NoSQL vs SQL

Databases 101

NoSQL vs SQL

Database - big box of things

SQL (SEQUEL)

Structured Query Language (1970)

- Data Definition
- Data Manipulation
- Transaction Control

SELECT COUNT (name) FROM Listeners;

Database Management Systems

- Oracle - 1979
- Microsoft - 1989
- MySql - 1995
- Postgres - 1996
- SqLite - 2000

Tables, tables, tables

- Scaling
- Sharding
- Schema changes

Vertical vs Horizontal scaling

TODO: image of buildings

NoSql

A meetup name

- Not using the relational model (not the SQL language)
- Open source
- Designed to run on large clusters
- Based on the needs of 21st century web properties
- No schema, allowing fields to be added to any record without controls

Types of NoSql

- Key-value
- Document
- Graph
- Wide Column
- Time series
- Search Engines

Key-value

Remote Dictionary (Map)

Usage

- Cache
- Simple data

DBMS

- Redis
- Memcache

Document oriented

Object == Document

Usage

- Higher performance for "item-based" queries
- Simpler scaling*
- Implicit schema - seems easier*

DBMS

- MongoDb
- DynamoDb
- Couchbase
- Firebase Realtime Db
- Firestore

Graph oriented

Object == Node + Connection

Usage

- Fraud detection
- Recommendation engines
- Identitity and Access management
- Message queues

DBMS

- Neo4j
- Dgraph
- Cosmos DB

Wide column (Columnar)

Object == Column (Column Family)

Usage

- Analytics
- Data warehouses
- Per-column queries

DBMS

- Bigtable
- Hbase
- Cassandra

Time series dbms (Columnar)*

Usage

- Sensor Data
- Tracking assets
- Predicting shopping behavior

DBMS

- InfluxDB
- Prometheus
- Graphite
- Objectbox TS*

Search engines (Document / Graph based)*

Usage

- Searching in documents
- Natural language questions
- Fuzzy search
- ... search :)

DBMS

- ElasticSearch
- Algolia
- Splunk

Cloud storage*

Usage

- It's not a DBMS as is
- File storage

DBMS

- Goocle Cloud Storage
- Amazon S3

To SQL or not to SQL?

Schemaless is our future!

~~Schemaless is our future!~~

Usually, there is an implicit schema

- User
  - id
  - name

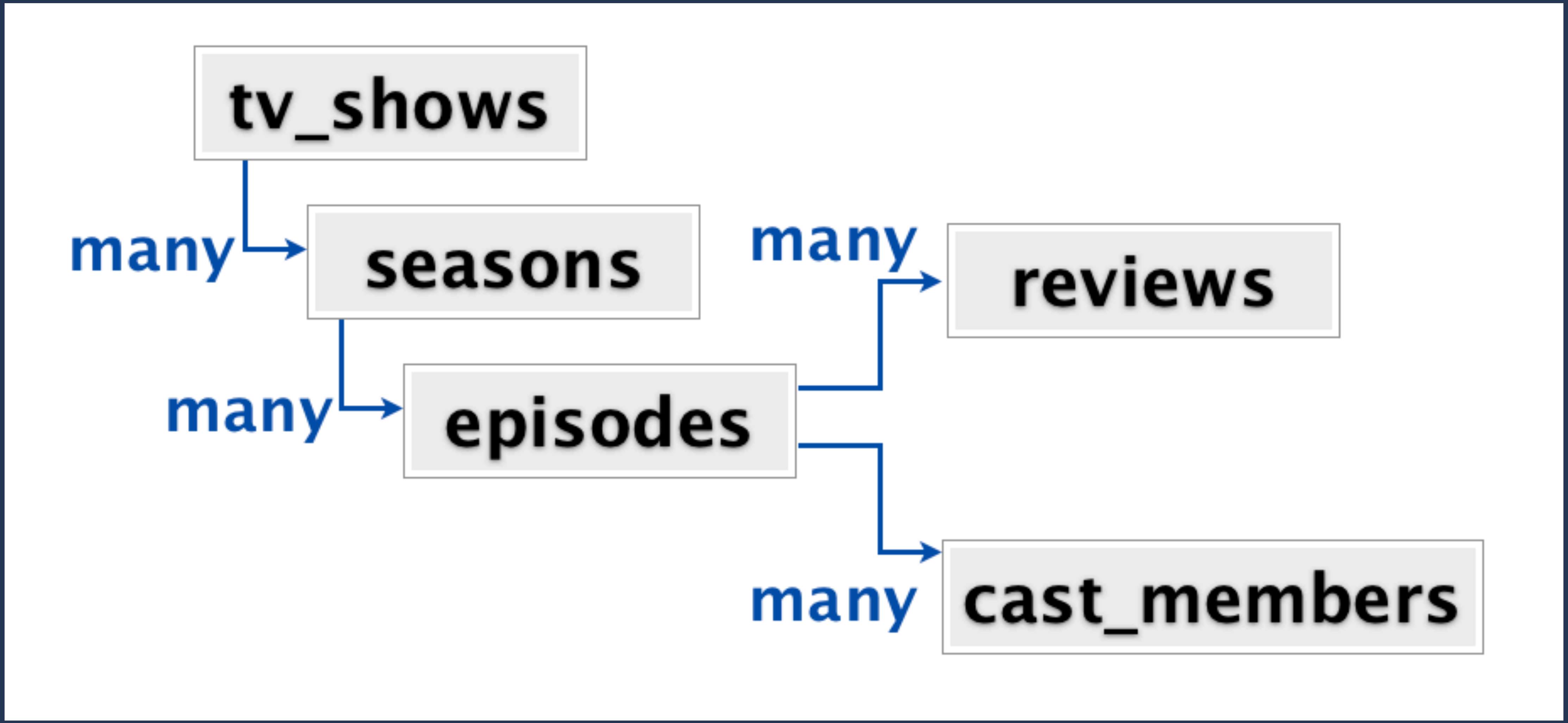
~~Schemaless is our future!~~

Usually, there is an implicit schema

- User
  - id
  - name
  - first name
  - last name

Migrations! Even in "schemaless" nosql...

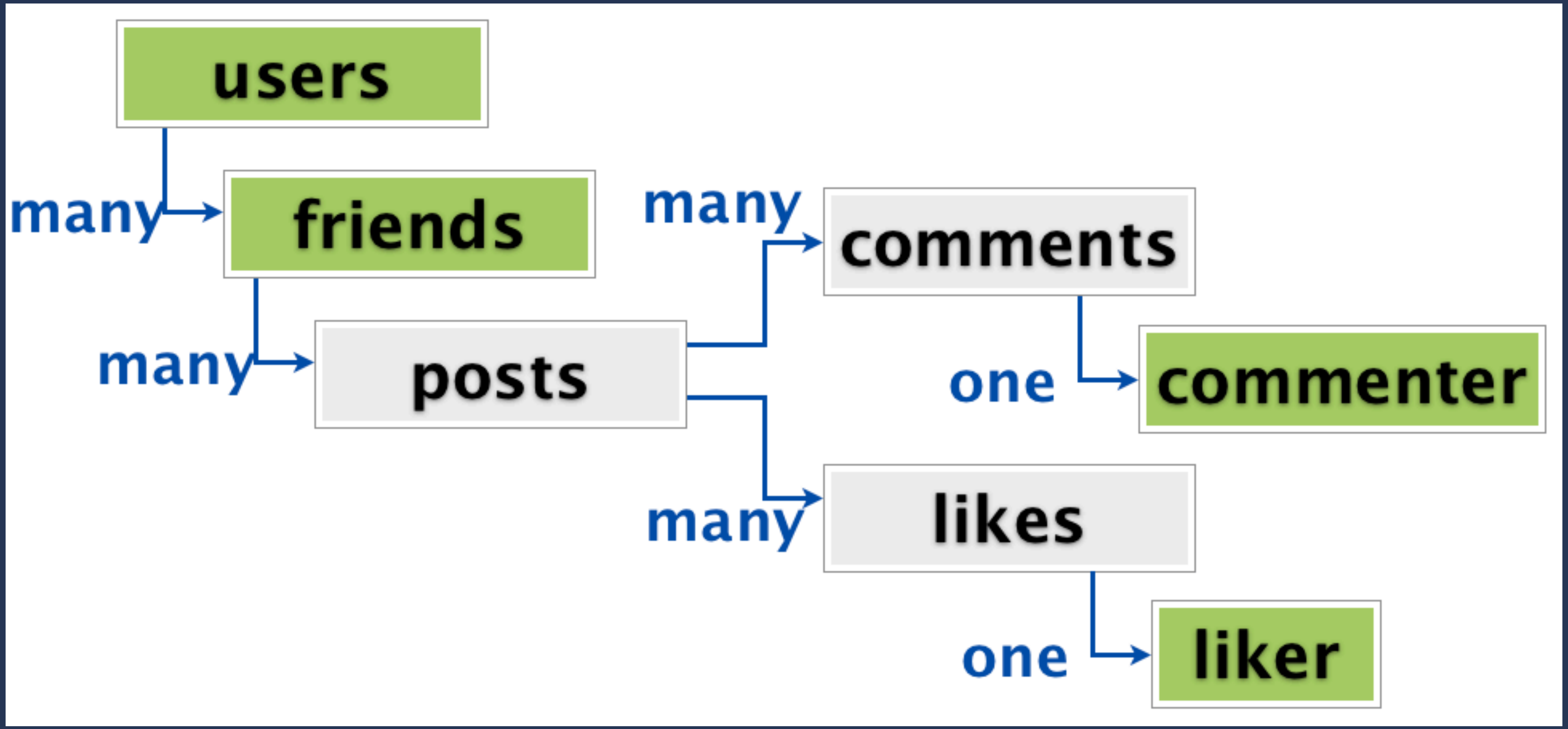
There is no silver bullet!



https://habr.com/ru/post/231213/

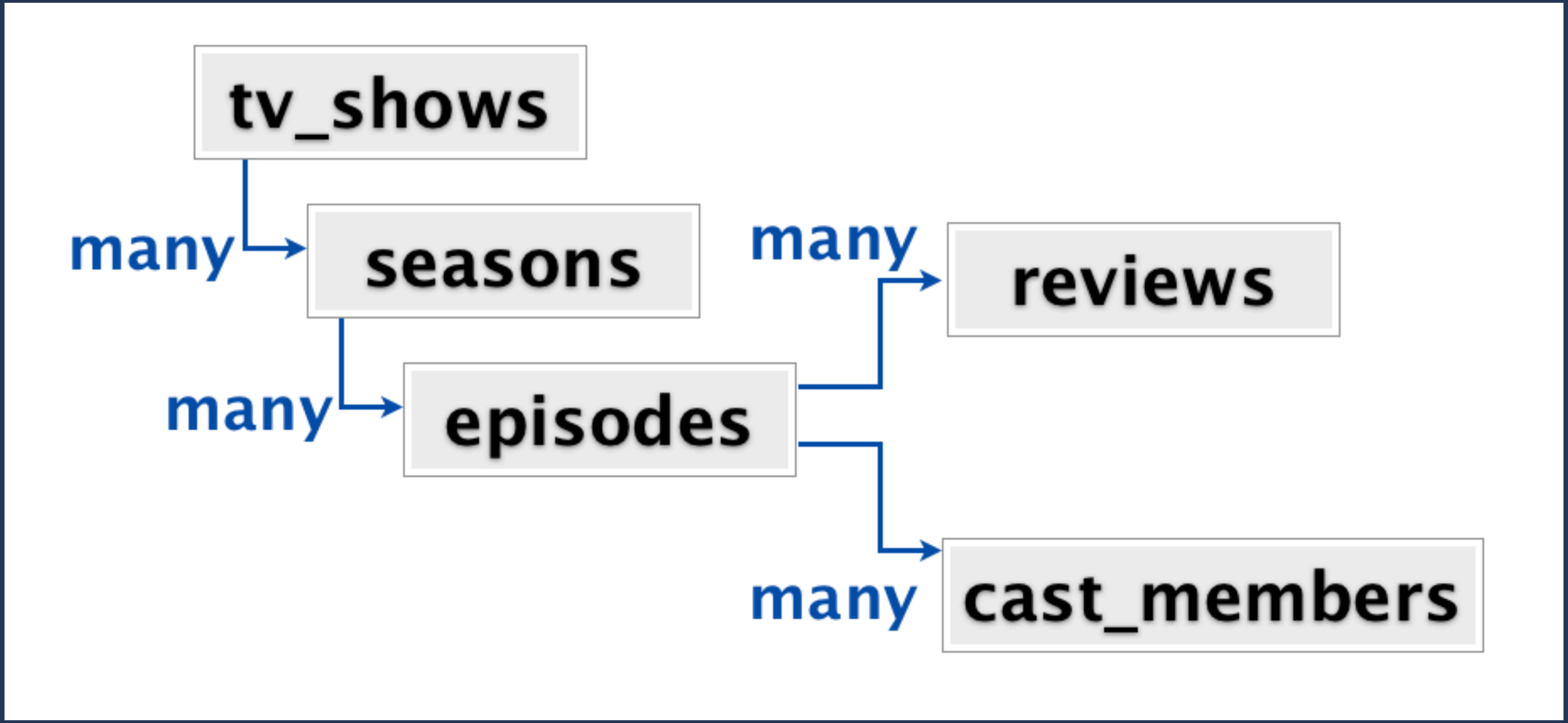
There is no silver bullet!

Users, Users, Users…



There is no silver bullet!

Where is my Actor?



How Twitter Uses Redis to Scale - 105TB RAM, 39MM QPS, 10,000+ Instances. September 2014

CocroachDB

ACID compliant databases allow for consistent transactions. Even in a distributed environment, CockroachDB provides the highest level of isolation - serializable.

OLTP and OLAP

- Online Transaction Processing
- Online Analytical Processing

Grand finale

- Need connections (relations)?
  - Relational DBMS
  - Graph DBMS
- Highly cohesive data without links to each other
- or just archiving a pile of stuff
  - Document DBMS
  - Object DBMS
- Specific use-cases for different NoSql dbms
  - Data analysis
  - Search
  - Time Series

Wait, but is that and Android Crew?

Android use cases

- *Need connections (relations)?*
  - Relational DBMS
  - Graph DBMS
- *Highly cohesive data without links to each other*
- or just archiving a pile of stuff
  - Document DBMS
  - Object DBMS

- Specific use-cases for different NoSql dbms
  - Data analysis
  - Search
  - Time Series

To SQL or not to SQL in Mobile?

SQL or NoSql - It doesn't matter![1]

But other things do

[1] Almost doesn't, as usual

Why don't we have Mobile Database Administrators?

Why don't we have Mobile Database Administrators?

We don't need to dance with CAP!

How to choose a database (dbms)?

- Vendor
- Business requirements
  - Local / Remote
  - Performance, number of objects
  - Security
- Tooling
- Team's experience
- Convenience of API
  - Queries
  - Relations
  - Reactive / listening to changes
- Footprint (size)
- Criticality of the project
  - Consider something new for short-term / side project

SqLite (pure) (2000)

- Relational
- Requires some SQL knowledge

Pros

- Time-proven
- Great tooling, db inspector
- Nearly doesn't add size to apk
- Encryption via SQLCypher

Cons

- Verbose
- Relations are relatively complex to build

Room - abstraction layer over SQLite

- Internally, it's still SqLite

Pros

- From Google - will likely live long time
- Great tooling
  - Syntax highlight
  - Compile time checks
- Adapters to everything
- Nearly doesn't add size to apk (50 KB)
- Encryption via SQLCypher

Cons

- Relations are relatively complex to build
- Not multiplatform
- Sometime breaks incremental compilation*
- Requires instrumented tests

SqlDelight

- Internally, it's still SqLite

Pros

- Great tooling
  - Sql autocompletion
  - Compile time checks
  - AS plugin
- Adapters to everything
- Nearly doesn't add size to apk
- Multiplatform
- Encryption via SQLCypher
- JVM unit tests

Cons

- Relations are relatively complex to build
- Mostly manual migrations

Realm

Pros

- Nice query building api
- Easy work with relations
- Lazy evaluation mechanism
- Works with managed objects
- Constistency with IOS app
- Supports encryption
- Has browser
- In-memory db for JVM unit tests
- Can be synced with cloud

Cons

- Entities must be inherited from RealmObjects
- Doesn't support Type Adapters (https://github.com/realm/realm-java/issues/1694)
- A couple of ways to shoot yourself in the foot
    - Managed objects are thread - dependendant
    - Realm instances should be opened / closed in the same thread
- 4 Mb per apk

ObjectBox

Pros

- Fast
- Nice query building api
- Easy work with relations
- Easy migrations
- Has a browser
- Test db for JVM unit tests
- For different platforms, incl. Flutter
- Can be synced with cloud

Cons

- 1,5 Mb per apk (not so bad)
- Small startup

Cloud Firestore

Pros

Pros

- Easy sync with the cloud
- Supports advanced queries
- High availability (99,999%)

Cons

- Relatively slow for offline usage

Shared Preferences

NoSql storage!!!

DataStore

New version of Key-value storage.

- Preferences DataStore
- Proto DataStore

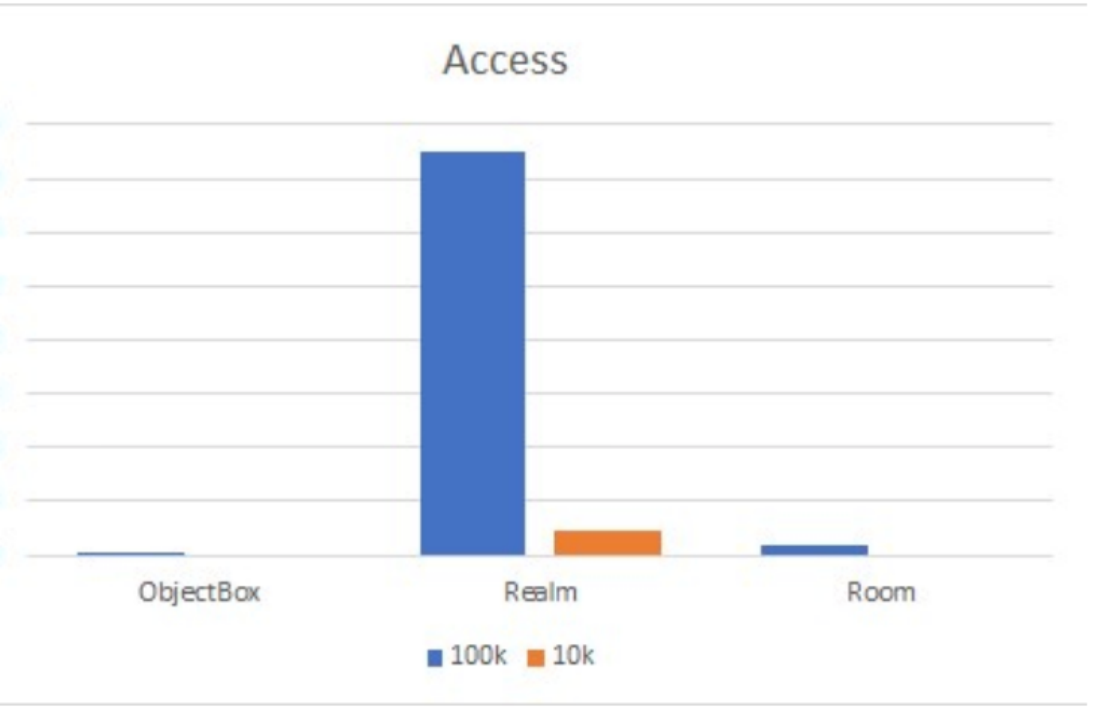
Others

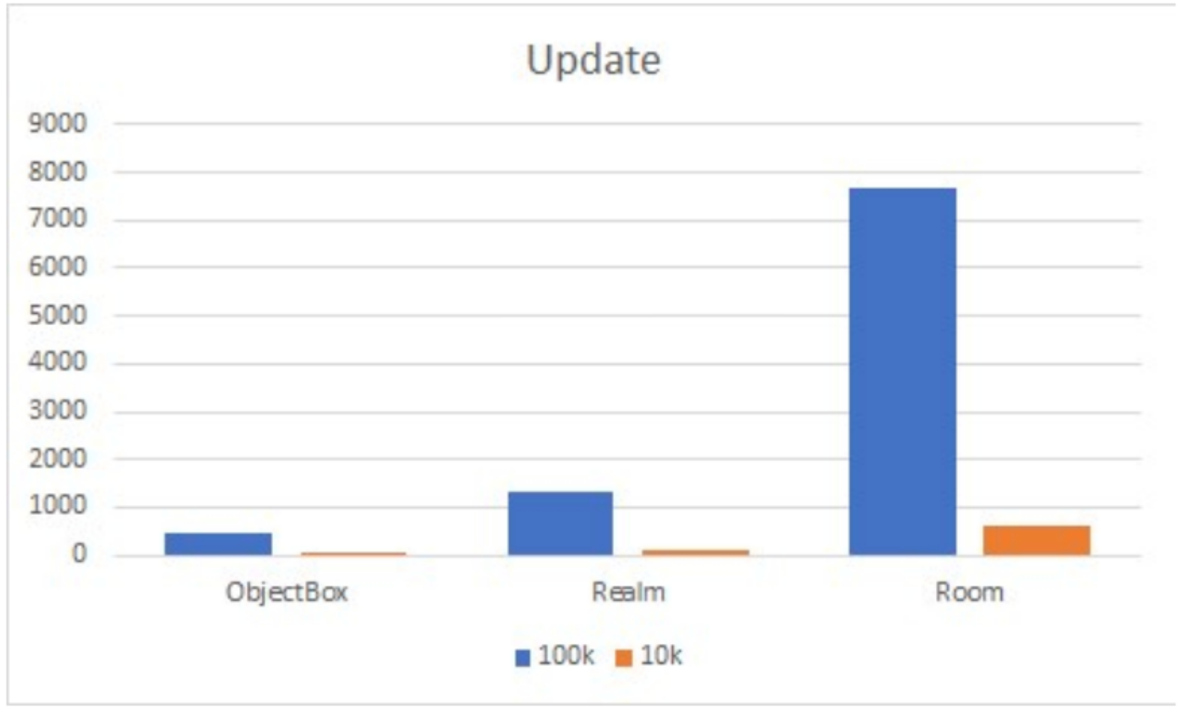
- Paper (NoSql-like) https://github.com/pilgr/Paper
- Couchbase-lite https://github.com/couchbase/couchbase-lite-android
- Tens of others are abandoned
- Case: Mail.ru Cloud Ios app and NoSql key-value storage: https://www.youtube.com/watch?v=-JBBlB0uTsU

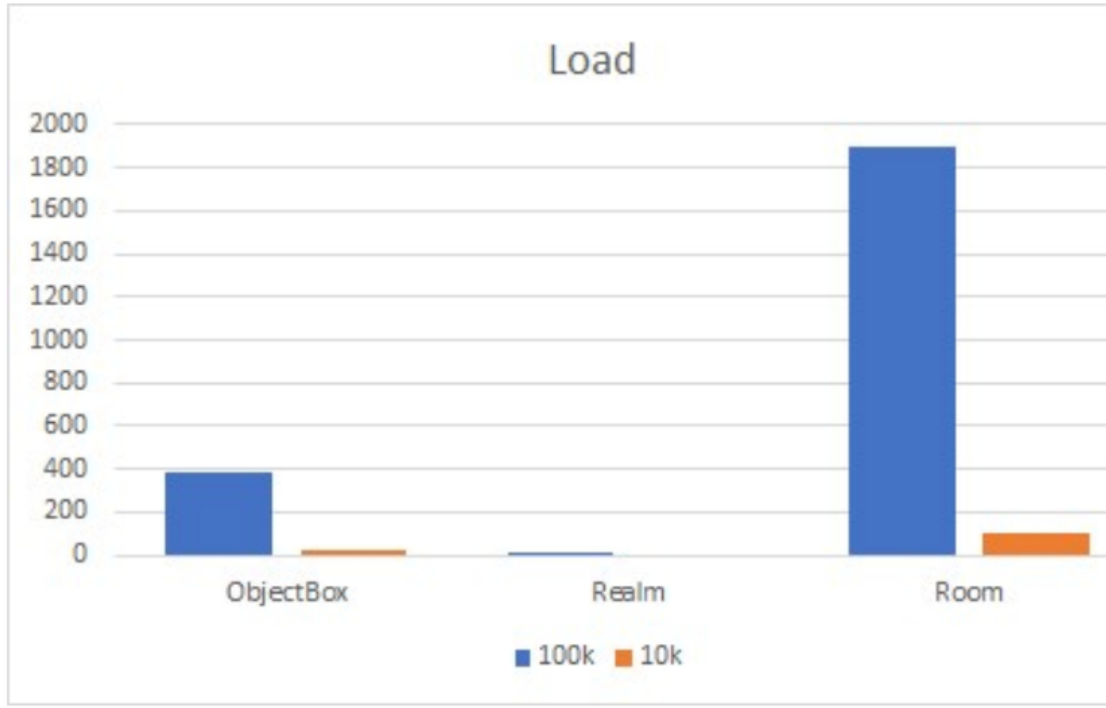
Perf matters

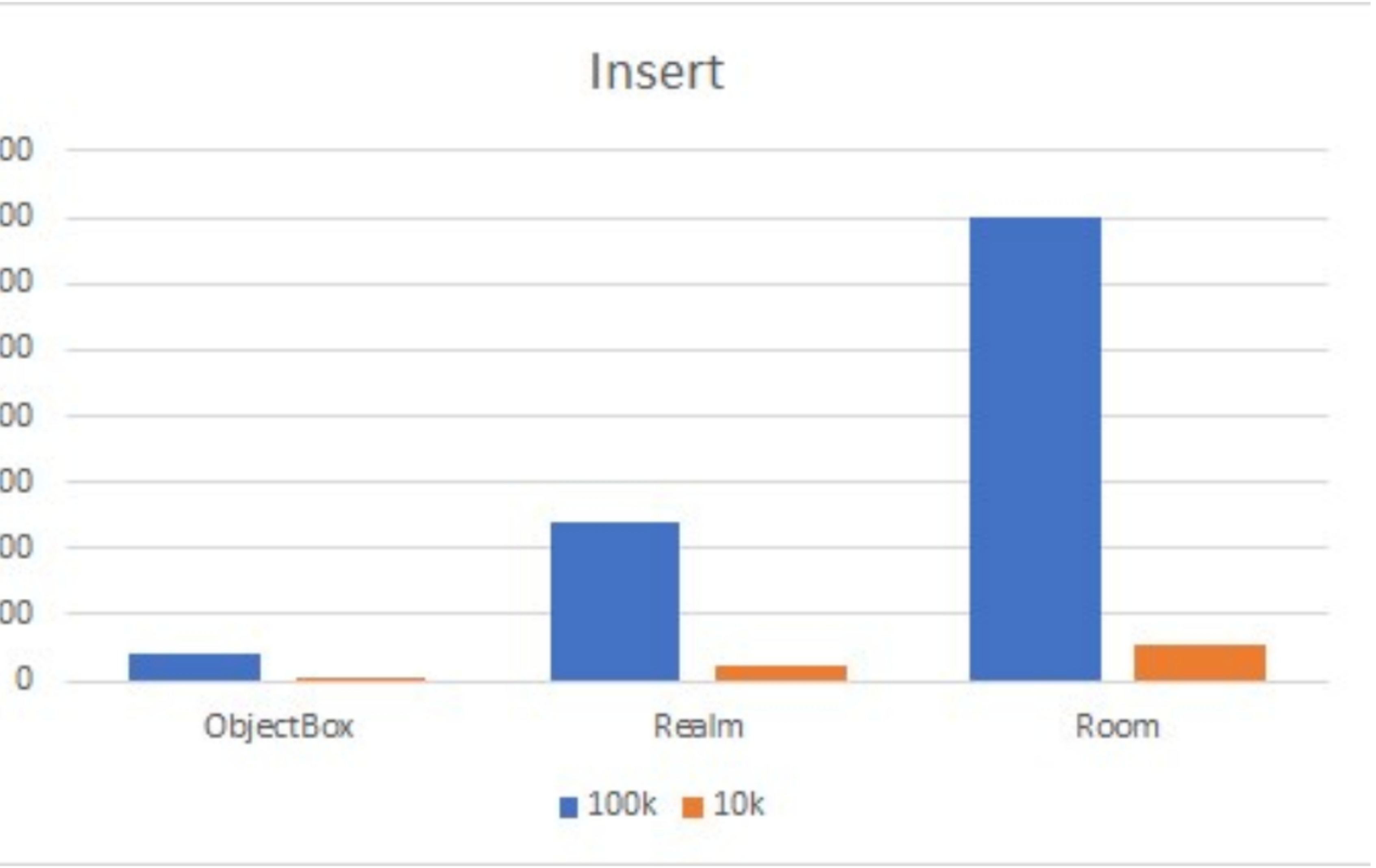
Open source benchmark: https://github.com/objectbox/objectbox-performance

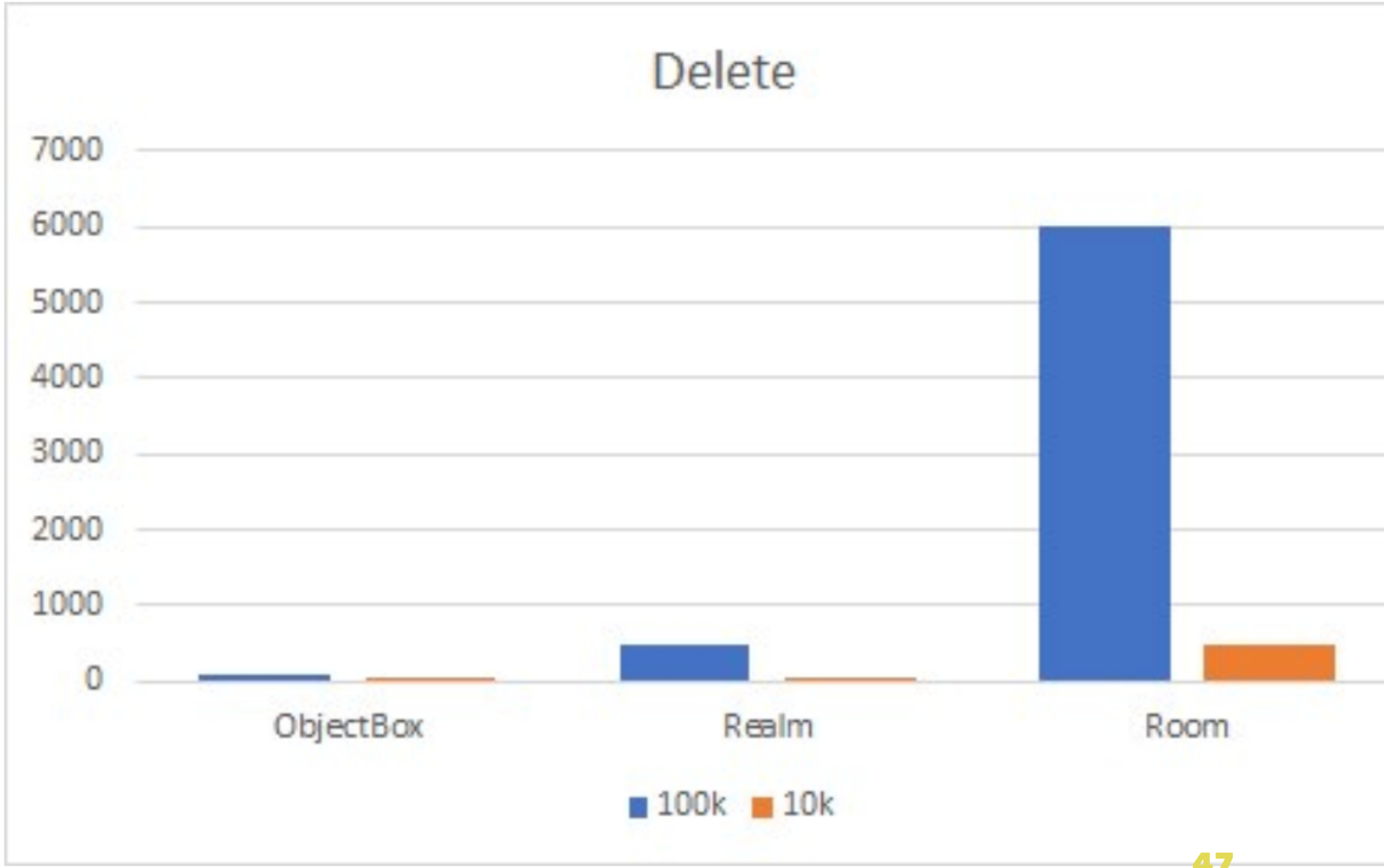
Third-party post: https://notes.devlabs.bg/realm-objectbox-or-room-which-one-is-for-you-3a552234fd6e











Performance for 100k/10k elements measured in ms

So what?

1. How much of relations do you need?
    1. Just quickly return cached requests
    2. Complex logic with offline work and relations between entities
2. How much of performance do you need?
    1. Storing a few hundreds or thousands objects
    2. Getting the data in bg thread vs reads from main thread
3. How often do you change your models?
    1. Can you just drop the data as cache?
    2. Do you often need to migrate schemas?
4. Do you need a remote sync?

There is no silver bullet!