Databázové systémy

NoSQL databázy

Martin Binder

Obsah

- Porovnanie RDBMS vs NoSQL
- Dátové modely NoSQL databáz
- Zopár príkladov použití jednotlivých databáz
- Konzistentnosť, Dostupnosť, Partitioning
- Dátové modelovanie v NoSQL databázach
- Zhrnutie

Relačné databázy - RDBMS

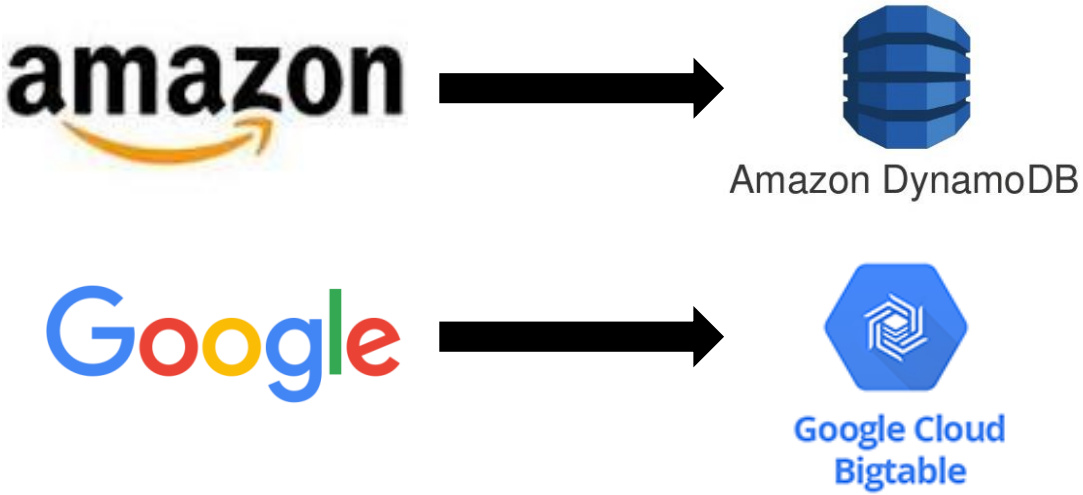
- Založené na relačnom modeli
  - Dáta organizované v tabuľkách so stĺpcami a riadkami
  - Entity v tabuľkách previazané pomocou kľúčov
- ACID transakcie, referenčná integrita, procedúry, indexy...
- Relačné operácie (relačná algebra) : SELECT, UNION, JOIN, INTERSECT, EXCEPT, eg.
- Dátové modelovanie: Normalizácia databázy

Potreba NoSQL

- Problémy SQL a jeho škálovania pri veľmi veľkých datasetoch
- Náročné nasadenie a vývoj aplikácií s využitím distribuovaných RDBMS
- Rast sociálnych sietí a ich požiadaviek na kapacitu a dostupnosť dát
- Big E-commerce
- Cloudové služby
- IoT, BigData analytika

História

- 60-70te roky – IBM IMS, M[umps], DBM
- 80te roky - Lotus Notes/Domino
- 2004 - The Dynamo Paper




mongoDB


cassandra


neo4j


redis


CouchDB
relax


HYPERTABLE


APACHE
HBASE

NoSQL buzzwords

- Non-relational
- open-source
- Cluster-friendly
- 21st century web driven
- Schema-less
- BASE, CAP
- Horizontal scaling
- Big data, IoT

Dátové modely NoSQL databáz

- Nevyžaduje schému
- Žiadne null hodnoty
- Žiadne implicitné dátové typy
- Dátový model závislý od aplikácie
- NoSQL databázy rozdeľujeme podľa 4 hlavných dátových modelov:
  - Key-Value
  - Document
  - Column Family (Wide Column Store)
  - Graph

Key-Value Datamodel

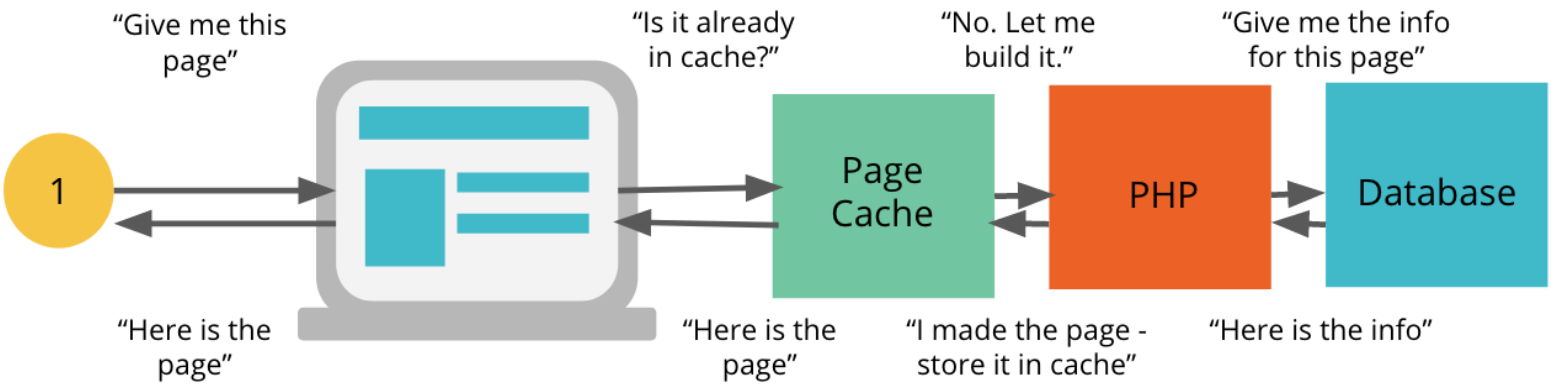
- Najjednoduchšie NoSQL db
- Hash table
- Optimalizácia na rýchlosť
- Index iba primárny kľúč
- Príklady:
  - Redis
  - Memcached
  - Riak

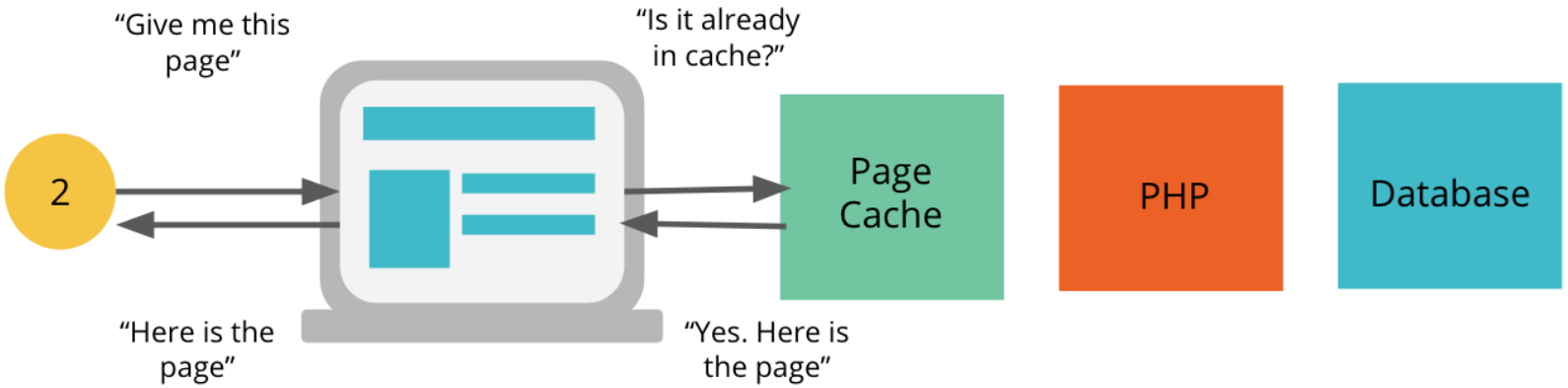
Key	Value
key1	12
key2	“Fero Nagy”
state12314	{ "id": "1", "state_slug": “goaler", "statename": “Goaler" }
op1	421412412412

Key-Value Datamodel

- In-memory databáza s perzistenciou na disku
- Iba základné operácie (SET, GET...)
- Časté prípady použitia:
  - Cache
  - Session Caching
  - Full Page Cache
  - Message Queue
  - Logging
- Pre zabezpečenie konzistencie môžeme využiť žurnálovanie a snapshoty

Full Page Cache



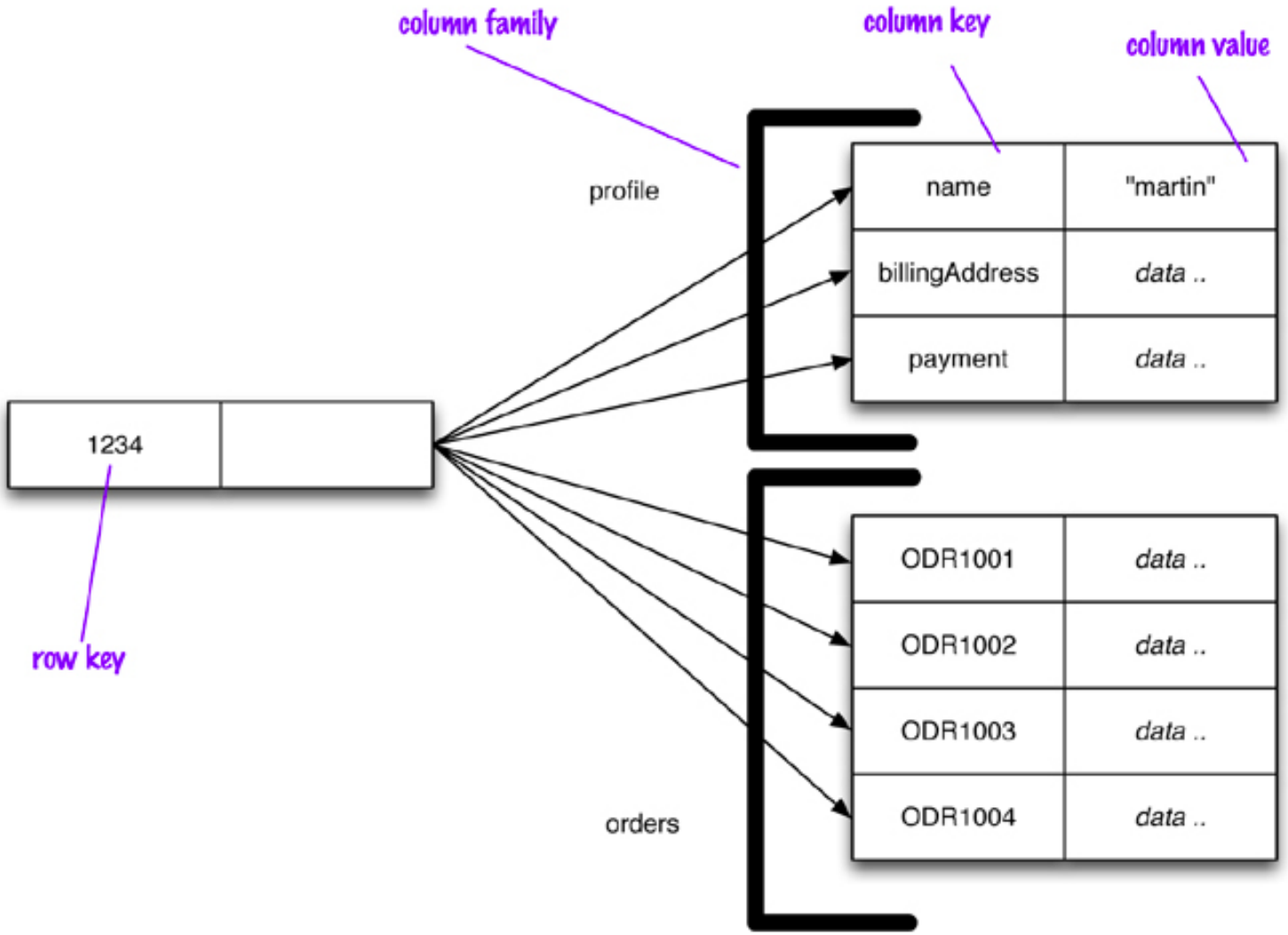


https://wpengine.com/support/cache/

Column Family Datamodel

- Dáta sú uložené podľa jedinečného kľúča
- Každý riadok môže mať rozdielny počet stĺpcov
- Stĺpce sú zaraďované do Column Family
  - Skupiny stĺpcov, ktoré logicky súvisia
- Optimalizované pre prácu s veľkými datasetmi
- Príklady:
  - Cassandra
  - Hbase
  - BigTable

Column Family Datamodel



Zdroj: https://sungsoo.github.io/2014/05/19/column-family-stores.html

Document Datamodel

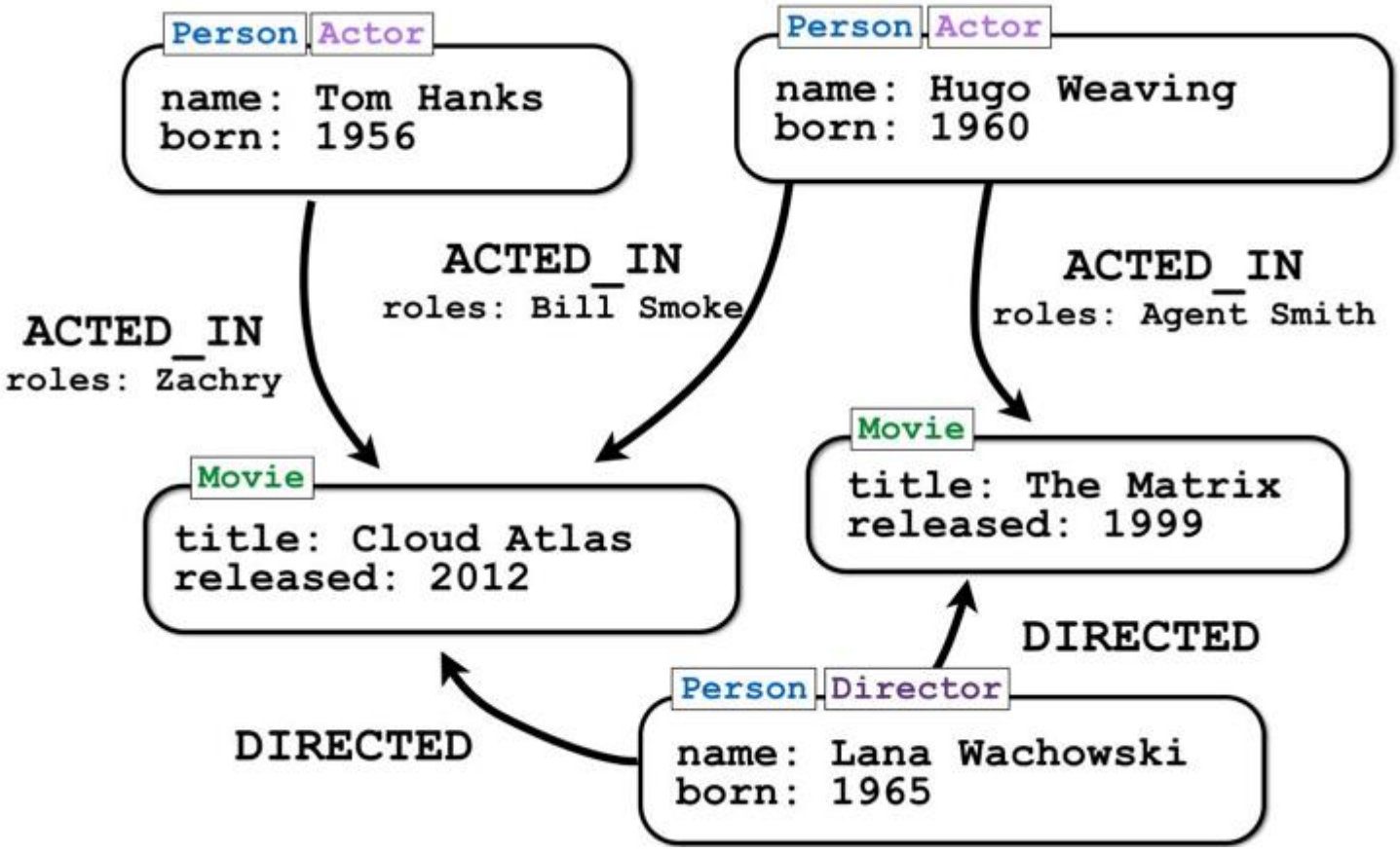
- Databáze je úložiskom čiastočne štruktúrovaných dát – dokumentov
  - Dokumenty nemaju pevnú schému
- Najčastejšie JSON, XML
- Rozšírenie konceptu Key-Value Store – funkčnosť databázy je čiastočne závislá od vnútornej štruktúry dokumentov
  - Podpora indexovania podľa metadát
- Pri moderných implementáciách sa hranica medzi Key-Value Store a Document DB zužuje

```
{
  "ID":3,
  "Order Date": "2012-04-23T18:25:43.511Z",
  "Customer":
  {"Fero",
    "FirstName":
    "LastName": "Nagy",
    "Telephone": "+421903123123"
  }
  "OrderLineItems":
  [
    {
      "ID": 1,
      "Product": 3,
      "Quantity": 200,
      "Price": 1.30
    },
    {
      "ID": 2,
      "Product": 4,
      "Quantity": 130,
      "Price": 3.45
    }
  ],
  "Payment Details":
  {
    "Card Type":"Visa",
    "Card Number": "32131766874",
    "Expiration Date": "03/22"
  }
}
```

Graph Datamodel

- Modelujú dáta na základe teórie grafov ako
  - Vrcholy – samotné modelované objekty
  - Hrany
    - Hrany môžu mať smer a názov
    - Na hrany je možné ukladať metadáta
    - Modelujú vzťahy medzi vrcholm
- Optimalizácia na operácie využívajúce prechádzanie grafmi
- Príklady: Neo4J

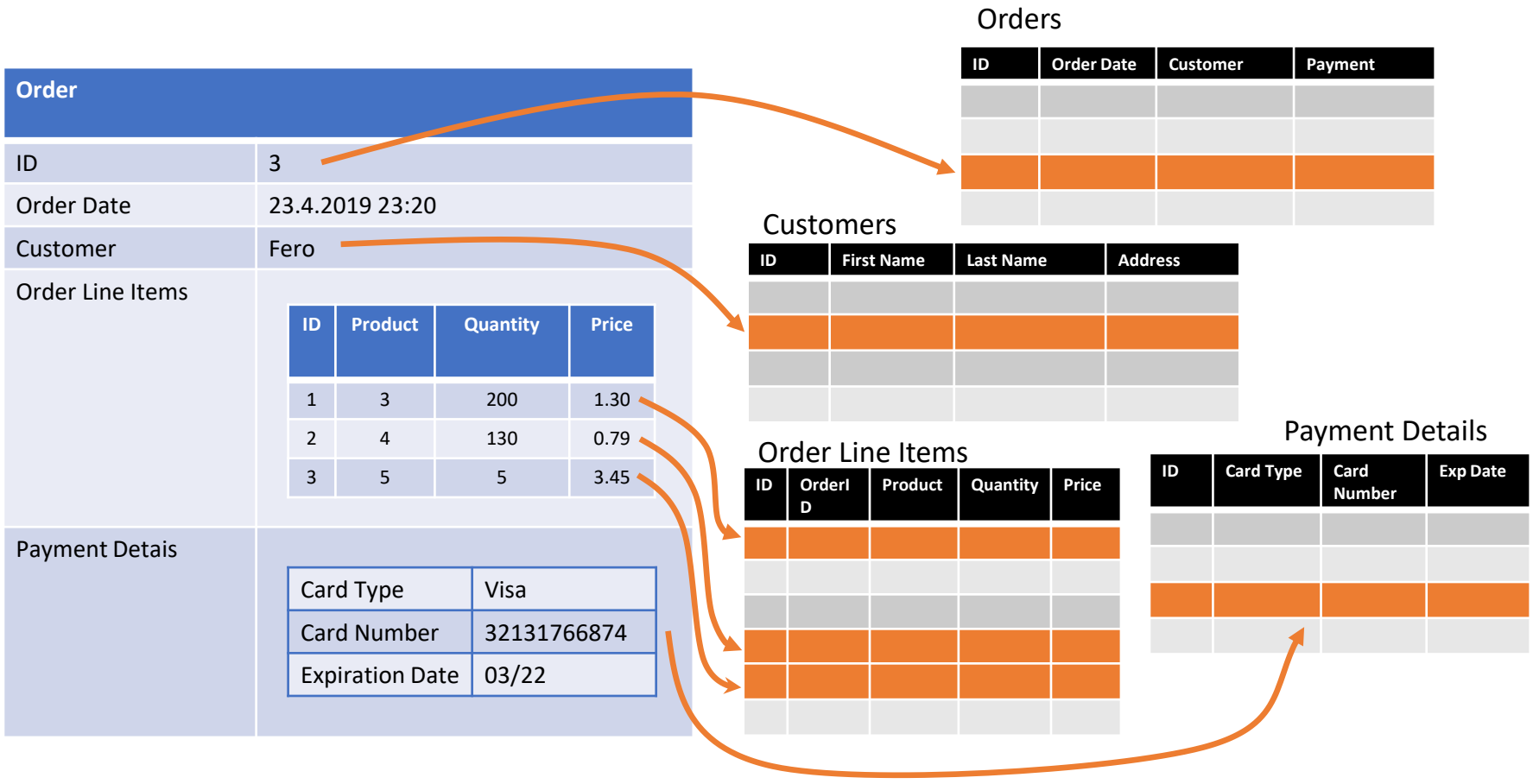
Graph Datamodel



Modelovanie dát

- Proces identifikovania logických entít (objektov) a ich vzťahov
- V RDBMS
  - normalizované tabuľky previazané cez primárne a cudzie kľúče
  - Dopyty sú založené na štruktúre tabuliek a vzťahy sú dopytované pomocou JOIN-ov
- V NoSQL
  - Modelovanie dát je Query-driven
  - Množina dopytov aplikácií priamo určuje štruktúru a organizáciu dát
  - 1 dopyt = 1 tabuľka
  - JOIN nie je väčšinou podporovaný

Modelovanie dát v RDBMS – normalizácia



Modelovanie dát v NoSQL – Query Driven

Order	
ID	3
Order Date	23.4.2019 23:20
Customer	Fero
Order Line Items	

ID	Product	Quantity	Price
1	3	200	1.30
2	4	130	0.79
3	5	5	3.45

Payment Detais	

Card Type	Visa
Card Number	32131766874
Expiration Date	03/22

Dopyt: All Orders by Customer

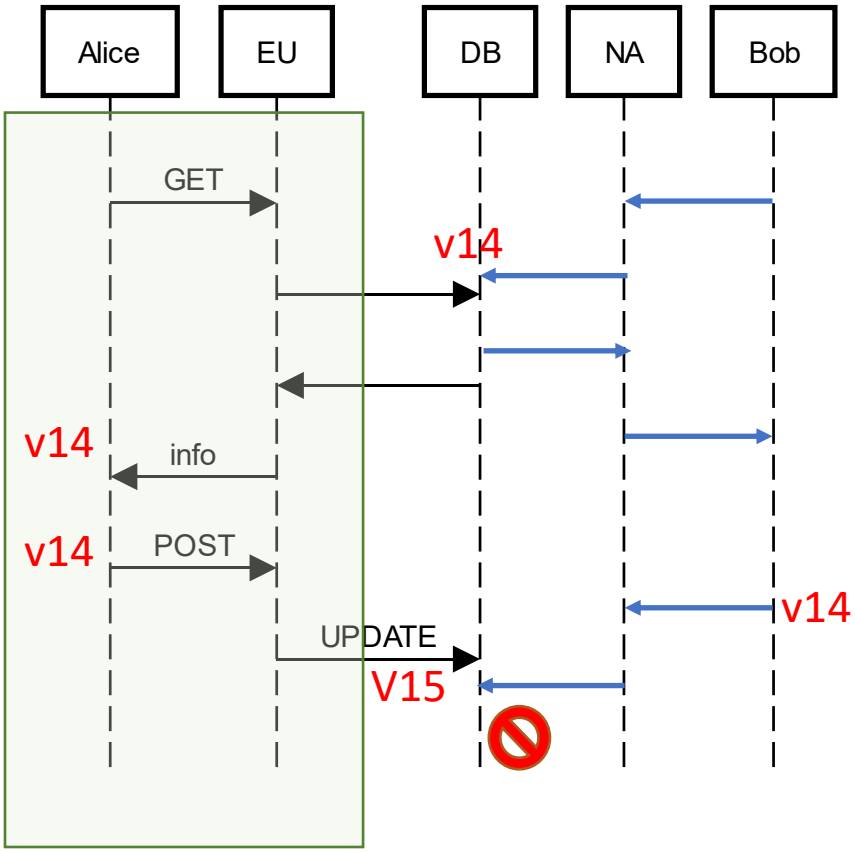
```
CREATE KEYSPACE order WITH replication =
{‘class’:
  ‘SimpleStrategy’, ‘replication_factor’ :
3};

CREATE TABLE order.orders_by_customer (
  customer_id uuid,
  order_id uuid,
  customer_firstname text,
  customer_lastname text,
  payment_cardtype text,
  payment_cardnumber text,
  payment_expirationdate text,
  order_lines list<order_line>,
  PRIMARY KEY ((customer_id), order_id) )
```

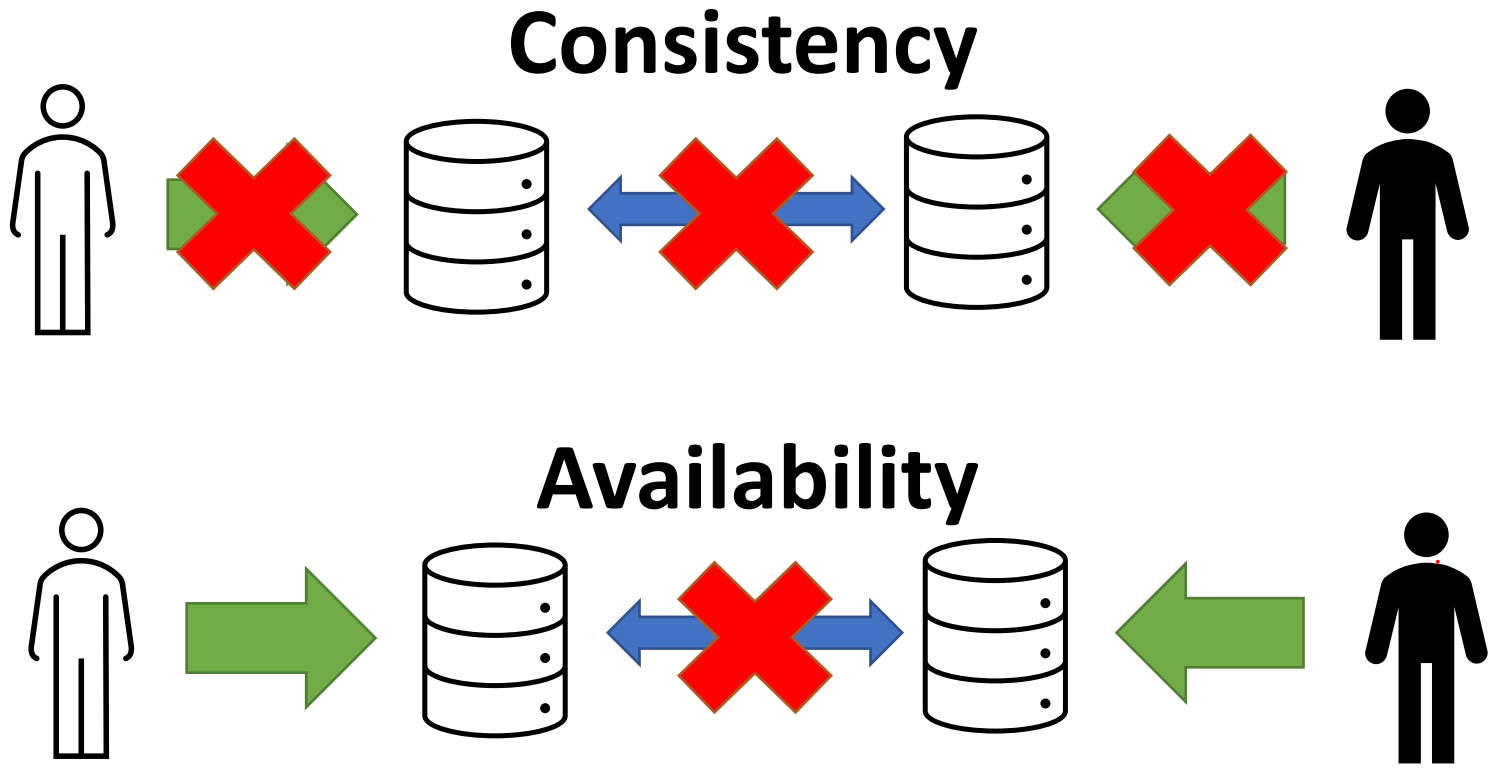
Modelovanie dát v NoSQL

- Zameranie na škálovateľnosť a rýchlosť
  - Preferovanie sekvenčného zápisu a čítania
- Prístup k modelovaniu závisí od aplikácie, jednotlivých dopytov a od využitej technológie

Konzistentnosť v NoSQL

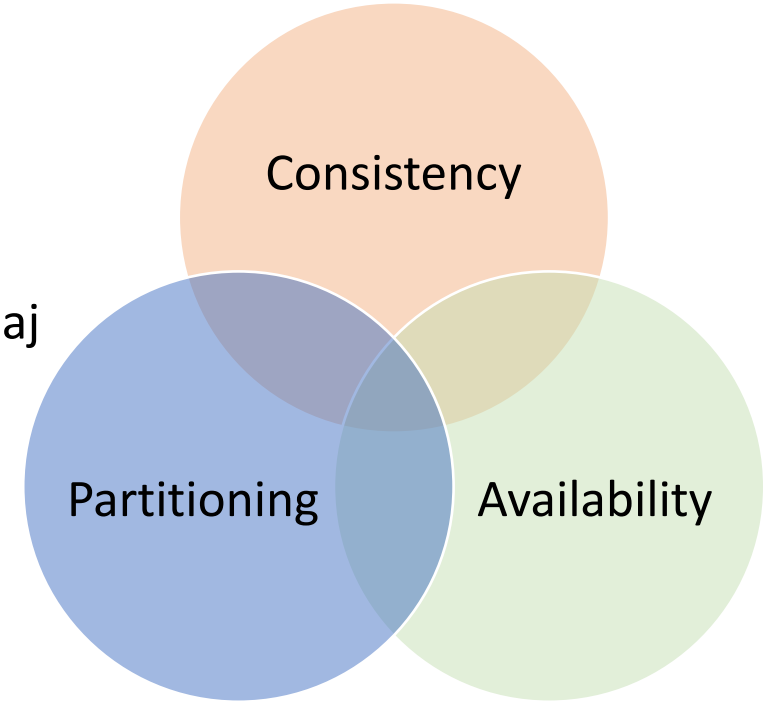


Konzistentnosť v NoSQL – distribuovaná db



Konzistentnosť v NoSQL

- CAP theorem
  - Consistency
    - všetci klienti vidia zhodné dáta
  - Availability
    - Dostupnosť databázy pri výpadkoch uzlov
  - Partition Tolerance
    - Zabezpečenie funkcie clustra aj pri výpadku prepojenia medzi uzlami



Konzistentnosť v NoSQL- PACELC

- Eventual Consistency
- Relaxed Durability
- Qorums
- Read-Your-Writes
- ...

DDBS	P+A	P+C	E+L	E+C
DynamoDB	✓		✓	
Cassandra	✓		✓	
Cosmos DB	✓		✓	
Couchbase		✓	✓	✓
Riak	✓		✓	
VoltDB/H-Store		✓		✓
Megastore		✓		✓
BigTable/HBase		✓		✓
MySQL Cluster		✓		✓
MongoDB	✓			✓
PNUTS		✓	✓	
Hazelcast IMDG[6][5]	✓	✓	✓	✓
FaunaDB[7]		✓	✓	✓

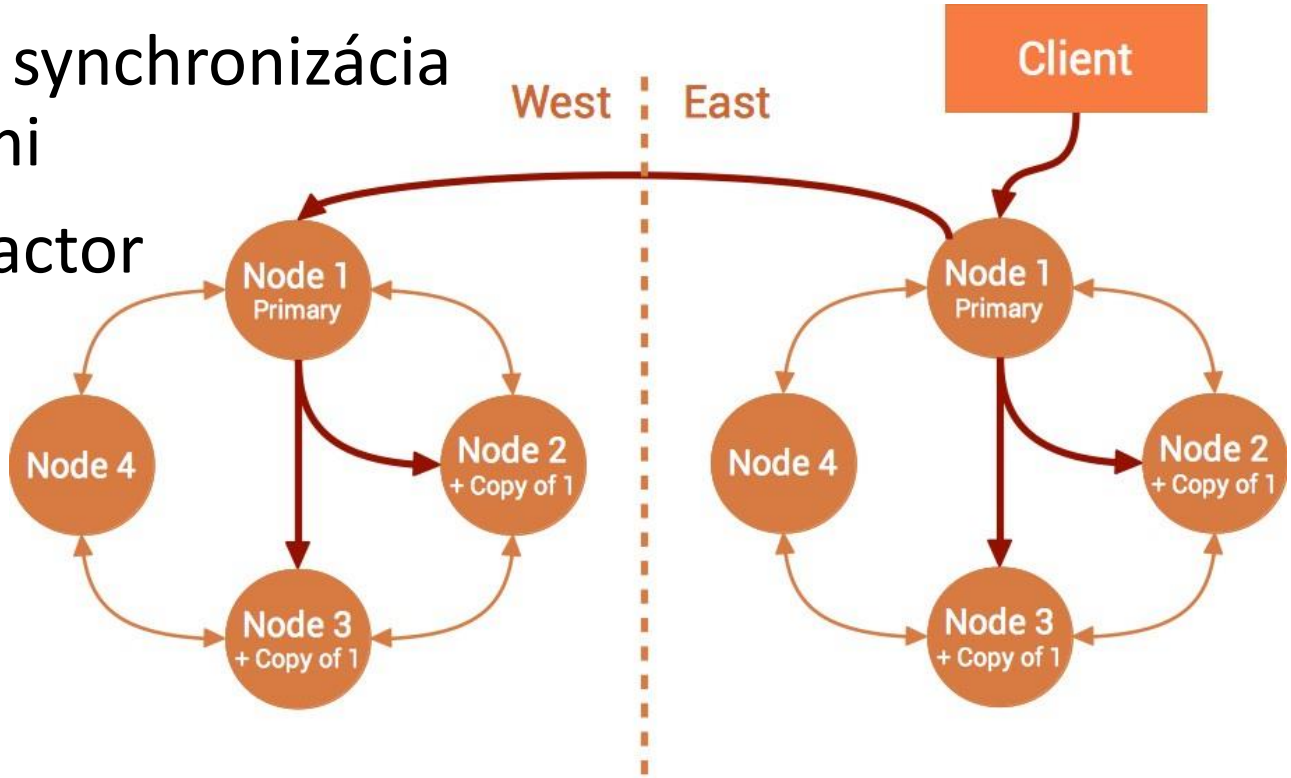
Zdroj: Wikipedia

Apache Cassandra

- Column Family databáza
- Facebook -> Open Source -> Apache
- **Replikácia, vysoká dostupnosť, jednoduché škálovanie**
- Vstavaný automatický clustering a redundancia
- Implementácia vychádza z Dynamo Paper a dátová štruktúra je podobna BigTable
- Rýchlosť dosahuje využívaním hlavne sekvenčného zápisu na disk

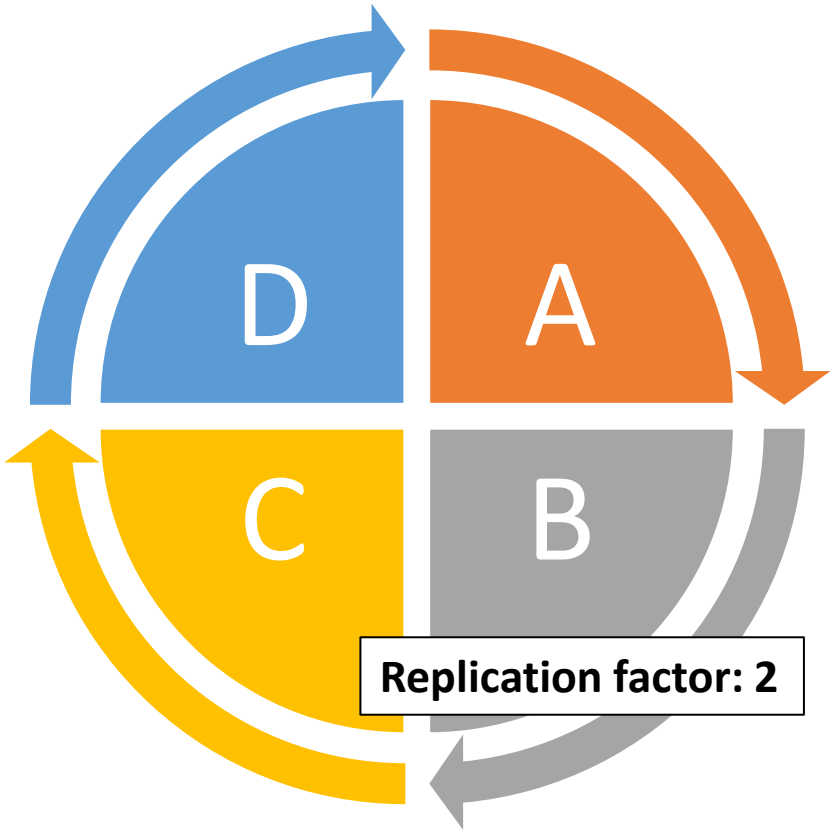
Replikácia

- Žiadny master uzol
- Klent zapisuje na lokálny uzol
- Asynchrónna synchronizácia naprieč uzlami
- Replication Factor



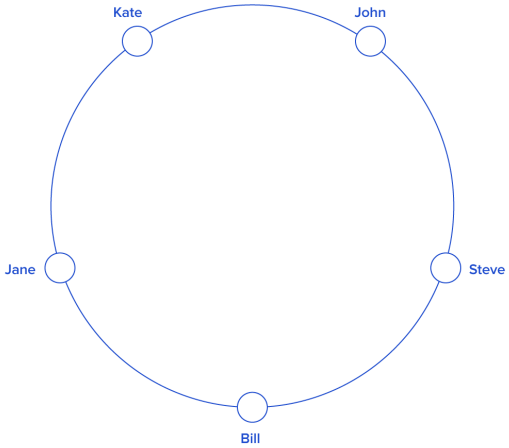
Partitioning

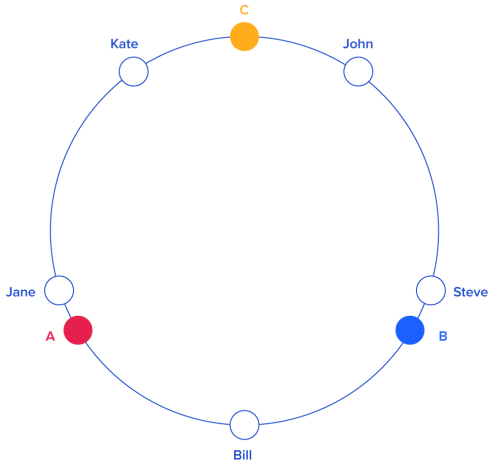
Partition Key	MD5 Hash
jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

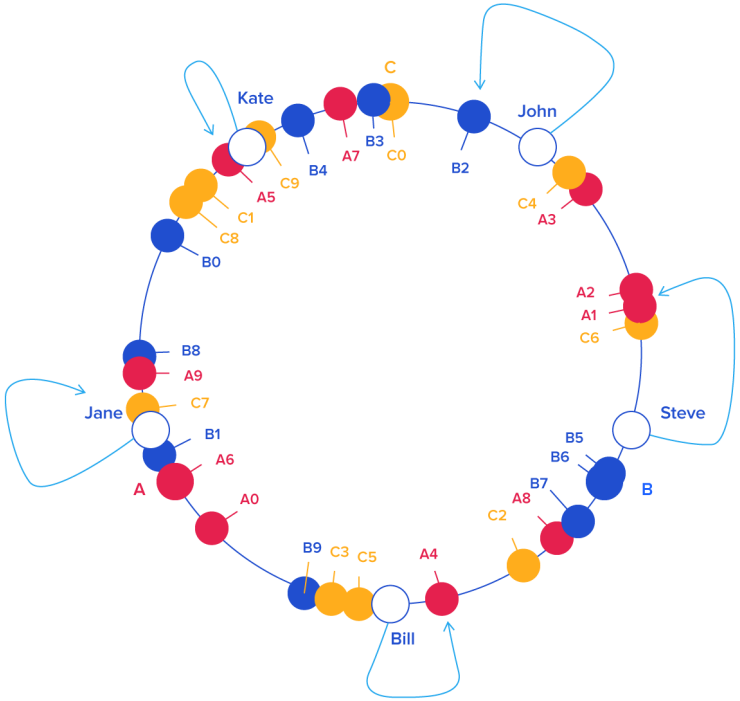


A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

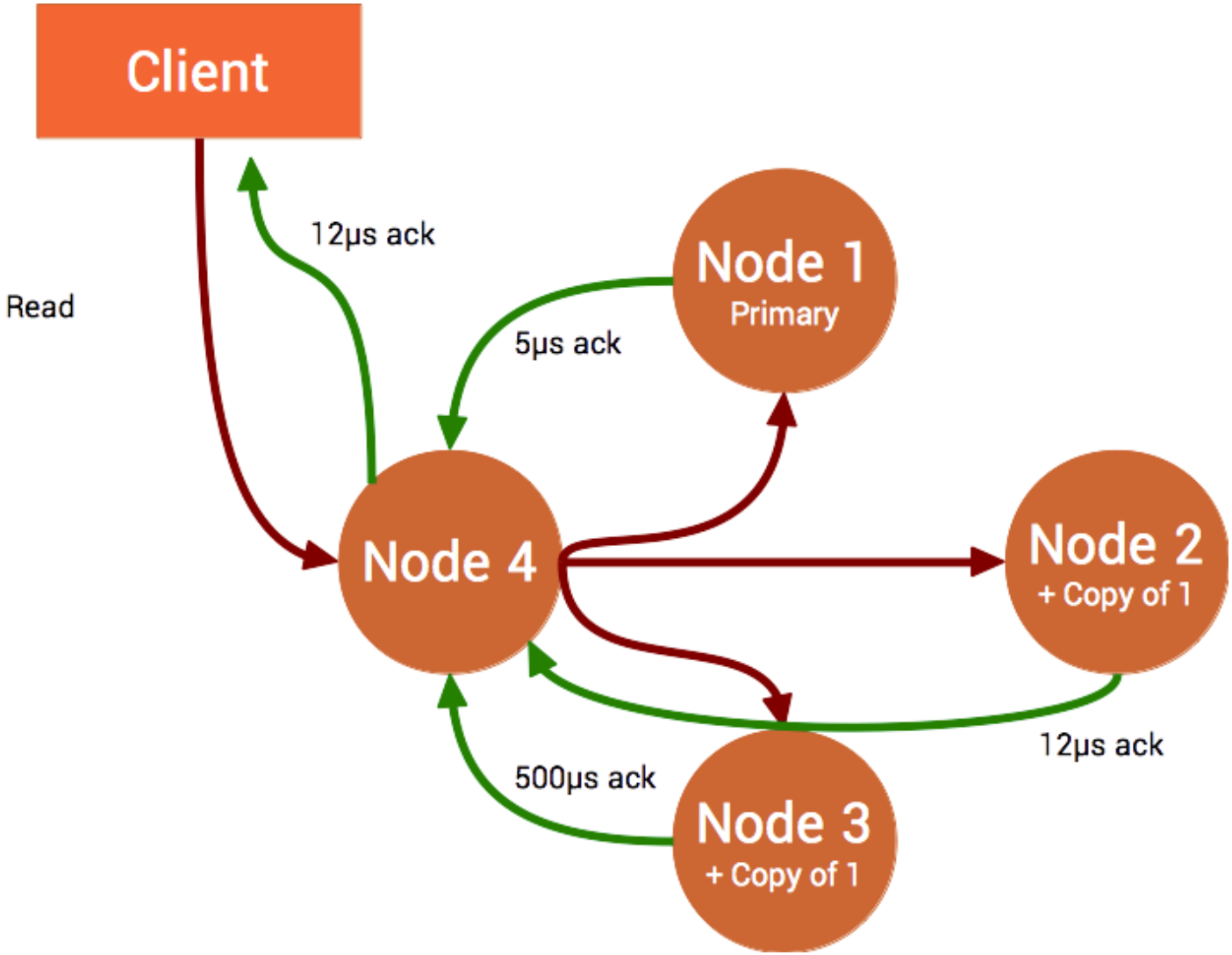
Consistent Hashing





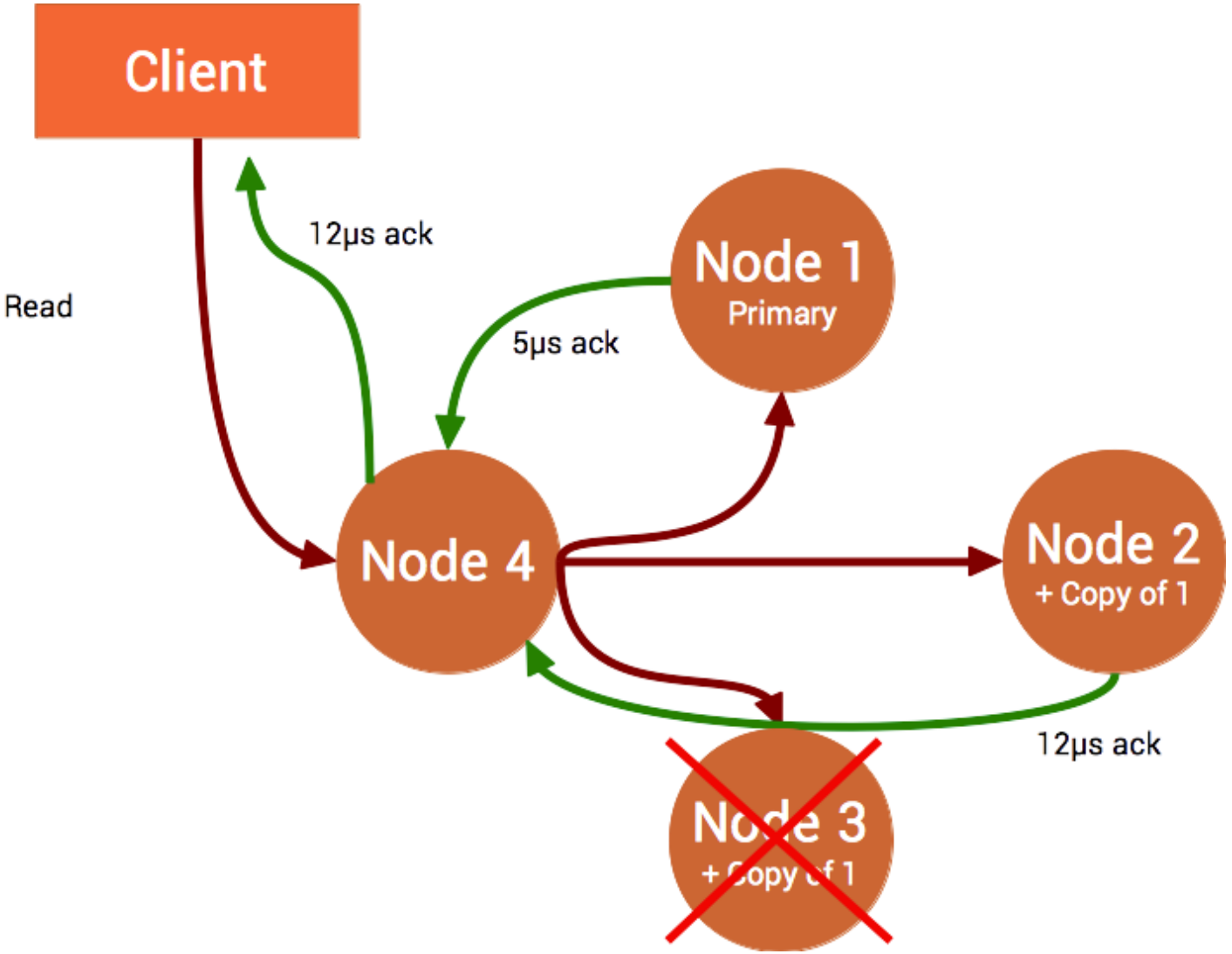


Coordinated reads



Zdroj: DataStax

QUORUM and availability

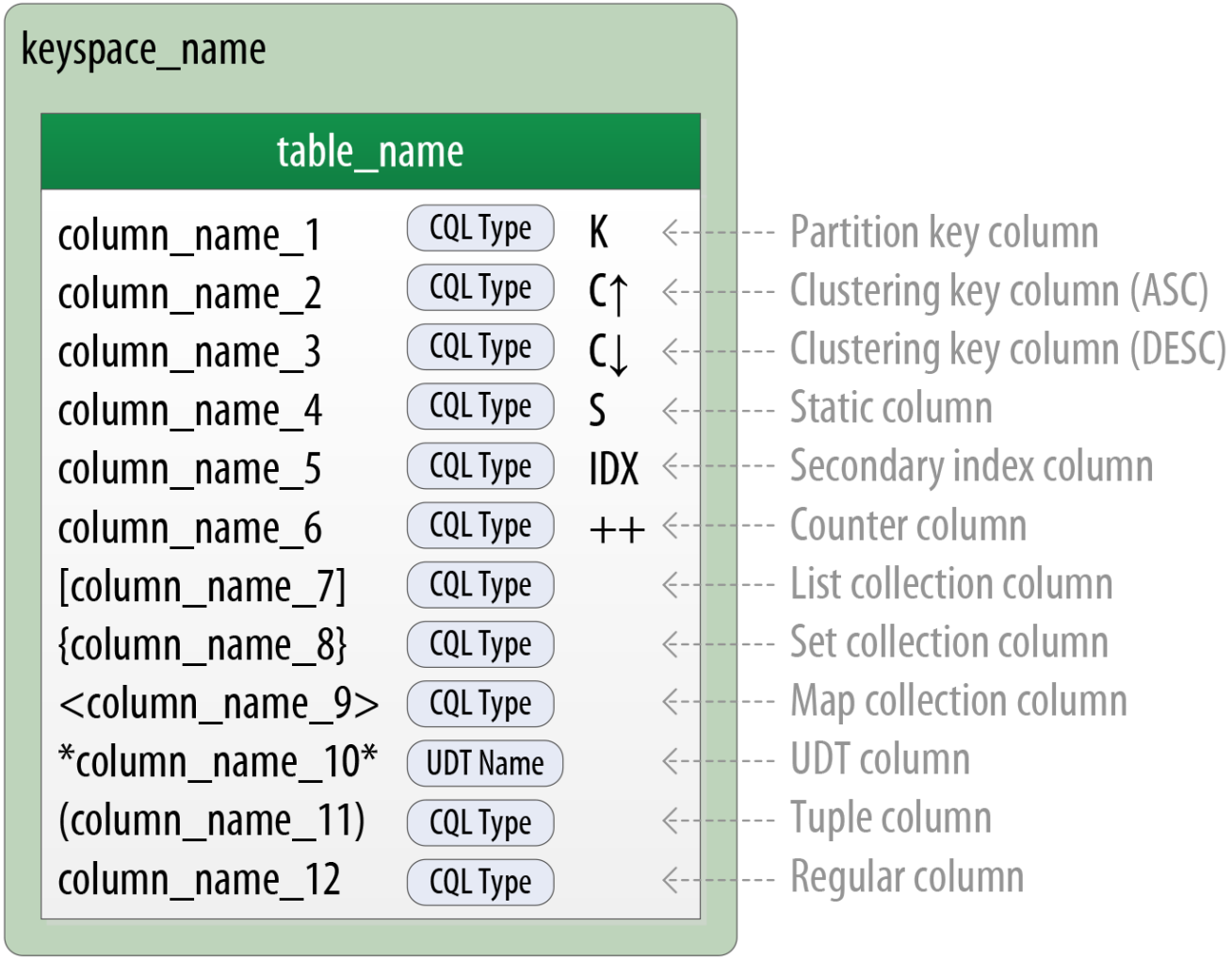


Zdroj: DataStax

Consistency Level

- Set with every read and write
- ONE
- QUORUM - >51% replicas ack
- LOCAL_QUORUM - >51% replicas ack in local DC
- LOCAL_ONE - Read repair only in local DC
- TWO
- ALL - All replicas ack. Full consistency

Dátový model



Príklad dátového modelu

```
CREATE KEYSPACE hotel WITH replication =
  {‘class’: ‘SimpleStrategy’, ‘replication_factor’ : 3};

CREATE TYPE hotel.address (
  street text,
  city text,
  state_or_province text,
  postal_code text,
  country text );

CREATE TABLE hotel.hotels (
  id text PRIMARY KEY,
  name text,
  phone text,
  address frozen<address>,
  pois set )
```

Príklad dátového modelu II

```
CREATE TABLE hotel.pois_by_hotel (
  poi_name text,
  hotel_id text,
  description text,
  PRIMARY KEY ((hotel_id), poi_name) )
  WITH comment = Q3. Find pois near a hotel’;

CREATE TABLE hotel.available_rooms_by_hotel_date (
  hotel_id text,
  date date,
  room_number smallint,
  is_available boolean,
  PRIMARY KEY ((hotel_id), date, room_number) )
  WITH CLUSTERING ORDER BY (room_number DESC);
```

Príklad dátového modelu III

```
CREATE KEYSPACE reservation
WITH replication = {‘class’:
  ‘SimpleStrategy’,
‘replication_factor’ : 3};

CREATE TYPE reservation.address
(
  street text,
  city text,
  state_or_province text,
  postal_code text,
  country text );
```

```
CREATE TABLE
reservation.reservations_by_confirmation (
  confirm_number text,
  hotel_id text,
  start_date date,
  end_date date,
  room_number smallint,
  guest_id uuid,
  PRIMARY KEY (confirm_number) )
  WITH comment = ‘Q6. Find reservations by
confirmation number’;

CREATE TABLE
reservation.reservations_by_hotel_date (
  hotel_id text,
  start_date date,
  end_date date,
  room_number smallint,
  confirm_number text,
  guest_id uuid,
  PRIMARY KEY ((hotel_id, start_date),
room_number) )
  WITH comment = ‘Q7. Find reservations by hotel
and date’;
```

```
CREATE TABLE user_tweets (
  userid uuid,
  added_date timestamp,
  tweetid uuid,
  name text,
  image_location text,
  PRIMARY KEY (user_id, added_date, tweetid)
) WITH CLUSTERING ORDER BY (added_date DESC, tweetid
ASC)
```

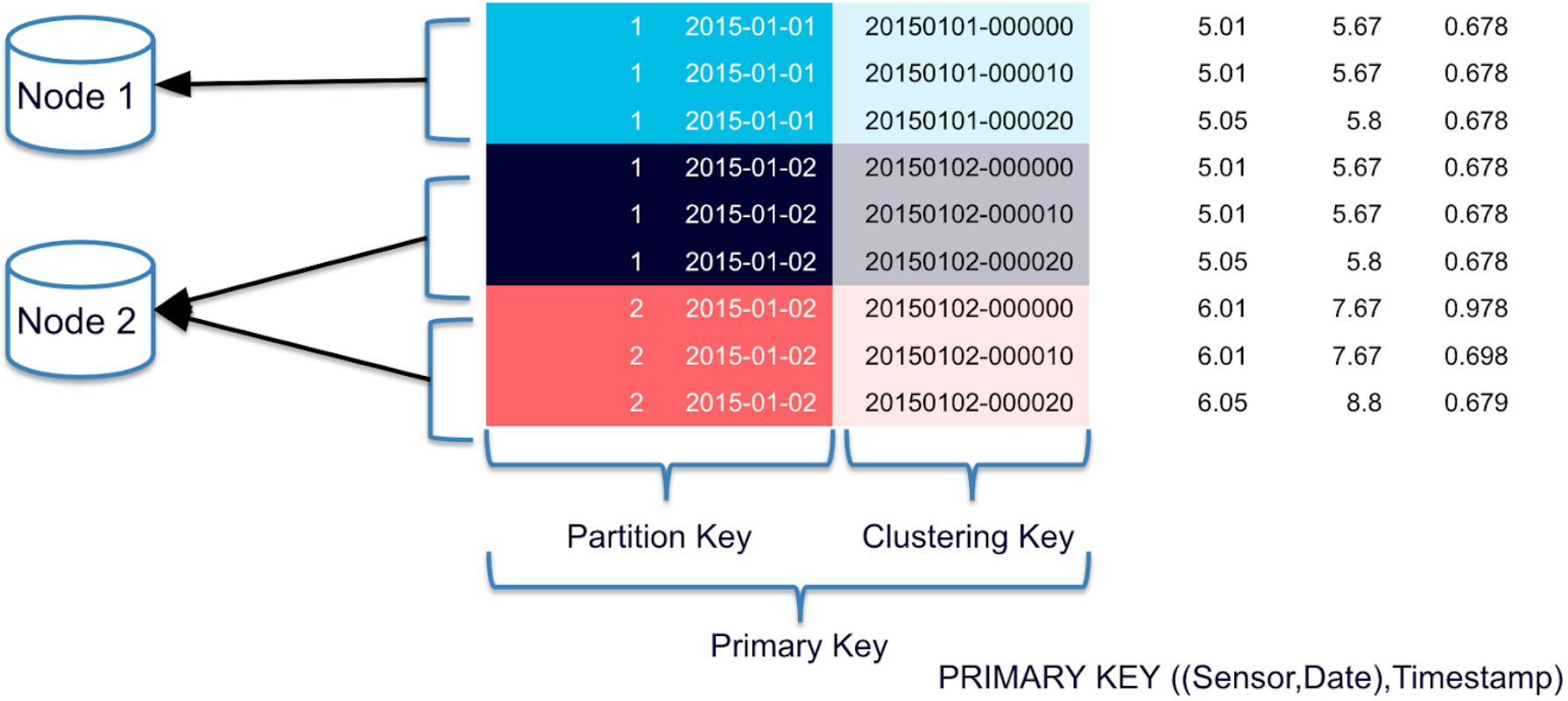
```
SELECT * FROM user_tweets WHERE userid = ???;

SELECT * FROM user_tweets WHERE userid = ??? AND added_date = ???;

SELECT * FROM user_tweets WHERE userid = ??? AND tweetid = ???;

SELECT * FROM user_tweets;

SELECT * FROM user_tweets ALLOW FILTERING;
```



Sensor #	Date	Timestamp	Metric1	Metric2	Metric3
1	2015-01-01	20150101-000000	5.01	5.67	0.678
1	2015-01-01	20150101-000010	5.01	5.67	0.678
1	2015-01-01	20150101-000020	5.05	5.8	0.678
1	2015-01-02	20150102-000000	5.01	5.67	0.678
1	2015-01-02	20150102-000010	5.01	5.67	0.678
1	2015-01-02	20150102-000020	5.05	5.8	0.678
2	2015-01-02	20150102-000000	6.01	7.67	0.978
2	2015-01-02	20150102-000010	6.01	7.67	0.698
2	2015-01-02	20150102-000020	6.05	8.8	0.679

PRIMARY KEY ((Sensor,Date),Timestamp)

Problémy NoSQL

- Redundancia dát
- Dodatočná implementácia pre zabezpečenie konzistencie (BATCH, Lightweight transakcie)
- On-Demand Analytika
- Chýbajúca štandardizácia
- Novorodenecké chyby, nevyspelý ekosystém a podpora
- Nie sú univerzálne

Kedy používať NoSQL databázy

- Veľké množstvo dát a problémy so škálovaním RDBMS
- Clustering
- Nezáleží nám na konzistentnosti a preferujeme dostupnosť a rýchlosť

Kedy nepoužívať NoSQL

- Pracujeme s malými datasetmi
- Vyžadujeme využívanie komplexných transakcií
- Implementácia aplikácií s voľnou špecifikáciou požiadaviek
- Nezáleží nám na globálnej škálovateľnosti
- Keď ste spokojní s aktuálnou db
- Keď nerozumiete špecifickým výhodám NoSQL databáz