MongoDB

{name: "mongo", type: "db"}

Database paradigms

- Relational (RDBMS)
- NoSQL
  - Key-value stores
  - Document databases
  - Wide column stores (BigTable and clones)
  - Graph database**S**
- Other

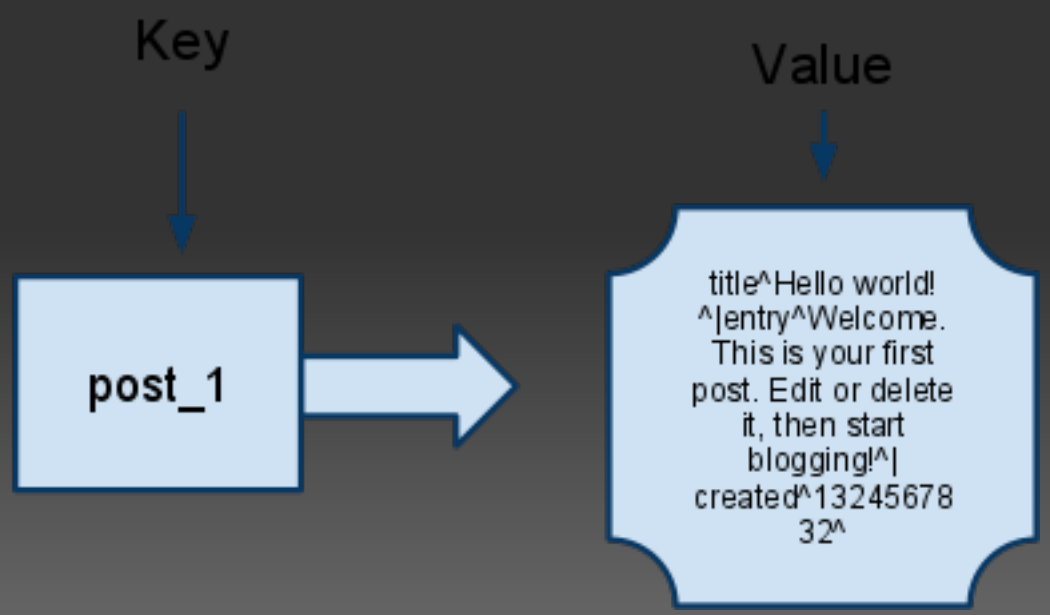
Relational Databases

- ACID (Atomicity Consistency Isolation and Durability)
- SQL
- MySQL, PostreSQL, Oracle, ...

id	title	entry	created
١	Hello world!	Welcome. This is your first post. Edit or delete it, then start blogging!	١٣٢٤٥٦٧٨٣٢

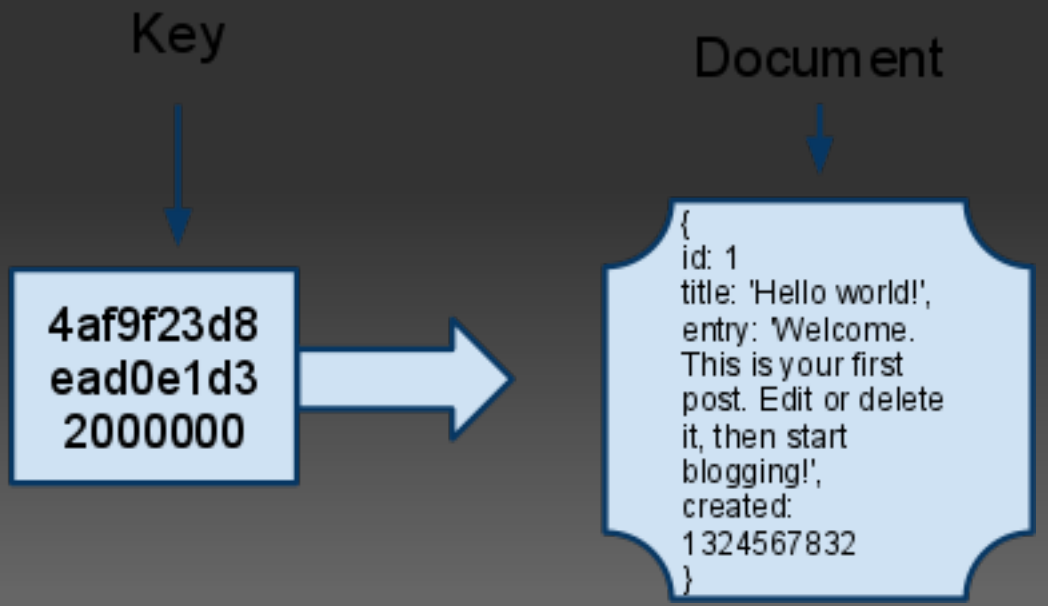
Key-value stores

- "One Key, one value, no duplicates, and crazy fast."
- It is a Hash!
- The value is binary object aka."blob" - the DB does not understand it and does not want to understand it.
- Amazon Dynamo, MemcahceDB, ...



Document databases

Key-value store, but the value is (usually) structured and "understood" by the DB.
Querying data is possible (by other means than just a key).
Amazon SimpleDB, CouhDB, MongoDB, Riak, ...



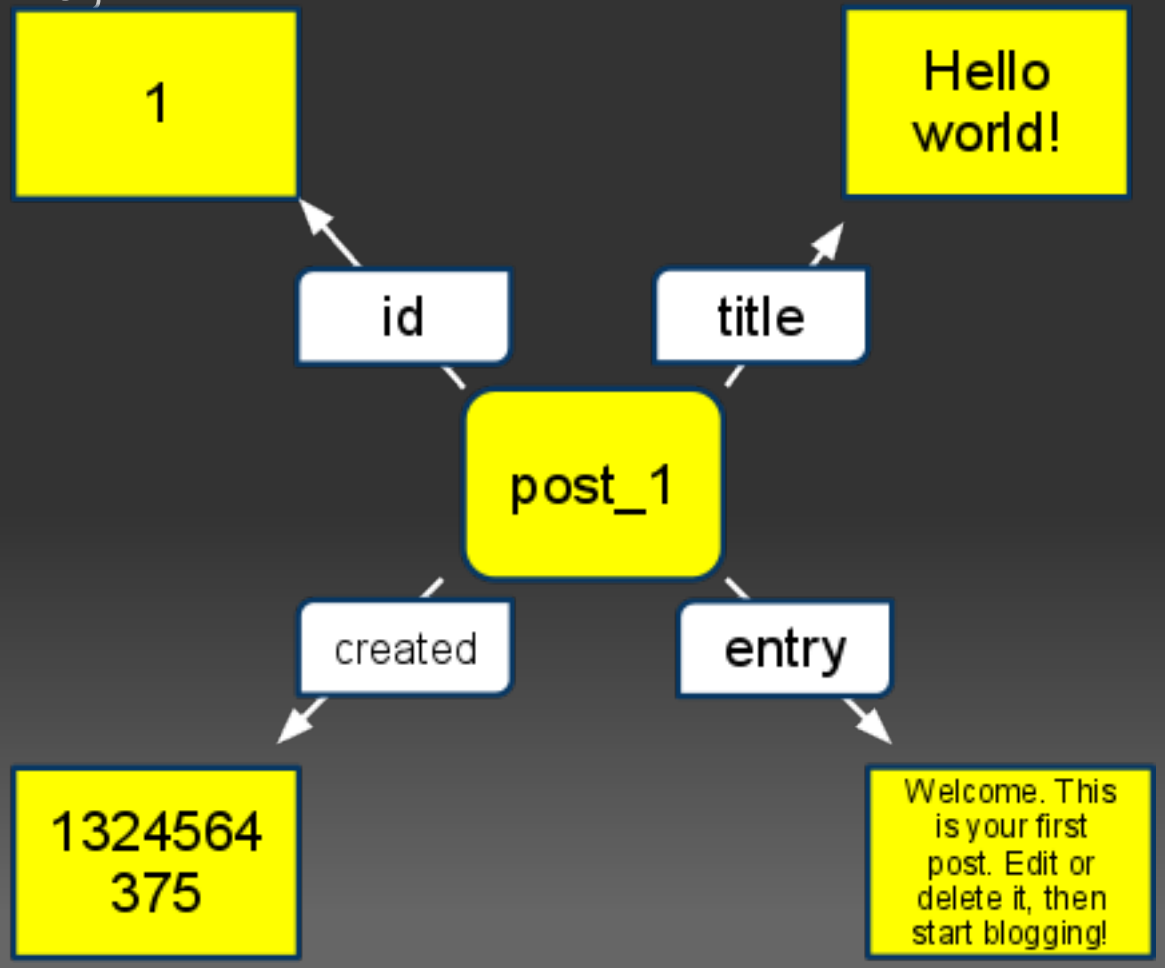
Wide column stores

Often referred as "BigTable clones"
"a sparse, distributed multi-dimensional sorted map"
Google BigTable, Cassandra (Facebook), HBase, ...

row-id	column family	title	time	value
post_١	١	title	١۴	Hello world!
post_١	١	entry	١٢	Welcome. This is your first post. Edit or delete it, then start blogging!
post_١	١	created	١۶	١٣٢۴۵۶٧٨٣٢

Graph databases

- "Relation database is a collection loosely conected tables" whereas "Graph database is a multi-relational graph".
- Neo4j, InfoGrid, ...



Relational databases have almost limitless indexing, and a very strong language for dynamic, cross-table, queries (SQL)
That's why ther handle all kinds of relationships well and dynamically.
NoSQL databases
might have limited support for dynamic queris and indexing
don't support JOIN like operations of SQL
but you can store sime relationships into document itself

Why NoSQL?

- Schema-free
- Massive data stores
- Scalability
- Some services simpler to implement than using RDBMS
- Great fit for many "web ٢.٠" services

Why NOT NoSQL?

DRBMS databases and tool are mature
NoSQL implementations often "alpha"
Data consistency, transactions
"Don't scale until you need it"

RDBMS vs, NoSQL

Strong consistency vs, Eventual consistency
Big dataset vs. HUGE datasets
Scaling is possible vs. Scaling is easy
SQL vs. Map-Reduce
Good availability vs. Very high availability

Questions?