 400 kB
- Updating smaller item costs less. (even for updating one attribute)
- Can create more indexes

1:M relationships



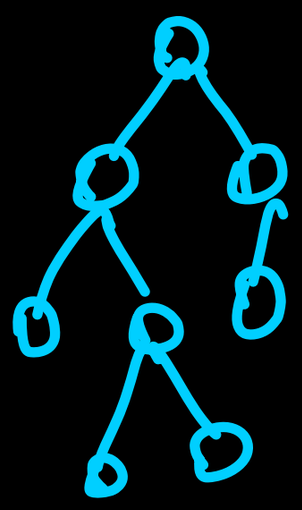
Partition key
eg. Order Id

Sort key
eg. ItemId

eg.
We can get all items in an order.

SRIHARI SRIDHARAN

Representing Hierarchical Data



A different sort key structure

CITY # ZONE # DEPT.

COUNTRY	DEPARTMENT	EMPLOYEE COUNT
India	Chennai # South # Finance	25
India	Chennai # North # Sales	33
India	Mumbai # South # HR	5

Partition Key

Sort Key

Attributes

Query → Get all departments

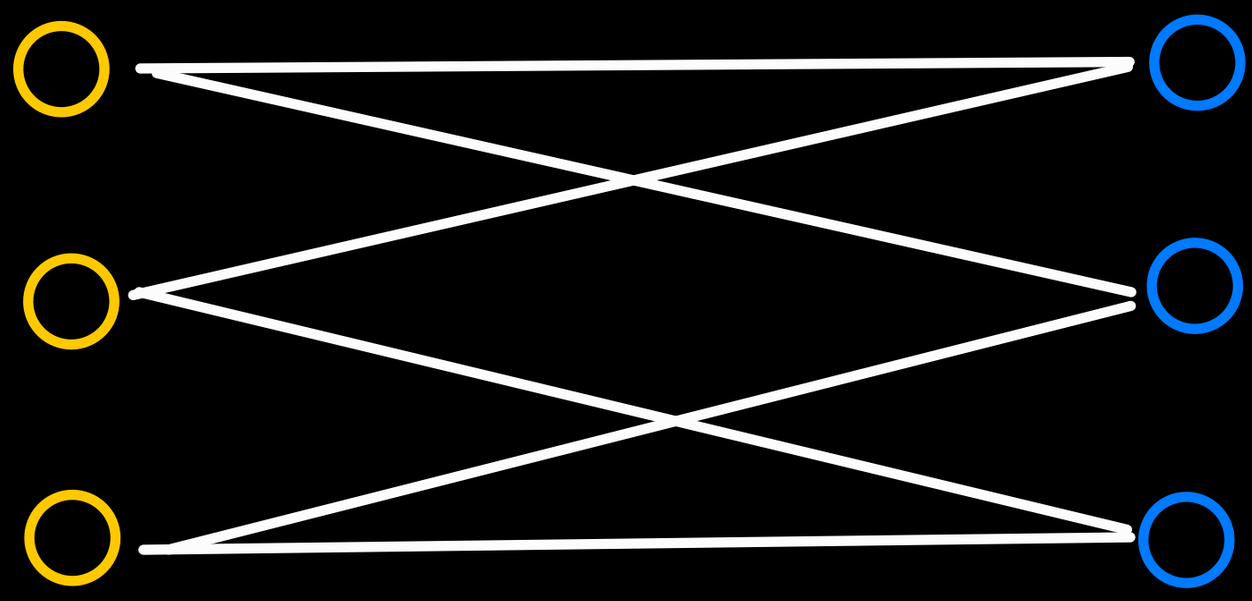
COUNTRY = India

DEPARTMENT Begins with "Chennai"

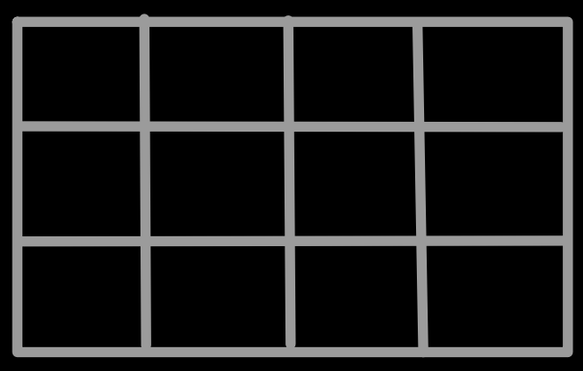
SRIHARI SRIDHARAN

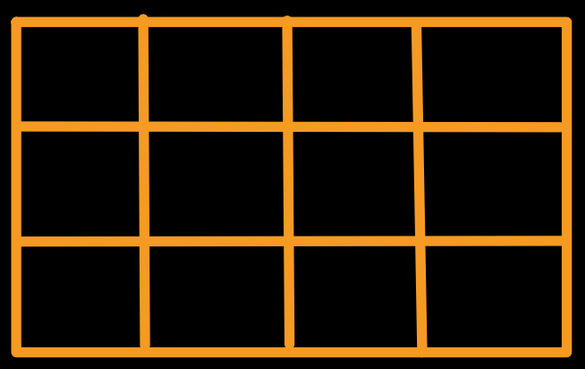
d. Representing Many-to-many relationships

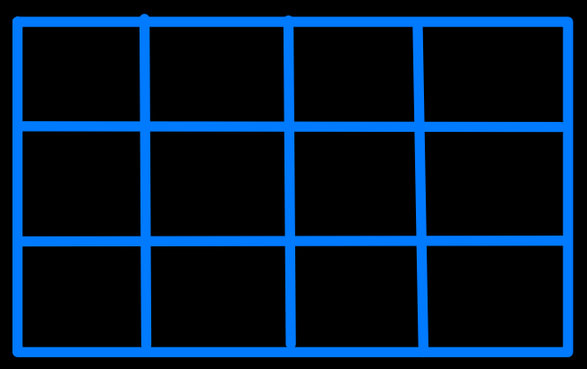
(eg) Students — Courses



RDBMS







STUDENTS — COURSES — STUDENTSTOCOURSES

Association table

Query uses joins

```
SELECT * FROM STUDENTS AS S
JOIN STUDENTSTOCOURSES STC
    ON S.Id = STC.StudentId
JOIN COURSES C ON C.Id = STC.CourseId
```

For M:M in DynamoDB (NoSQL)
we just need a single table

SRIHARI SRIDHARAN

NoSQL (DynamoDB)

Other attributes

Id	Sk	Attributes
S1	S1	Name: Srihari ...
C1	C1	Name: DynamoDB ...
S1	C1	Grade: A ...

Student

Course

Student to Course

Similarly we can have data for other students and courses

Same partition key and sort key

Note:

DynamoDB is schemaless

Important → Use Global Secondary Index to model many to many

Normal table

It was from the perspective of students

GSI

It is from the perspective of courses

NOTE: DynamoDB creates copy of data in a new table for a GSI.

SRIHARI SRIDHARAN

Get all info about students } Single query
Get all info about courses }

No Joins ⟶ Efficient

Key take aways

- One table instead of many.
- Different types in one table.
- Single attribute used for multiple fields.

e. <u>Concatenated attributes</u>

use case → query all orders delivered in July 2021

→ concatenated attributes

OrderId	UserId	Status	Date	Status_Date
1	1 ↑ GSI PK	DELIVERED	2021-01-02	This column stores the status and date as concatenated string
		SHIPPED	2021-05-03	
		CANCELLED	2021-07-01	
		DELIVERED	2021-07-01	

<u>Query</u> : Status_Date BEGINS_WITH "DELIVERED#2021-07"

SRIHARI SRIDHARAN

f. Hot keys

What ?

Some partitions getting uneven load - R/W

How to deal with them?

- Choice of partition keys
- Adaptive Capacity
- Shard Limits

Avoid hot keys

- Ensure uniform load on all partitions
  - ↳ Hard to achieve
- use caching
- Select good partition key

How to select a partition key ?

x Boolean :(Bad choice (only 2 values)

x Numeric values → limited range → Bad.

✓ UUID - good choice for a partition key

OK ✓ Numbers - You need a good range of

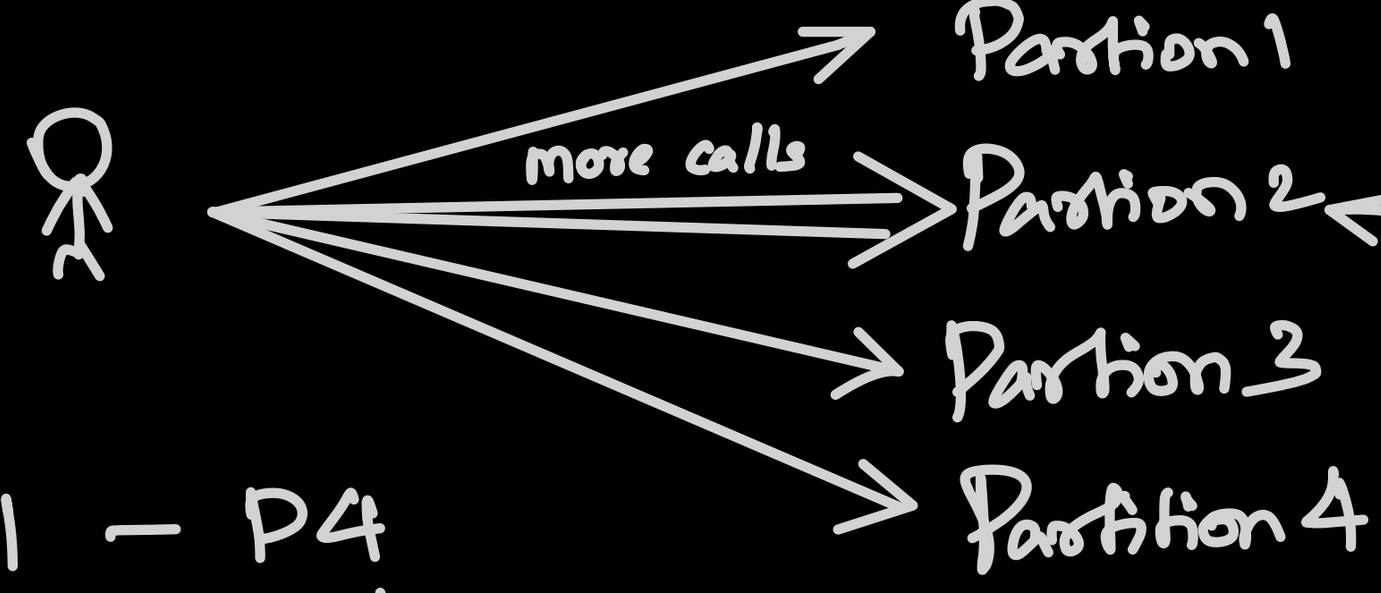
SRIHARI SRIDHARAN

A table with 20000 RCUs - assume it receives read requests of 5000 RCUs

Anything to worry?

Total 20000 RCU

↳ It depends on partitioning



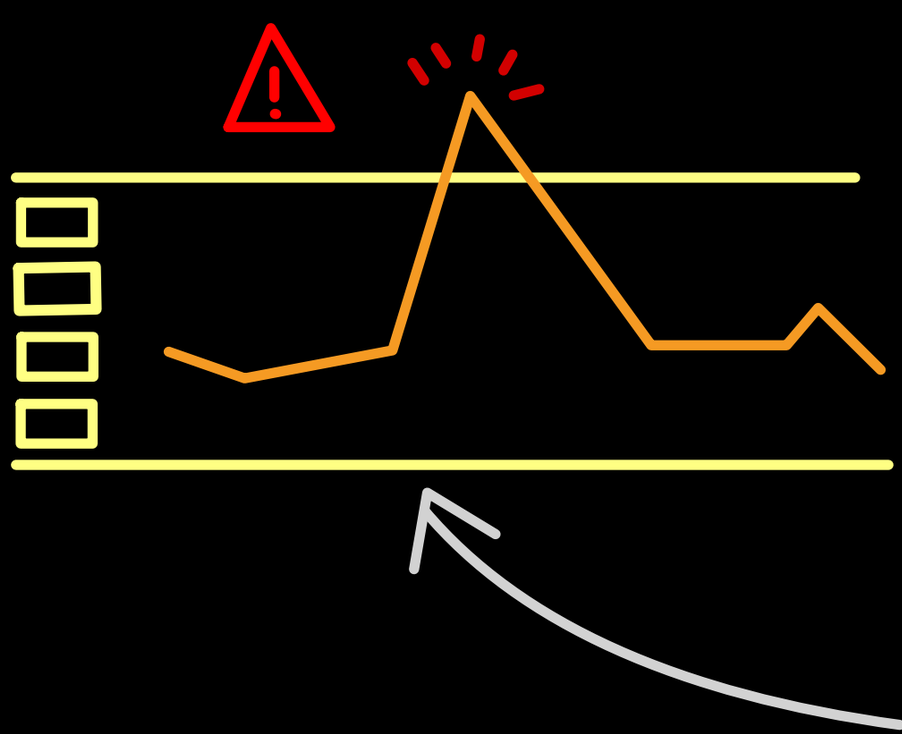
P1 – P4

5000 RCU each.

HOT PARTITION

Possible that this partition exceeds capacity

Adaptive Capacity



It is hard to achieve even load and DynamoDB shifts some capacity to partitions under load.

Works if consumed capacity is less than the provisioned capacity.

SRIHARI SRIDHARAN

Shard Limits

Even with adaptive capacity there are limits. It will work as long as total Consumed RCUs is less than 3000 RCUs and WCUs is less than 1000 WCUs on a single partition.

g. Write Sharding

Some items can be popular than the others ⟶ Can receive too many write requests

eg: Election Voting

↳ Some candidates are more popular - may need more than 1000 WCUs.

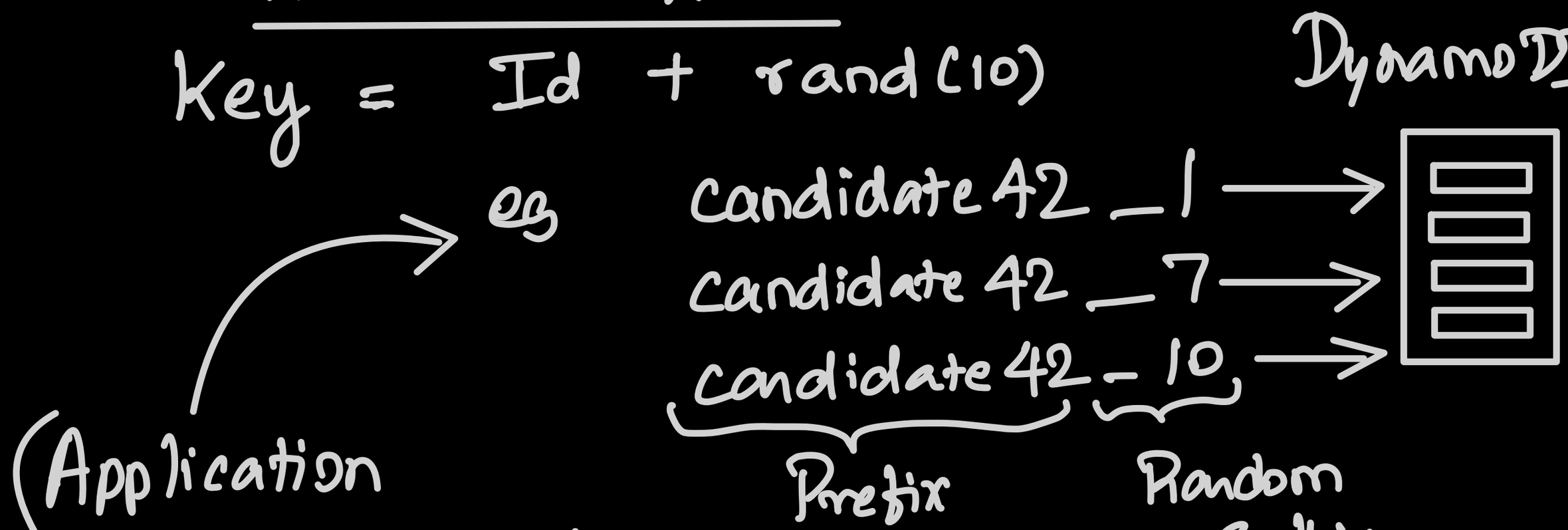
Solution: Split into multiple items and aggregate in the application.

- Random Suffixes
- Calculated Suffixes

SRIHARI SRIDHARAN

Random Suffixes

Key = Id + rand(10)



Calculated Suffixes

Key = date() + orderId() + rand(10)

eg: 21.07.2021 # 23 # 9

21.07.2021 # 23 # 7

h. Reducing costs

As costly as we use RCUs and WCUs

* Send less data to DynamoDB
  - Store big items in S3
  - Use data compression
  - Attributes Projection
  - Split big items into multiple tables

SRIHARI SRIDHARAN

* Exploit temporal access patterns for time series data

- Recent Data ⟶ More RCUs and WCUs
- Older Data ⟶ Lesser RCUs and WCUs
- Very old data ⟶ Move to S3

i Using Scans appropriately

Avoid scans ⟶ Use queries instead

Avoid spikes in read requests from scans ⟶ Can affect production traffic

↓ Read using small page sizes

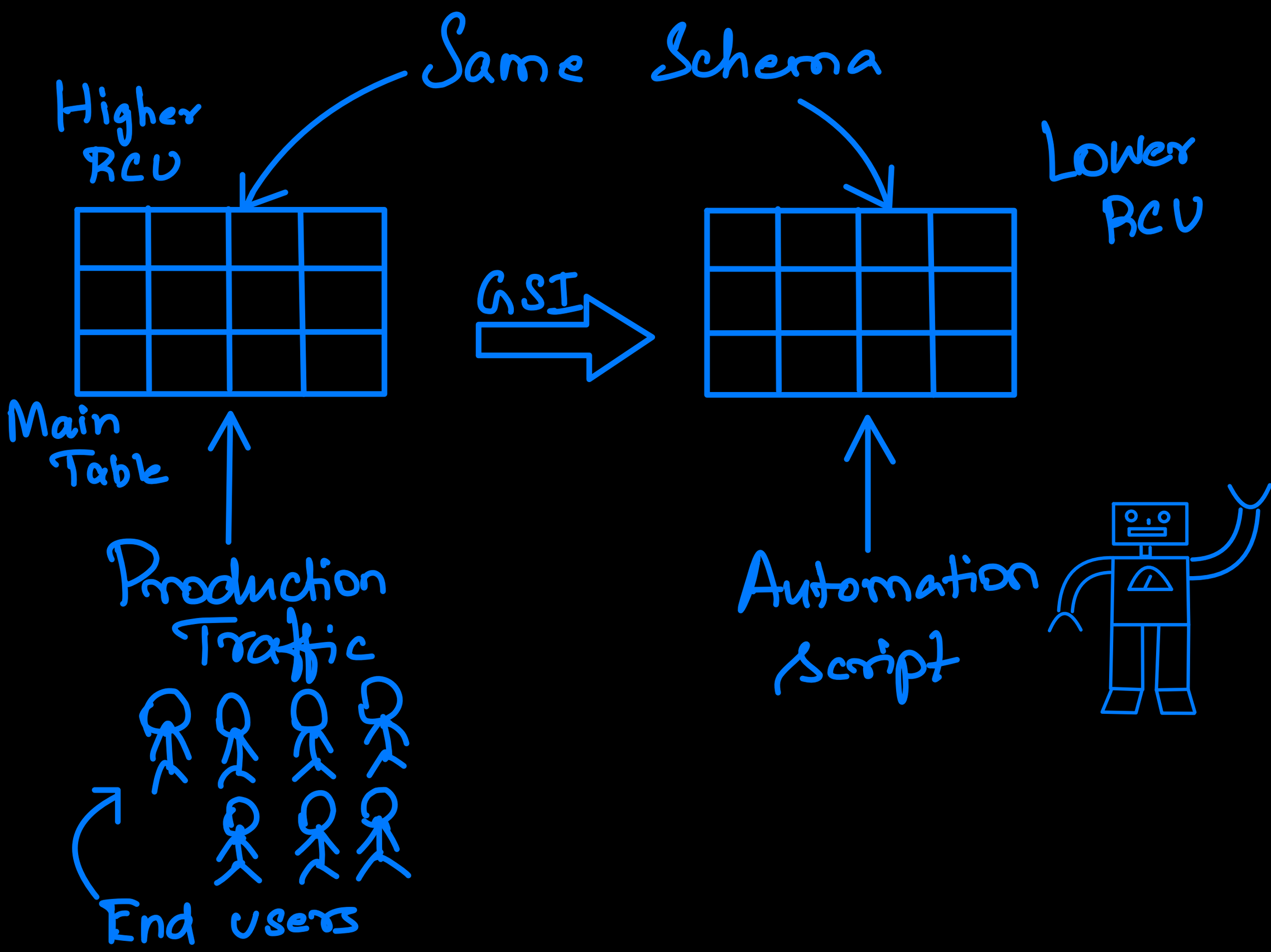
↓ Pause between reading chunks of data

Parallel Scans ⟶ Allows to read data faster

↳ Does not put all load on a single partition.

SRIHARI SRIDHARAN

Shadow Table



- Time-to-live

DynamoDB removes item from the table when they expire

No extra cost

Allows to save money

No need to manually remove items

SRIHARI SRIDHARAN

TTL use cases

- Legal requirements - Don't use TTL when you need to retain data for legal and compliance
- Session Data
- Temporary Data

TTL can be set at the table level

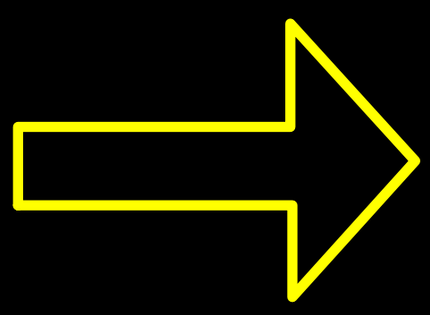
k. Sparse Indexes

Filter a small subset of items

↳ Have a specific criteria

Have a GSI

On that attribute



Only items with this attribute are copied

Not all the items in the main table have this attribute.

l. Caching with DynamoDB

Why? Avoid hot partitions

Decrease latency

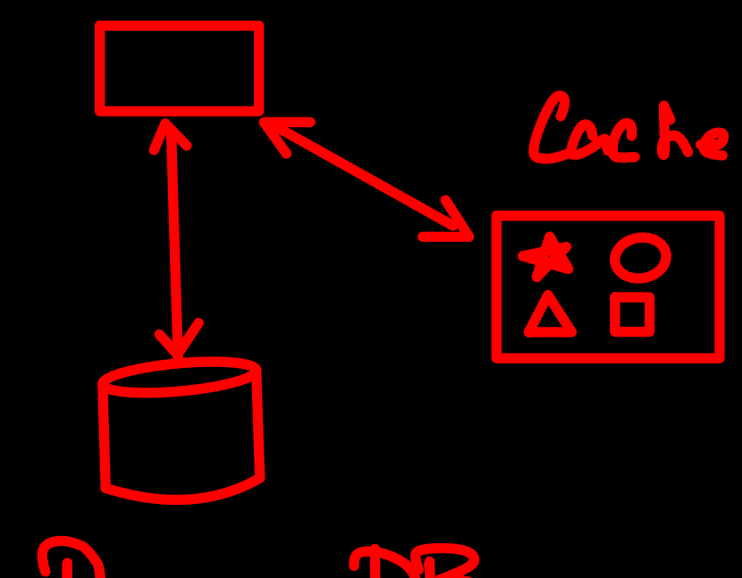
Read-intensive applications

SRIHARI SRIDHARAN

DAX - DynamoDB Accelarator

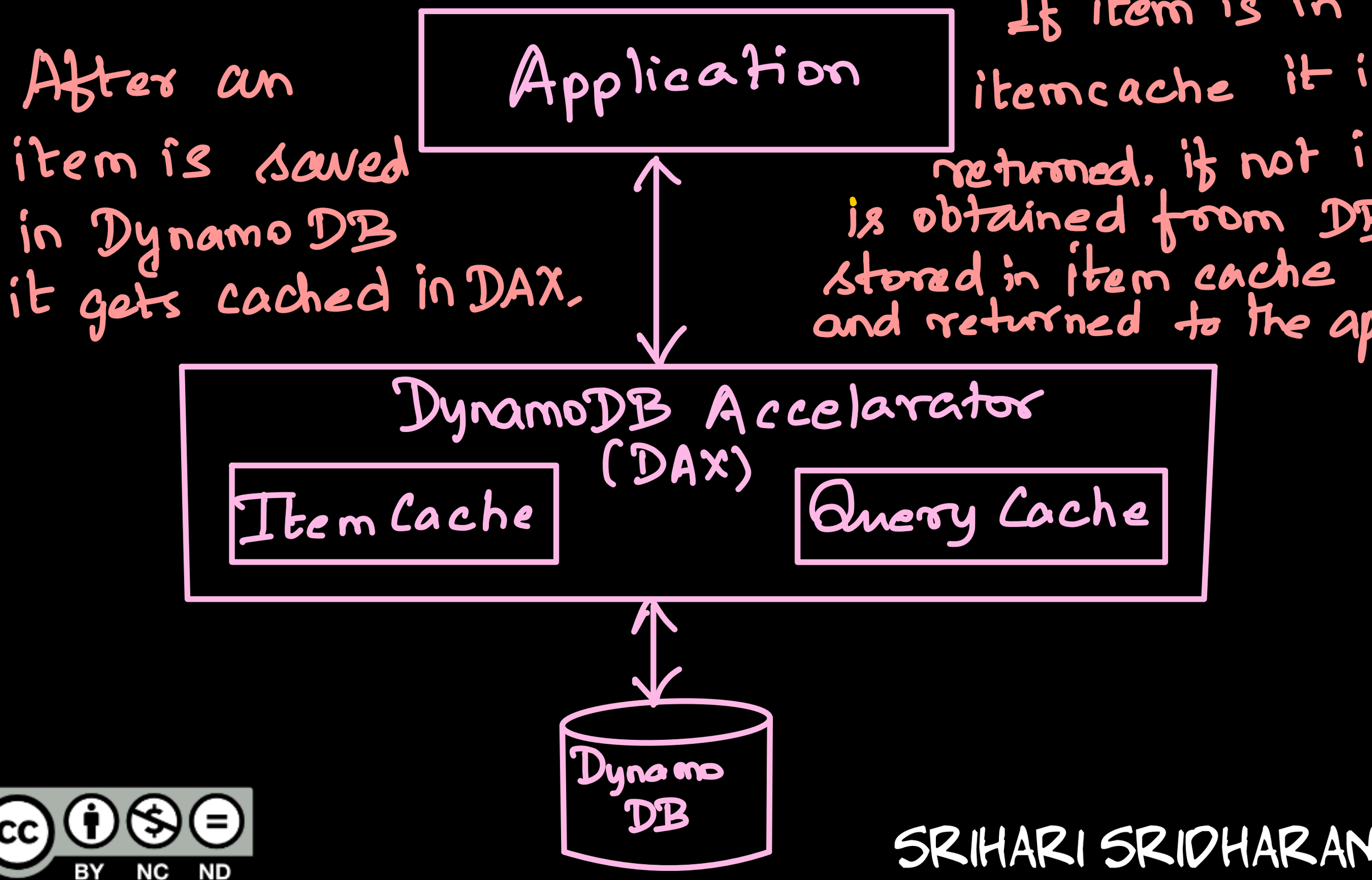
- In-memory accelaration
- sub millisecond latency
- Fully managed
- Scalable upto 10 nodes
- Sits between DynamoDB and the application
- Easy to use and same API

Before DAX



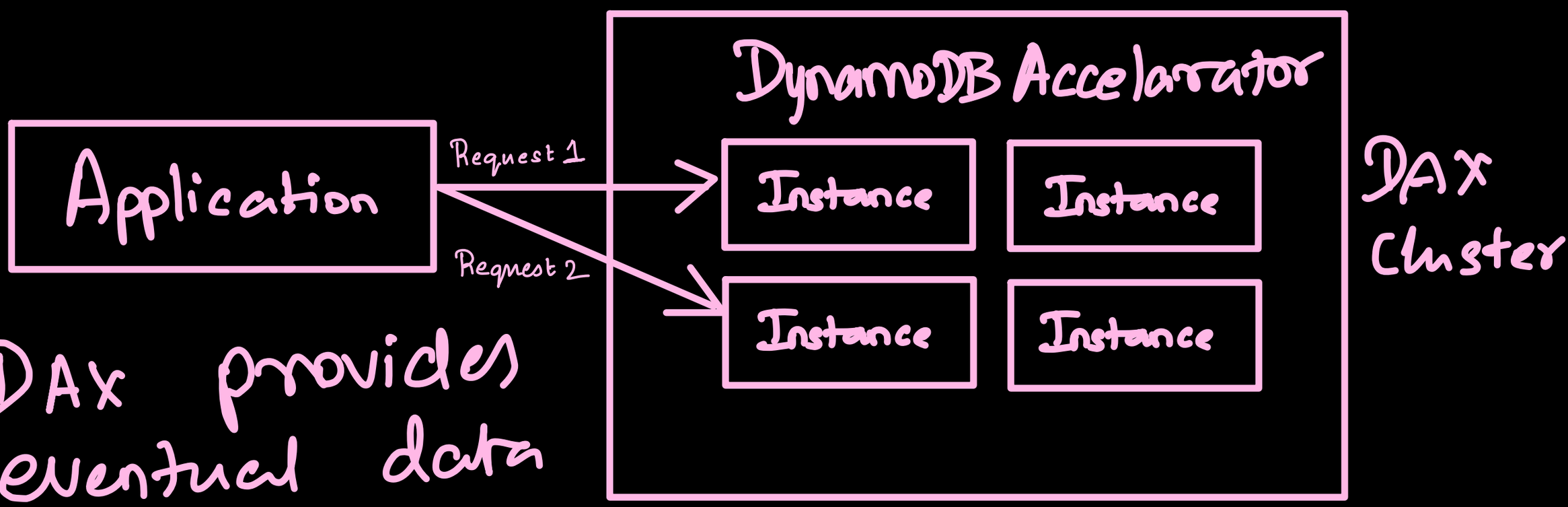
Application managed Caching ☹

This is how DAX Works



SRIHARI SRIDHARAN

DAX has multiple instances



DAx provides eventual data consistency "If your application requires strong consistency you cannot use DAx"

NOTE: Application can bypass DAx and bulk write into DynamoDB, just that items will not be cached.

A DAx cluster can only be accessed by an application that is running in the VPC.

m. DynamoDB and Aws Lambda

λ Lambda - FaaS, Serverless.

↳ can process DynamoDB streams

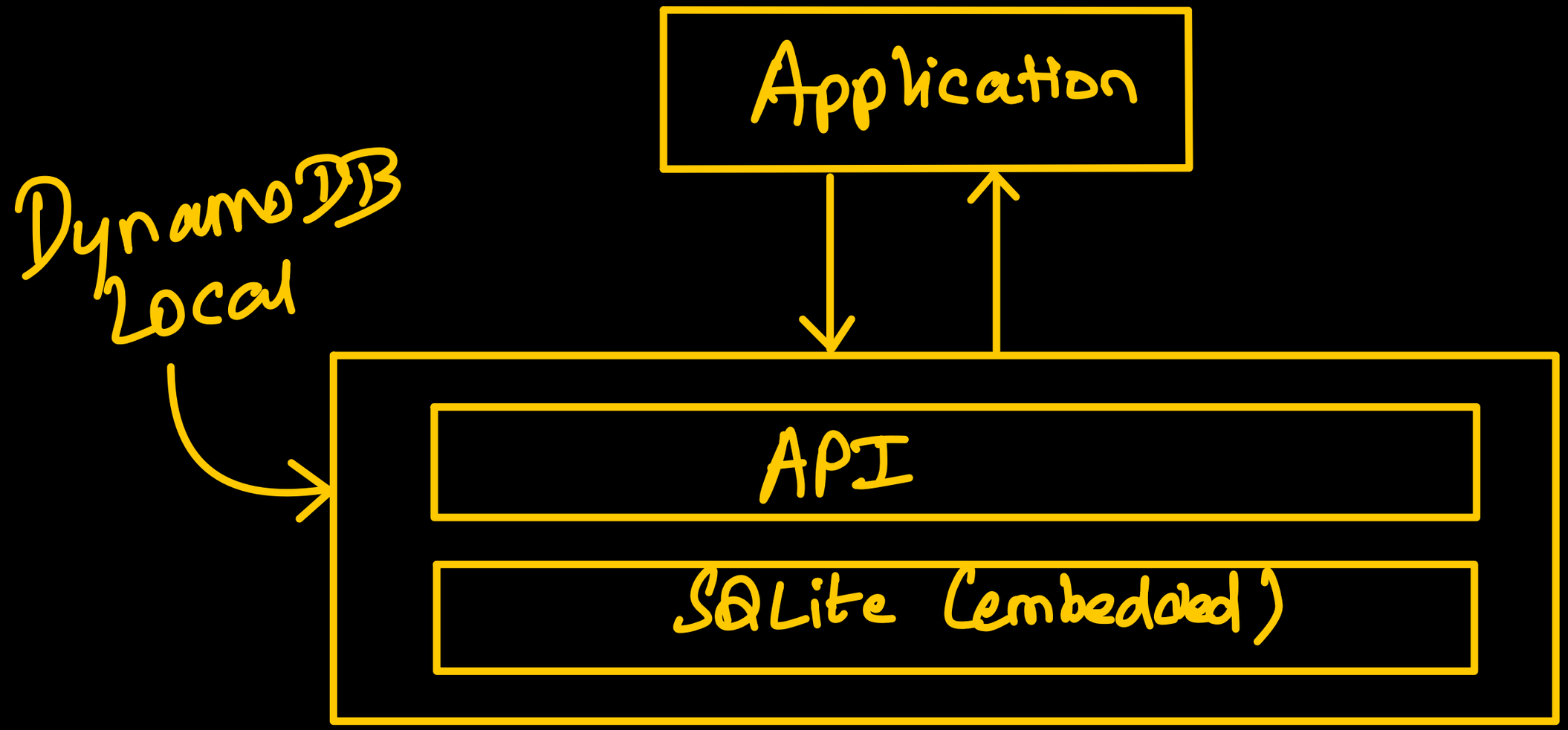
Each shard can be processed by an instance of λ

SRIHARI SRIDHARAN

n. Local Development Environment

Why run locally ?

- Fast
- Convenient - Low cost
- Tests can be run at the same time



2 options - DynamoDB Local , Docker Image

Thanks for reading !!

All the best !!

Warm regards

SRIHARI SRIDHARAN