Denormalizing the Non-relational Data Model

Leonard Lobel
CTO, SLEEK TECHNOLOGIES

lennilobel.wordpress.com

Handling Relational Workloads

Cosmos DB is non-relational

No JOINs

No relational constraints

Paradigm shift

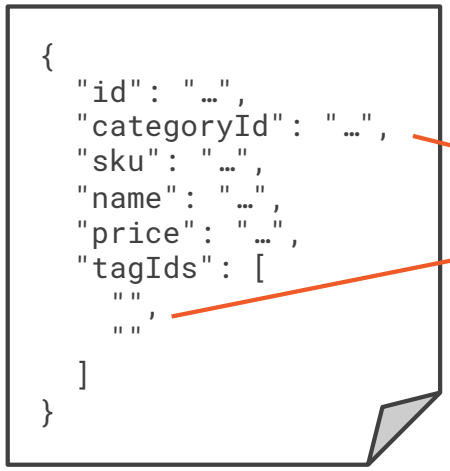
Need to use different techniques to implement relations between entities

Denormalize

Duplicate specific properties

Satisfy queries with a single request

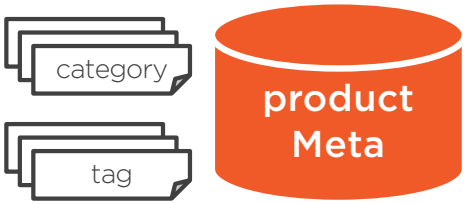
Querying Products



Category name?

Tag names?

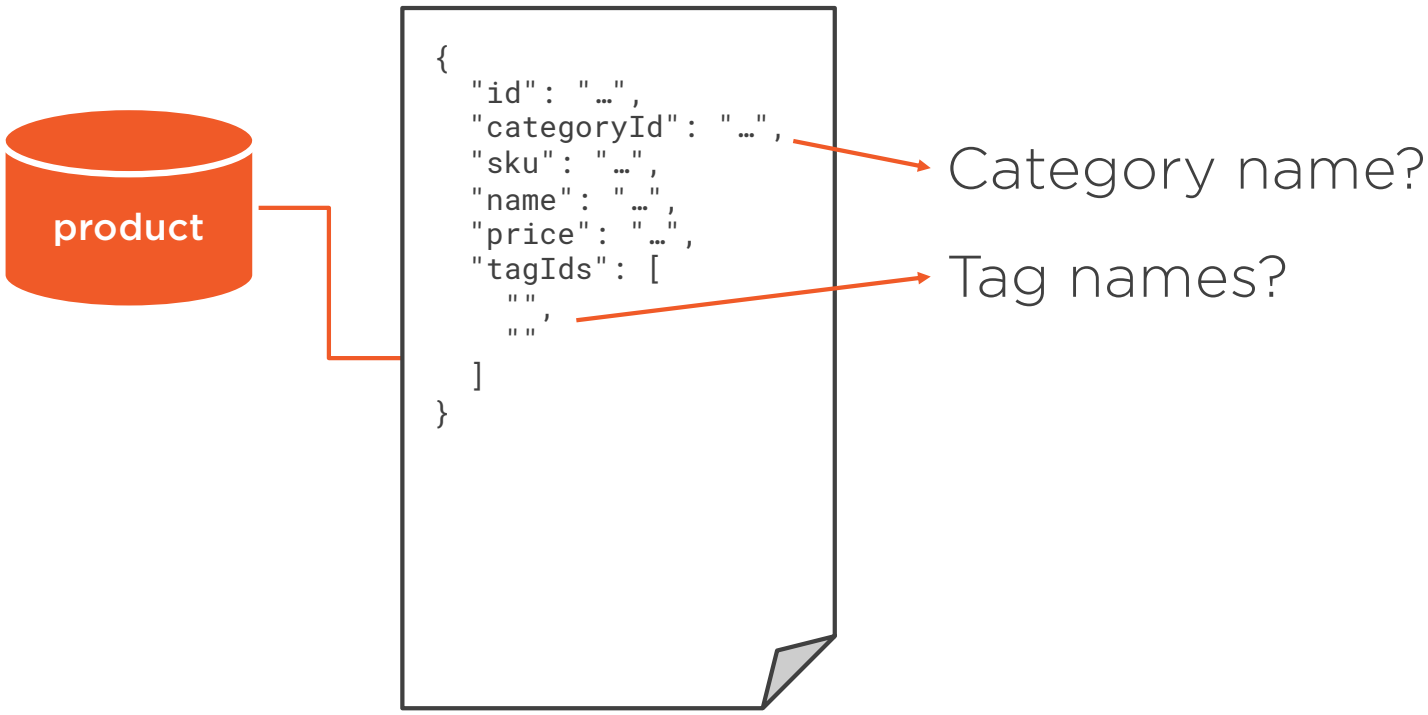

```
SELECT * FROM c WHERE c.categoryId = '<categoryId>'
```



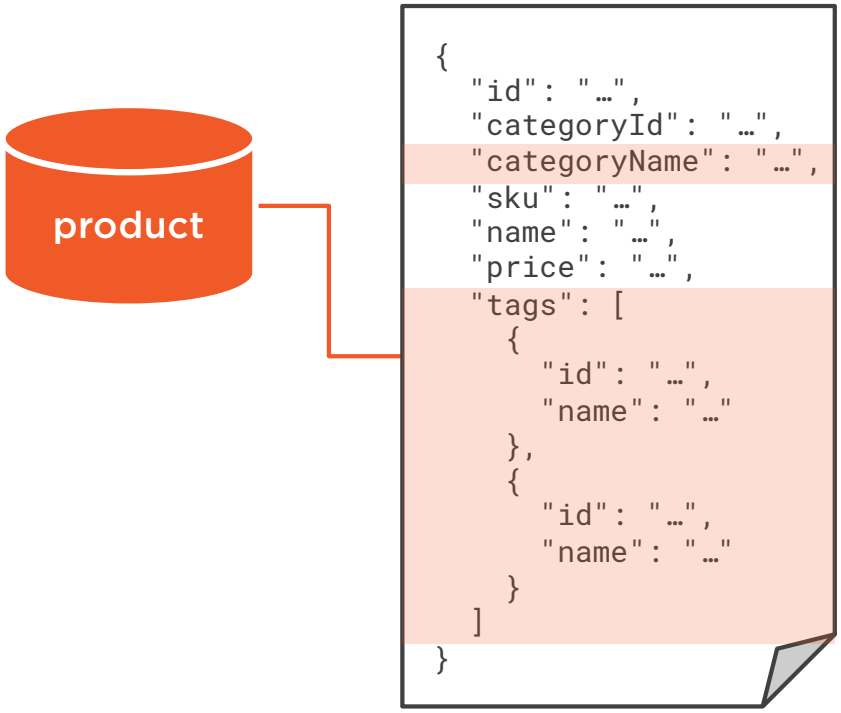

```
SELECT c.name FROM c WHERE c.type = 'category'
 AND c.id = '<categoryId>'
```

```
SELECT c.name FROM c WHERE c.type = 'tag'
 AND c.id IN '<tagId1>, <tagId2>, …, <tagIdN>'
```

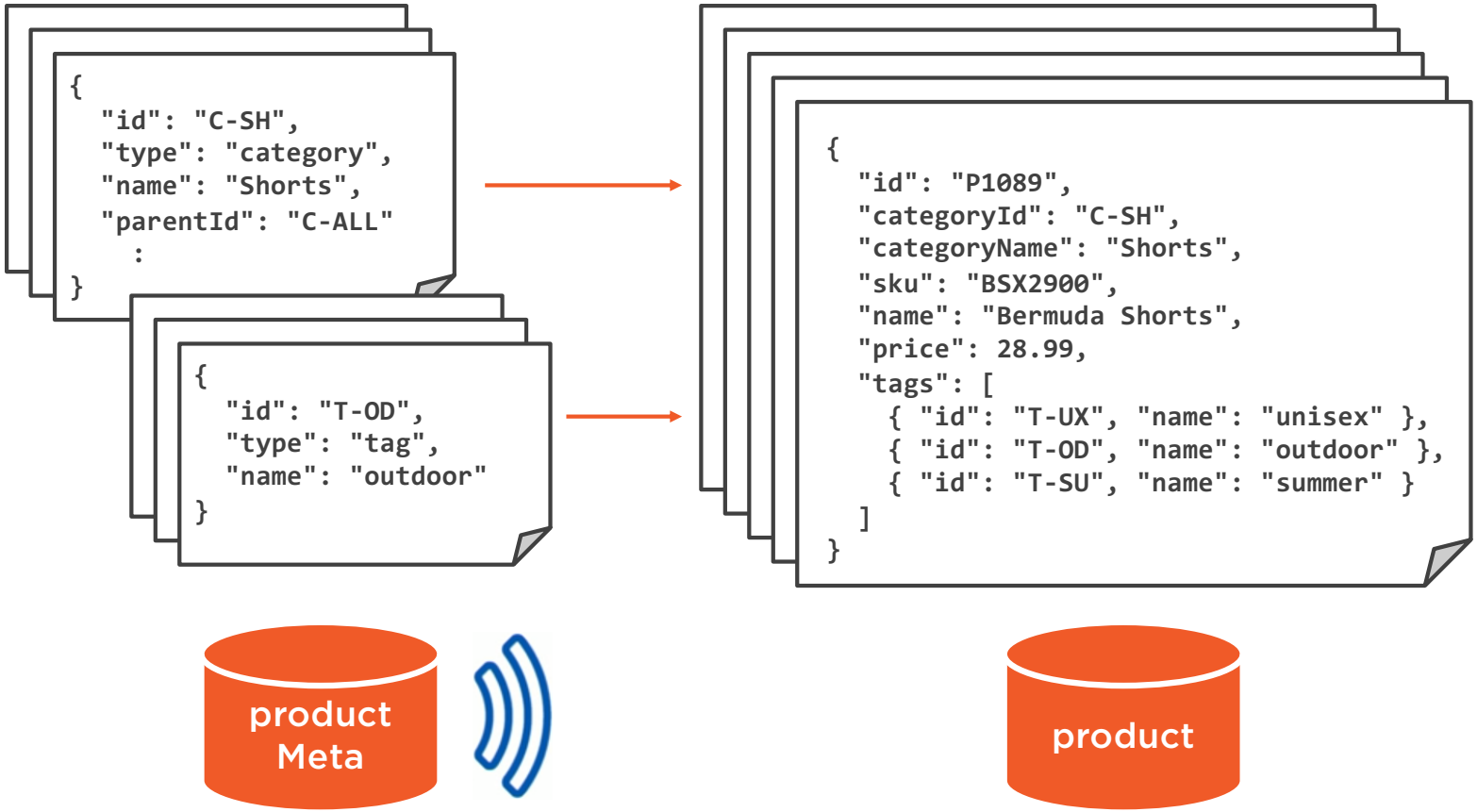
Denormalizing Products



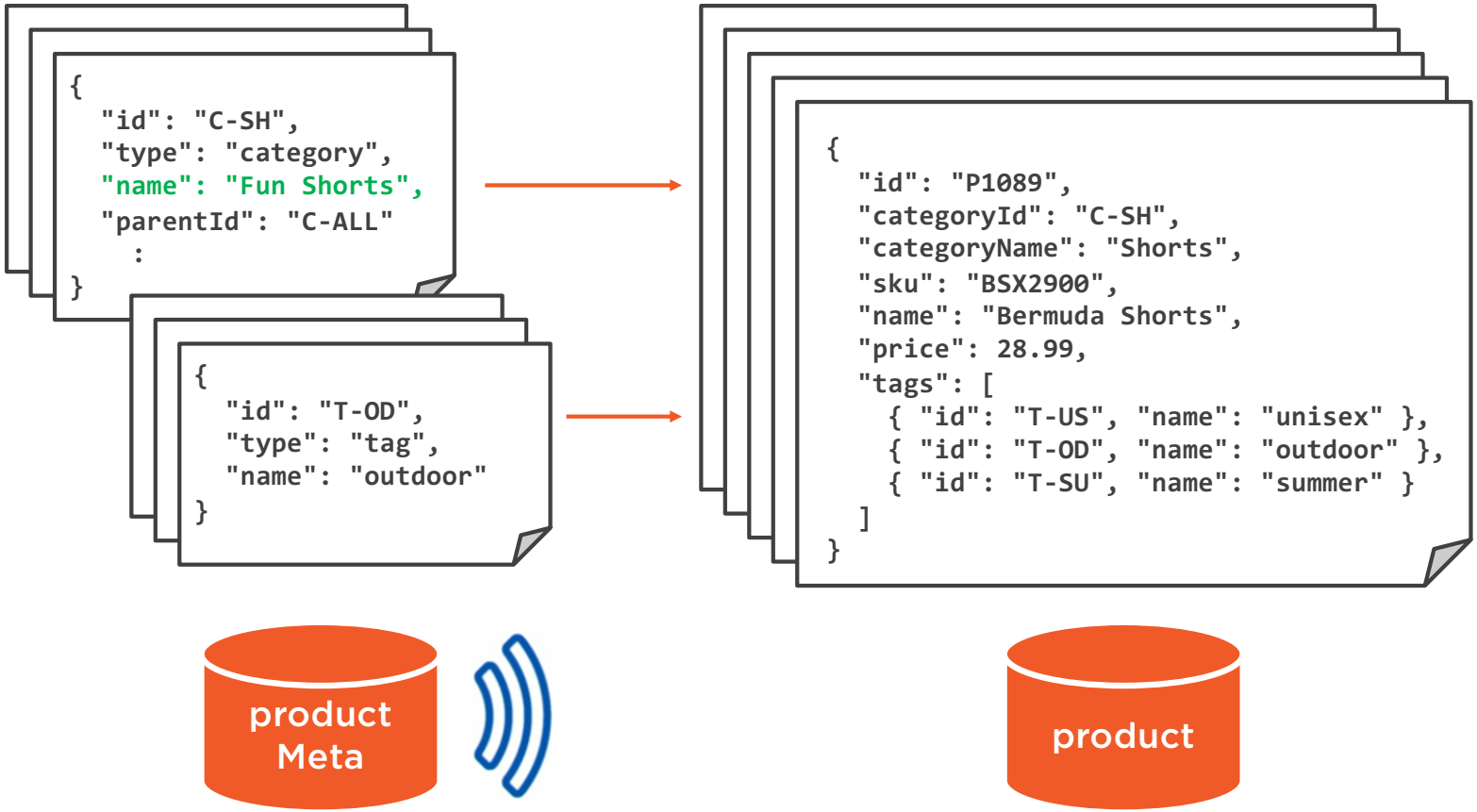
Denormalizing Products



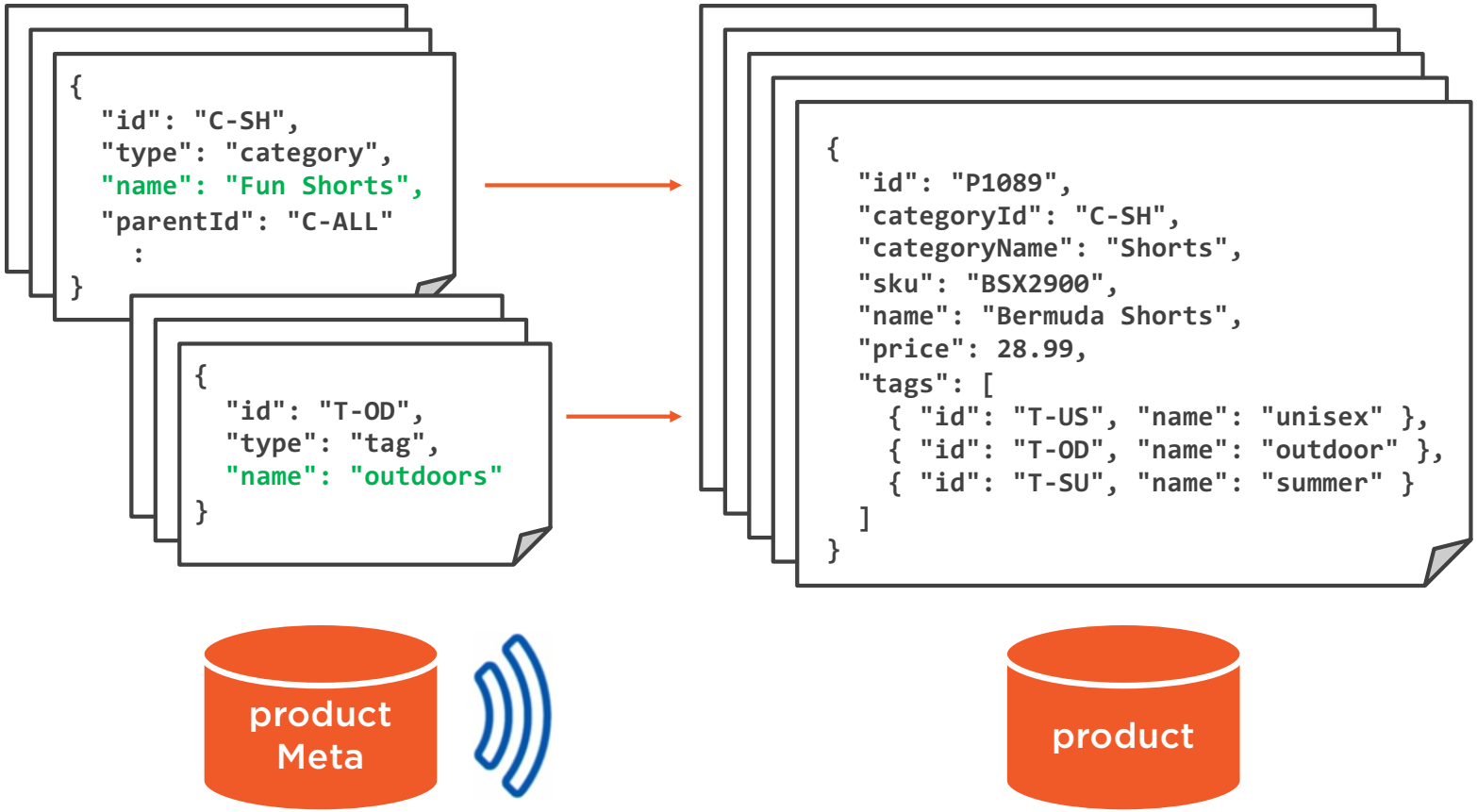
Denormalizing Products



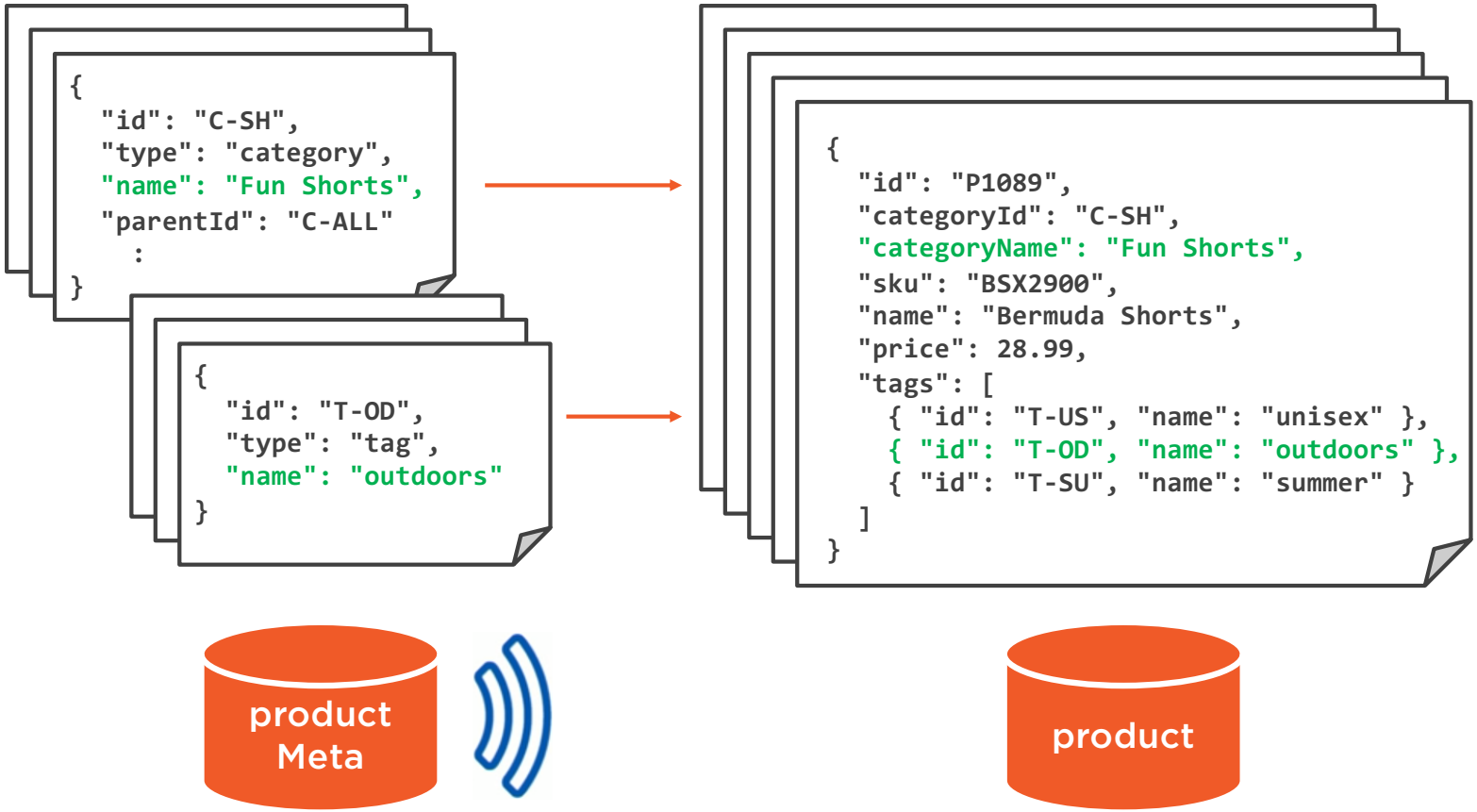
Denormalizing Products



Denormalizing Products



Denormalizing Products



Demo

Denormalization Microservice

Summary

Why denormalize?

- Optimize the data model for common query patterns
- Ensure most queries can be satisfied by a single request

Denormalization microservice

- Product category and tag names