EUC 2017 Stockholm - 06/2017

Building a database from scratch

barrel

- benoît chesneau
- craftsman working on *P2P and custom data endpoints technologies*
- **opensource** only
- **enki multimedia** : the corporate interface

about me

barrel

- versatile data endpoint
- micro-services, message solutions are all based about custom data endpoints
- need for a simple solution that allows you to bring the data near your service or locally.

why barrel?

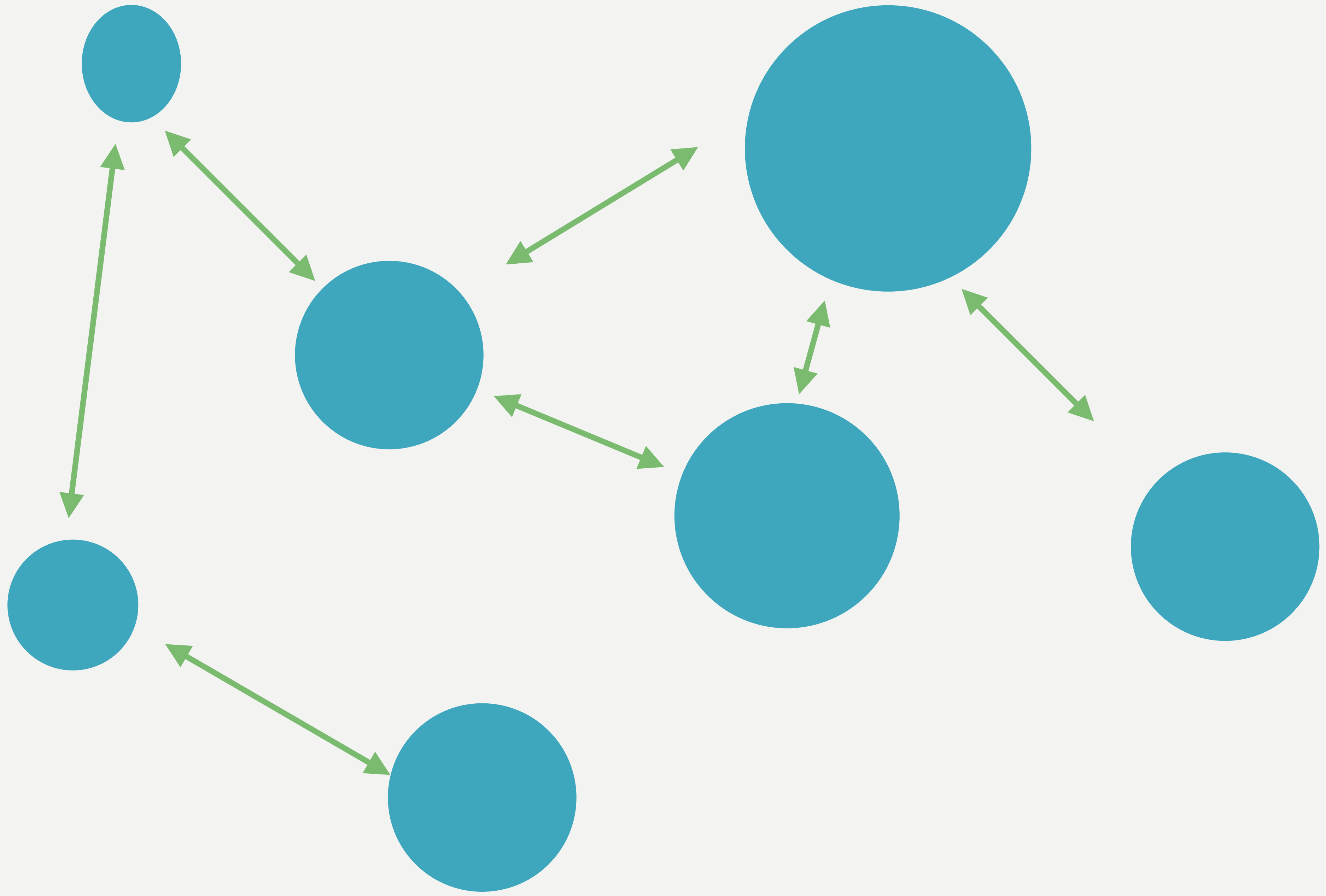
barrel

- a modern **database**
- **documents**, with time and attachments
- distributed, **local first**
- bring a **view of your data near your application**
- **automatic indexing**
- focus on simplicity

what is barrel?

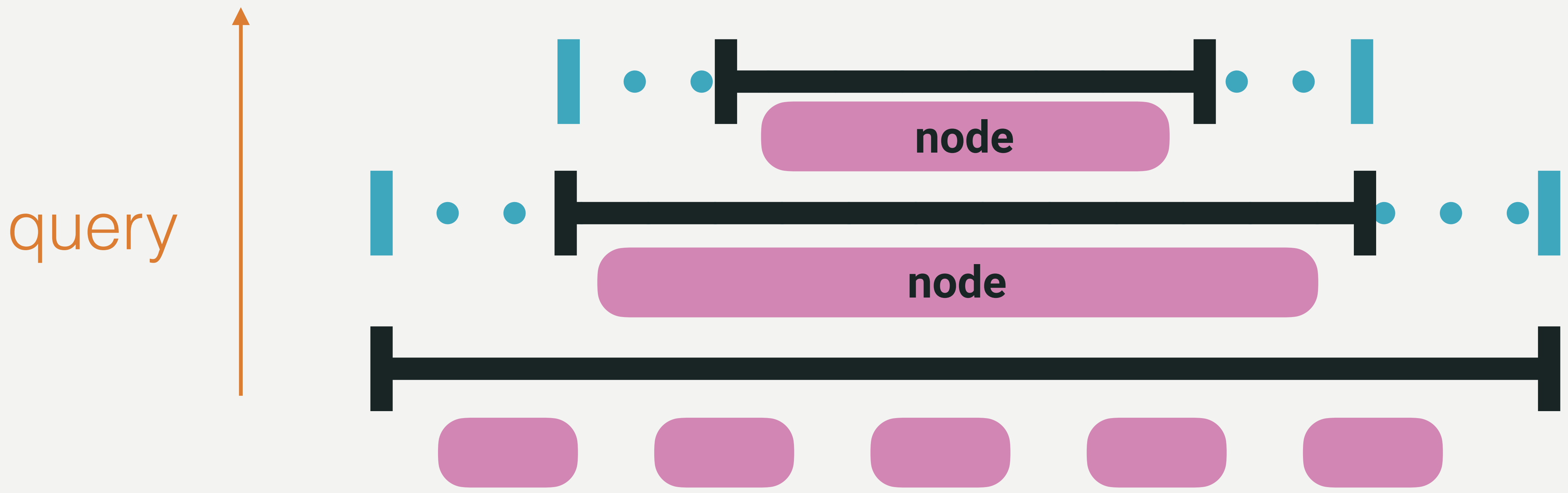
barrel

distributed: P2P



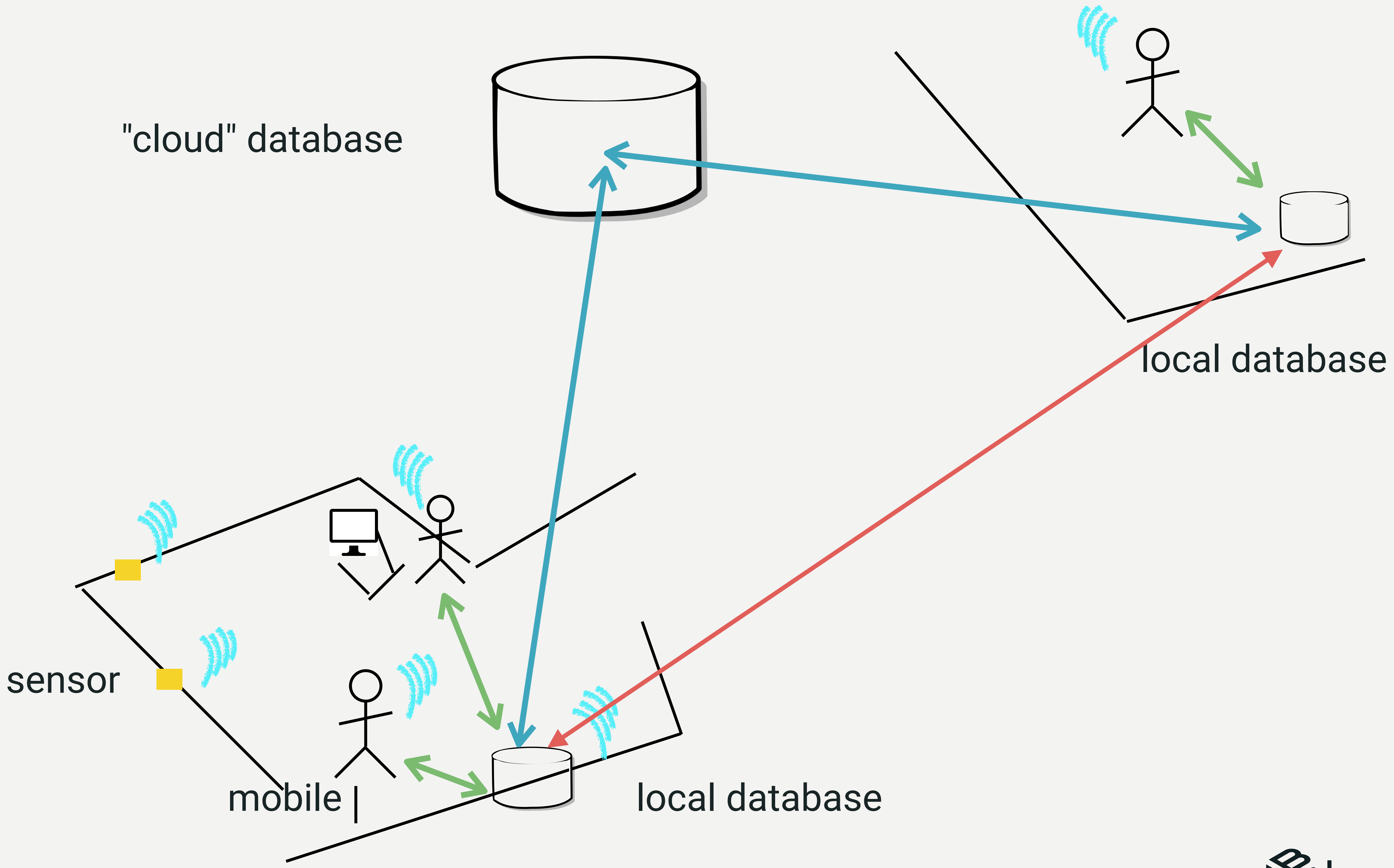
barrel

a partial view of the data



barrel

a partial view of the data

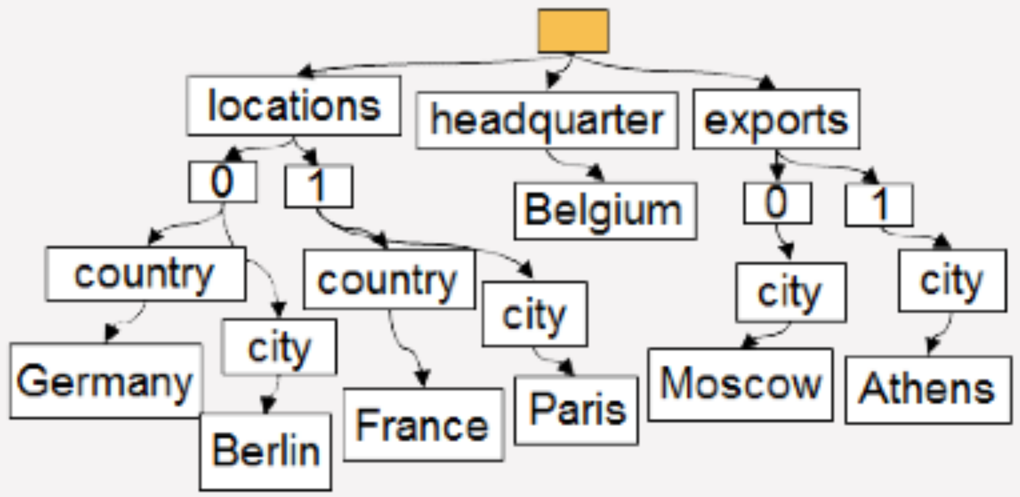


barrel

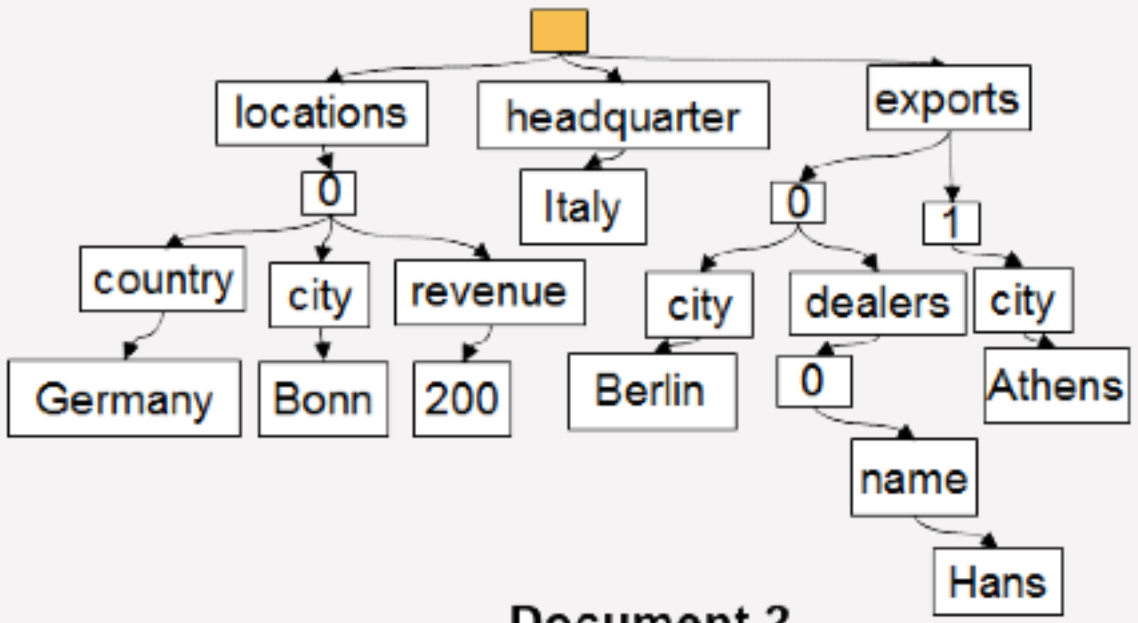
agnostic indexing

```
{ "locations": [
    { "country": "Germany", "city": "Berlin" },
    { "country": "France", "city": "Paris" }
  ],
  "headquarter": "Belgium",
  "exports":[{ "city": "Moscow" },
             { "city": "Athens"}]
};
```

```
{ "locations": [
    { "country": "Germany",
      "city": "Bonn","revenue": 200
    }],
   "headquarter": "Italy",
   "exports": [
     { "city": "Berlin","dealers":[{"name": "Hans"}]},
     { "city": "Athens" }]
};
```



Document 1



Document 2

- barrel can be embedded in your own Erlang application:
    - local database
    - no need to cache
- platform release: HTTP/Erlang pod to store and query the documents

platform

barrel

problems to solve

barrel

- stateful
- different queries return different results
- update expectations
    - read your own write?

database complexity

barrel

- processes don’t share anything
  - how do we have multiple writers and multiple readers
  - actor model
- no integer atomic operations
- IO operations are “slow”
  - until you get nifs

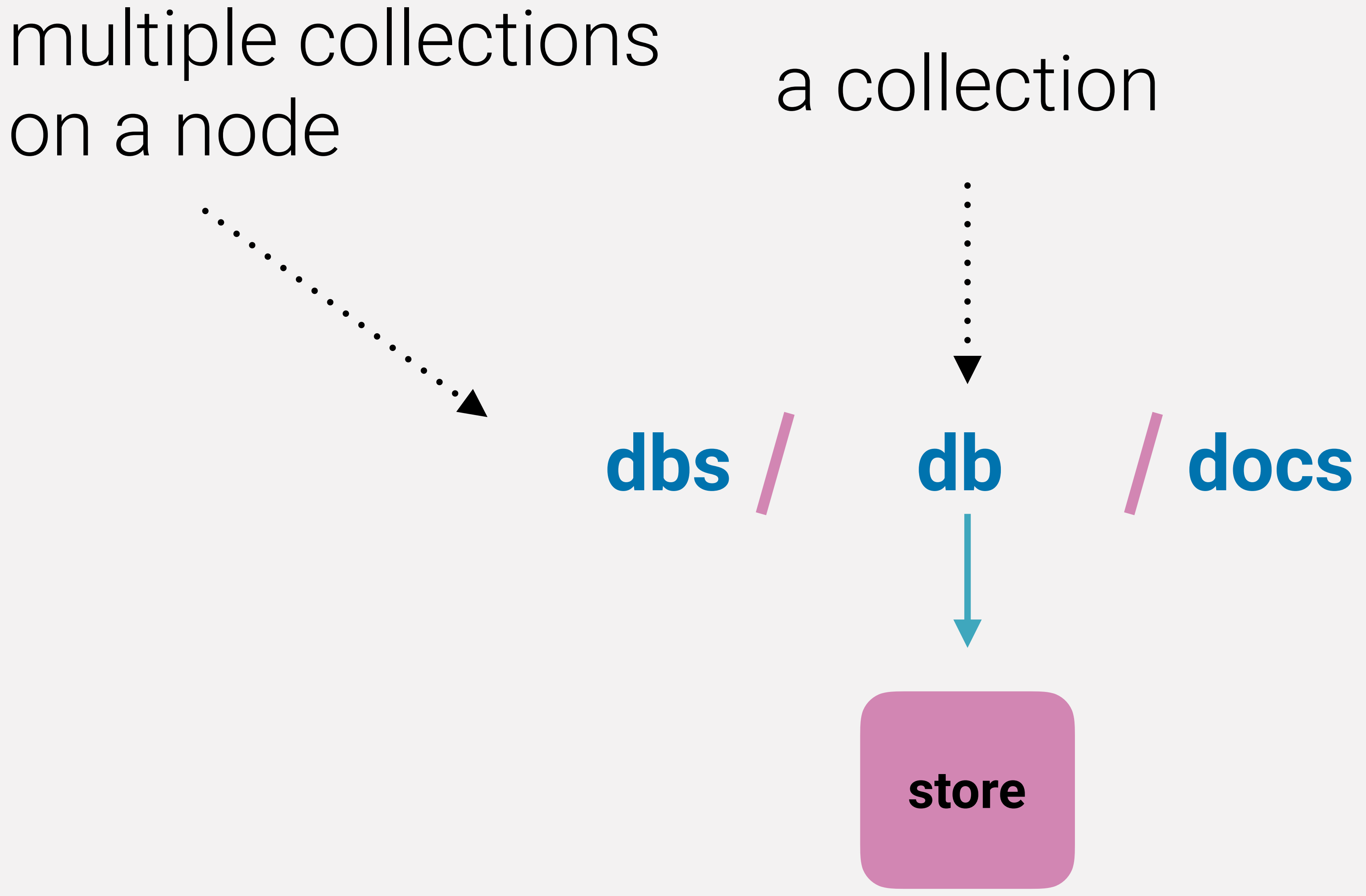
erlang constraints

barrel

- build over existing storage solutions:
  - key/value interface
  - allows atomic batch updates
  - ordered set
- 1 collection, 1 storage
- collections are small

decisions

barrel



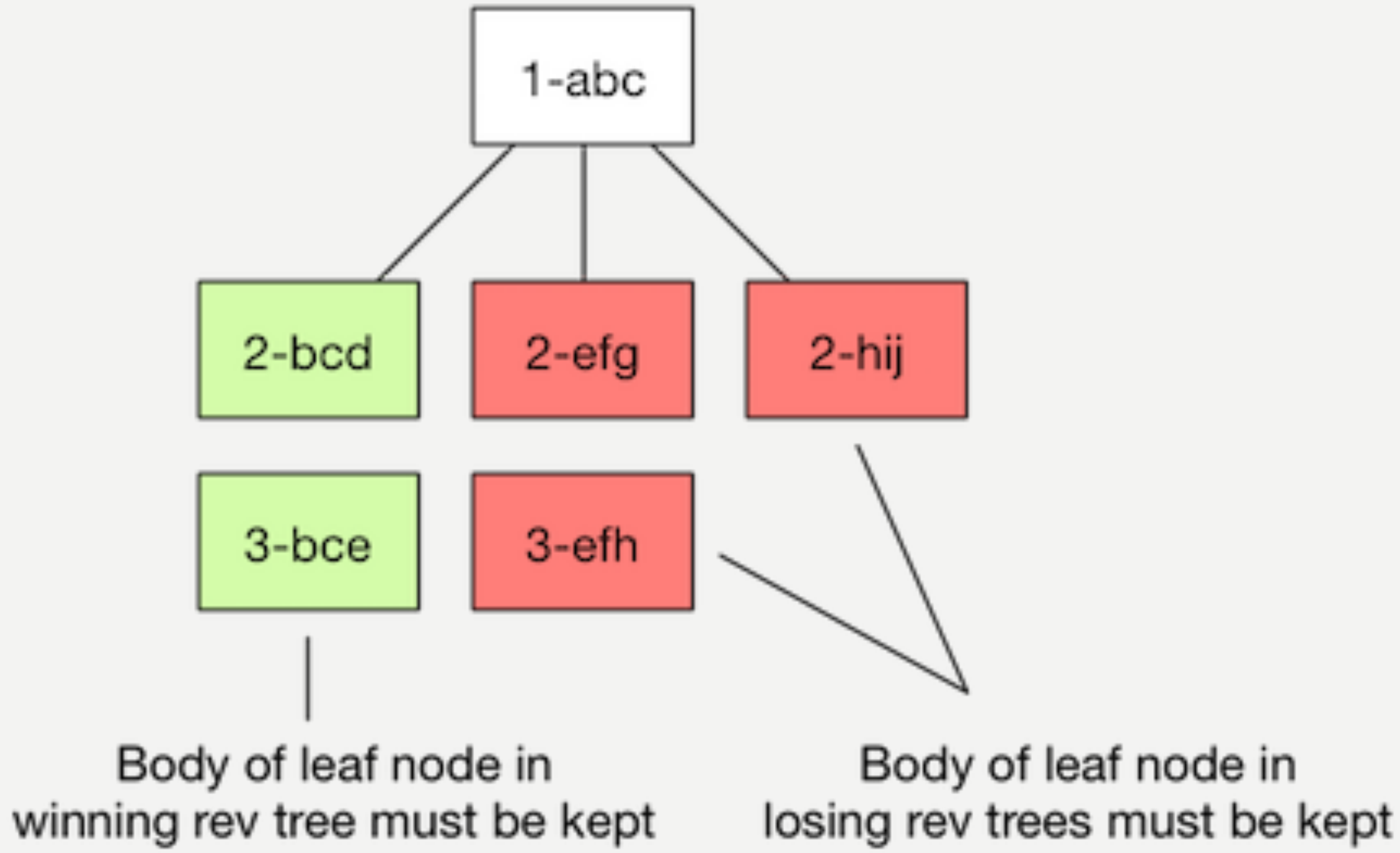
hierachical

barrel

- document:
    - map in erlang
- revision tree:
    - https://oceanstore.cs.berkeley.edu/publications/papers/pdf/hh_icdcs03_kang.pdf

storing a document

barrel


1-abc
2-bcd
2-efg
2-hij
3-bce
3-efh
Body of leaf node in winning rev tree must be kept
Body of leaf node in losing rev trees must be kept

revision tree

barrel

indexing

- 2 modes: lazy and consistent
  - lazy: indexed asynchronously based on the changes feed
  - consistent
- support maps, filter, chain opererations based on paths

barrel

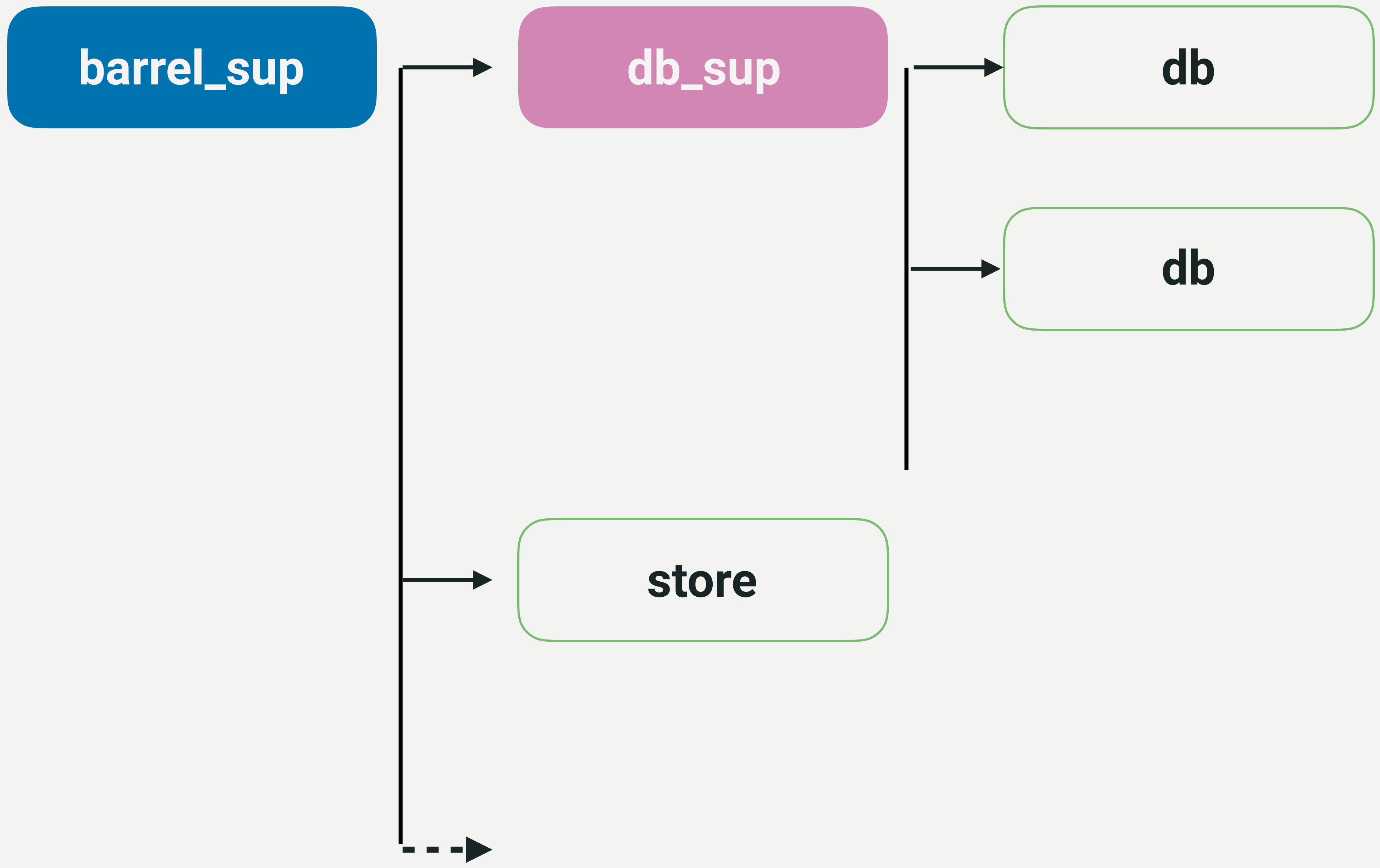
internals

barrel

- using rocksdb for the storage
    - http://gitlab.com/barrel-db/erlang-rocksdb
- used for memory and disk. optimised for SSD.
- dirty nifs

rocksdb

barrel



db supervision

barrel

- writes are queued on the main db process
- store a canonical version of doc
- states of the database is shared between other processes via ETS
  - readers are getting last db state via ets

write process (current)

barrel

prevent delayed jobs

- write more operations at once
  - selective receive
- group operations based on the document ID (merge)
- from 40 RPS to 1000 RPS on a node with 4GB of ram and 2 cores)

write process (current)

barrel

- By ID, Changes queries
- get latest DB state from ETS
- everything happen on the reader process
- coming: backpressure
  - share the db state across a pool of readers
  - remove the state from ETS

readers

barrel

- testing dispatching of write operations on different processes:
  - https://arxiv.org/pdf/1509.07815.pdf
- testing optimistic writes
- back pressure:
  - short circuit to not accept more write than the node can sustain
  - based on the running transaction and metrics
  - similar to safety valve: https://github.com/jlouis/safetyvalve

write process rewrite

barrel

- just appending data to the storage we never read from old index values
- inside the DB process for consistent write
- a process listening on db updates events (using a simple gen_server, no gen_event)
- index policies to index each json segment to retrive via their valur or hash to support value or range queries.

indexing process

barrel

- over HTTP
  - cowboy 2
- over TCP using teleport and Erlang serialisation (coming):
  - https://gitlab.com/barrel-db/teleport
  - allows embedded mode

replication

barrel

add some instrumentation

barrel

- how to not block without counting
- first try: statsd client sending to an UDP endpoint counter/gauge/histogram updates
- we run out of processes & file descriptors
- asynchronous sending: better.
- how to make generic?

instrumenting

barrel

- add hooks
  - https://github.com/benoitc/hooks
  - prometheus plugin and wombat support (EE version)
  - internal metrics sytem
    - https://gitlab.com/barrel-db/lab/instrument

```
barrel_start_transaction(Trans, DbName) ->
  erlang:put(barrel_transaction_start_time, erlang:monotonic_time()),
  prometheus_counter:inc(barrel_db_transactions, [DbName, Trans]).
```

instrumenting

barrel

roamap

barrel

- **0.9 release: 2017/06/13**
    - https://gitlab.com/barrel-db/barrel-platform
- add **documentation** (june 2017)
- optimise writing
- atomic updates
- enrich query engine.

roadmap

barrel

?

barrel

contact

twitter: @barreldb
web: https://barrel-db.org

barrel