goto;

GOTO
AARHUS 2022

#GOTOaar

Scalable SQL databases

"NewSQL"

Jon Tirsen

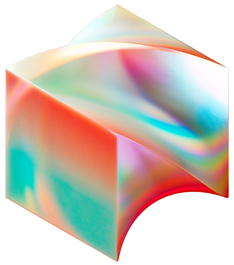



Previously

ThoughtWorks®

Google

Currently



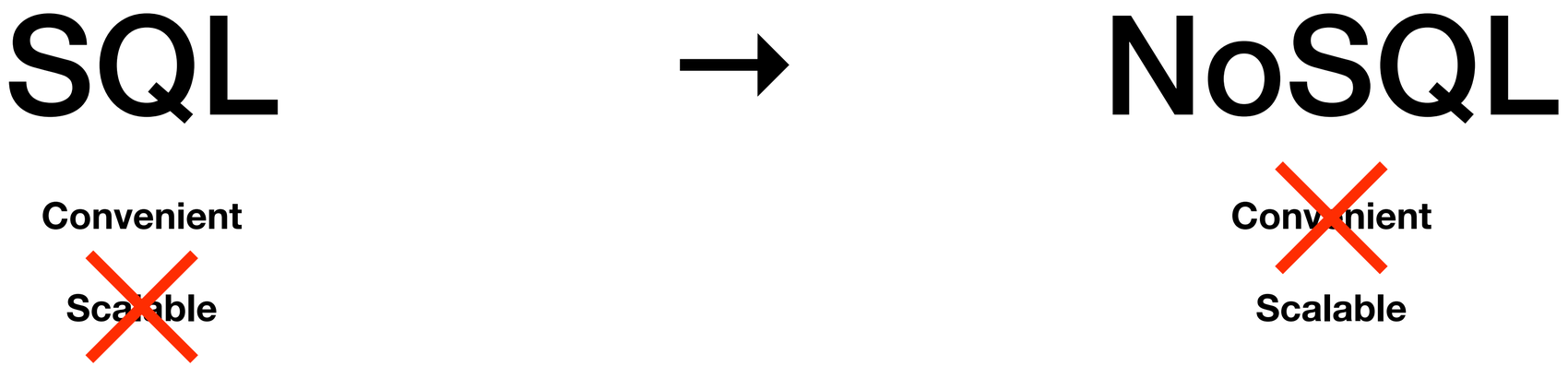
BLOCK

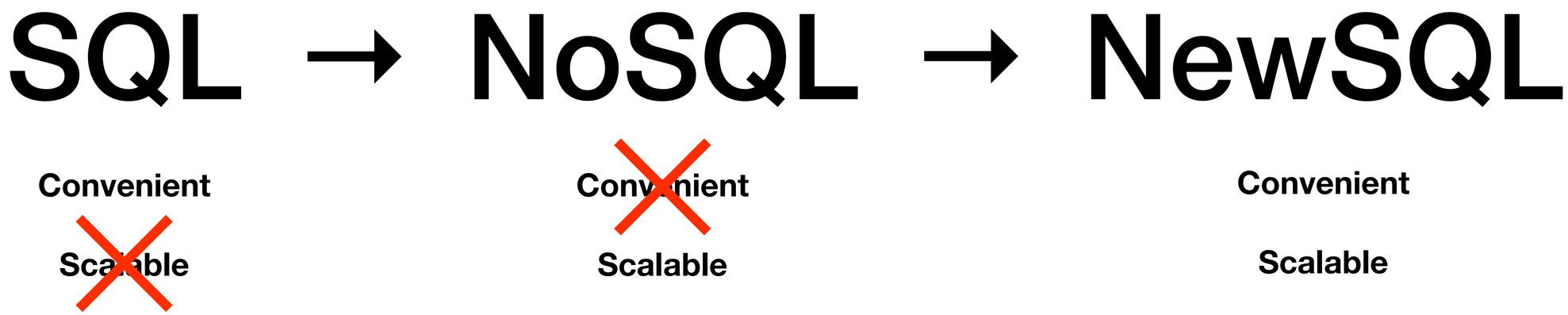
Square

Cash App

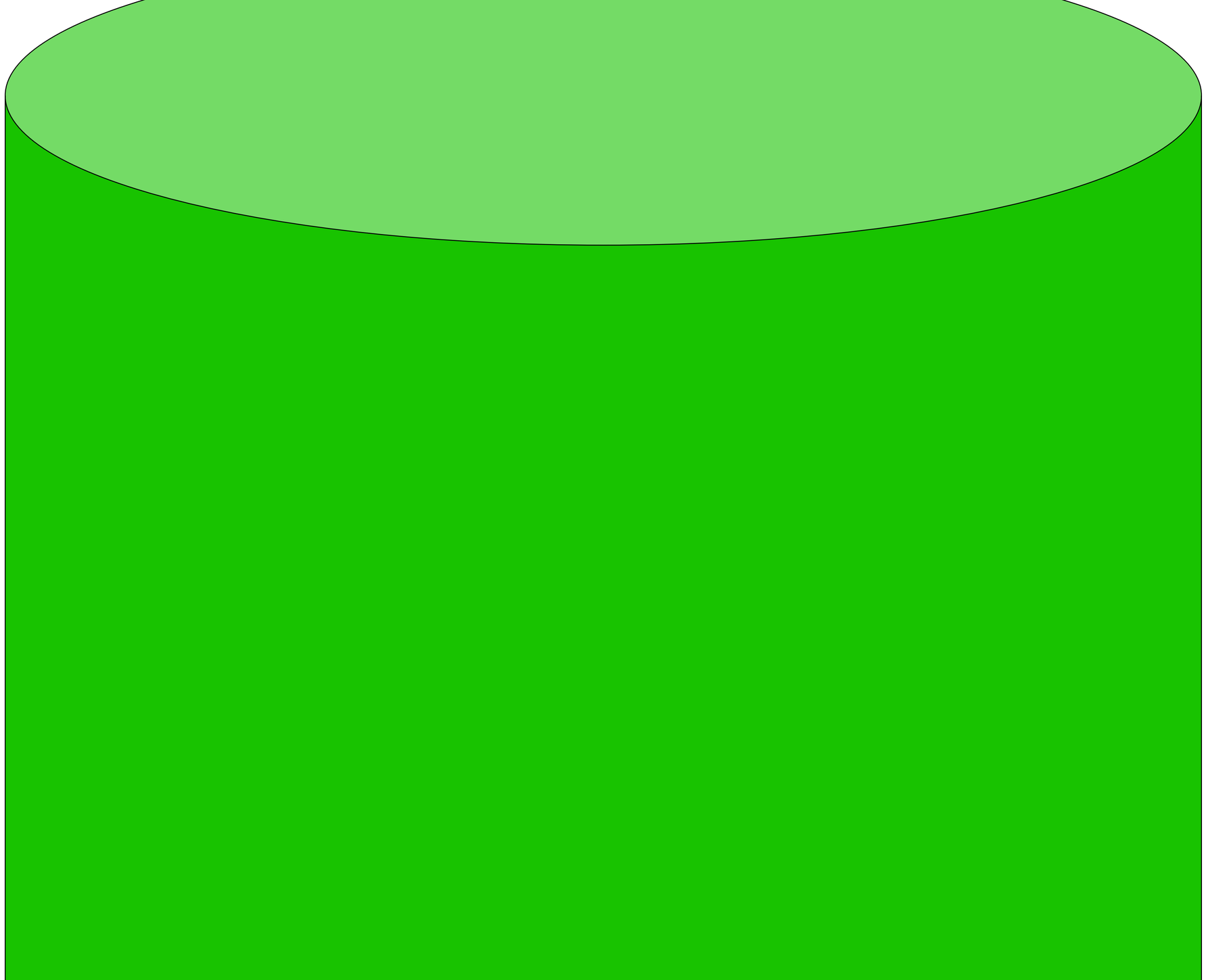
SQL

Convenient


SQL
Convenient
Scalable
NoSQL
Convenient
Scalable


SQL
Convenient
Scalable
NoSQL
Convenient
Scalable
NewSQL
Convenient
Scalable

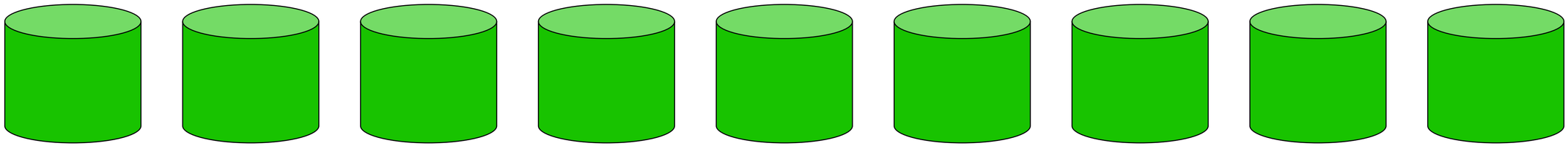
Scalable



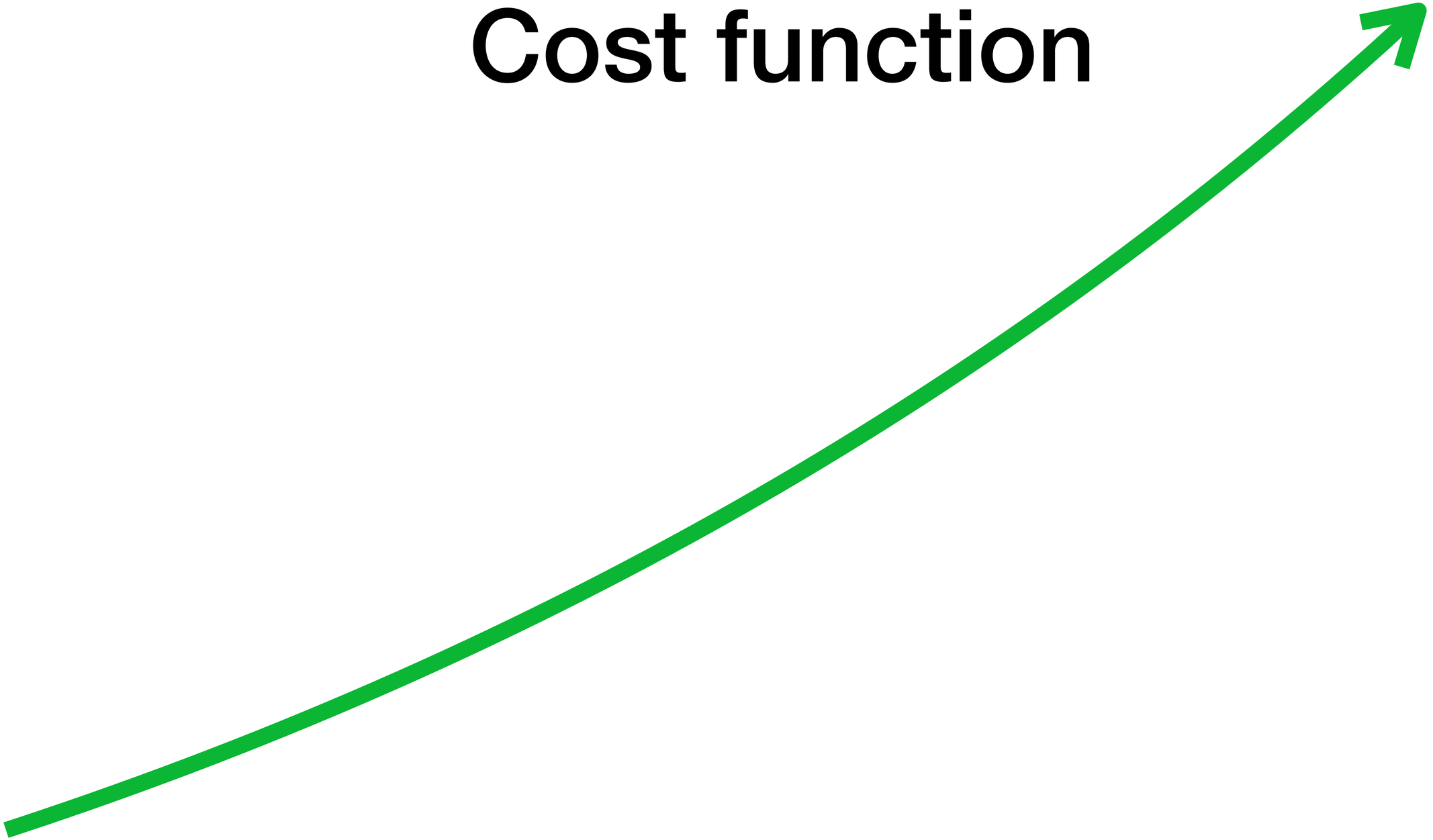

BOOM!

Cost function

Horizontal scaling



Cost function



Convenient

NoSQL

Business logic

Aggregations

App

Joins

Transactions

Database

Data storage

SQL / NewSQL

Business logic

Aggregations

Joins

Transactions

Data storage

Why SQL?

```
SELECT customers.id,
       SUM(order_lines.price) AS total_spend
FROM customers
JOIN orders
  ON customers.id = orders.id
JOIN order_lines
  ON orders.id = order_lines.order_id
GROUP BY customers.id
ORDER BY total_spend;
```

Mainstream

PranaDB

SQL Revival

Declarative

What - not How

```
SELECT customers.id,
       SUM(order_lines.price) AS total_spend
FROM customers
JOIN orders
  ON customers.id = orders.id
JOIN order_lines
  ON orders.id = order_lines.order_id
GROUP BY customers.id
ORDER BY total_spend;
```

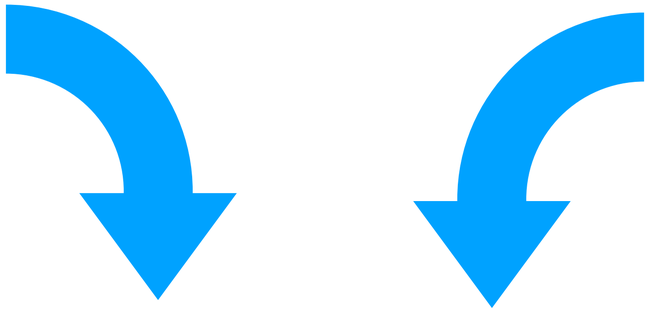
SQL

ACID transactions

Joins/Aggregations/
Windowing functions

Indexes/Constraints

Relational modeling



NoSQL

Distributed

Auto partitioning/sharding

High scale

High availability

Global ACID

Global indexes

Global constraints

Global joins

NewSQL


Clustrix
TiDB
Vitess
Cloud
Spanner
Cockroach DB
Amazon
Aurora
YugaByte DB

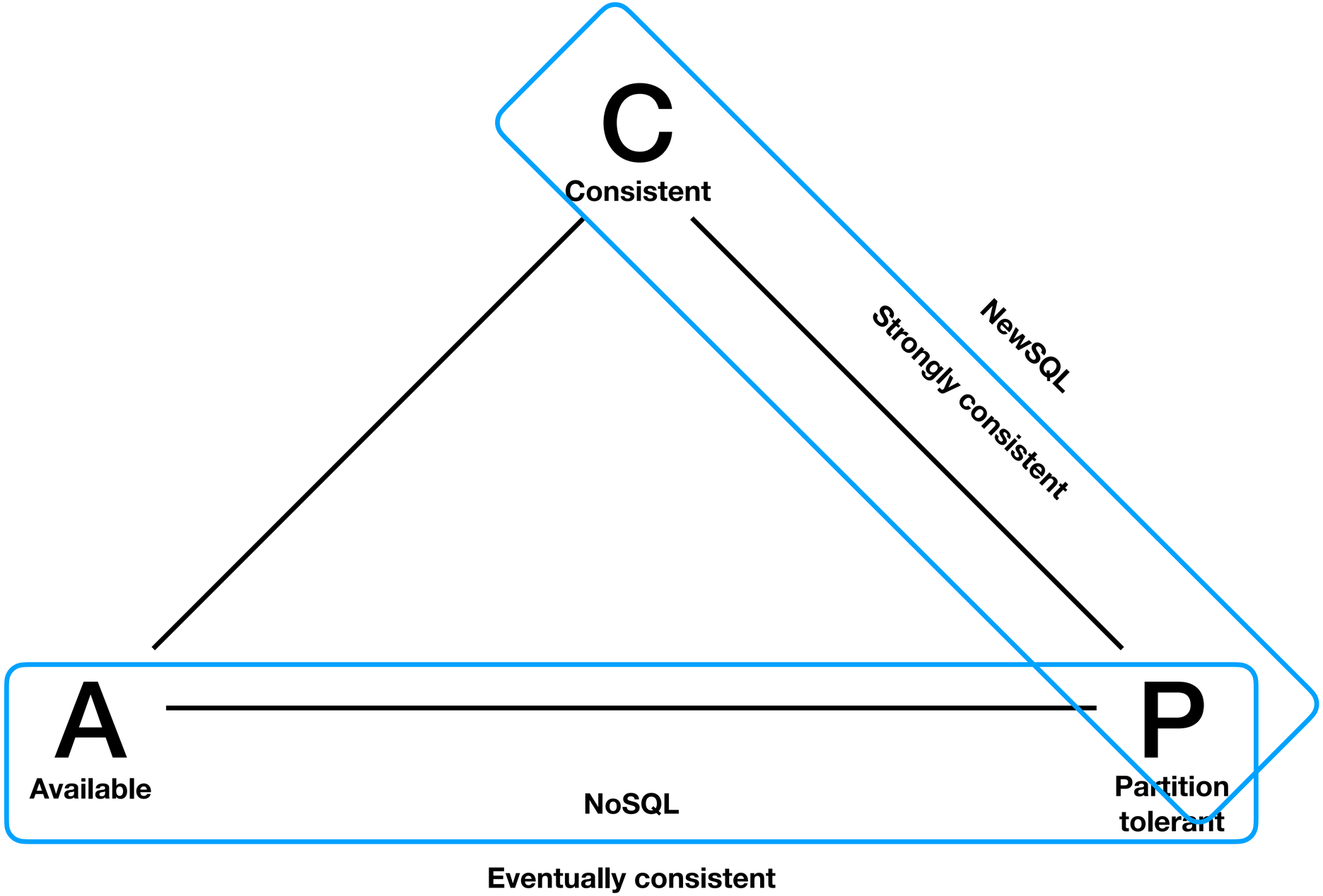
Wanna try one?

```
> docker run -d -p 4000:4000 pingcap/tidb

> mysql -h 127.0.0.1 -P 4000 -u root

...

mysql>
```

What has changed?

CPU
RAM
Disks
Network
Everything is better
faster
cheaper
more reliable
more of it

Everything is better
Data structures
Infrastructure
Algorithms


C
Consistent
NewSQL
Strongly consistent
A
Available
NoSQL
P
Partition tolerant
Eventually consistent

Distributed ACID

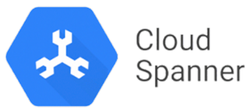
2 phase commit

RAFT/PAXOS for high availability

Log structured merge trees: "read the past"

Spanner style

Sync time

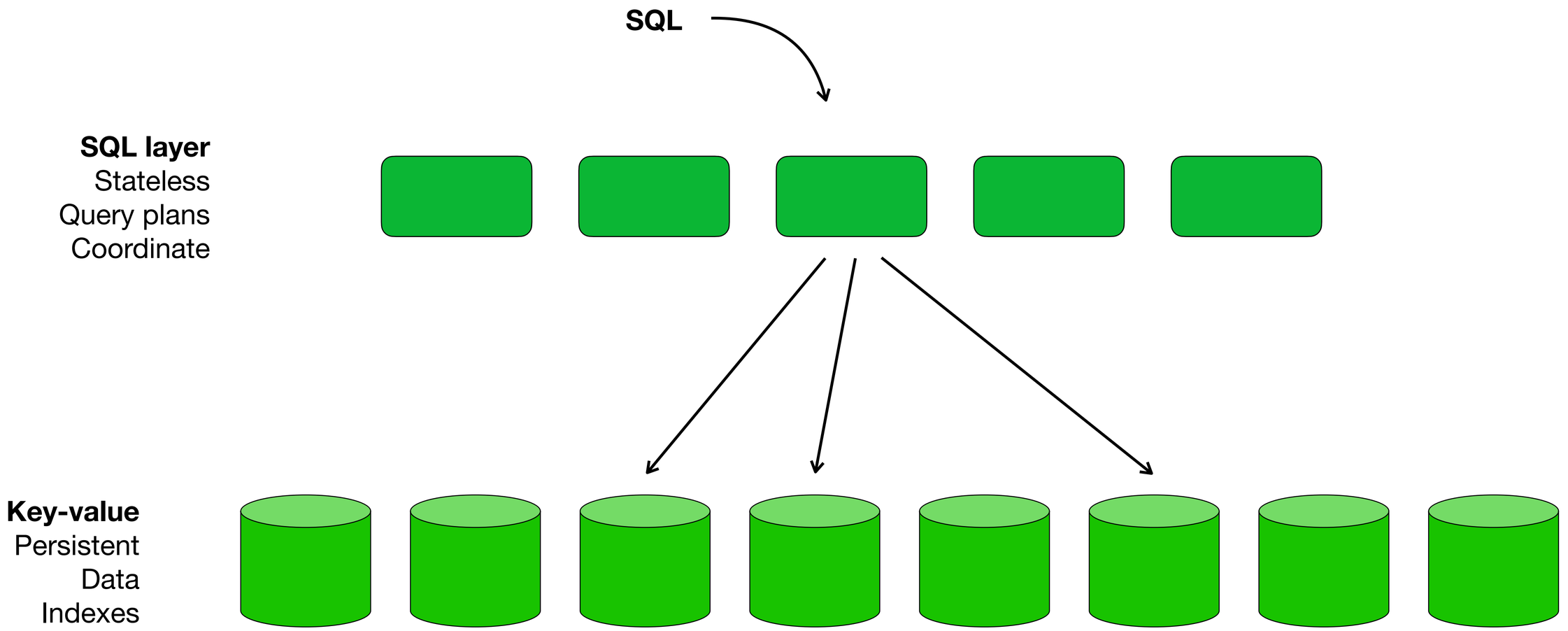


Percolator style

Timestamp oracle

Local ACID

Built on top of key-value



When to use

Transactional

10ms - 100ms

High scale

writes per second

queries per second

storage volume

>1TB

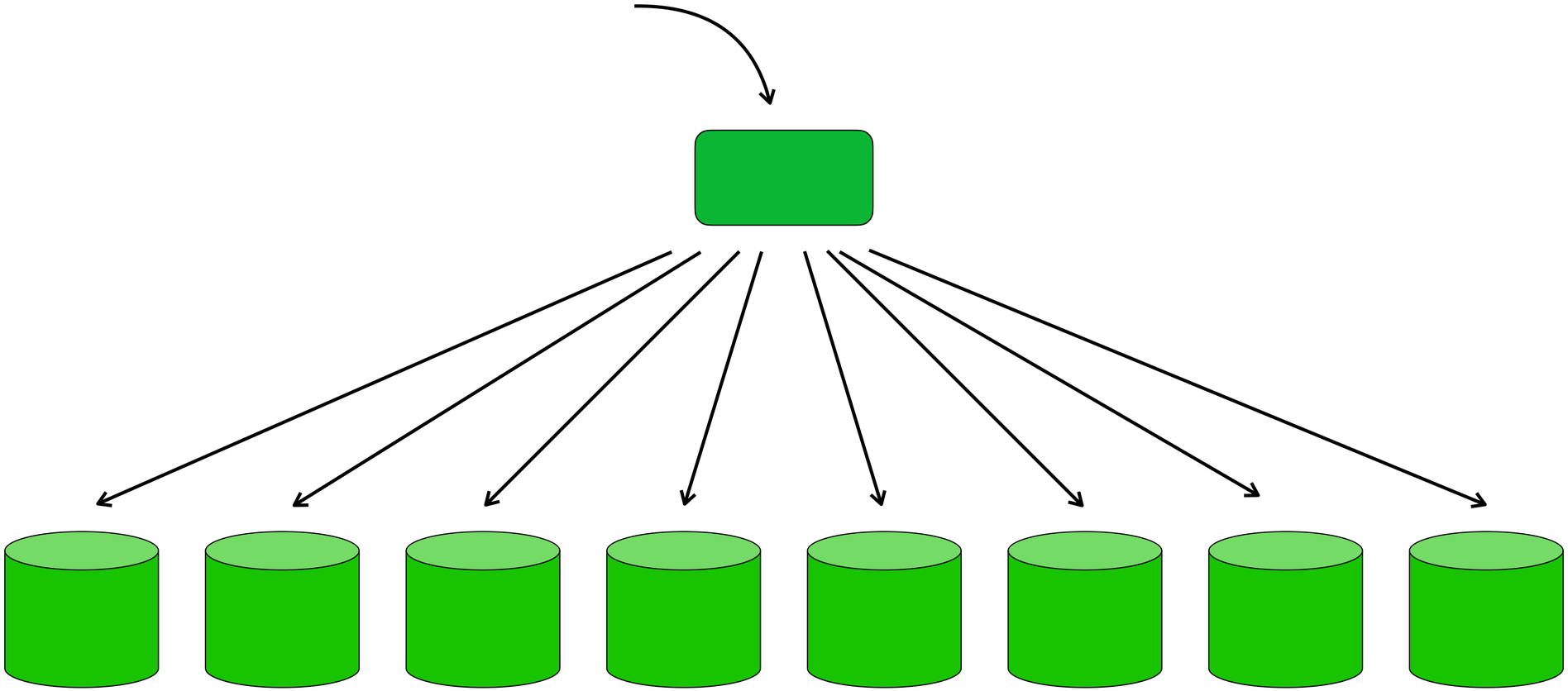
Do you need strong consistency?

Do you need SQL?

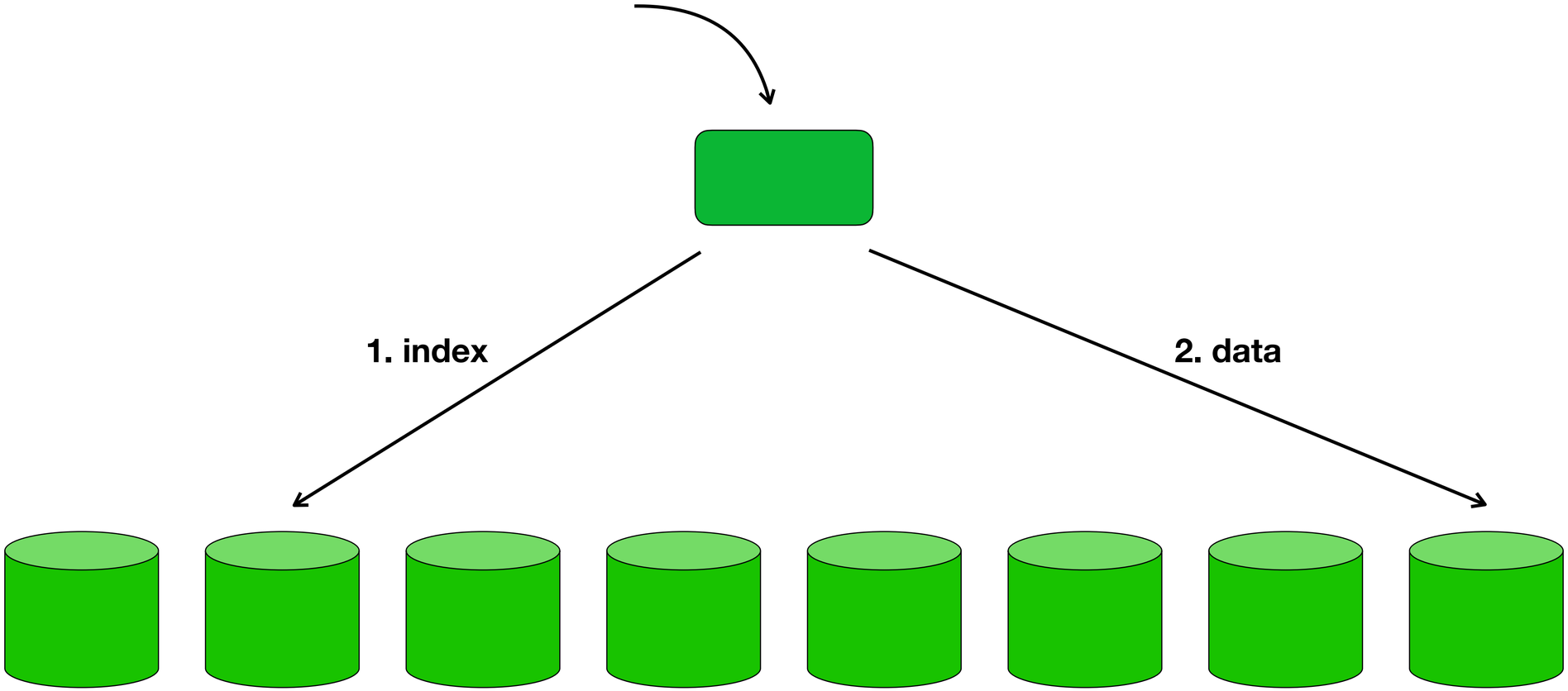
But! Plan ahead!

Issues

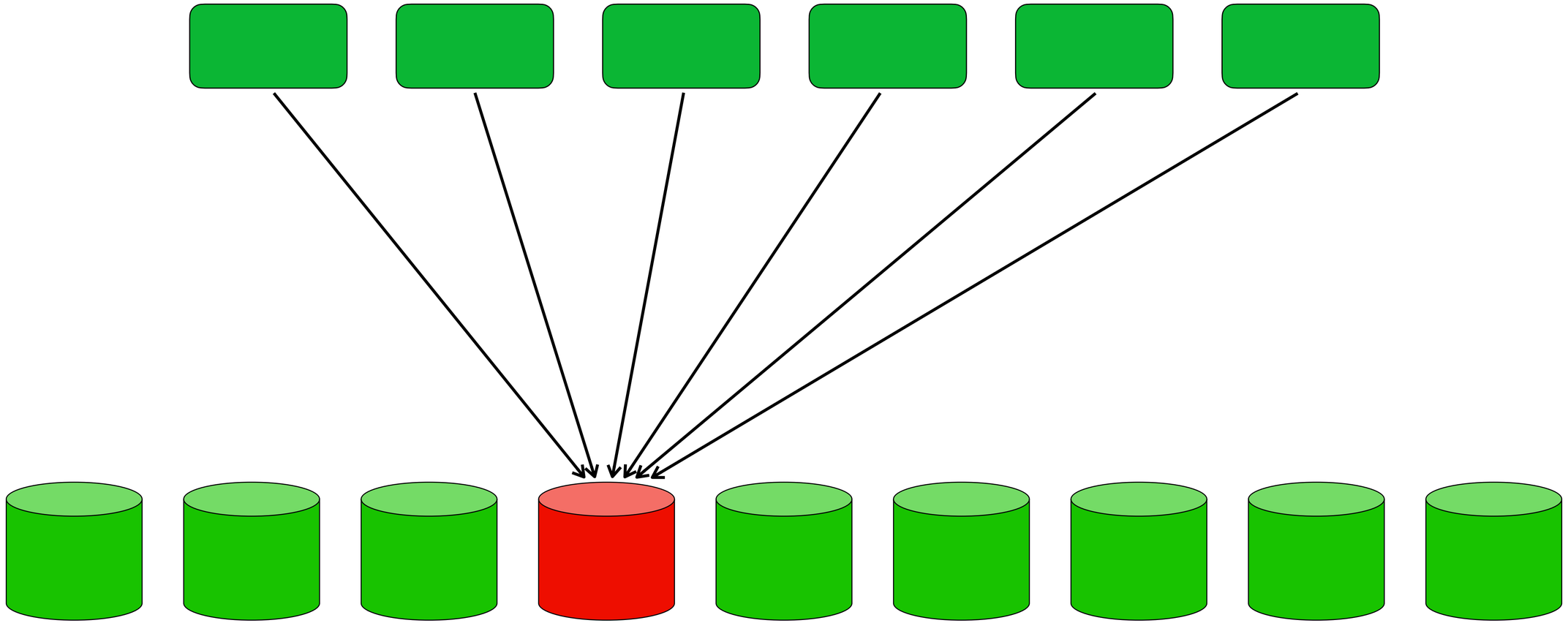
Avoid fan outs



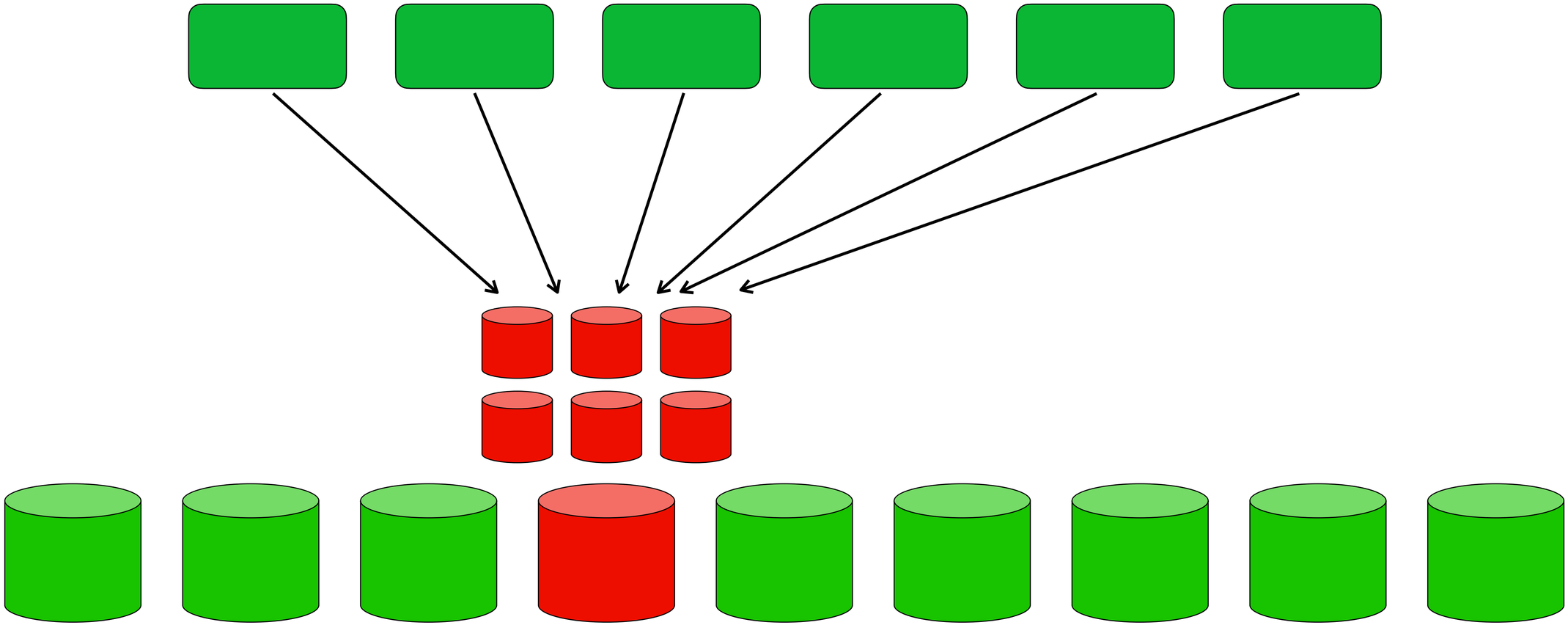
Use global indexes



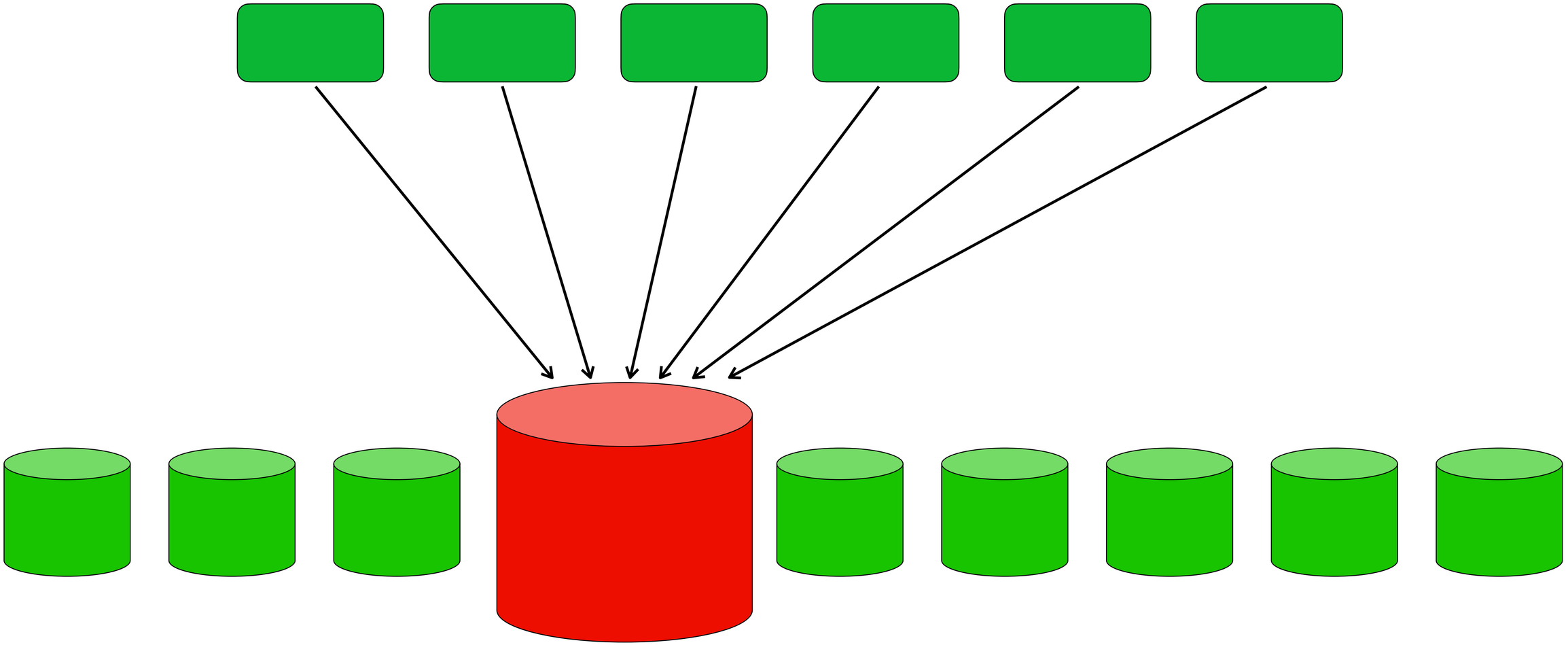
Avoid hot spots



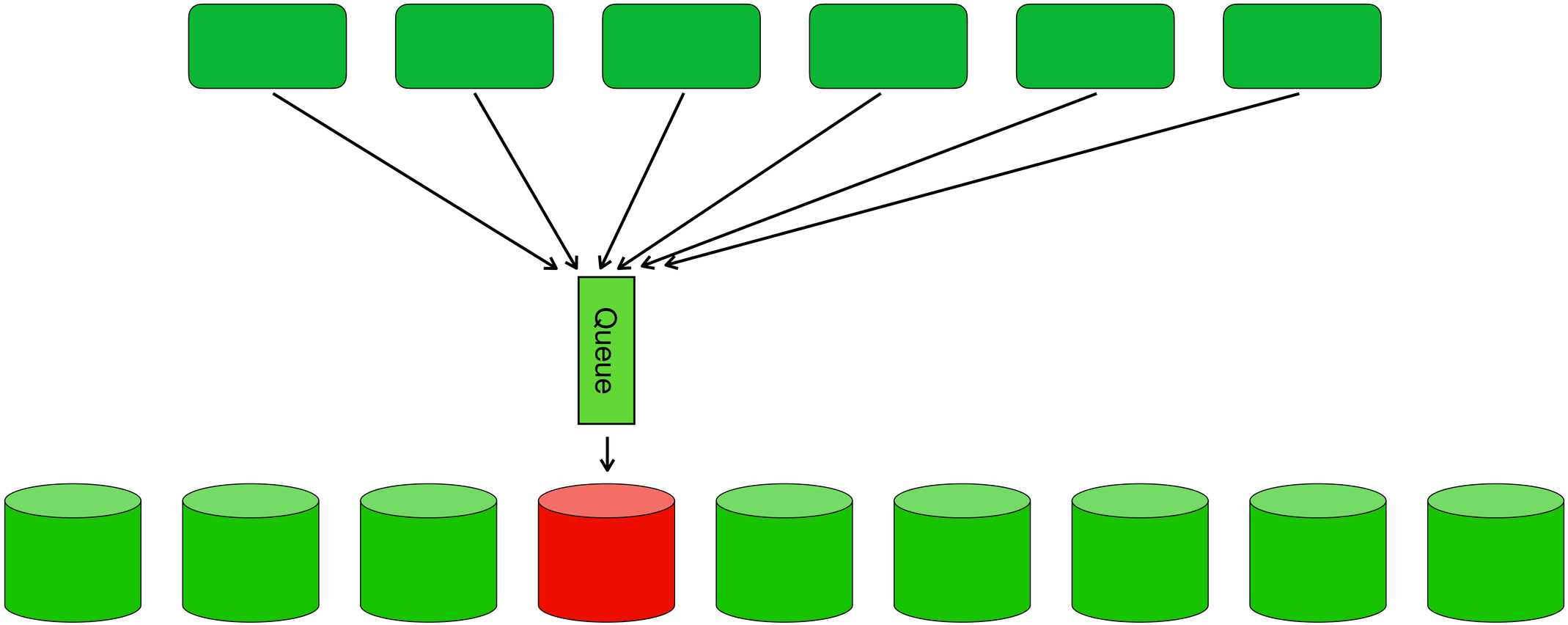
Solve read hot spots

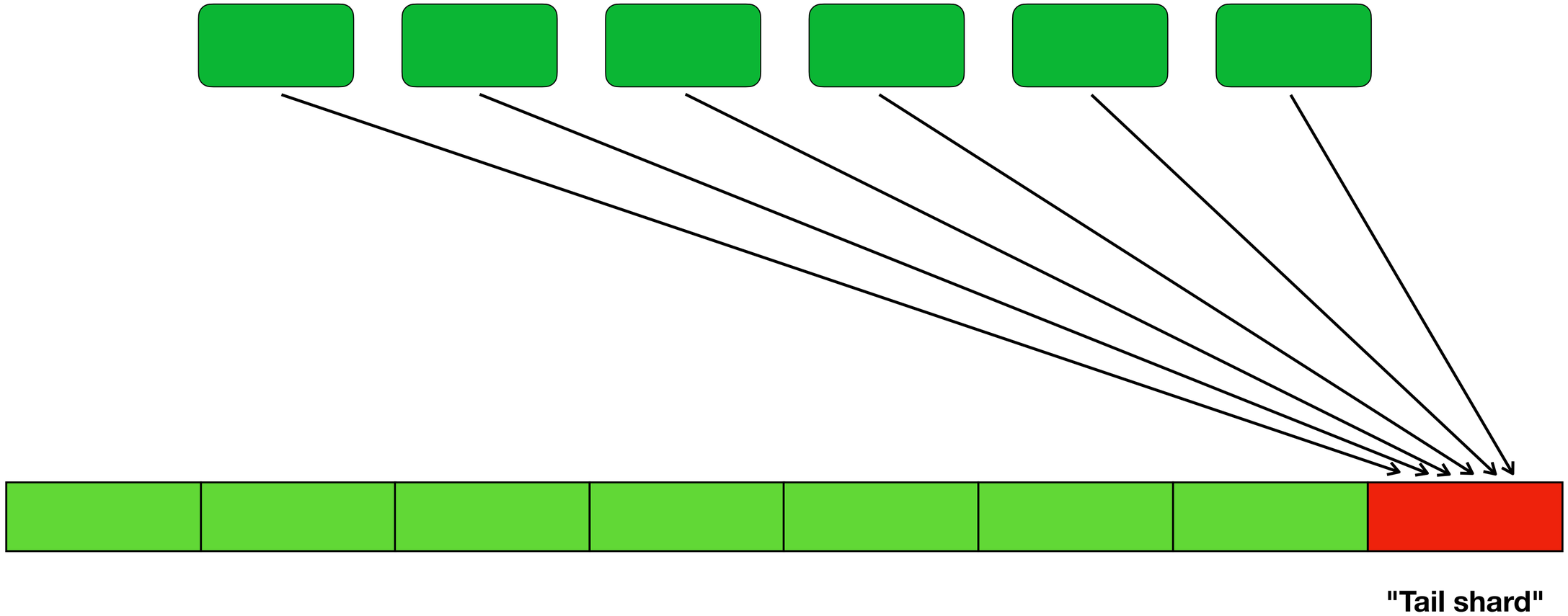


Solve write hot spots



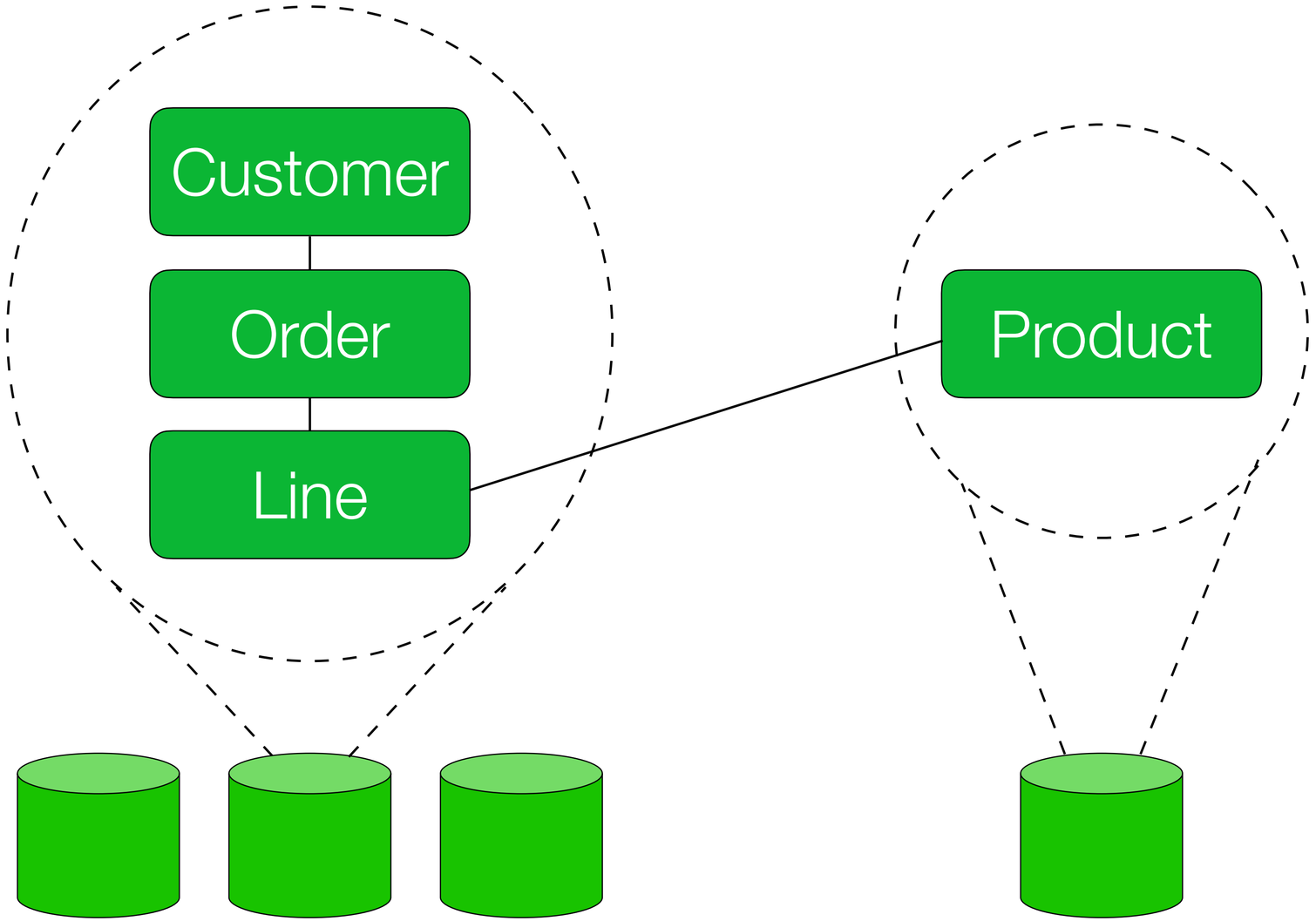
Solve write hot spots





Randomize IDs

Entity groups



TiDB

```
CREATE TABLE customers(
  id BIGINT,
  PRIMARY KEY(id)
);

CREATE TABLE orders(
  id BIGINT,
  customer_id BIGINT
) PRIMARY KEY(customer_id, id);

CREATE TABLE order_lines(
  id BIGINT,
  customer_id BIGINT,
  product_id BIGINT
) PRIMARY KEY(customer_id, id);
```

```
CREATE TABLE products(
  id BIGINT,
  PRIMARY KEY(id)
);
```

Spanner

```
CREATE TABLE customers(
  id BIGINT,
  PRIMARY KEY(id)
);

CREATE TABLE orders(
  id BIGINT,
  customer_id BIGINT
) PRIMARY KEY(customer_id, id)
  INTERLEAVE IN PARENT customers;

CREATE TABLE order_lines(
  id BIGINT,
  customer_id BIGINT,
  product_id BIGINT
) PRIMARY KEY(customer_id, id)
  INTERLEAVE IN PARENT customers;
```

```
CREATE TABLE products(
  id BIGINT,
  PRIMARY KEY(id)
);
```

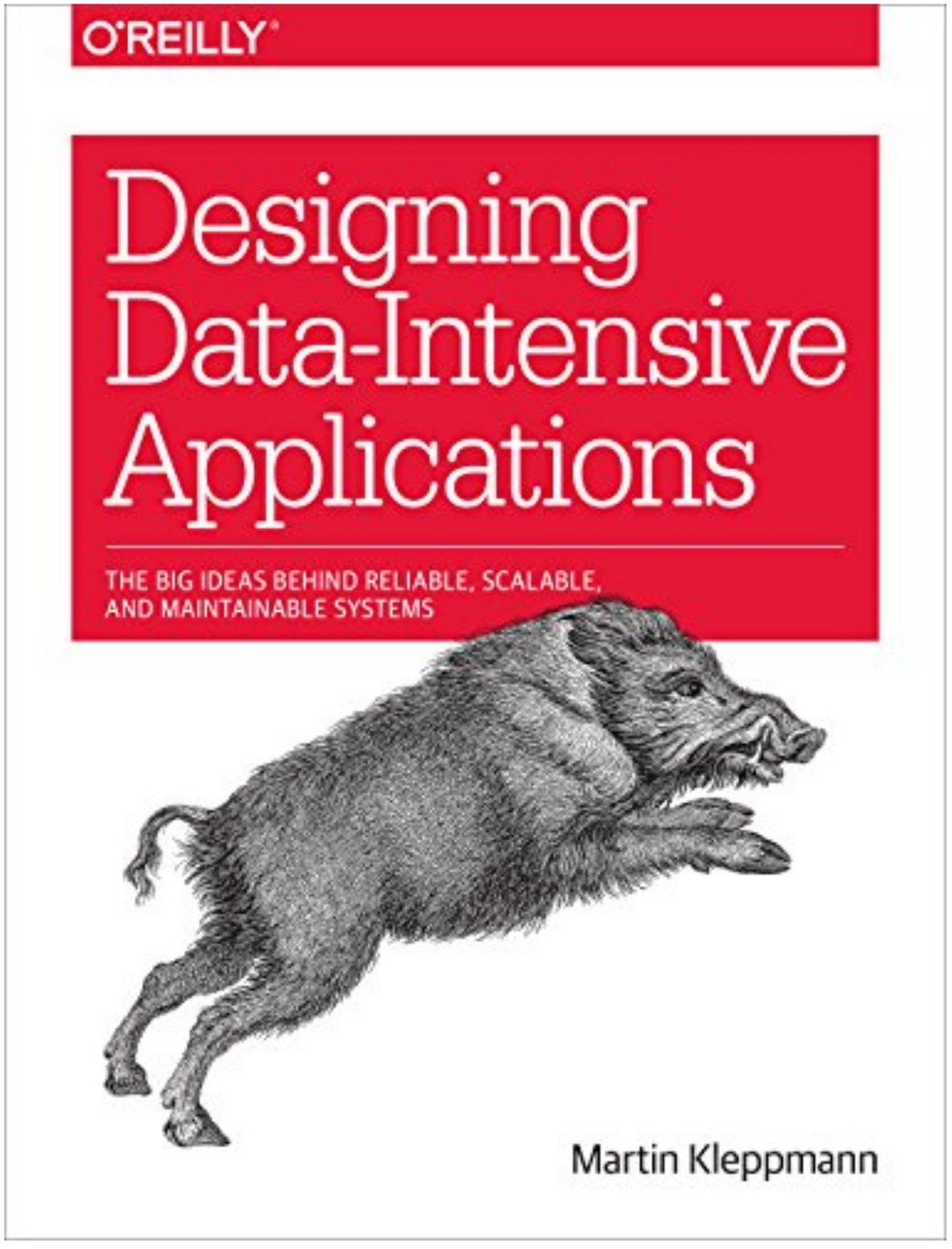
Scalable SQL built on recent innovations in distributed computing

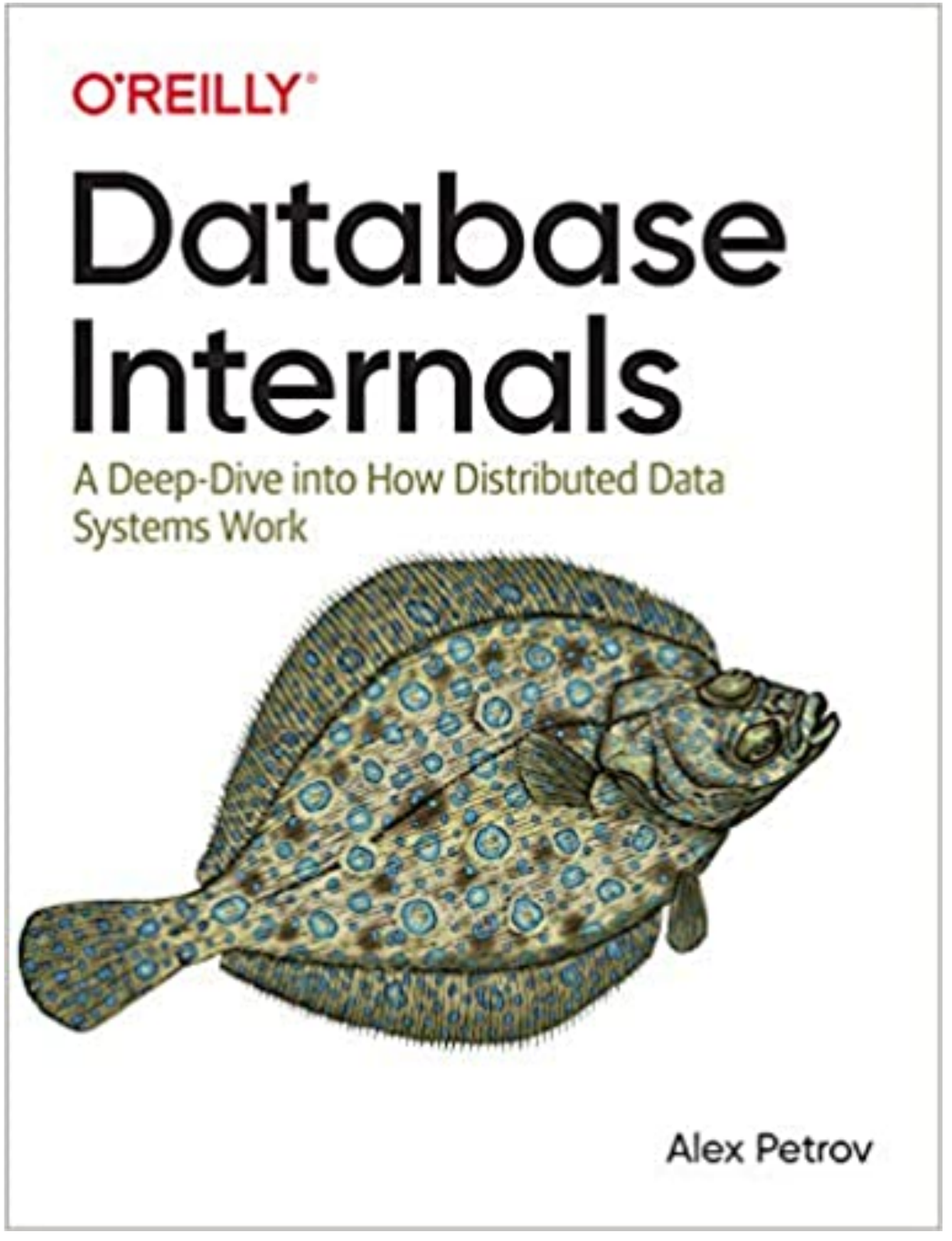
Use it for scaling out transactional workloads

NewSQL

Watch out for write hot spots and wide fan outs

Thank you!
Jon Tirsen
@tirsen
linkedin.com/tirsen


O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann


O'REILLY
Database Internals
A Deep-Dive into How Distributed Data Systems Work
Alex Petrov

goto;

DON'T FORGET TO **RATE THE SESSIONS**

#GOTOaar

Rate a minimum of **5 sessions** and claim your **reward** at the Registration Desk at the Trifork Hall