Data Modeling with Amazon DynamoDB

Rob McCauley

DynamoDB Specialist Solution Architect
Amazon Web Services

Amazon DynamoDB: Built to Scale

Performance at scale

- Consistent, single-digit millisecond read and write performance
- Nearly unlimited throughput and storage

Enterprise ready

- Data encryption at rest
- Global replication
- Up to 99.999% availability SLA

No servers to manage

- Fully managed, scale-to-zero serverless database
- Massive scalability

Built-in integration with others AWS services

- Logging, monitoring, and analytics
- Applications that span multiple AWS services

But, DynamoDB doesn't have JOINs!

Agenda

1. Why Model?
2. Customer Orders case study
3. Demo

"Show me order 234"

SQL JOIN

Relational Database

customers

orders

order_lines

products

Get-Item

Get-Item

Query

Get-Item

Get-Item

DynamoDB

customers

orders

order_lines

products

Relational to DynamoDB:

Single-Table philosophy

Deciding how to combine many entities into one set with SQL

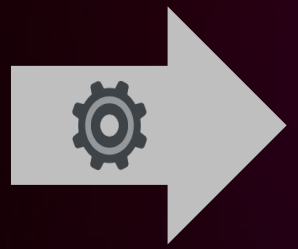
customers

orders

order_lines

products

SQL?



DynamoDB Table

DENORMALIZED with JOIN

Get-Item (cust-2, ord-2-line-3)

Orders

A single, complete record is returned

cust-2	ord-2-line-3	2022-12-30	pr-300

STACKED with UNION

Query (cust-2)

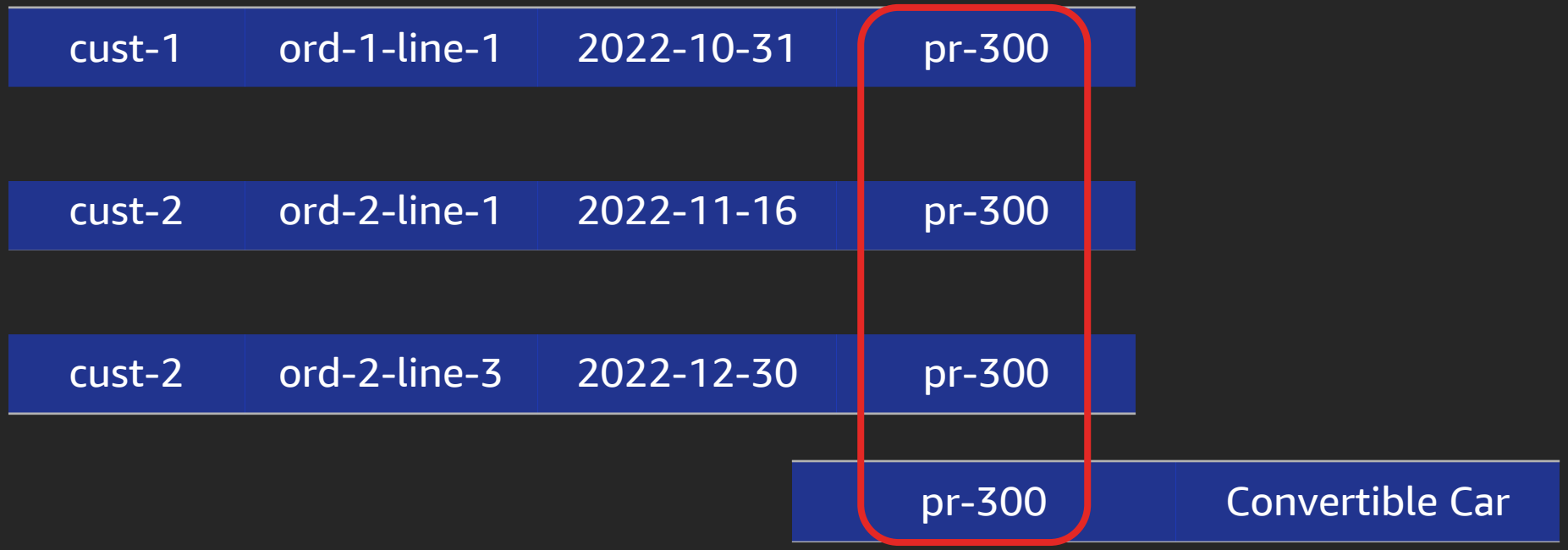
616-838-3000	cust-2

Returns an Item Collection

cust-2	ord-2-line-1	2022-11-16	pr-300
cust-2	ord-2-line-2	2022-12-26	pr-400
cust-2	ord-2-line-3	2022-12-30	pr-300

STACKED with UNION

Query GSI (pr-300)



Demo:
Relational data in a single DynamoDB table

Quick Recap

1. DynamoDB can support your relational access patterns
2. Define a model for your access patterns
3. Best practice: Consider single table design

Thank you!

Rob McCauley
mccaul@amazon.com
robmccauley