### Ordered Key-Value Store

#### What this presentation is not?

- A comparison of database systems
- A how to guide to build your own ordered key-value store
- A how to guide to build a micro-blogging software
- A sponsored talk about a particular database
- An experience report of using an OKVS in production...

#### What is this presentation?

- Caveat Emptor.
- How to make more informed decisions when it comes to database systems?
- Getting started with Ordered Key-Value Stores programming
History

I did not review all of database literature since 1974.

1974: "SEQUEL: A Structured English Query Language" by Chamberlin et al.

1991: Berkeley DB, an ordered key-value store (later acquired by Oracle, and forked by Bloomberg)

2008: Cassandra, "speed is all that matters"

2013: FoundationDB, an ordered key-value store that can scale

2017: NewSQL: TiDB, CockroachDB and Spanner

2018: Apple open sourced FoundationDB

SQL Premise (1974)

"sub-language for both the professional programmer and the more infrequent data base user"
User survey (2016)

Infrequent database users as still disappointed.

What is a database?

- Query language
- Query optimizer
How we choose a database?

- Off-the-shelf solution claiming to be no-code or low-code.
- Vendor support.
- Pop-culture.

⇒ No ownership.
⇒ Hidden costs.

How to choose a database?

- Consider all the features and how they fit into the architecture,
- Create a list of candidates.

It is all about making tradeoffs.
Describe entities, relations and structure.

Prototype queries.

Benchmarks and tests.

**OKVS Concepts**

1. Bytes
2. Key and value
3. Key is unique
4. Keys are ordered
5. Range and prefix
6. Lexicographic order
7. Packing and unpacking
8. Lexicographic packing
9. **Ordered Mapping of Objects**
10. Space and Subspace
11. 👍 **Key Composition**
12. Copying
13. Fractal

**API: Basics**

```
(pack object) → bytevector?
(unpack bytevector) → object
(okvs-in-transaction okvs proc)
```
(okvs-ref okvs key) → bytevector?
(Okvs-set! okvs key value)

(okvs-range okvs start end [reverse? [limit]]) → procedure?
(okvs-clear! okvs key)
(okvs-clear-range! okvs start end)

_api: Cursor navigation_

(cursor-search okvs key) → <cursor> + position
(cursor-next? cursor) → boolean?
(cursor-previous? cursor) → boolean?
(cursor-key-ref cursor) → bytevector?
(cursor-value-ref cursor) → bytevector?

⭐_Minimal Viable Database_

(pack object) → bytevector?
(unpack bytevector) → object
(okvs-in-transaction okvs proc)

(okvs-ref okvs key) → bytevector?
(okvs-set! okvs key value)

(okvs-range okvs start end [reverse? [limit]]) → procedure?
(okvs-clear! okvs key)
(okvs-clear-range! okvs start end)

_api: FoundationDB only!
- Watches callback: similar to PostgreSQL
- Atomic operations: add, and, or, xor...
- ...

Compendium
- time-series
- so called "relational" database ie. row store or record store
- triple store / quad store / generic tuple store / versioned generic tuple store
- space filling curve ⇒ geometric queries
- property graph / hyper graph / atom space
- approximate string matching
- inverted index / backward index / full-text search
- ranked set / priority list / leader board

❤ tl;dr:

Managed essential complexity

Small procedural interface: binary tree with a cursor

Extensible: many higher level abstraction are possible

Usable in your favorite programming language

💔 tl;dr:
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low-level</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Little or no documentation</td>
</tr>
<tr>
<td>Benchmarks</td>
<td>No independent benchmarks</td>
</tr>
<tr>
<td>Tests</td>
<td>No independent tests</td>
</tr>
</tbody>
</table>

https://okvs.dev