



NoSQL databáze

Pavel Cenek

OptimSys, s.r.o.

TO SQL OR NOSQL



THAT'S THE QUERY

imggenerator.net

Relační databáze

- Tabulka = soubor záznamů stejného typu
 - Týká se jedné entity (prvku reálného světa)
 - Řádky tabulky = jednotlivé záznamy
- Sloupce tabulky = atributy - vlastnosti entity
 - Datový typ (číslo, text, datum, logická hodnota, atd.), omezení hodnot
 - Primární klíč - soubor hodnot jednoznačně identifikující konkrétní záznam
 - Cizí klíč - odkazuje na konkrétní záznam v jiné tabulce, tj. nese informaci o vztazích (**relacích**) mezi jednotlivými záznamy



Relační databáze


- Pevná struktura dat
- Jazyk SQL
 - Jazyk pro tvorbu dotazů do relačních databází
 - Vytvořen v IBM
 - Poprvé představen v roce 1974
- MySQL/MariaDB, PostgreSQL, Oracle DB, Microsoft SQL server, SQLite, ...
- Podnikové aplikace: ERP, HR, účetnictví
- Bankovní aplikace



Nové tisíciletí, nové druhy aplikací

- Webové, mobilní, IoT (client–server)
- Požadavky
 - Podporovat velký počet současně pracujících uživatelů (~10.000 až ~1.000.000)
 - Globální užívání
 - Být neustále dostupné, bez výpadků
 - Zpracovávat polo- a nestrukturovaná data a obrovské objemy dat
 - texty, obrázky, videa, logy, telemetrie, ...
 - Poskytovat časté aktualizace a neustále nové funkce

Jak si stojí relační databáze

Velký počet současných uživatelů	<ul style="list-style-type: none">• Relační DB byly navrženy tak, aby je bylo možné provozovat na jednom serveru• Zvýšit kapacitu = koupit více procesorů, paměti, úložiště (tzv. vertikální škálování)	
Bezvýpadkový provoz	<ul style="list-style-type: none">• HA se řeší provozem na více serverech a replikací dat• Relační DB systémy to „nemají v genech“	
Data bez pevné struktury	<ul style="list-style-type: none">• Relační DB systémy to „nemají v genech“	
Agilní vývoj	<ul style="list-style-type: none">• Pevné DB schéma• Změna aplikace = změna struktury všech dat	

Odbočka: JSON

- **JavaScript Object Notation**
- Pravidla syntaxe JSON:
 - Data jsou v párech
jméno : hodnota
 - Data jsou oddělena čárkami
 - Složené závorky obsahují objekty
 - Hranaté závorky obsahují pole

```
{  
  "velikost": "střední",  
  "počet": 2,  
  "ingredience": [  
    "sýr",  
    "šunka",  
    "kukuřice"  
  ]  
}
```

Odbočka: ECMAScript/JavaScript

- ECMAScript: obecný objektově orientovaný programovací jazyk
- Určen pro provoz v rámci **hostitelského prostředí**
 - Hostitelské prostředí definuje např. operace pro vstup a výstup
 - Hostitelské prostředí definuje specifické objekty pro manipulaci s hostitelským prostředím
- ECMA-262
<https://tc39.es/ecma262/>
- JavaScript = ECMAScript + hostitelské prostředí „okno webového prohlížeče“
 - `window.close()`
 - `window.alert("Hello world!")`
 - `window.location`

NoSQL databáze

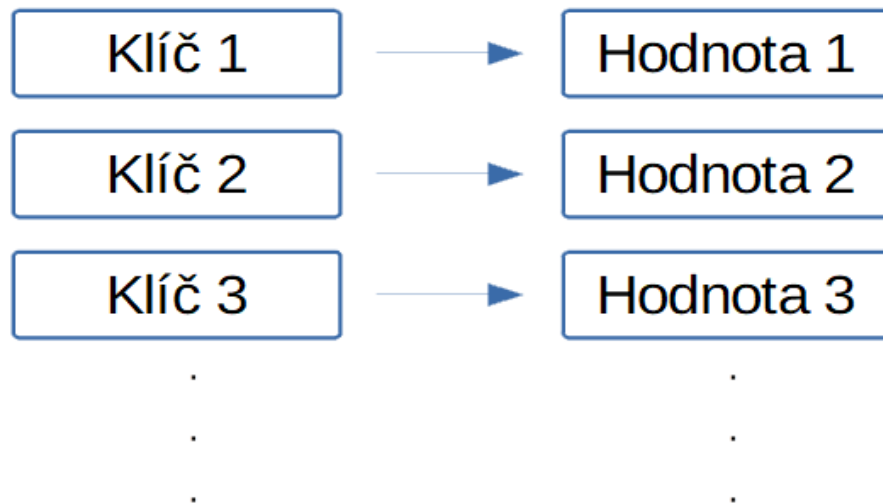
- non-SQL, not only SQL
- přesněji NoREL – „nerelační“

” Databáze, které dokáží zpracovávat velké objemy rychle se měnících nestrukturovaných dat jinými způsoby než relační databáze (SQL) s řádky a tabulkami “

<https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-nosql-database>

NoSQL databáze „klíč–hodnota“

- Do databáze se obvykle ukládá dvojice: klíč a jeho hodnota
- Na základě znalosti klíče jsme schopni z databáze získat uloženou hodnotu
- Hodnota je pro DB černá skříňka
- Redis, Amazon DynamoDB, Google Bigtable



Dokumentové NoSQL databáze

- Do databáze se ukládají dokumenty ve formátech JSON, XML apod.
- Každý ukládaný dokument může mít jinou strukturu
- MongoDB, Elasticsearch

```
{
  "first_name": "Jiří",
  "family_name": "Černý",
  "address": {
    "street": "Tokijská 15",
    "city": "Poděbrady",
    "zip": "29001"
  },
  "email": "cerny@email.cz",
  "phone": "+420728938271",
  "company": {
    "id": "23571483",
    "name": "Jeduvmedu, s.r.o."
  }
},
{
  "first_name": "Antonín",
  "middle_name": "František",
  "family_name": "Skalický",
  "email": "afskalicky@seznam.cz",
  "phone": "+420608428344"
},
{
  "first_name": "Eva",
  "family_name": "Vymětalová",
  "email": "evym@google.com"
}
```

Sloupcové NoSQL databáze

- Wide Column Store, Column Families
- Ke každému klíči je možné uložit více hodnot odpovídající příslušnému sloupci
- Každý klíč může mít vyplněné hodnoty jiných sloupců
- Používá tabulky, řádky a sloupce, ale na rozdíl od relační databáze se názvy a formát sloupců mohou v jednotlivých řádcích téže tabulky lišit
- Apache HBase (LinkedIn, Facebook, Spotify), Google Bigtable, Apache Cassandra (Facebook)

	Země	Produkt	Prodáno
1	ČR	čokoláda	3000
2	ČR	zmrzlina	5000
3	Rakousko	čokoláda	3000
4	Itálie	těstoviny	8000

uložení po řádcích

řádek 1	ČR
	čokoláda
	3000
řádek 2	ČR
	zmrzlina
	5000
řádek 3	Rakousko
	čokoláda
	3000
řádek 4	Itálie
	těstoviny
	8000

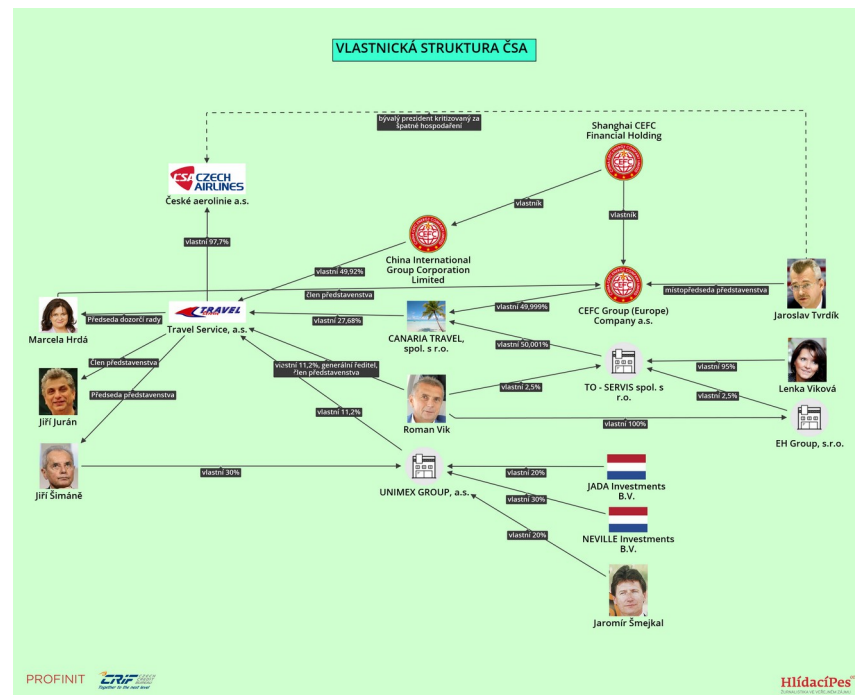
uložení po sloupcích

Země	ČR
	Rakousko
	Itálie
Produkt	čokoláda
	zmrzlina
	čokoláda
	těstoviny
Prodáno	3000
	5000
	3000
	8000


ČR:1,2 Rakousko:3 Itálie:4
čokoláda:1,3 zmrzlina:2 těstoviny:4
3000:1,3 5000:2 8000:4

Grafové NoSQL databáze

- Do databáze se ukládají
 - uzly a jejich vlastnosti
 - vztahy mezi těmito uzly (hrany)
- Rychlé hledání uzlů a vztahů v rozsáhlém grafu na základě grafových algoritmů
- Neo4j, FlockDB (Twitter)

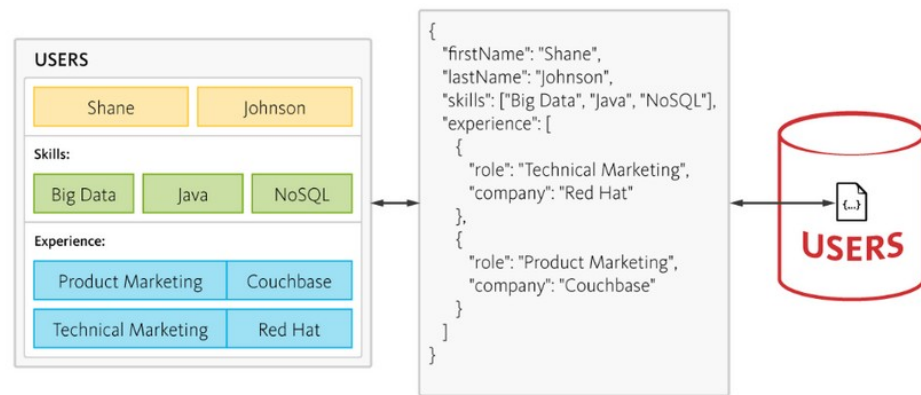
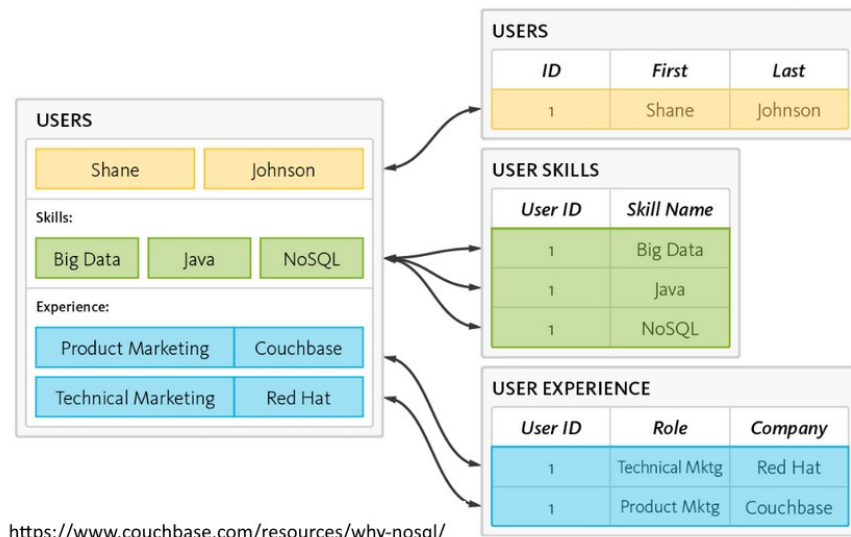


Jak si stojí NoSQL databáze

Velký počet současných uživatelů	<ul style="list-style-type: none">• NoSQL DB jsou typicky navrženy tak, aby bylo možné zvyšovat výkon přidáváním serverů (tzv. horizontální škálování)	
Bezvýpadkový provoz	<ul style="list-style-type: none">• Replikace + Partitioning = HA• NoSQL DB to „mají v genech“	
Data bez pevné struktury	<ul style="list-style-type: none">• Jsou na to dělané	
Agilní vývoj	<ul style="list-style-type: none">• Nemají pevné DB schéma• Data s rozdílnou strukturou lze přidávat zcela přirozeně	

Jak si stojí NoSQL databáze

- Vysoce efektivní **pro určité typy úloh**
- NoSQL databáze obecně zpracovávají data tak, jak jsou prezentována
 - Eliminuje se tak nutná transformace dat (tzv. objektově-relační mapování - ORM)





ACID vs BASE

- **Atomicity**
 - Všechny příkazy, které tvoří transakci, jsou považovány za jeden celek a buď uspějí, nebo selžou společně
- **Consistency**
 - Změny provedené v rámci transakce jsou v souladu s omezeními databáze
- **Isolation**
 - Transakce probíhají ve vzájemně izolovaném prostředí. Transakce se navzájem neovlivňují, i když běží souběžně.
- **Durability**
 - Jakmile je transakce dokončena a změny jsou zapsány do databáze, zůstanou zachovány (i po výpadku napájení)



ACID vs BASE

- **Basically Available**
 - Databáze je vždy dostupná díky rozprostření dat a jejich replikaci napříč uzly databázového clusteru (na úkor okamžité konzistence)
- **Soft State**
 - Hodnoty dat se mohou v průběhu času měnit, i když nedochází k žádnému zápisu, protože systém aktualizuje data, aby byla konzistentní
- **Eventually Consistent**
 - Data nemusí být konzistentní okamžitě, ale nakonec se konzistentními stanou. Čtení dat je však možné kdykoliv, i když výsledek nemusí odrážet skutečnost.

MongoDB

- **Humongous** = ohromný, enormní
- <https://www.mongodb.org/>
- Open source
- Dokument = JSON objekt
 - uložený v binárním formátu BSON
- Dokumenty jsou organizovány do kolekcí
- Množina kolekcí tvoří databázi
- mongosh - command line interface (CLI) pro MongoDB
 - Příkazy = ECMAScript
 - MongoDB je hostitelské prostředí
 - Objekt `db` – aktuální zvolená databáze
 - Objekt `db.jmenoKolekce` – přístup k dokumentům příslušné kolekce

Ukládání dat v MongoDB

```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "skills": ["Big Data", "Java", "NoSQL"],
  "experience": [
    {
      "role": "Technical Marketing",
      "company": "Red Hat"
    },
    {
      "role": "Product Marketing",
      "company": "Couchbase"
    }
  ]
}
```

- Silné stránky MongoDB
 - Zanořené struktury (subdokumenty)
 - Seznamy (pole) – relace 1:N
- Slabé stránky MongoDB
 - „joiny“ (složené dotazy s cizími klíči)
- Vícenásobné dotazy
- Nenormalizovaná data
 - Všechna data pro danou úlohu pohromadě

SQL → MongoDB

```
CREATE TABLE people (  
  id MEDIUMINT NOT NULL  
    AUTO_INCREMENT,  
  user_id Varchar(30),  
  age Number,  
  status char(1),  
  PRIMARY KEY (id)  
)
```

```
db.createCollection("people")
```

- není potřeba, kolekce se vytvoří automaticky při zápisu prvního dokumentu
- každý dokument má automaticky generovaný primární klíč v atributu `_id`

SQL → MongoDB

```
ALTER TABLE people  
ADD join_date DATETIME
```

```
ALTER TABLE people  
DROP COLUMN join_date
```

- Neexistuje ekvivalent, není třeba měnit strukturu databáze

```
db.people.updateMany(  
  {},  
  { $set: { join_date: new Date() } }  
)
```

```
db.people.updateMany(  
  {},  
  { $unset: { "join_date": "" } }  
)
```

SQL → MongoDB

CREATE INDEX

idx_user_id_asc_age_desc

ON people(user_id, age **DESC**)

```
db.people.createIndex(  
  {
```

```
    {
```

```
      user_id: 1,
```

```
      age: -1
```

```
    }
```

```
  )
```

DROP TABLE people

```
db.people.drop()
```

SQL → MongoDB

```
INSERT INTO people(user_id, age, status)
VALUES ("bcd001", 45, "A")
```

```
SELECT * FROM people
```

```
SELECT user_id, status FROM people
```

```
db.people.insertOne({
  user_id: "bcd001",
  age: 45,
  status: "A"
})
```

```
db.people.find()
```

```
db.people.find(
  {},
  { user_id: 1, status: 1, _id: 0 }
)
```

SQL → MongoDB

```
SELECT * FROM people  
WHERE status = "A" AND age = 50
```

```
SELECT * FROM people  
WHERE status = "A" OR age = 50
```

```
SELECT * FROM people  
WHERE age > 25 AND age <= 50
```

```
db.people.find(  
  { status: "A", age: 50 }  
)
```

```
db.people.find(  
  { $or: [ { status: "A" }, { age: 50 } ] }  
)
```

```
db.people.find(  
  { age: { $gt: 25, $lte: 50 } }  
)
```


Dotazovací operátory

\$eq	=	\$exists	dokument obsahuje uvedený atribut
\$gt	>	\$type	atribut dokumentu má uvedený typ
\$gte	>=	\$mod	modulo hodnoty má požadovaný výsledek
\$lt	<	\$regex	hodnota vyhovuje regulárnímu výrazu
\$lte	<=	\$text	dokument je ve výsledku textového vyhledávání
\$ne	≠	\$where	hodnota splňuje daný výraz v jazyce JavaScript
\$in	je ve výčtu	\$bitsAllSet	hodnota má na uvedených bitových pozicích hodnotu 1
\$nin	není ve výčtu	\$all	pole obsahuje všechny uvedené prvky
\$and	A	\$elemMatch	některý prvek pole splňuje všechny uvedené podmínky
\$not	NEGACE	\$geoIntersects	hodnoty mají geometrický průsečík s uvedenou gemoetrickou hodnotou
\$nor	ANI	\$near	geoprostorový objekt je v blízkosti uvedeného bodu
\$or	NEBO	\$rand	vygeneruje náhodné reálné číslo v rozmezí 0 až 1

SQL → MongoDB

```
SELECT * FROM people
WHERE status = "A"
ORDER BY user_id DESC
LIMIT 5
SKIP 10
```

```
INSERT INTO people(user_id, age, status)
VALUES ("bcd001", 45, "A")
```

```
db.people.find({
  status: "A"
}).sort({
  user_id: -1
}).limit(5).skip(10)
```

```
db.people.insertOne({
  user_id: "bcd001",
  age: 45,
  status: "A"
})
```

SQL → MongoDB

```
UPDATE people  
SET age = age + 3  
WHERE status = "A"
```

```
DELETE FROM people  
WHERE status = "D"
```

```
db.people.updateMany({  
  status: "A"  
},  
{  
  $inc: { age: 3 }  
})
```

```
db.people.remove({  
  status: "D"  
})
```

Agregační pipeline

- Skládá se z jedné nebo více fází, které zpracovávají dokumenty
- Každá fáze provede na vstupních dokumentech určitou operaci. Může například filtrovat dokumenty, seskupovat dokumenty a vypočítávat hodnoty
- Výsledek jedné fáze je vstupem pro fázi následující

```
db.people.find({
  status: "A"
}).sort({
  user_id: -1
}).limit(5).skip(10)

db.people.aggregate([
  {$match: { status: "A" }},
  {$sort: { user_id: -1 }},
  {$skip: 10},
  {$limit: 5 }
])
```

Agregační pipeline

- fáze **\$group**
 - Seskupí dokumenty a spočítá (statistické) hodnoty pro skupinu
 - Výstupem je jeden dokument pro každou skupinu.
- fáze **\$unwind**
 - Z jednoho dokumentu obsahujícího pole (seznam) vytvoří N dokumentů, každý obsahující jeden prvek pole
- fáze **\$project**
 - Přidá nebo odebere atributy dokumentu

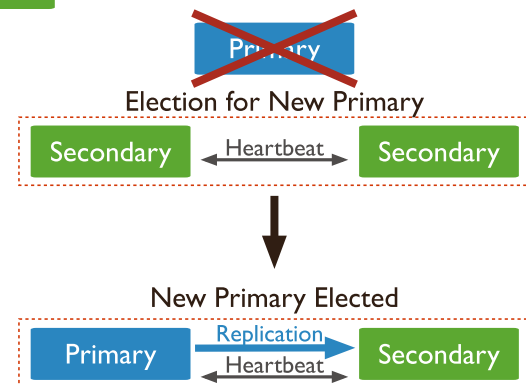
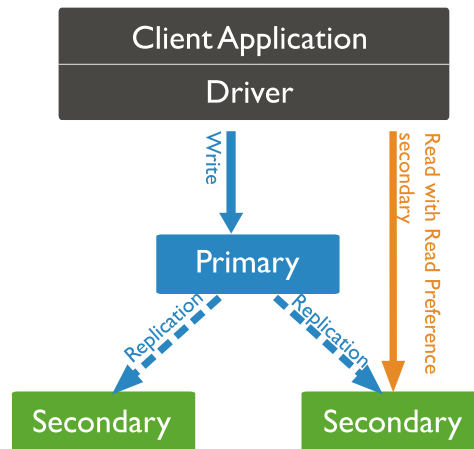
```
db.people.aggregate([
  {
    $group: {
      _id: "$status",
      num: { $sum: 1 },
      avg_age: { $avg: "$age" },
    }
  }
])
```

Indexy

- *Slouží k optimalizaci vyhledávání a filtrování dat v kolekcích*
- *Vytváření indexů umožňuje rychlejší vyhledávání ale na druhou stranu indexy zaberou více místa na disku*
- *Existují různé typy indexů, jako například jednoduché indexy, složené indexy a geografické indexy*
[Chat GPT]
- Pokud filtruji dokumenty podle více atributů, jednoduchý index nepomůže
- Statistické údaje ve fázi **group** spočítá MongoDB z indexu
- `db.collection.explain`
 - `queryPlanner`
 - `executionStats`
 - `allPlansExecution`

Replikace

- *Replica set*
 - několik instancí MongoDB obsahujících stejná data
- Slouží k zajištění vysoké dostupnosti a odolnosti systému proti výpadkům
- Repliky primárního uzlu slouží jako záloha pro případ selhání primárního uzlu
- Repliky mohou být použity pro čtení dat, což může výrazně zlepšit výkon aplikace



Sharding

- Data jedné databáze jsou rozdělena na části (shardy)
- Každá část je uložena na jiném serveru (uzlu)
- Data se mezi uzly rozdělují na základě klíče definovaného při vytváření kolekce
- Data by měla být rozdělena rovnoměrně

