BAZY DANYCH WYKŁAD 5

NO-SQL DATABASE

CO TO JEST NOSQL

- NoSQL obejmuje szeroką gamę różnych technologii baz danych, które zostały opracowane w odpowiedzi na wymagania stawiane w budowaniu nowoczesnych aplikacji:
  - Programiści pracują z aplikacjami, które tworzą ogromne ilości nowych, szybko zmieniających się typów danych - strukturyzowanych, częściowo ustrukturyzowanych, nieustrukturyzowanych i polimorficznych.
  - Systemy kaskadowe rozwoju oprogramowania już są prawie przeszłością. Małe zespoły pracują w krótkich sprintach, szybko iterując i przesyłając kody co tydzień lub dwa, niektóre nawet kilka razy dziennie.
  - Aplikacje, które kiedyś służyły skończonym odbiorcom, są teraz dostarczane jako usługi, które muszą być zawsze dostępne, dostępne z wielu różnych urządzeń i skalowane globalnie do milionów użytkowników.
  - Organizacje przechodzą obecnie do architektur skalowalnych poziomo, wykorzystujących oprogramowanie typu open-source, przetwarzanie w chmurze zamiast dużych monolitycznych serwerów i infrastruktury pamięci masowej.
  - Relacyjne bazy danych nie zostały opracowane w taki sposób, aby sprostać wyzwaniom związanym ze skalowalnością i sprawnością, z którymi borykają się współczesne aplikacje, ani też nie zostały zbudowane w celu wykorzystania dostępnej dziś mocy magazynowania i przetwarzania danych.

CO TO JEST NO SQL

Nie należy mylić z Brak SQL dokładnie skrót oznacza Not Only SQL

Są to NIERELACYJNE bazy dany (bez tabel)

Wyróżniły się ze względu na zastosowania:

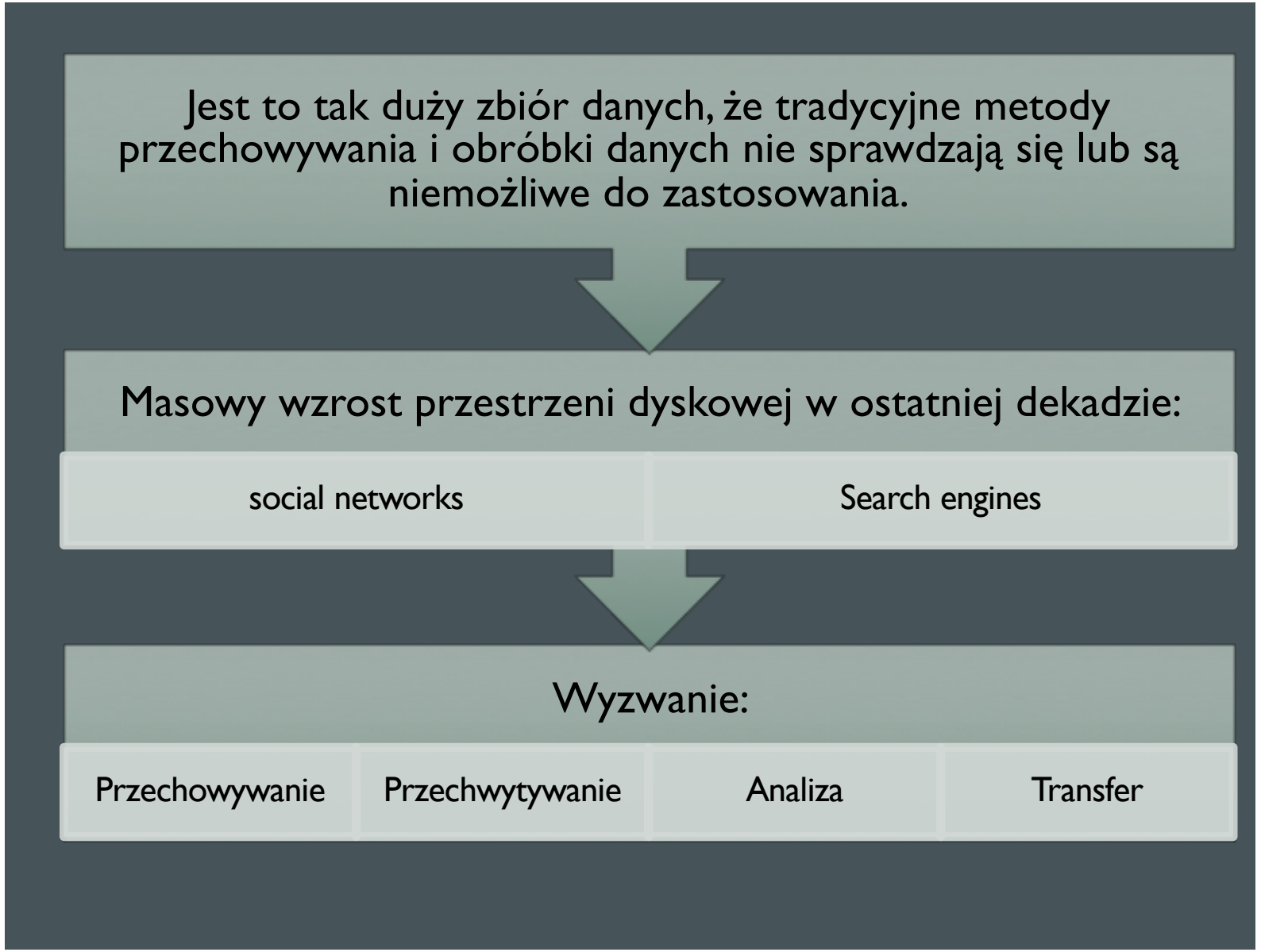
- Big-data
- Real-time web apps

Jest wiele typów baz NoSQL np. documents, key-value store

RODZAJE BAZ DANYCH NOSQL

- Dokumentowe bazy danych - łączą klucze ze złożoną strukturą danych zwaną dokumentem. Dokumenty mogą zawierać wiele różnych par key-value lub key-array, a nawet dokumenty zagnieżdżone.
- Zbiory grafowe służą do przechowywania informacji o sieciach danych, takich jak połączenia społecznościowe. Zbiory grafowe to Neo4J i Giraph.
- Zbiór klucz-wartość to najprostsze bazy danych NoSQL. Każdy pojedynczy element w bazie danych jest przechowywany jako nazwa atrybutu (lub "klucz") wraz z jego wartością. Przykładami zbiorów klucz-wartość są Riak i Berkeley DB. Niektóre zbiory key-value,, takie jak Redis, pozwalają każdej wartości mieć typ, na przykład "integer", który zwięszka funkcjonalność.
- Zbiory szerokokolumnowe, takie jak Cassandra i HBase, są zoptymalizowane pod kątem zapytań dotyczących dużych zestawów danych i przechowują kolumny danych razem zamiast wierszy.

CO TO JEST BIG-DATA



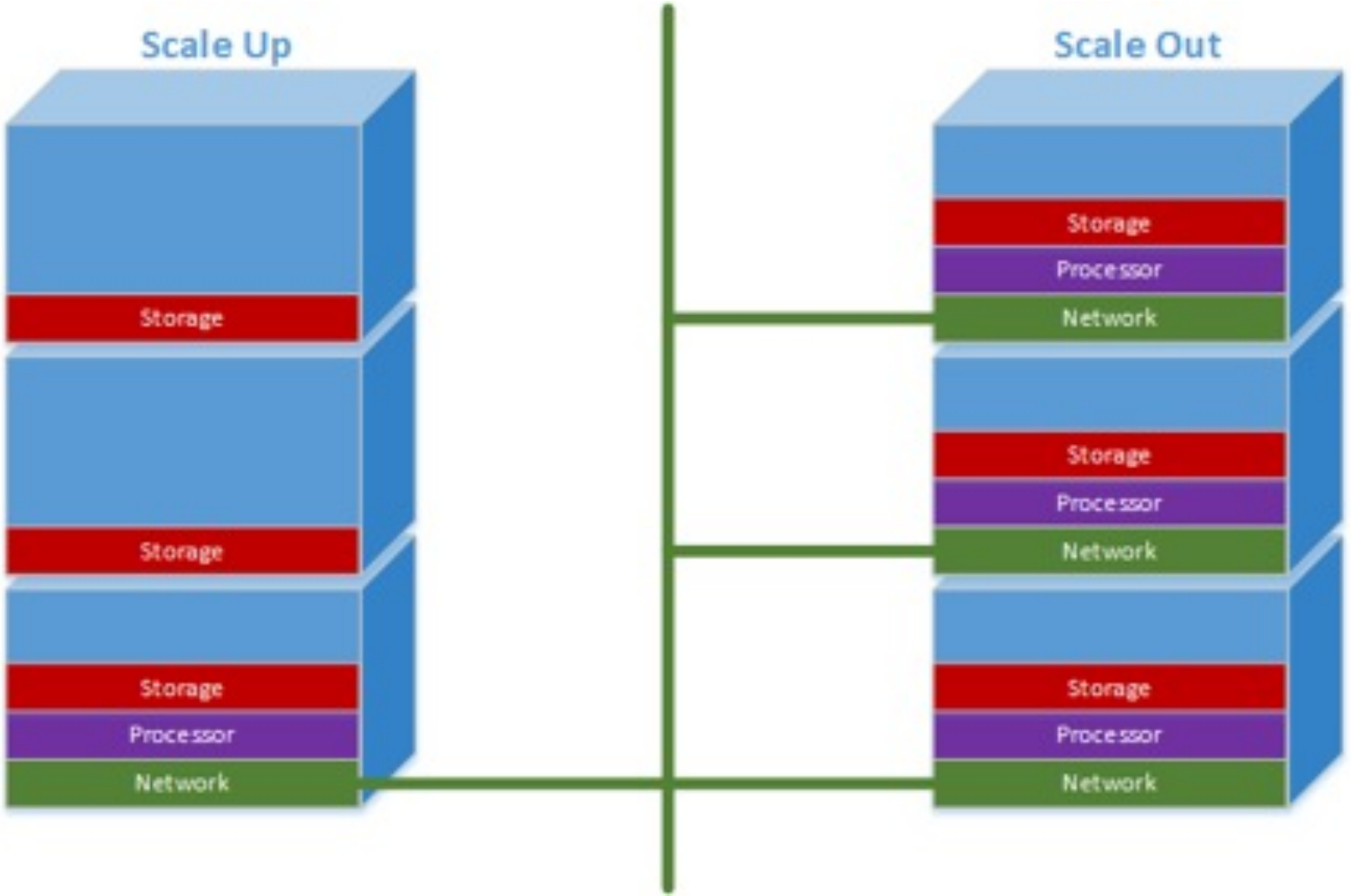
PRZEWAGA NOSQL NAD RDBMS

- Przeznaczone do Big Data – szybkość
- Model danych – niezwykle elastyczny – nie ma predefiniowanego schematu
- Struktura danych – NoSQL radzi sobie z niestrukturalnymi danymi, wygląd i strukturę danych planuje się po stronie aplikacji
- Tańsze w zarządzaniu
- Programowanie obiektowe, które jest łatwe w użyciu i elastyczne
- Skalowanie – poziome w przeciwieństwie do RDBMS, które skalują się dobrze pionowo (Geograficznie rozproszona architektura skalowalna zamiast kosztownej, monolitycznej architektury).

DYNAMICZNY SCHEMAT

- Relacyjne bazy danych wymagają zdefiniowania schematów przed dodaniem danych. Na przykład możesz chcieć przechowywać dane o swoich klientach, takie jak numery telefonów, imię i nazwisko, adres, miasto i stan - baza danych SQL musi wiedzieć, co przechowujesz z góry.
- Jest to kiepsko dopasowane przy podejściu do zwinnego programowania, ponieważ za każdym razem, gdy uzupełniasz nowe funkcje, schemat bazy danych często musi się zmieniać. Jeśli więc zdecydujesz, kilka iteracji do rozwoju, że chcesz przechowywać ulubione przedmioty klientów oprócz ich adresów i numerów telefonów, musisz dodać tę kolumnę do bazy danych, a następnie zmigrować całą bazę danych do nowy schemat.
- Jeśli baza danych jest duża, jest to bardzo powolny proces, który wymaga znacznych przestojów. Jeśli często zmieniasz dane, które przechowuje Twoja aplikacja - ponieważ szybko przeprowadzasz iterację - przestój ten może być częsty. Nie ma również możliwości, za pomocą relacyjnej bazy danych, efektywnego adresowania danych, które są całkowicie nieuporządkowane lub nieznane z góry.
- Bazy danych NoSQL zostały zbudowane w celu umożliwienia wstawiania danych bez wstępnie zdefiniowanego schematu. Ułatwia to wprowadzanie istotnych zmian w aplikacji w czasie rzeczywistym, bez martwienia się o przerwy w świadczeniu usług - co oznacza, że tworzenie jest szybsze, integracja kodu jest bardziej niezawodna i potrzeba mniej czasu administratora bazy danych. Programiści zwykle musieli dodać kod po stronie aplikacji, aby wymusić kontrolę jakości danych, na przykład nakazując obecność określonych pól, typów danych lub dopuszczalnych wartości. Bardziej wyrafinowane bazy danych NoSQL umożliwiają stosowanie zasad sprawdzania poprawności w bazie danych, co pozwala użytkownikom wymuszać zarządzanie danymi, przy zachowaniu korzyści płynących z dynamicznego schematu.

SKALOWANIE PIONOWE I POZIOME



PRZEWAGA RDBMS NAD NOSQL

1. Korzystniejsze dla danych relacyjnych
2. Normalizacja – pozbycie się redundancji – (baza mniejsza i szybsza)
3. Używają SQL – łatwy, stara, dobra technologia
4. Integralność danych – nie można usunąć powiązanych danych
5. ACID – atomicity niepodzielność, consistency – spójność, isolation – izolacja, durability- trwałość

NOSQL VS. SQL

TYP	Jeden typ (SQL database) z małymi wariacjami	Różne typy: key-value stores, document databases, wide-column stores, and graph databases
Historia powstania	Opracowany w 1970 roku, aby poradzić sobie z pierwszą falą aplikacji do przechowywania danych	Opracowany pod koniec 2000 roku, aby radzić sobie z ograniczeniami baz danych SQL, w szczególności skalowalnością, wielowymiarowymi danymi, geo-dystrybucją i technologiami zwinnymi rozwoju oprogamowania

NOSQL VS. SQL

Przykłady	MySQL, Postgres, Microsoft SQL Server, Oracle Database	MongoDB, Cassandra, HBase, Neo4j
Development Model	Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)	Open-source
Supports multi-record ACID transactions	Yes	Mostly no. MongoDB 4.0, scheduled for Summer 2018*, will add multi-document transactions. Sign up for the beta today.

NOSQL VS. SQL

Data Storage Model	Poszczególne rekordy (np. "Pracownicy") są przechowywane jako wiersze w tabelach, a każda kolumna przechowuje określoną część danych o tym rekordzie (np. "Menedżer", "data zatrudnienia" itp.), Podobnie jak arkusz kalkulacyjny. Powiązane dane są przechowywane w osobnych tabelach, a następnie łączone, gdy wykonywane są bardziej złożone zapytania. Na przykład "biura" mogą być przechowywane w jednej tabeli, a "pracownicy" w innej. Gdy użytkownik chce znaleźć adres służbowy pracownika, silnik bazy danych łączy się razem w tabelach "pracownik" i "biuro", aby uzyskać wszystkie niezbędne informacje.	Różni się w zależności od typu bazy danych. Na przykład zbiory klucz-wartość działają podobnie do baz danych SQL, ale mają tylko dwie kolumny ("klucz" i "wartość"), a bardziej złożone informacje są czasami przechowywane jako BLOB w kolumnach "wartości". Bazy danych dokumentów całkowicie eliminują model tabeli i wiersza, przechowując wszystkie istotne dane razem w jednym "dokumencie" w JSON, XML lub innym formacie, który może hierarchicznie zagnieżdżać wartości.

NOSQL VS. SQL

Schemat	Struktura i typy danych są ustalane z góry. Aby przechowywać informacje o nowym elemencie danych, cała baza danych musi zostać zmieniona, w tym czasie baza danych musi zostać przejść do trybu offline.	Zwykle dynamiczny, z pewnymi regułami sprawdzania poprawności danych. Aplikacje mogą dodawać nowe pola w locie, w przeciwieństwie do wierszy tabeli SQL, różne dane mogą być przechowywane razem w razie potrzeby. W przypadku niektórych baz danych dodawanie nowych pól jest nieco trudniejsze.

NOSQL VS. SQL

Skalowanie	Istnieje możliwość rozprzestrzeniania baz danych SQL na wiele serwerów, ale generalnie wymagana jest znaczna inżynieria dodatkowa, a podstawowe funkcje relacyjne, takie jak JOIN, integralność referencyjna i transakcje są zazwyczaj tracone.	W poziomie, co oznacza, że w celu zwiększenia wydajności administrator bazy danych może po prostu dodać więcej serwerów comodity lub instancji w chmurze. W razie potrzeby baza danych automatycznie rozsyła dane między serwerami.

NOSQL VS. SQL

Data Manipulation	Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE…	Through object-oriented APIs
Spójność	Zapewnia silną spójność	Zależy od produktu. Niektóre zapewniają silną spójność (np. MongoDB, z przestrajalną spójnością odczytu), podczas gdy inne oferują chwilową spójność (np. Cassandra).

MONGO DB

- MongoDB to baza dokumentów open-source'owa i wiodąca baza danych NoSQL.
- MongoDB jest napisany w C ++.
- MongoDB to wieloplatformowa, zorientowana na dokumenty baza danych zapewniająca wysoką wydajność, wysoką dostępność i łatwą skalowalność. MongoDB zbudowano w celu zbierania i dokumentowania.

POJĘCIA

Baza danych to fizyczny pojemnik na kolekcje. Każda baza danych otrzymuje własny zestaw plików w systemie plików. Pojedynczy serwer MongoDB zwykle ma wiele baz danych.

Kolekcja to grupa dokumentów MongoDB. Jest to odpowiednik tabeli RDBMS. Zbiór istnieje w pojedynczej bazie danych. Kolekcje nie wymuszają schematu. Dokumenty w kolekcji mogą mieć różne pola. Zazwyczaj wszystkie dokumenty w zbiorze mają podobny lub pokrewny cel.

Dokument to zestaw par klucz-wartość. Dokumenty mają dynamiczny schemat. Dynamiczny schemat oznacza, że dokumenty z tego samego zbioru nie muszą mieć tego samego zestawu pól lub struktury, a wspólne pola w dokumentach kolekcji mogą zawierać różne typy danych.

NAZEWNICTWO

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

PRZYKŁADOWY DOKUMENT

```
{
        _id: ObjectId(7df78ad8902c)
        title: 'MongoDB Overview',
        description: 'MongoDB is no sql database',
        by: 'tutorials point',
        url: 'http://www.tutorialspoint.com',
        tags: ['mongodb', 'database', 'NoSQL'],
        likes: 100,
        comments: [
                {
                        user:'user1',
                        message: 'My first comment',
                        dateCreated: new Date(2011,1,20,2,15),
                        like: 0
                },
                {
                        user:'user2',
                        message: 'My second comments',
                        dateCreated: new Date(2011,1,25,7,45),
                        like: 5
                }
        ]
}
```

_id to 12-bajtowa liczba szesnastkowa, która zapewnia unikatowość każdego dokumentu. Możesz podać _id podczas wstawiania dokumentu. Jeśli tego nie zrobisz, MongoDB zapewni unikalny identyfikator dla każdego dokumentu. 12 bajtów:

- pierwsze 4 bajty dla bieżącego znacznika czasu,
- kolejne 3 bajty dla identyfikatora komputera,
- kolejne 2 bajty dla identyfikatora procesu serwera MongoDB i
- pozostałe 3 bajty, są prostymi wzrastającymi wartościami.

DLACZEGO WARTO KORZYSTAĆ Z MONGODB?

- Przechowywanie dokumentów - dane są przechowywane w postaci dokumentów w stylu JSON.
- Indeks na dowolny atrybut
- Replikacja i wysoka dostępność
- Auto-sharding
- Bogate zapytania
- Profesjonalne wsparcie MongoDB

GDZIE MOŻNA KORZYSTAĆ Z MONGODB?

- Big Data
- Zarządzanie treścią
- Infrastruktura mobilna i społeczna
- Zarządzanie danymi użytkownika
- Data Hub

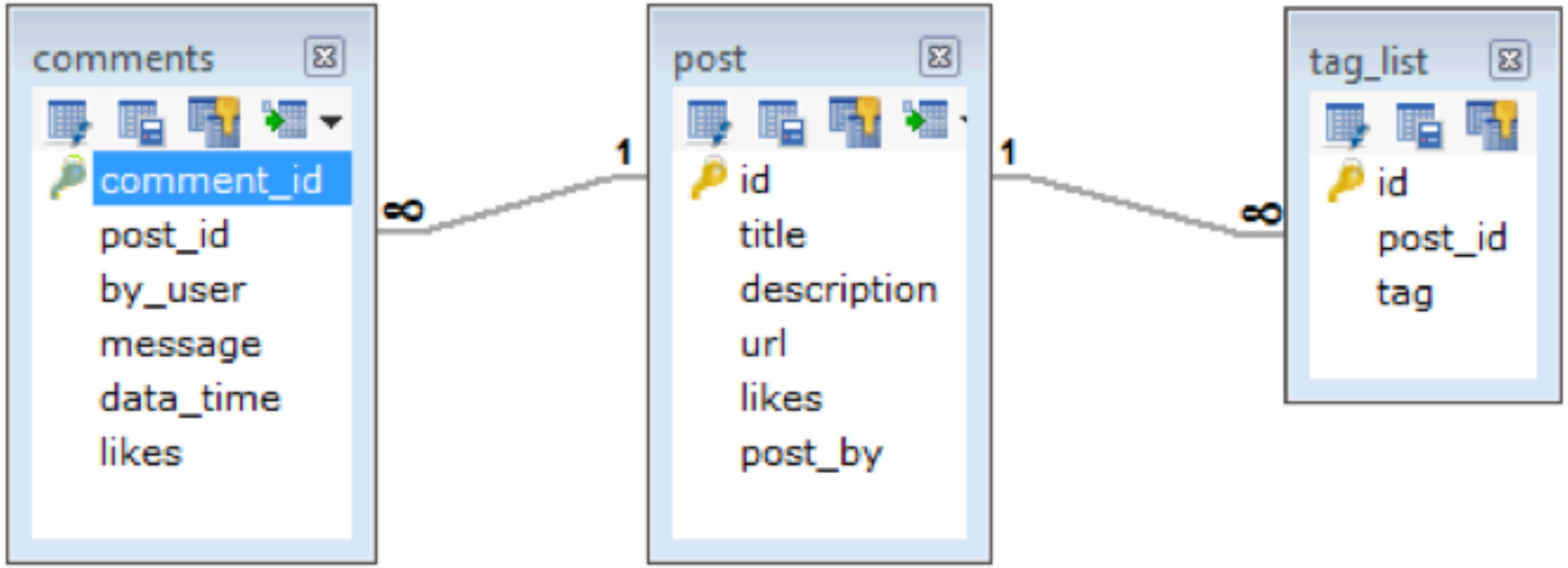
CECHY

- Dane w MongoDB mają elastyczne pliki schema.documents w tej samej kolekcji. Nie muszą mieć tego samego zestawu pól lub struktury, a wspólne pola w dokumentach kolekcji mogą zawierać różne typy danych.
- Kilka wskazówek do projektowania schematu w MongoDB
  - Zaprojektuj swój schemat zgodnie z wymaganiami użytkownika.
  - Połącz obiekty w jeden dokument, jeśli użyjesz ich razem. W przeciwnym razie rozdziel je (ale upewnij się, że nie powinno być potrzeby łączenia).
  - Powiel dane (ale rozsądnie), ponieważ miejsce na dysku jest tanie w porównaniu do czasu obliczeń.
  - Wykonaj JOIN podczas pisania, a nie czytania.
  - Zoptymalizuj swój schemat do najczęstszych przypadków użycia.
  - Wykonuj złożoną agregację w schemacie.

PRZYKŁAD

- Załóżmy, że klient potrzebuje projektu bazy danych na swoim blogu / stronie .Witryna ma następujące wymagania.
  - Każdy wpis ma unikalny tytuł, opis i adres URL.
  - Każdy post może mieć jeden lub więcej tagów.
  - Każdy wpis ma nazwę wydawcy i całkowitą liczbę polubień.
  - Każdy wpis zawiera komentarze nadesłane przez użytkowników wraz z ich imieniem, wiadomością, czasem i polubieniami.
  - W każdym poście może być zero lub więcej komentarzy.

RDBMS


comments
comment_id
post_id
by_user
message
data_time
likes
1
∞
post
id
title
description
url
likes
post_by
1
∞
tag_list
id
post_id
tag

MONGODB

```
{
        _id: POST_ID
        title:TITLE_OF_POST,
        description: POST_DESCRIPTION,
        by: POST_BY,
        url: URL_OF_POST,
        tags: [TAG1,TAG2,TAG3],
        likes:TOTAL_LIKES,
        comments: [
                {
                        user:'COMMENT_BY',
                        message:TEXT,
                        dateCreated: DATE_TIME,
                        like: LIKES
                },
                {
                        user:'COMMENT_BY',
                        message:TEXT,
                        dateCreated: DATE_TIME,
                        like: LIKES
                }
        ]
}
```