If NoSQL is your answer, you are ~~probably~~ asking the wrong question.

Hi, my name is

Lukas Kahwe

Smith

and I am **not** a

troll


@lsmith

@lsmith77

SQL

For the conference we're looking for someone who can be a part of the 'Databases - short talks' presentations to give a lightning talk about SQL.

SQL

NoSQL

No SQL?

Not Only SQL?

Time for some pseudo math

Given that most relational databases have an SQL interface

RDBMS - SQL = NoSQL ?

NoSQL + SQL = RDBMS ?

Key-Value-Store + Complex Queries = NoSQL ?

I am about to blow your mind

I have created a database API for **unstructured hierarchical documents on top of an RDBMS** with an SQL-like interface

And I wasn't even the first person to do it ..

Can we agree to **stop using** the term **NoSQL** ?

Using NewSQL is not any better!

Instead talk about **RDBMS**,

Doc-Stores, **Key-Value-Stores**, **Graph-Databases** ..

There is clearly a **growing** number of **people** considering **alternatives** to **RDBMS**

Actually **Key-Value-Stores** have been popular **for** many **years** already

So clearly the current trend is **not** just **about sharding** unstructured **documents** in an **elastically scaling** cluster

Its also not about getting rid of SQL, after all the **CAP** theorem **doesn't talk** about what **query languages** to use

RDBMS - relational model = ?

We still love RDBMS for what their good for

Storing and **retrieving**

structured **relational** small to

large data sets with high

consistency and **reliability**

And yes we also still love SQL

Running **ad-hoc** search **queries** and **schema updates** even if we are not rocket scientists

http://2012.nosql-matters.org/cgn/slides/#olaf_bachmann

And while JSON queries might be easier parsed by computers

SQL is definitely **easy** to parse **for humans**

So what do we **really** worry about .. ?

ACID, Asynchronous, Atomic Updates, BigData, Binaries, CAP Theorem, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, BigData, Binaries, CAP Theorem, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, **BigData**, Binaries, CAP Theorem, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, BigData, Binaries, **CAP Theorem**, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, BigData, Binaries, CAP Theorem, Column Oriented, **Elastic Scaling**, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, BigData, Binaries, CAP Theorem, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, **FullText Search**, Graphs, Low Latency, In Memory, MapReduce, Master-Master, Sharding, Smart Clients

ACID, Asynchronous, Atomic Updates, BigData, Binaries, CAP Theorem, Column Oriented, Elastic Scaling, Eventual Consistency, Failover, FullText Search, Graphs, Low Latency, In Memory, MapReduce, Master-Master, **Sharding**, Smart Clients

RDBMS vendors **got stuck** as a result of their own success

Innovators Dilemma

"The two biggest issues that stand out to me are that current leading relational databases **never solved scale out** very well, and **online operations are too expensive.** [..] The innovation hasn't been in the language, but in the design of database engines themselves."

Brian Aker (Drizzle, previously MySQL)

http://blog.krow.net/2013/03/mysql-vs-nosql-vs-postgres-vs-sql.html

The good news is that

RDBMS vendors have **been unstuck**

However its hard to **move** on **from** a **monolithic** **architecture**, Drizzle is a radical attempt at trying it

In many ways its easier to

start from scratch

But lets look at some **examples** of things that have happened in the **RDBMS** world

VoltDB, ScaleBase and NuoDB promise **elastic scaling** on top of an **RDBMS** with SQL and ACID* compliance

http://voltdb.com/tao-volt/five-principles.php
http://www.scalebase.com/solution/
http://www.nuodb.com/explore/sql-cloud-database-how-it-works/

Both **PostgreSQL** and **MySQL** have native **support** for **JSON** (and XML)

PostgreSQL performs reads **on par** with **MongoDB**, especially for larger data sets

http://www.slideshare.net/stormdb_cloud_database/postgres-xc-askeyvaluestorevsmongodb
http://jathanism-event-notes.readthedocs.org/en/latest/scale11x/talks/postgres_as_a_schemaless_db.html

MySQL Server and MySQL Cluster allow by passing SQL via the **Memcache** protocol

http://blog.ulf-wendel.de/downloads/nosql_in_mysql.pdf

MySQL and **PostgreSQL**
both have forks providing
Multi-Master replication

http://www.enterprisedb.com/products-services-training/products-overview/xdb-replication-server-multi-master
http://www.codership.com/content/using-galera-cluster

OQGRAPH Engine to store graphs in **Maria DB**

http://openquery.com/products/graph-engine

MySQLND plugins for **client** side **query caching**, transaction aware **load balancing**

http://php.net/mysqlnd_qc
http://php.net/manual/en/book.mysqlnd-ms.php

Conclusion

SQL is not a scalability enemy

SQL is **not** a **scalability enemy**

How **big** is your **data** ..

really?

Operational **experience** only comes **over time**, having **something solid** that **just works** matters

There can be **innovation** in the **RDBMS world**

Using an RDBMS is fine

So is using **Doc-Stores**, **Key-Value-Stores**, **Graph DBs**

Using **NoSQL** is **not fine**

Just pick the **right** tool **for the job** ..

Duh!

Fin

Everything is a search - scale it...

Simon Willnauer
Elasticsearch Inc.

9 min 45 seconds left....

```
# $ whoami

curl -s -XPUT 'http://localhost:9200/hacker_index/hacker/1?pretty=true' -d '{
  "name" : "Simon Willnauer",
  "profession" : [ "Co-Founder Lucene Hacker @ ElasticSearch",
                   "Lucene Core Committer since 2006 and PMC Member"],
  "passion" : "Information Retrieval, Machine Learning, Concurrency",
  "freetime" : "Runner, Swimmer, Father & Berlin Buzzwords Co-Organizer",
  "twitter" : "https://www.twitter.com/s1m0nw"
}'
```

9 min 15 seconds left....

```
##########################################################################
# What is a Search?
##########################################################################

# “how tall is george w. bush” (who cares?)

# “and how tall is his father”

# “foo fighters”

# “high cholesterol diet”

# “Champions League game tonight”

# “most prominent tags on twitter in the last week”

# “aggregated counts of origin countries accessing my website this month”

# “shoe size distribution across the user signed up last month”

# “more results like the first result of my previous search”
```

9 min 15 seconds left.... to scale...

```
##########################################################
# What is Lucene?
##########################################################

$ echo “I wish I could talk as fast as Jim Webber!” | Lucene

{
   "could": [{ "term_freq":1, "doc_id":0 , pos: [4]}],
   "fast": [{ "term_freq":1, "doc_id":0, pos: [7]}],
   "jim": [{ "term_freq":1, "doc_id":0, pos: [9], payload: [“firstname”] }],
   "talk": [{ "term_freq":1, "doc_id":0, pos: [5] }],
   "wish":[{ "term_freq":1, "doc_id":0, pos: [2] }],
   "webber": [{ "term_freq":1, "doc_id":0, pos: [10], payload: [“surname”]}]
}

$ echo “I wish I could be as entertaining as Jim Webber!” | Lucene

{
   "could": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   "entertaining": [{ "term_freq":1, "doc_id":1 }],
   "fast": [{ "term_freq":1, "doc_id":0 }],
   "jim": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   "talk": [{ "term_freq":1, "doc_id":0 }],
   "wish":[{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   "webber": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }]
}
```

What can you do with it?

```
##################################################################
# 2 Basic Operations
##################################################################

$ seekExact(“webber”)

{
   "could": [{ "term_freq":1, "doc_id":0 }],
   "fast": [{ "term_freq":1, "doc_id":0 }],
   "jim": [{ "term_freq":1, "doc_id":0 }],
   "talk": [{ "term_freq":1, "doc_id":0 }],
   "wish":[{ "term_freq":1, "doc_id":0 }],
   --> "webber": [{ "term_freq":1, "doc_id":0 }]
}

$ seekCeil(“enter”)

{
   "could": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   --> "entertaining": [{ "term_freq":1, "doc_id":1 }],
   "fast": [{ "term_freq":1, "doc_id":0 }],
   "jim": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   "talk": [{ "term_freq":1, "doc_id":0 }],
   "wish":[{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }],
   "webber": [{ "term_freq":1, "doc_id":0 }, { "term_freq":1, "doc_id":1 }]
}
```

Recap please?

```
# 2 Basic Operations on a sorted dictionary

# Supporting incremental update & Near Real Time

# Massive set of language / text processing tools

# Pre-computed statistics allows to score by relevance give a query

# Proximity information in the index

# Highly optimized read-only / immutable data-structures

# But.... it’s just a library and you need to write code to use it...
```

Here comes ElasticSearch

```
# Create an index - 3 physical Lucene indices & 2 copies each for redundancy
curl -XPUT 'http://localhost:9200/phrase_index/' -d '{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 1
        }
    }
}'
```

```
# start indexing
curl -s -XPUT 'http://localhost:9200/phrase_index/comment/1?pretty=true' -d '{
  "phrase" : "I wish I could talk as fast as Jim Webber!",
  "date" : “2013-04-10”,
}

curl -s -XPUT 'http://localhost:9200/phrase_index/comment/2?pretty=true' -d '{
  "phrase" : "I wish I could be as entertaining as Jim Webber!",
  "date" : “2013-04-11”
}
```

A powerful tool easily accessible...

```
# Lookup by ID
curl -XGET 'http://localhost:9200/phrase_index/comment/1' # NoSQL you know!

# simple plain search
curl -XGET 'http://localhost:9200/phrase_index/comment/_search?q=Jim Webber'
```

```
# or use the powerful Query DSL
curl -XPOST 'localhost:9200/twitter/_search?pretty=true' -d '{
    "query": {
        "filtered" : {
            "query" : {
                "match": { "phrase" : "jim webber" }
            },
            "filter" : {
                "range" : {
                    "created-at" : {
                        "from" : "now-1d",
                        "to" : "now",
                    }
                }
...
```

Back to the searches

```
##########################################################
# What is a Search?
##########################################################

# “how tall is george w. bush” (who cares?)
 - named entity recognition + payloads ✓
# “and how tall is his father”
 - context sensitive queries (you need to sell your own phone I guess? )
# “foo fighters”
 - phrase searches ✓
# “high cholesterol diet”
 - TF-IDF Ranking, BM25, Custom Boosting etc. ✓
# “Champions League game tonight”
 - combined with other tools like OpenNLP ✓
# “most prominent tags on twitter in the last week”
 - date range filter & term faceting ✓
# “aggregated counts of origin countries accessing my website this month”
 - date range filter & term faceting ✓
# “shoe size distribution across the user signed up last month”
 - date range filter & Histogram Faceting ✓
# “more results like the first result of my previous search”
 - MoreLikeThis ✓
```

That's it!

Datomic

(in 15 minutes)

Data, meet facts

Entity Attribute Value

Data, meet facts

Entity	Attribute	Value
Thomas	likes	pizza

Data, meet facts

Entity	Attribute	Value
Thomas	likes	pizza
Uri	capital is	Altdorf

Data, meet facts

Entity	Attribute	Value
Thomas	likes	pizza
Uri	capital is	Altdorf

identifier

named schema attribute

valid value for attribute

Querying

Find name of cantons
where German
is spoken

```
[:find ?name
 :where
 [?c :canton/name ?name]
 [?c :canton/languages ?l]
 [?l :language/name "German"]]
```

Find name of cantons
where capital and
canton has same name

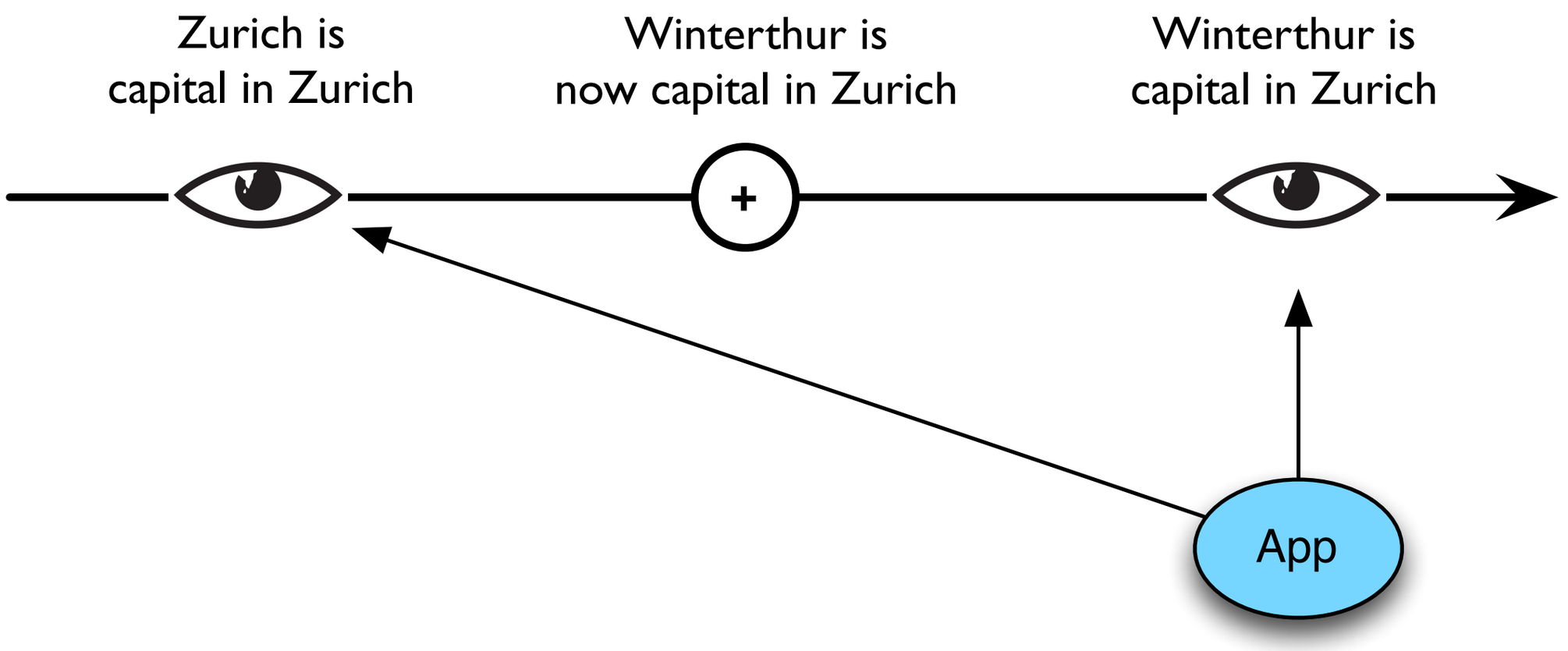
```
[:find ?name
 :where
 [?c :canton/name ?name]
 [?c :canton/capital ?name]]
```

Count all cantons
where capital and
canton has same name

```
[:find (count ?c)
 :where
 [?c :canton/name ?name]
 [?c :canton/capital ?name]]
```

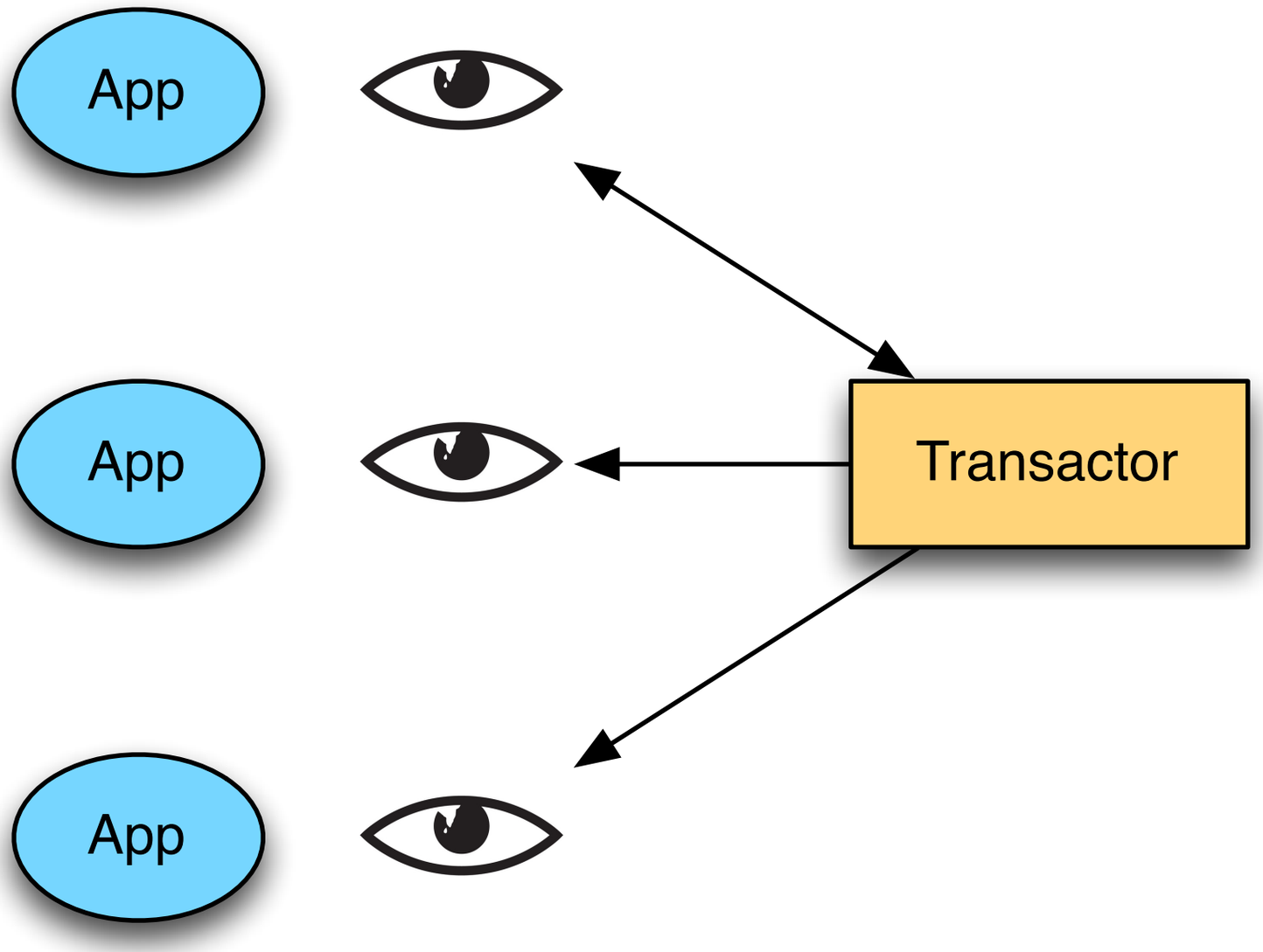
(it's 12)

Temporal database



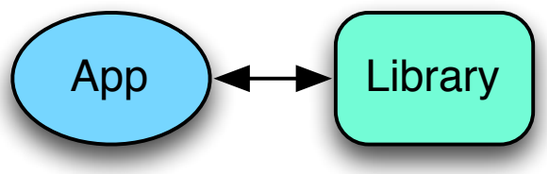
Free debugging and auditing

Database as a value

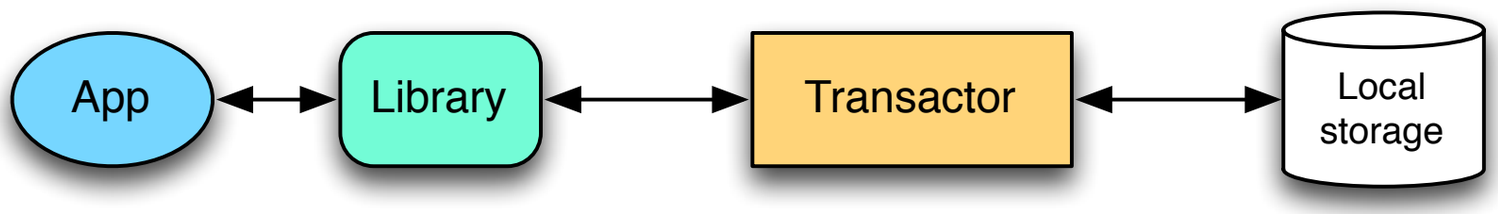


Architecture

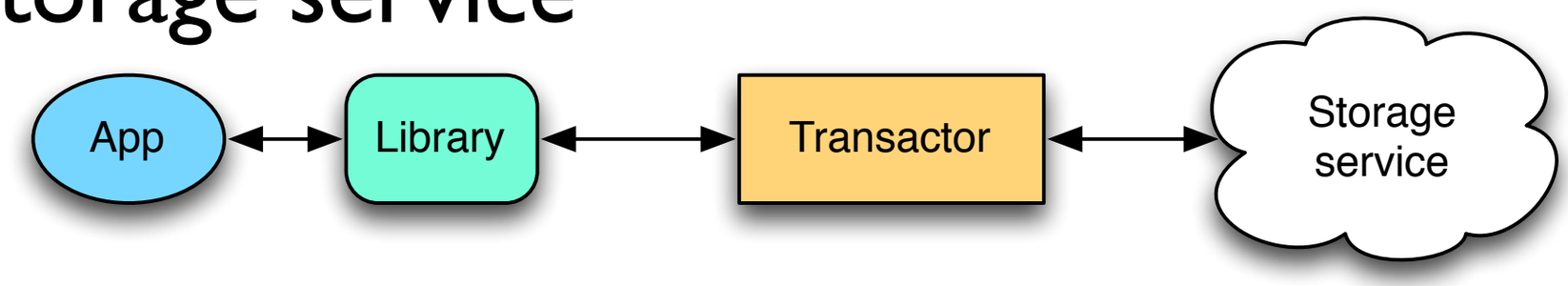
- In-process



- On-disk



- Storage service



Summary

- Data is facts
- Query with datalog
- Temporal data
- Architecture separates facts and storage
- Database as a value

Bi-temporal database