BRILLIX DBACES
delivering brilliant DBAs covering your database

NO SQL

* NoSQL Databases *
Which and when





ILTAM USER'S ASSOCIATION

Hello World

- Alon Spiegel
- Mamram grad.
- Programmer since 1995
- DBA since 1997
- Co founder and CEO since 2007 – Brillix
- VP products since 2014 – Brillix/DBAces

alon@brillix.co.il

Let's talk about

- What is NoSQL
- NoSQL Architecture - Distributed database architecture principals
- NoSQL Flavors – exploring the leaders
- On-premise vs managed service
- Community vs Enterprise
- Adoption plan and choosing the right one

Thinking out of the box

- 1998 Carlo Strozzi RDBMS without SQL API.
- 2009 NoSQL movement emerges
- Storzzi refers to it as NoREL
- Eric Evans (Rackspace employee) reintroduces the term in an event that discusses open-source distributed dbs





So, what is NoSQL

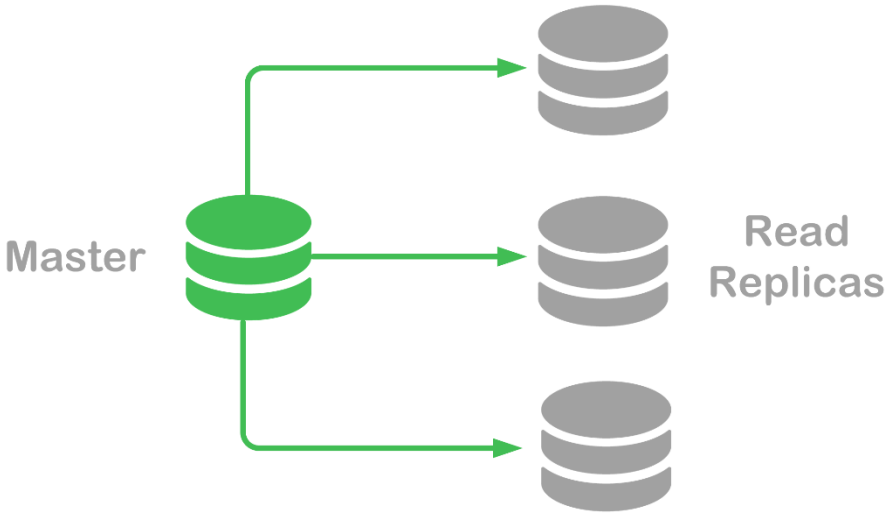
- Not Only SQL
- Modern web-scale database
- Key principals



- Often more characteristics

Non relational	Distributed
Open-source	Horizontally scalable
schema-free	
easy replication support	
simple API (CRUD+)	
eventually consistent (not ACID)	
huge amount of data	

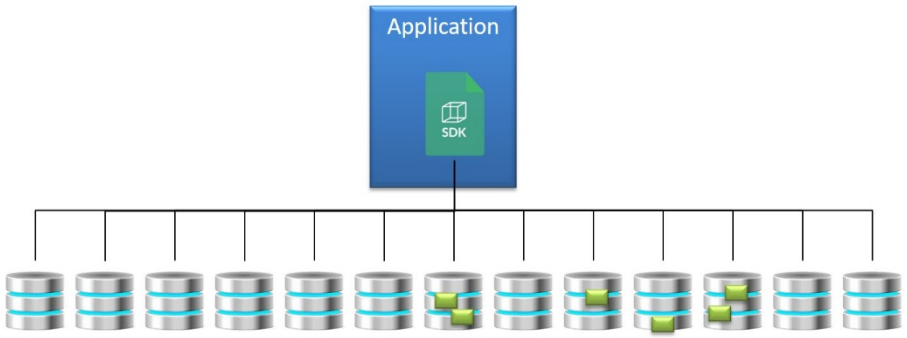
http://nosql-database.org/



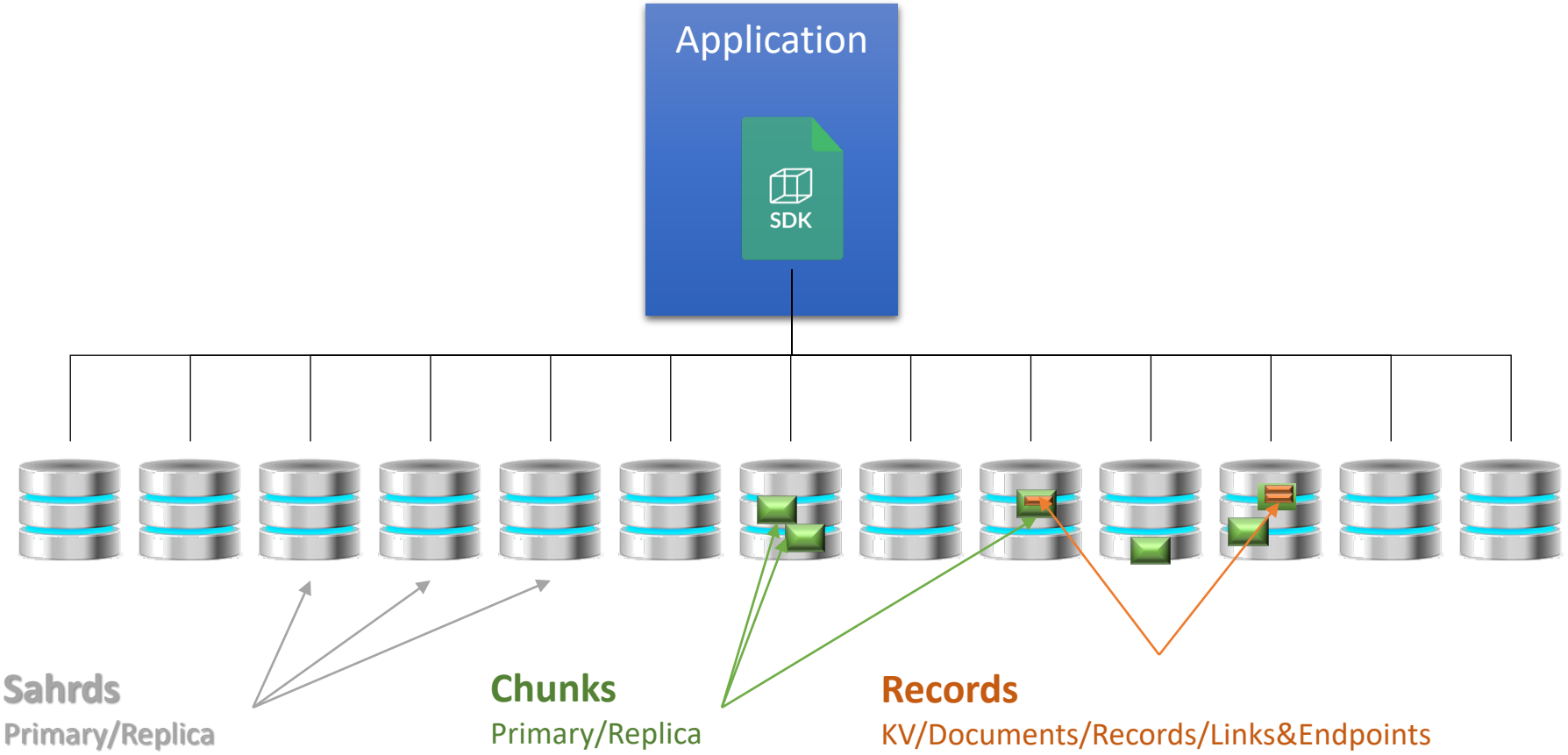
ARCHITECTURE

General layout

- “Records” are distributed among chunks in shards
- Shards are usually installed one per host
- Shards usually host a mix of primary chunks and replica chunks
- Some implementations separate primary from replica chunks by the shard

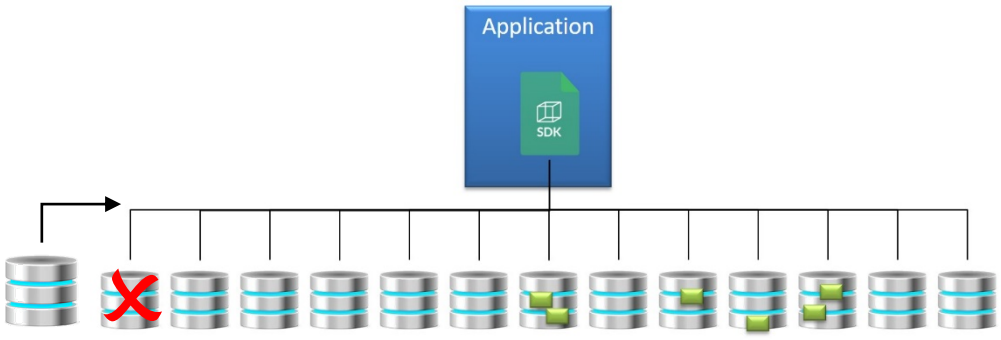


General layout



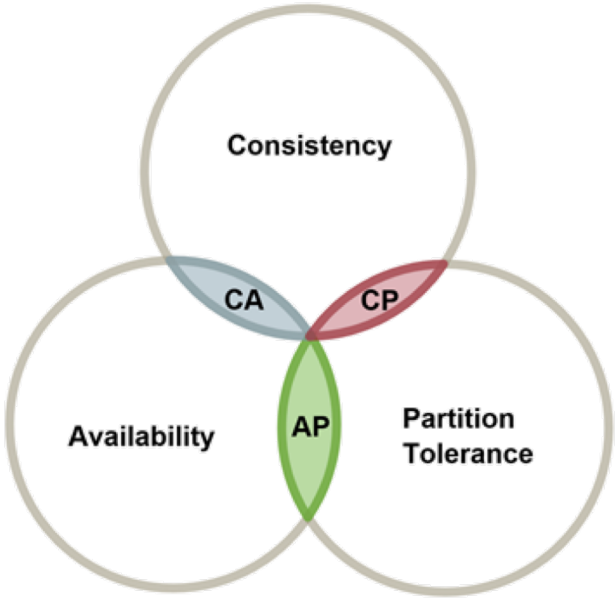
General layout – scaling (out)

- The process of adding or removing a shard from the cluster
- The process is complete after a rebalance – redistribute (long) operation



Brewer's (CAP) Theorem

- Consistency - no logical data corruption
- Availability - service always available
- Partition tolerance - protect against data discrepancies when network islands form



CAP Theorem – in practice

- In real life, the network will eventually fail:
  - Inconsistent (AP): Return what every data node holds (Outdated or not yet propagated)
  - Unavailable (CP): Don't return failing node's data
- MongoDB is AP allows to play with two settings: Write concern and Read reference
- Couchbase is CP; automatic failover (AP)


SUMMARY

… So why NoSQL

- Workload diversity – flexible design and data modeling
- Scalability – resource elasticity
- Performance – Avoid monolithically limitations
- Durability and continuous availability – no single point of failure – no matter what

… So why NoSQL

- Manageability – Minimum operational complexity and specialized personnel
- Cost – Cluster of commodity servers with cheap/free licensing
- Strong Community – An important factor to choosing a NoSQL platform

RDBMS vs NoSQL comparison

Types	One type (SQL Database) with minor variations	Many different types including key-value stores, document database, wide column stores and graph databases
Development history	Developed in 1970s to deal with first wave of data storage applications	Developed in the 2000s to deal with limitations of SQL databases, particularly concerning scale, replication and unstructured data storage
Schemas	Structure and data types are fixed in advance	Typically dynamic. Records can add new information on the fly unlike SQL tables
Scaling	Vertically	Horizontally
Data manipulation	SQL – mostly ANSI and very similar among all the vendors	Through object-oriented API
Examples	Oracle, MSSQL, MySQL, PostgreSQL	MongoDB, Cassandra, Aerospike, Couchbase, DynamoDB



EXPLORING NOSQL FLAVORS

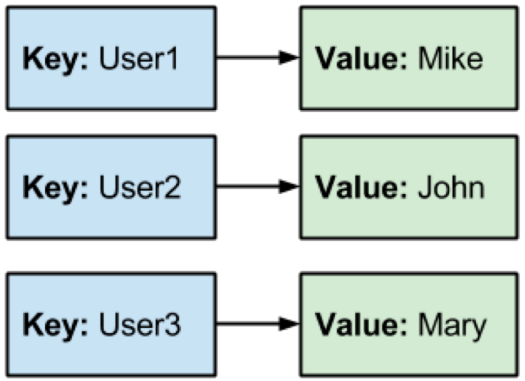
4 flavors of NoSQL databases

- Key-value store (KVS)
- Document database
- Wide Column Store - Column Families
- Graph database

NoSQL Database Models	
Volatile Storage	Memcached (an in-memory key/value store
Persistence Storage	
Key/Value Databases	Voldemort, Redis, Scalaris
Columnar Databases	HBase, Hypertable, Cassandra
Document Databases	MongoDB, CouchDB
Graph Databases	InfoGrid, Neo4j

Key-value store

- Designed for scaling to huge amounts of data handling massive loads
- A hash table of unique keys that point to a data store. A very easy model to implement.
- Only keys are indexed
- No complex data model



Key-value store - Aerospike

- Hybrid memory architecture
- Key – always in memory
- Value – always on disk
- UDF – User defined functions with
- Optimized for SSD storage
- Strong consistency
- Fast as a cache, reliable as a database





Aerospike – Popular uses

- AdTech
- Travel caching
- Online gaming











Key-value cache - Redis



- Key-value cache and store
- Referred to as data structure server: Strings, hashes, lists, sets, sorted sets, bitmaps and hyperloglogs
- Interval checkpoint to disk or log appends
- Very good and fast as long as dataset is contained in machine's RAM

Redis - Popular uses

- Main feeds
- Queues
- Caching
- Follow





Instagram
Fast beautiful photo sharing





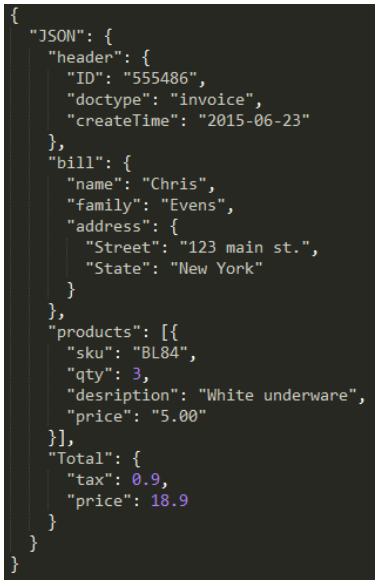


Document databases

- Support a complex, multi-value, nested data model.
- Non PK indexes – updated online or batch
- Document level operations
- Recognize special data-formats such as “spatial”
- Update partial / entire document

Document databases

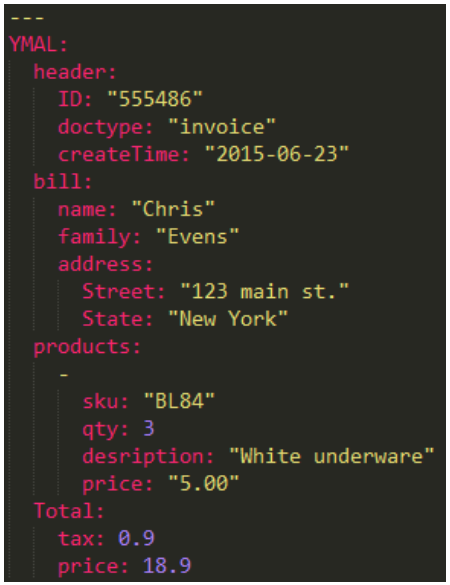
- Organized similarly to key/value store
- Value is a semi-structured document



```
{
  "JSON": {
    "header": {
      "ID": "555486",
      "doctype": "invoice",
      "createTime": "2015-06-23"
    },
    "bill": {
      "name": "Chris",
      "family": "Evens",
      "address": {
        "Street": "123 main st.",
        "State": "New York"
      }
    },
    "products": [{
      "sku": "BL84",
      "qty": 3,
      "desription": "White underware",
      "price": "5.00"
    }],
    "Total": {
      "tax": 0.9,
      "price": 18.9
    }
  }
}
```



```
<XML>
  <header>
    <ID>555486</ID>
    <doctype>invoice</doctype>
    <createTime>2015-06-23</createTime>
  </header>
  <bill>
    <name>Chris</name>
    <family>Evens</family>
    <address>
      <Street>123 main st.</Street>
      <State>New York</State>
    </address>
  </bill>
  <products>
    <p1>
      <sku>BL84</sku>
      <qty>3</qty>
      <desription>White underware</desription>
      <price>5.00</price>
    </p1>
  </products>
  <Total>
    <tax>0.9</tax>
    <price>18.9</price>
  </Total>
</XML>
```



```
---
YMAL:
  header:
    ID: "555486"
    doctype: "invoice"
    createTime: "2015-06-23"
  bill:
    name: "Chris"
    family: "Evens"
    address:
      Street: "123 main st."
      State: "New York"
  products:
    -
      sku: "BL84"
      qty: 3
      desription: "White underware"
      price: "5.00"
  Total:
    tax: 0.9
    price: 18.9
```

JSON
BSON
XML
YMAL
...

Document based - mongoDB



- Collection of BSON objects
- Secondary indexes (online)
- Fully consistent reads
- Provides embedded docs and linking (joins)
- Uses a locking mechanism
- Master-slave replication
- 10 checkpoints a second

MongoDB - Popular uses

- Weather alerts
- Travel arrangement
- Personalization
- Custom CMS for readers









Document based – Couchbase



- A hybrid of CouchDB and memcached
- Buckets of JSON documents
- Views of Map/Reduce (batch) for indexing and aggregations.
- Multi-Master replication with XDCR
- Integration with Hadoop, Elasticsearch, etc’

Couchbase - Popular users

- Product listing cache
- Session/Token store cache
- Follow
- Answers on Demand – big data analytics on consumer purchase behavior and product sales trends

Couchbase

ebay

LinkedIn

nielsen

Document database - DynamoDB



- Managed service by Amazon
- Store 3 geographical distributed replicas
- Read consistency – customizable
  - Eventually or Strong
- PK single hashed attribute (e.g. User ID)
  or composite hash-range (e.g. User ID, timestamp)

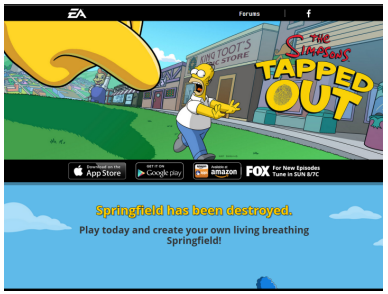
DynamoDB - Popular uses

- Recommendation platform

- “Kids, you tried your best and you have failed miserably. The lesson is, never try.”
- (Homer J. Simpson)

amazon DynamoDB

The New York Times





Wide column store

- Extensible record store OR Column family store
- Can hold a lots of columns (millions)
- Each key points to a “row”, each column is a tuple of “column name” and a value or a triplet adding a timestamp

Key	Value
Name	Preston

Wide column store

- A column family (table) is a list of values that reside together on disk (like a table in RDBMS).
- Granularity – update individual column

Row ID	Columns...		
1	**Name**	**Website**	
	Preston	www.example.com	
2	**Name**	**Website**	
	Julia	www.example.com	
3	**Name**	**Email**	**Website**
	Alice	example@example.com	www.example.com

Wide column store

- A Super column is a list of columns – It is actually one nesting level
- Super columns do not have timestamps

Row ID	Columns...	
3	Website	
	Subdomain	www
	Domain	example.com
	Protocol	http://

Wide column store - Cassandra







- Born at Facebook, now under Apache
- A hybrid of Amazon's Dynamo and Google's Bigtable
- SQL like query language: CQL
- Write to one / majority nodes (Tunable consistency)
- Use gossip protocol for intra-cluster communication
- Row-level write isolation
- Use secondary indexes for low cardinality searches

Wide column store – Popular uses

- Logging, tracking and notifications
- Fraud detection
- Social signals
- Ad serving
- Audit (fight spam and abusive)
- Churn tracking, follow requests











Wide column store – HBase

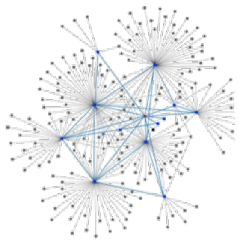


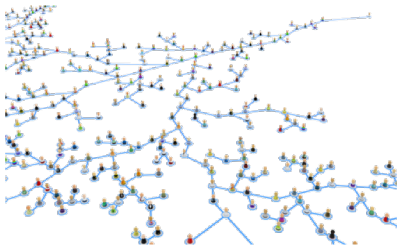


- Part of Hadoop under Apache
- Strictly consistent reads and writes
- Intra-cluster communication is handled by zookeeper
- ACID level semantics on per-row basis
- Secondary indexes alternates
  - Filter query
  - Periodic-update secondary indexes – using MapReduce
  - Dual write index
  - Summary tables – using MapReduce
  - Coprocessor secondary index – using internal triggers

Graph database

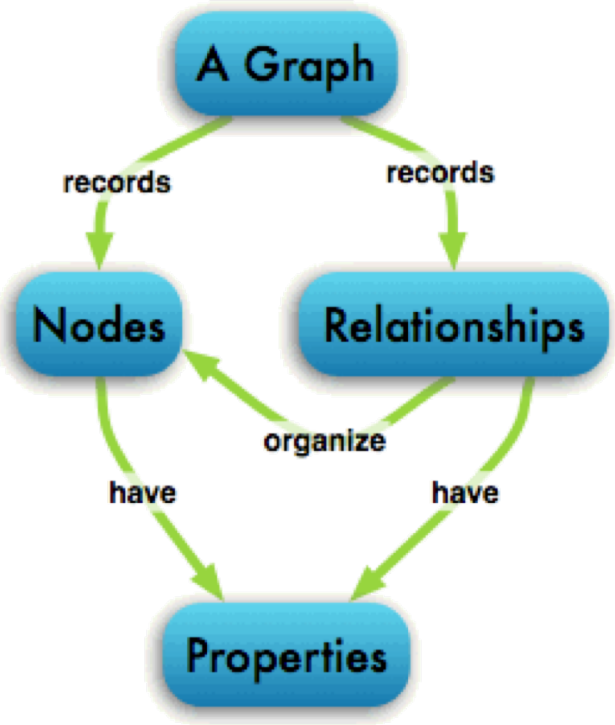
- Data is stored in a flexible graph model instead of rigid structure of tables (rows and columns)
- In the graph each node can be connected to any number of nodes such as network topologies and public transportation system





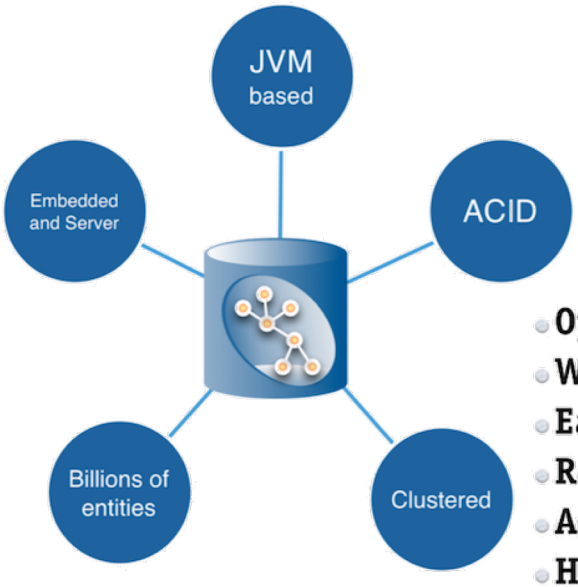
Graph database - Property graph

- Entities (nodes)
  - Attributes (KV)
  - Labels, roles, constraints, indexes, metadata
- Relationships
  - Start node & End node
  - Attributes (KV), Named, typed, Directional (BiDi navigation)



Graph database – neo4j

- First graph database
- Fully ACID compliant
- Visualization



Neo4j is a NoSQL Graph Database

- Open source
- Welcoming UI
- Easy data modeling
- Readable queries
- Active community
- High performance
- optional schema

http://neo4j.com/developer/guide-data-visualization/

Graph database - Popular uses

- Online recommendations
- Social networks
- Graph based search of content
- Spatial (Geo)

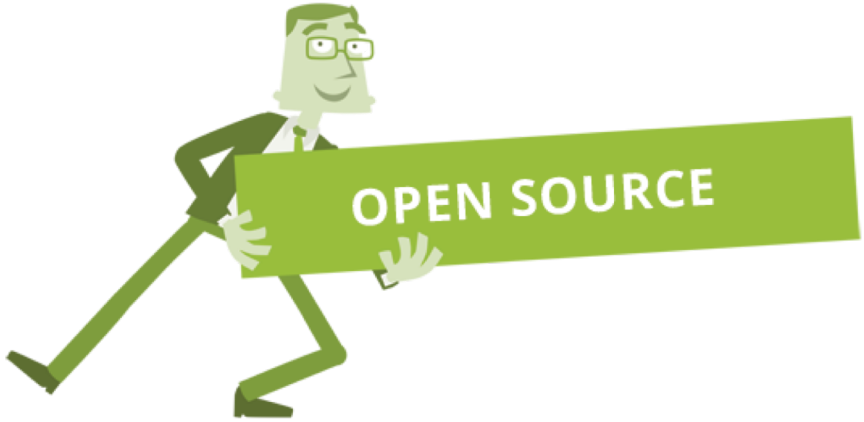
neo4j
ebay
Walmart
NATIONAL GEOGRAPHIC
Lufthansa Systems
TOMTOM



DEPLOYMENT

Comparison: On-premise vs Cloud

	On-premise	Managed service (cloud)	Cloud VMs
Ease/speed of deployment	♦	◈	♦
Ease/speed of administration	♦	◈	♦
Ease/speed of scaling	♦	◈	♦
Upfront cost	♦	◈	◈
Dynamically change cost	♦	◈	◈
Fixed resources cost	◈	♦	♦
Customization	◈	♦	◈
Log analysis	◈	♦	◈
Ops/DevOps resources	♦	◈	♦
Amount of vendors	◈	♦	◈



COMMUNITY VS ENTERPRISE

Comparison: CE vs EE

	Community edition	Enterprise edition
License cost	**Free**	**Per: Node / RAM / Core / Data volume**
Features	**Basic**	**Enhanced**
Security	**Basic**	**Enhanced**
Support	**None**	**SLA**
Scalability	**Low (limited cores / nodes / RAM support) – High**	**High**



Choosing the right one out of the bunch

ADOPTION PLAN

Adoption plan

- Organizations usually opt for one of the followings:
  1. Convert a lesser important module to NoSQL
  2. Convert a main module to NoSQL
  3. Convert a suitable module to NoSQL
  4. Convert the entire system to NoSQL
- Organizations usually begin with CE (free) and later upgrade to EE (Availability, scalability, security, SLA and professional services)

Adoption plan

- RDBMS is not dead
- Migrate only modules that would really benefit
- Choose the best NoSQL engine to your application module
- Note that programming against a NoSQL dictates different paradigms than relational
- Using several NoSQL engines for different functionalities is advised
- Beware of falling into a DevOps pit





PROGRAMMING PARADIGM SHIFT

Write preference

- Option in the source code (per op)
- Can write to memory of primary
  - Add write to secondary
- Can write to storage of primary
  - Add write to secondary
- If forcing write to secondary and secondary is missing – write may fail

Read preference

- Option in the source code (per op)
- Read from primary
- Read from secondary
  - Force
  - Preferred
  - Read from majority (to minimize probability if inconsistent reads)

Eventually consistent

- Read from primary replica is consistent
- But what happens if that replica fails?
- Persist writes to primary and secondary replica
- Only when necessary

Scalable writes

- Avoid multiple writes to the same shard
  - Sequence or timestamp as range shard key
  - Updates to the same items – hotspot

Minimize use of cluster wide ops

- Secondary indexes are usually co-located with data
- Querying them performs a cluster wide search
- Instead model your data to use cluster key whenever possible

PK and secondary indexes

- Primary keys are used for sharding
- Most databases allow secondary indexes
- Secondary indexes are collocated with data