A Tale of Two Databases with DynamoDB and RDS

Ryan Lewis
CLOUD ENGINEER
@ryanmurakami ryanlewis.dev

Overview

Relational Introduction Service

A Database has to start somewhere

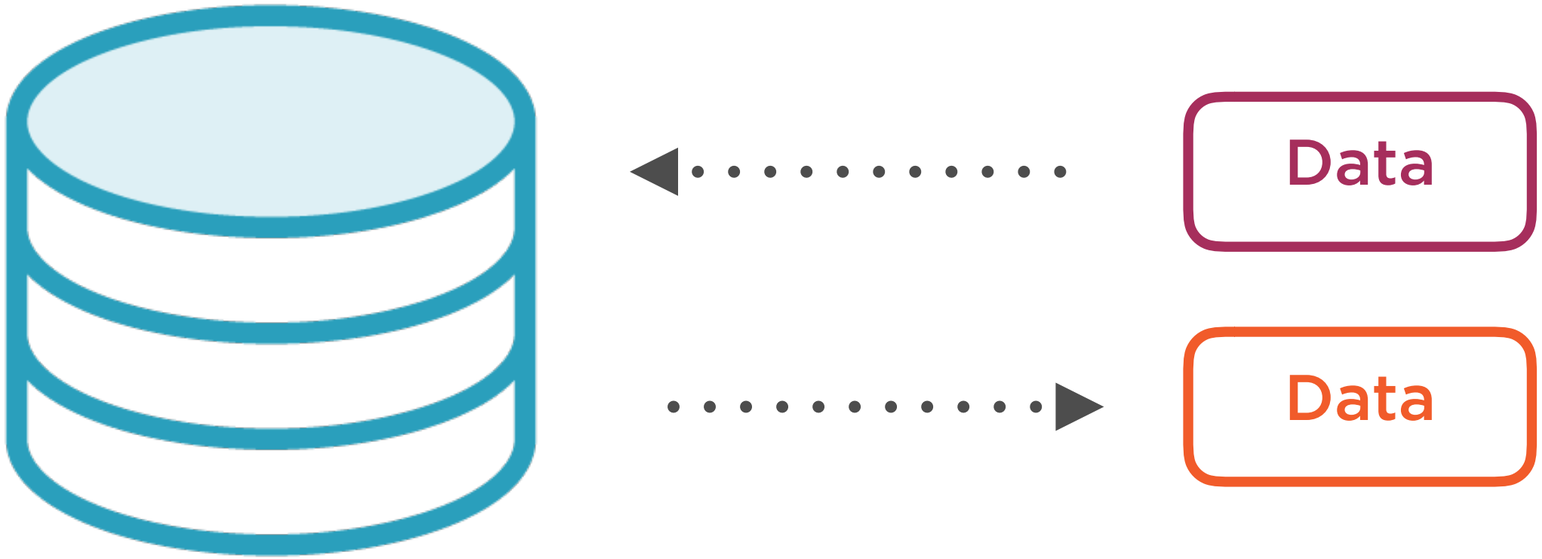
Database Connection Protocol

Back to the DynamoDB Basics

Bringing data to the Table

Relational Database Service (RDS) Overview

Databases



Database Admin

Database Developer

Software Updates

Performance

Backups

Relational Database Service

Managed database instances in AWS running on EC2

RDS Managed Task Examples

Software upgrades

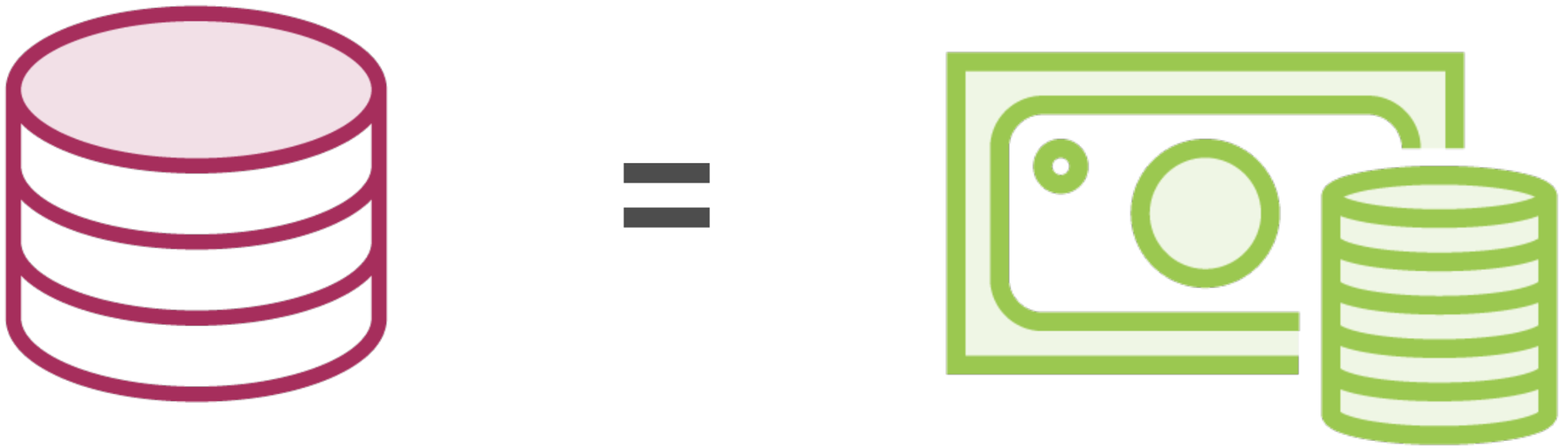
Nightly database backups

Monitoring

RDS Instance Architecture

EC2 Instance Type

*can be changed after the instance has been created



RDS Backups

Occurs daily

Configurable backup window

Backups stored 1 - 35 days

Restore database from backup

Multi-AZ Deployment

Database replication to different Availability Zone

Automatic failover in case of catastrophic event

Database Read Replica

Non-production copy of database

Eventual consistency with source

Useful for running queries on data

Will not be used as failover

Choosing a Database Engine in RDS

RDS Database Options

What database type are you using locally?

How much do you want to spend?

What database type do you have the most experience with?

Which database client do you like the most?

Creating a Database in RDS

Objective

Create a PostgreSQL Database Instance

Connecting to a Database in RDS

Objective

Connect to Postgres Database with Postico

PostgreSQL Clients

pgAdmin

Free, Cross-platform

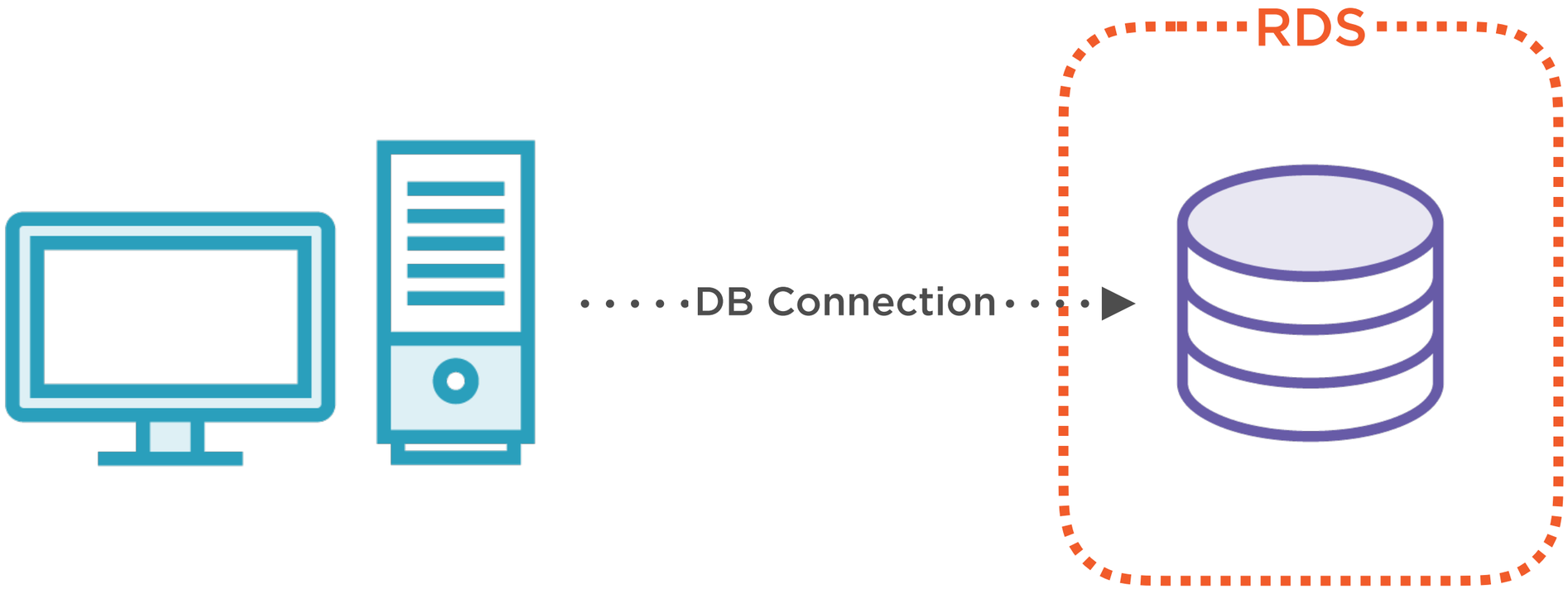
https://www.pgadmin.org/

Postico

Free Trial, Mac Only

https://eggerapps.at/postico/

Interacting with RDS in Code


RDS
DB Connection

Object Relational Mapping (ORM)

Converts between database records and in-code "Objects"

Sequelize

Node.js ORM library

Supports PostgreSQL, MySQL, and more

DynamoDB Overview

Why Do We Need a Database?

Persistance between application restarts

Scalability when activity increases

Database Services in AWS

RDS

Relational

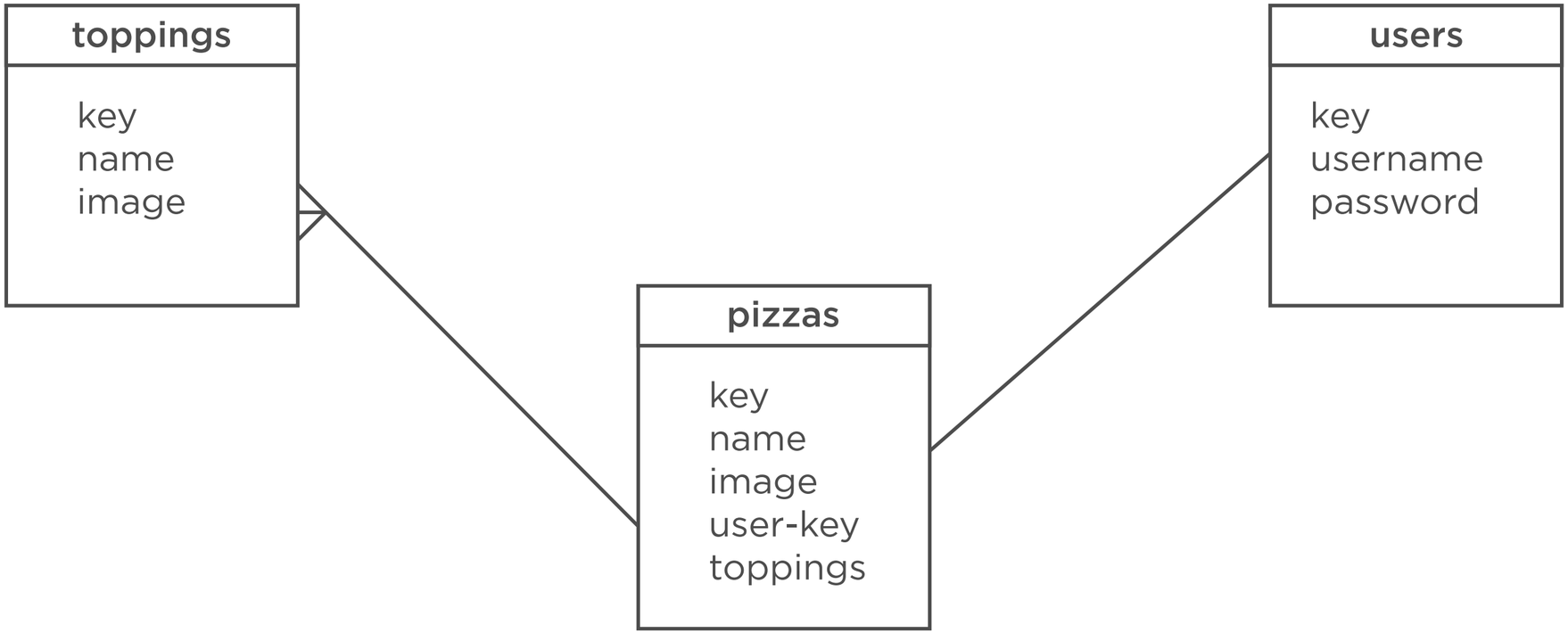
SQL

DynamoDB

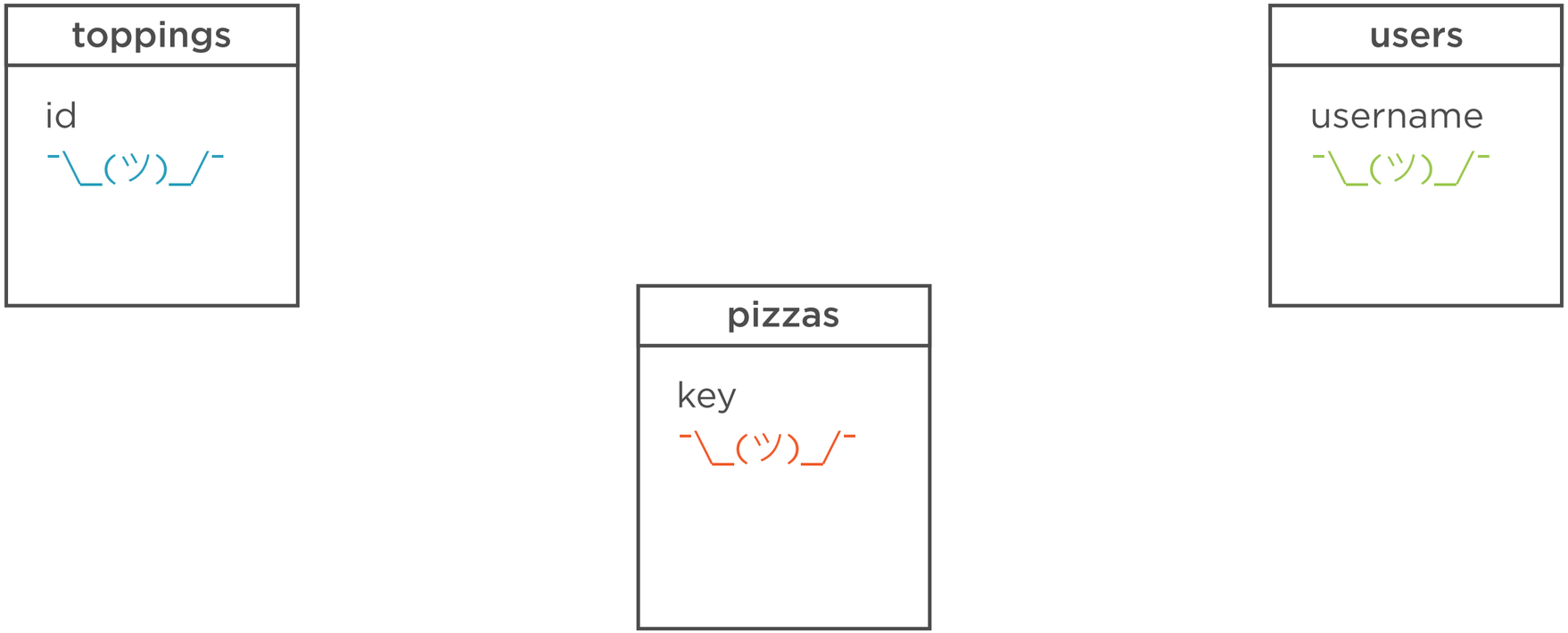
Non-relational

NoSQL

Relational Database Design Example



DynamoDB Design Example



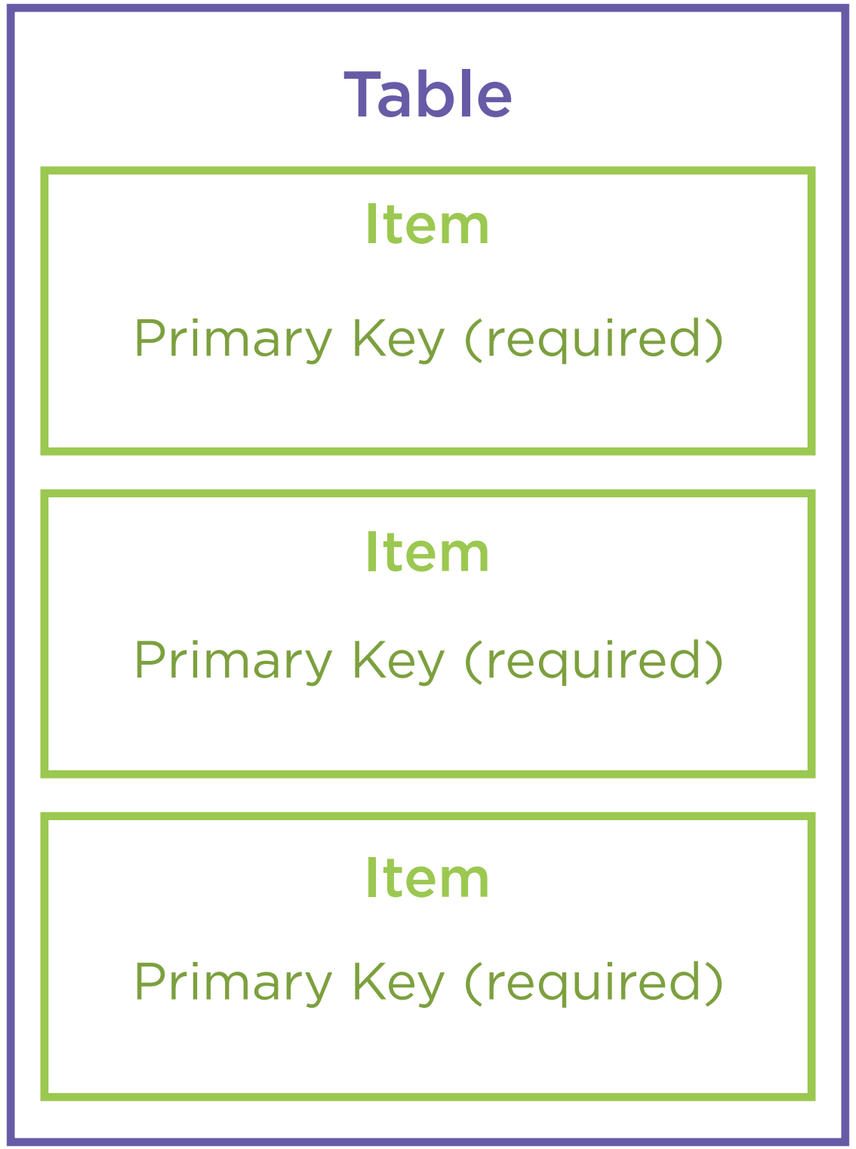
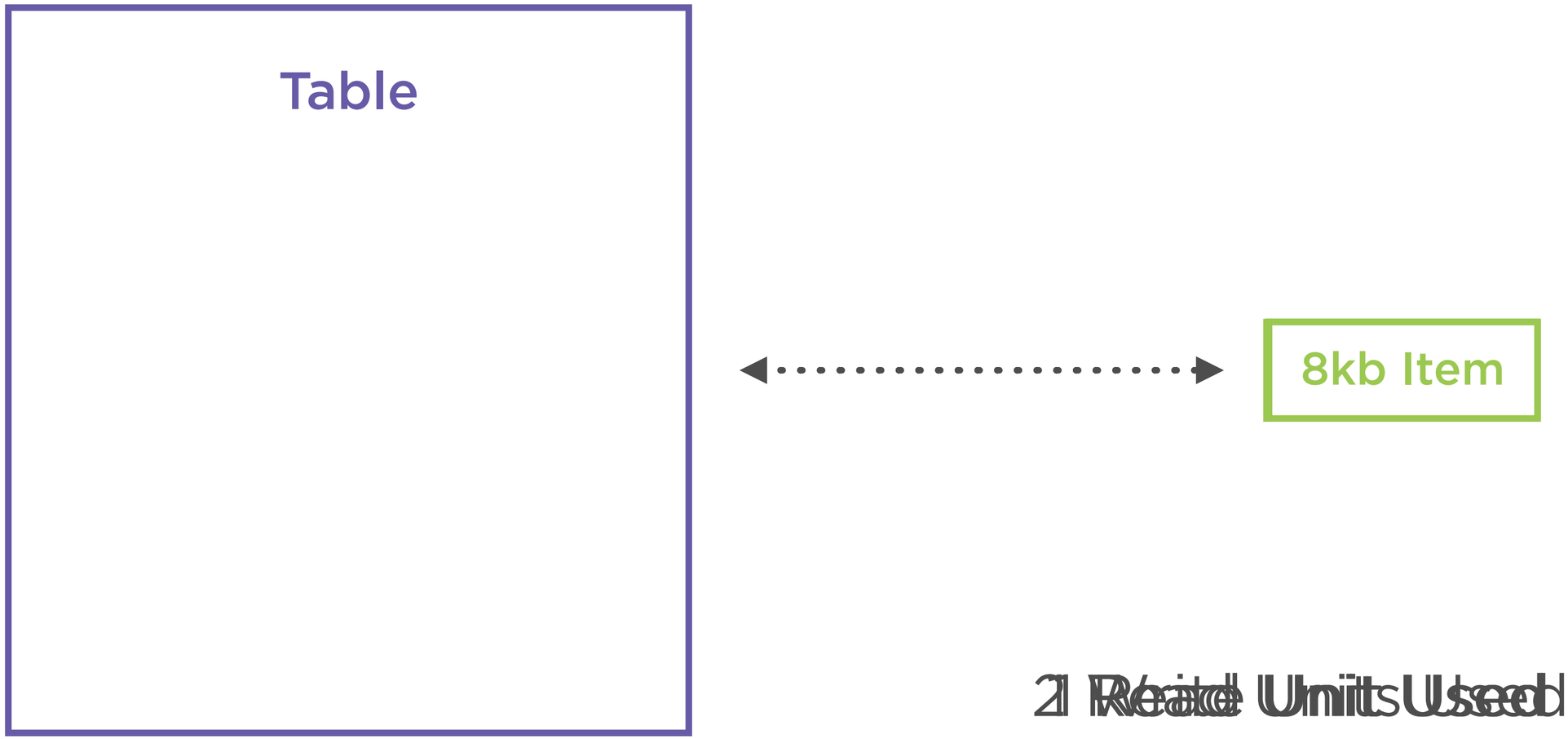


Table
Item
Primary Key (required)
Item
Primary Key (required)
Item
Primary Key (required)

Provisioned Throughput Capacity

Read/Write operations per second provisioned for your DynamoDB table

Provisioned Throughput Example



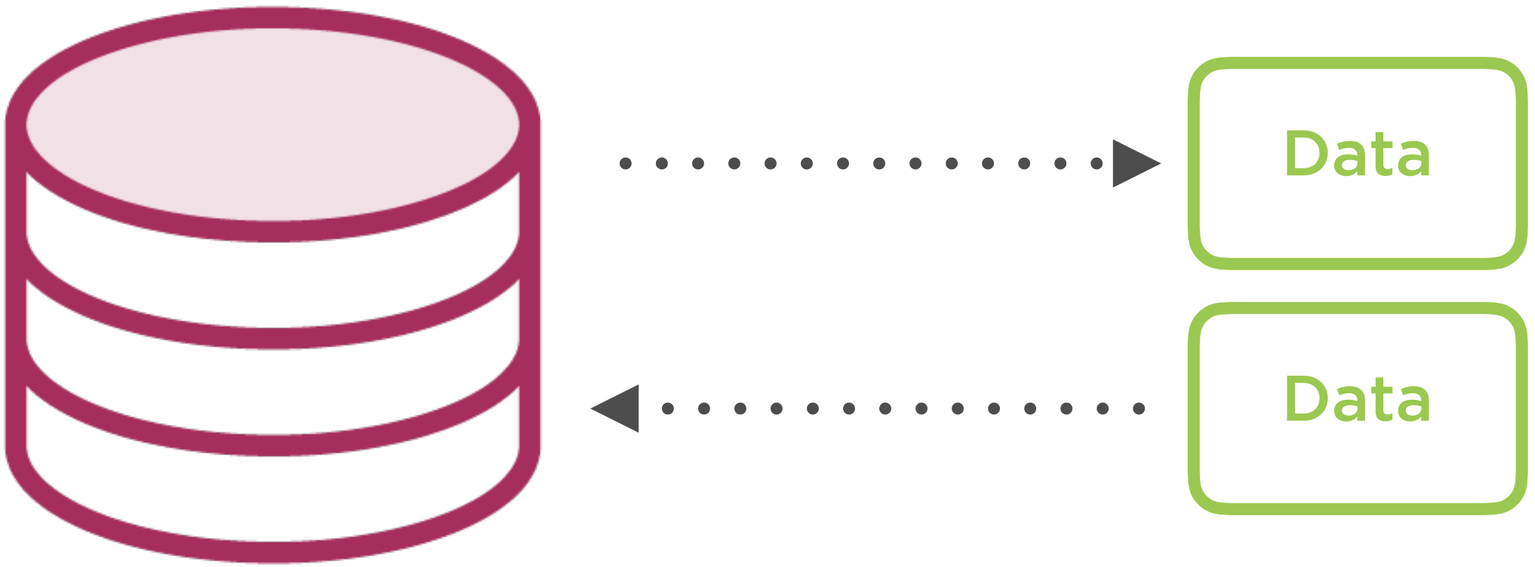
DynamoDB will throttle or deny requests that exceed the table's provisioned throughput capacity

Deciding Between RDS and DynamoDB

Relational or Non-relational?

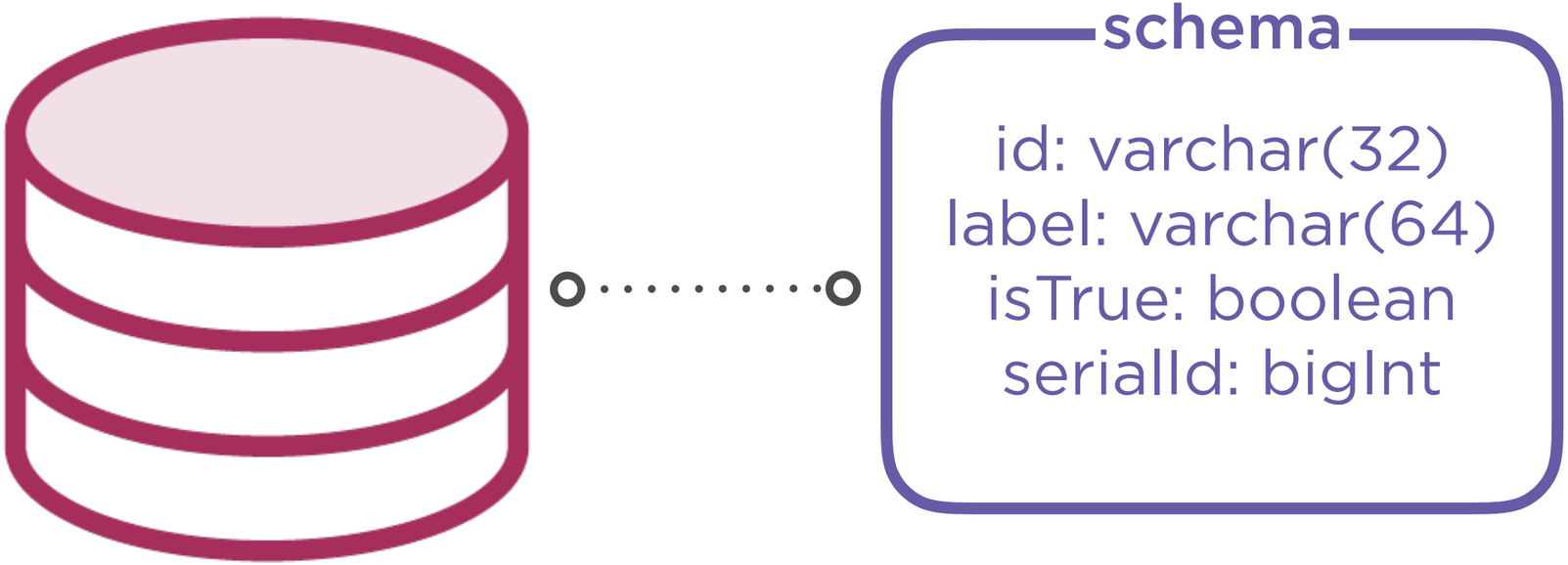
SQL or NoSQL?

Relational DB



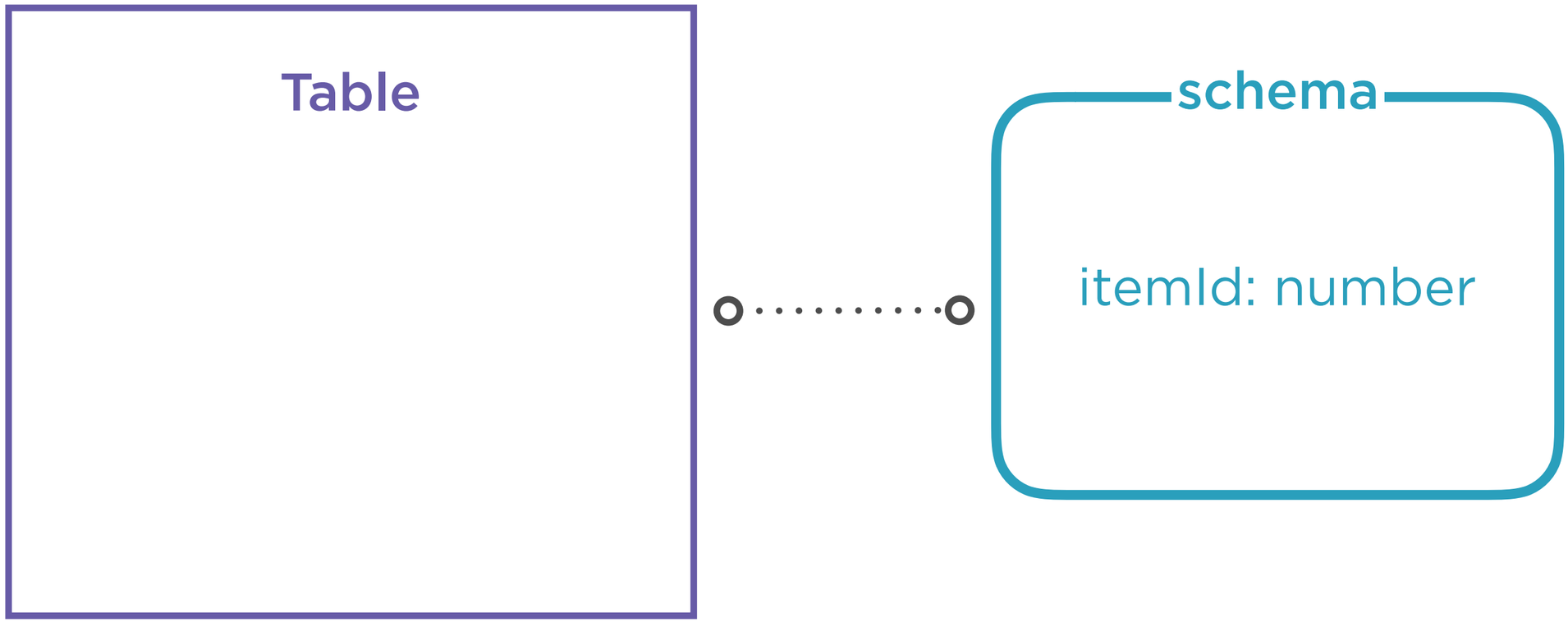
Efficient Data Transfer & Storage

Relational DB



Strict Record Schema

DynamoDB



No Schema; Only Primary Key Restriction

DynamoDB has Storage Flexibility

Relational DB

```
SELECT id,
       label
FROM my_table
WHERE isTrue = true
```

Easy Querying with SQL

DynamoDB

Limited Query Properties

RDS has
Query Flexibility

Storage Flexibility vs Query Flexibility

DynamoDB RDS

Creating a Table in DynamoDB

Objective

Create DynamoDB Tables for Toppings and Users

Connecting to DynamoDB with Code

Objective

Create Module to Interact with DynamoDB

To run on EC2 add
AmazonRDSFullAccess and
AmazonDynamoDBFullAccess
to the *pizza-ec2-role*

and

allow access to RDS instance from
pizza-ec2-sg by adding to the RDS
instance security group

Conclusion

Summary

Relational database fundamentals

Pepperoni Pizzabase

The key to the table

Dynamo launched

Up Next:

Automate Your App with Elastic Beanstalk and CloudFormation

Elastic Beanstalk & CloudFormation

Automated Application Deployment

Infrastructure as Code