Lecture 9: NoSQL

Professor Robert J Brunner
Quinn Jarrell

How was lab?

- Do not use sort by, only order by

Lab ∞

- No lab over spring break
- Now your projects are the only thing to do
- Each week you will be expected to show concrete progress
  - Get graded down if you don’t
- We don’t expect you to do work over spring break but we do expect progress by next lecture

Lab ∞ - continued

- We’re changing the structure of the lab
- Split yourselves into groups of 2, each group of 2 will do one report

Lab ∞ Schedule

- Mar 19 – 25, 2017 - Spring break
  Mar 26 – Apr 1, 2017 - Research
  Apr 2 – 8, 2017 - Research/Begin writing paper
  Apr 9 – 15, 2017 - Write Paper
  Apr 16 – 22, 2017 - Peer Review
  Apr 23 – 29, 2017 - Revise Paper
  Apr 30 – May 6, 2017 - Everything is due

A little bit of history

- Most databases became SQL like in the 1980s
  - There were still non SQL like databases all the time
- In 2006 Google published their BigTable paper
  - It was not SQL
  - It was designed to scale to petabytes of data (1000s of gigabytes)
  - It scaled across hundreds to thousands of machines
  - Solved scaling by relaxing availability
- In 2007 Amazon published their Dynamo paper
  - Again not SQL
  - Similarly solves the problem of scaling
  - Solved scaling through relaxing consistency
- By 2009 there were tons of systems like these
- Now when you have hundreds of nodes, NoSQL is the normal solution

NoSQL

- Databases which may not be relational and can scale to tons and tons of servers
- Sacrifice SQL compatibility to get higher read/write/storage rates
- Only needed when data cannot be managed a few servers

SQL vs NoSQL

- Anything that does not use SQL or does not provide the same features
- SQL systems are typically good at consistency
  - If I write to a row, all reads will get that write
  - This slows down everything
- The vast majority of databases (not only SQL) are ACID
  - Atomic
  - Consistent
  - Isolated
  - Durable
- ACID just means that it has the same properties as a global variable in a single threaded program.

NoSQL Types

- Key Value Store
- Document Oriented
- Columnar Storage

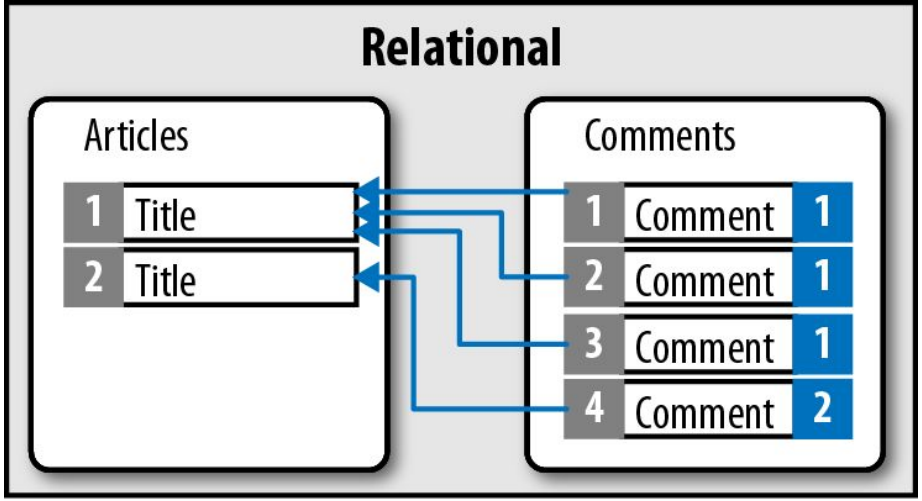
Key Value Store

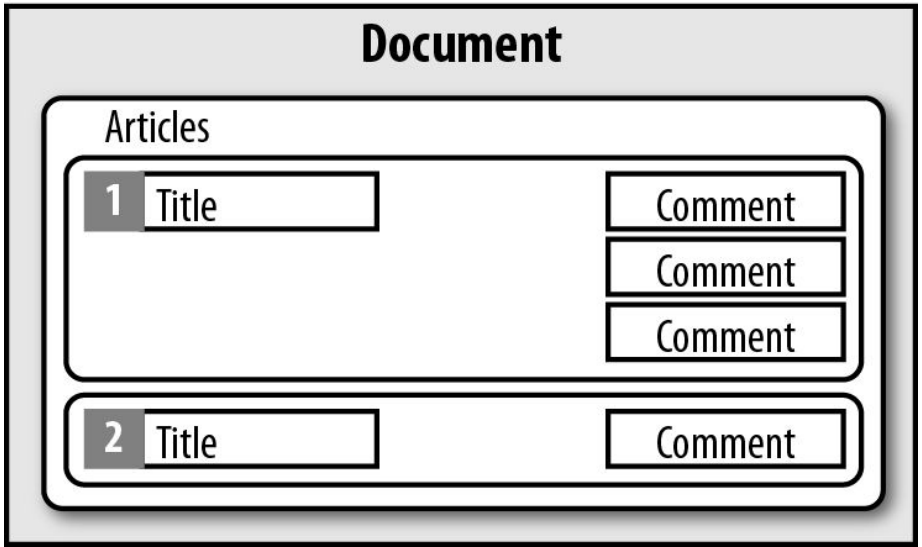
- These store key value pairs really really well
- Can be used as a very good cache
- Some document stores are key value stores under the hood

```
{
    Key1: val1,
    Key2: val2
}
```

Document Oriented Databases

- They store complex structures like
  - JSON
  - XML
  - YAML
- These work really well when most queries are for one item instead of aggregations
- Typically provide their own unique query languages
- These extend the idea of key value stores to more complex types


Relational
Articles
1 Title
2 Title
Comments
1 Comment 1
2 Comment 1
3 Comment 1
4 Comment 2


Document
Articles
1 Title
Comment
Comment
Comment
2 Title
Comment

Document Vs Key Value Databases

Both address objects with a key

- Document DBs cluster documents within collections
- Key value stores mainly have only one collection
- Key value stores are faster (Smaller values and less structure)
- Document DBs support more extensive query languages in general
- If you do not need complex objects, use a key value store

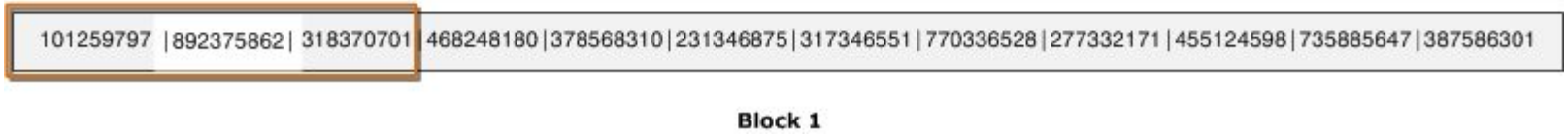
Columnar/Column based Databases

- Columns are stored together instead of rows
- A row is can be split amongst many machines
- Makes aggregations really fast since a single column normally resides on one machine
- Does not support joins
- Parts of a row can be stale

Row based databases

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

Columnar based databases



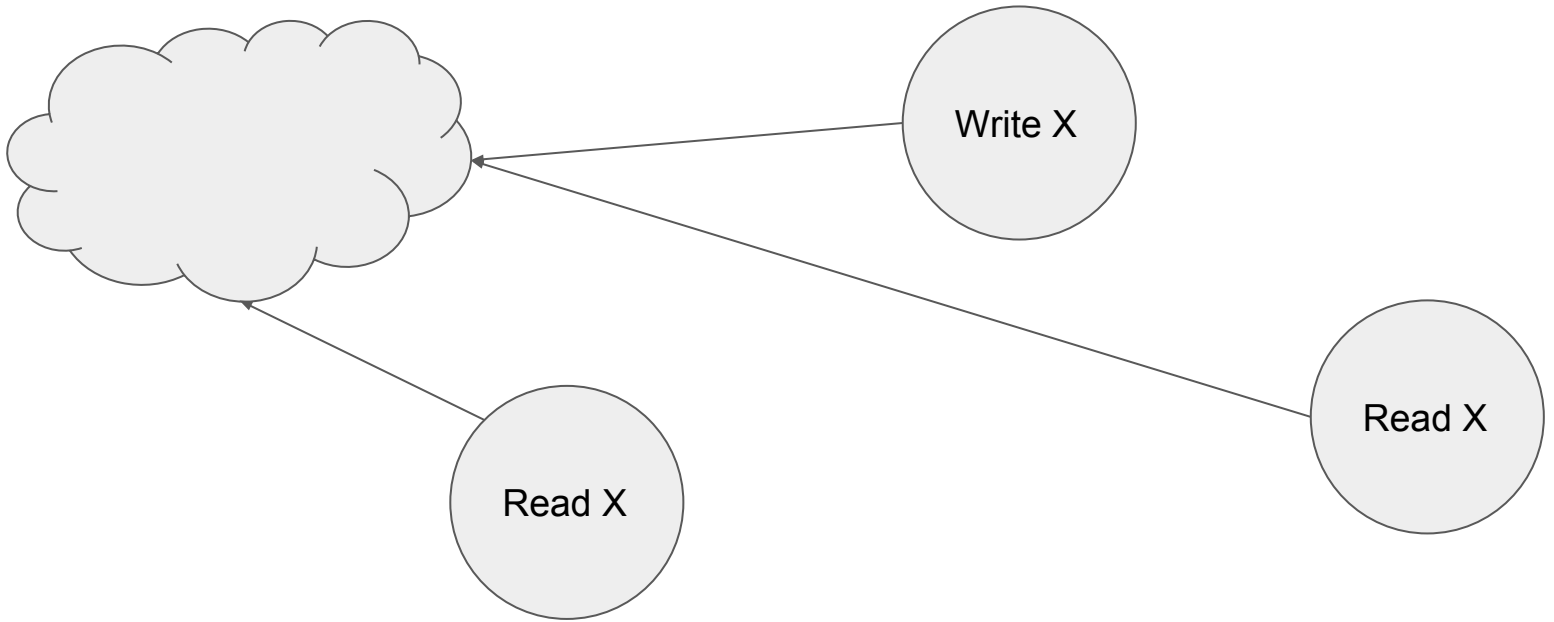
Source: AWS

Relaxing Constraints

- All of the above types can be implemented using a normal SQL database a backend
- They can and are also implemented as ACID databases
- But sometimes you can deal with things not being ACID
- What if we specified you did not need strong consistency?

Eventual Consistency

- Making everything consistent immediately means clients need to queue
- Say you have 3 clients, one writing and the other two reading



Eventual Consistency

- Making everything consistent immediately means clients need to queue
- Say you have 3 clients, one writing and the other two reading

The true ordering is

T=0	T=1	T=2
Read X	Write X	Read X

Eventual Consistency

- But you could receive this order because of network delays

T=0	T=1	T=2
Read X	Read X	Write X

Eventual Consistency

- Or if you have two servers, one could receive the true ordering and one the out of order ordering

Server 1 sees

T=0	T=1	T=2
Read X	Write X	Read X

Server 2 Sees

T=0	T=1	T=2
Read X	Read X	Write X

Eventual Consistency

- But sometimes we can afford old values being read for a little while. This means we can read and write at the same time.

Server 1 sees

T=0	T=1	T=2
Read X	Write X	Read X

Server 2 Sees

T=0	T=1	T=2
Read X	Read X	Write X

CAP Theorem

- Consistency
  - All reads receive the most recent write or error
- Availability
  - Every read/write recieves a non error
- Partition Tolerance
  - Everything keeps working if the network starts dropping messages

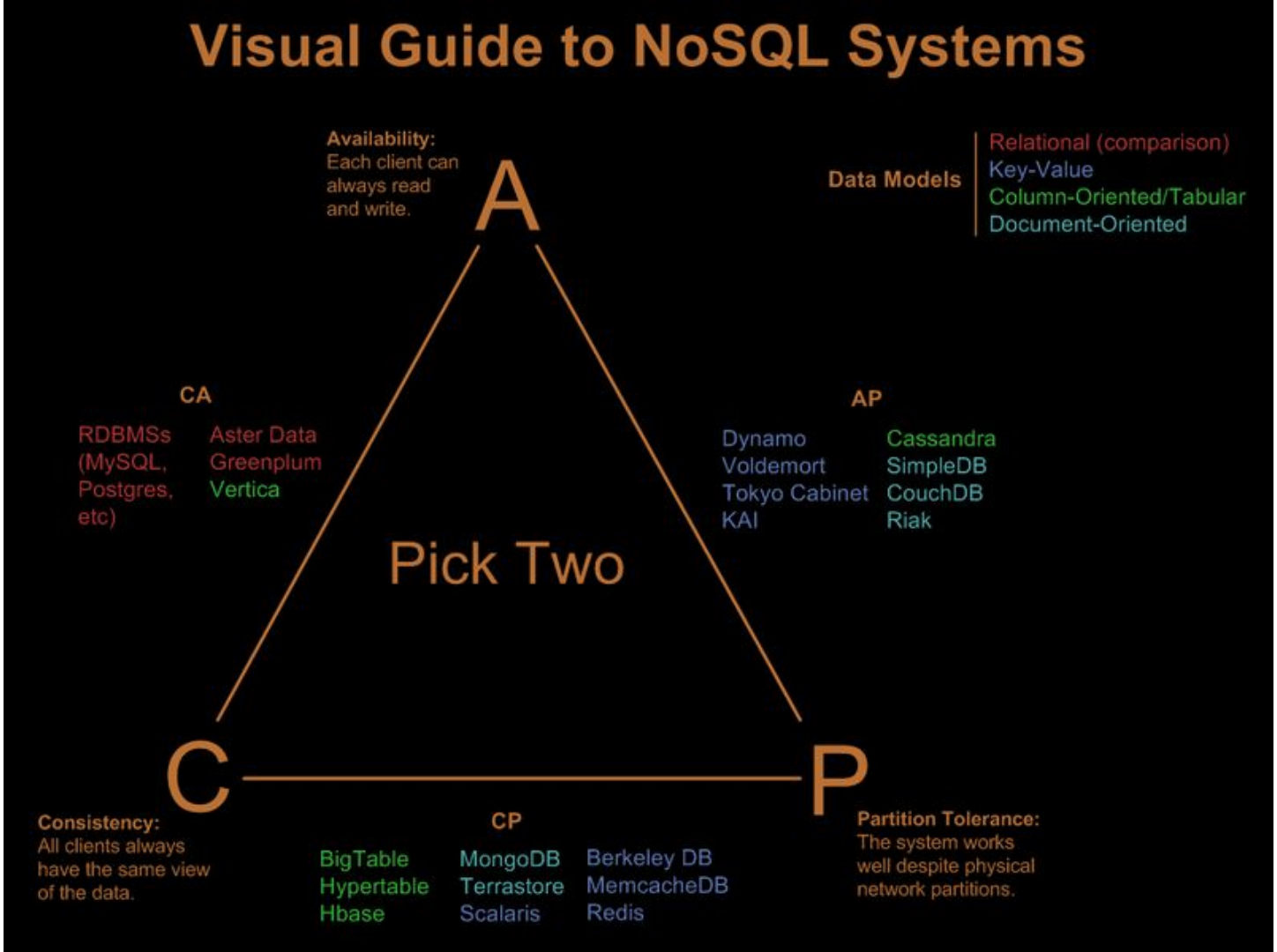
Pick 2

CAP Theorem

- Consistency
  - All reads receive the most recent write or error
- Availability
  - Every read/write recieves a non error
- Partition Tolerance
  - Everything keeps working if the network starts dropping messages

Pick 2

- Each of these have a non strict version
- But you cannot guarantee all 3 in all scenarios


Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

From Ofirm

CA Systems

- Consistency and availability
  - They will always respond with the latest write
- Most SQL databases are CA systems.
- SQL Systems
  - MySQL
  - MSSQL
  - SQLite
  - PostgreSQL

CP Systems

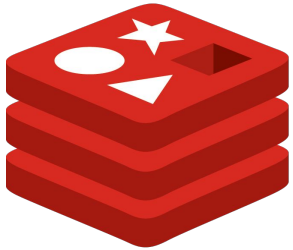
Consistency and Partition Tolerance

- Will give you the latest write or give you an error if not possible
- Can survive half the network going down
- HBase, BigTable, MongoDB

- Is a CP system
- Used in HDFS
- Linear and modular scalability.
- Strictly consistent reads and writes.
- Automatic and configurable sharding of table

- Everything is still a table
- Can return an error since it is CP

- Is a CP system
- Extremely easy to set up
  - The defaults are insecure
- Document Oriented DB
  - Only stores JSON objects
  - No longer a simple table
- Is a key value store

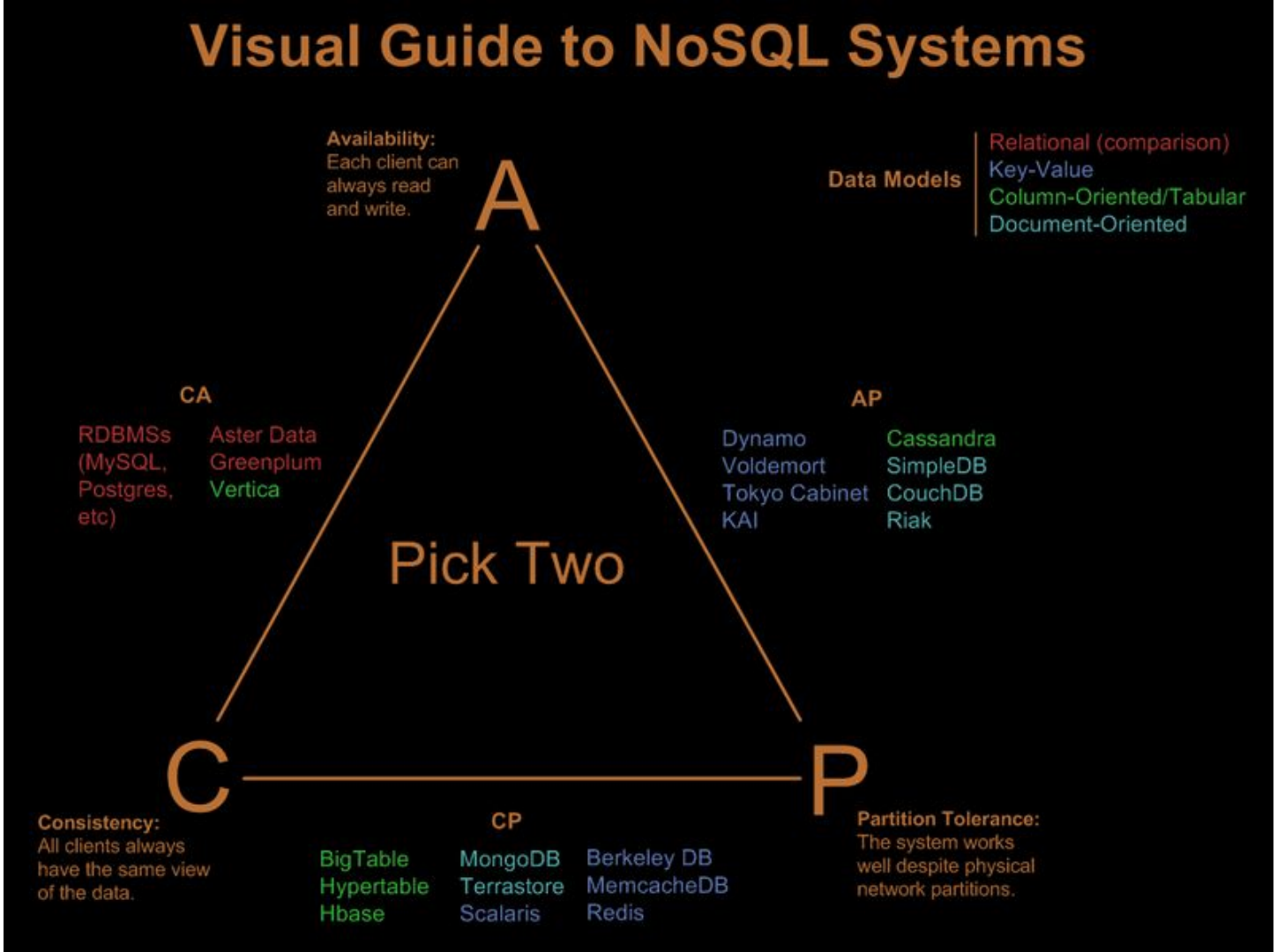


redis

- CP System
- Is a key value store
- Lets you write datastructures into memory and share them

AP Systems

- Available and partition tolerant
- Never returns an error even when half the network is dead


Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

From Ofirm

- Is an AP system
- Is column oriented
- Super high availability and super high throughput
    - Used by reddit, facebook and others

- Is an AP system
- Is a key value store
- Can be faster than MongoDB but sacrifices consistency

When to use SQL vs NoSQL

- By default use an SQL database
- Use NoSQL when you need more than one server AND you have a super high write rate
  - NoSQL happens when you can sacrifice CA and need some other pair from CAP
- PostgreSQL is bulletproof, use it by default
  - It is super fast
  - It is super reliable
  - It can scale amazingly
  - It has been extremely battle tested