Projekt NIERelacyjnej Bazy danych

System zarządzania miejscami w akademiku

Filip Maciejak (260375)

Piotr Grygoruk (260299)

Jakub Szwedowicz (243416)

Spis treści

[1. Podsumowanie projektu 3](#_Toc120136818)

[1.1. Analiza projektu 3](#_Toc120136819)

[1.1.1. Początkowe założenia 3](#_Toc120136820)

[1.1.2. Początkowe ograniczenia 3](#_Toc120136821)

[1.2. Kierunki rozwoju 3](#_Toc120136822)

# Faza wstępna

## Wybór i uzasadnienie technologii NoQSL

### Wstęp

Relacyjne bazy danych są starszym rozwiązaniem od nierelacyjnych baz danych i uwidacznia się to w sposobie ich implementacji. Projektując relacyjną bazę danych trzeba ściśle opracować poszczególne tabele oraz typy danych. Wraz ze wzrostem ilości danych oraz liczby relacji – komplikuje się struktura bazodanowa co utrudnia modyfikacje tabel oraz wprowadzanie nowych. Ponad to zostały one zaprojektowane z myślą o działaniu na jednym urządzeniu w czasach gdy nie rozważało się zastosowań w systemach rozproszonych i łatwo skalowalnych (np. chmurowych). Prowadzi to do sytuacji, w której rozbudowa maszyny hostującej bazę danych może nastąpić tylko wertykalnie poprzez wymianę podzespołów na lepsze. Wynikiem tego jest powstanie pewnej granicy opłacalności powyżej której wymiana hardware-u na mocniejszy nie opłaca się.

Wiele z tych problemów zostało zaadresowane przez twórców nierelacyjnych baz danych i właśnie z myślą o takich systemach były one projektowane. Pośród ich cech szczególnych można wyróżnić elastyczny sposób przechowywania danych w postaci dokumentów. Często są one realizowane jako JSON-y, gdzie każdy dokument stanowi niezależną informację, co tym samym pozwala na łatwą dystrybucję bazy danych na wiele urządzeń (wzrost horyzontalny). Dane w JSONach są przechowywane w postaci par klucz-wartość co pozwala na ławy odczyt danych. Taka struktura przechowywania danych prowadzi jednak do ograniczeń względem możliwych typów danych – te bardziej złożone nie są obsługiwane.

Można wyróżnić kilka różnych typów nierelacyjnych baz danych, są to między innymi:

* Klucz-wartość (key-value database) – podstawowy rodzaj bazy danych, w której dane przechowywane są jako klucz i wartość. Wartości mogą być różnego typu jak liczby całkowite czy łańcuchy znaków natomiast w ogólności są to proste typy danych.
* Dokumentowe (document database) – jest to rodzaj bazy klucz-wartość, gdzie wartością jest dokument (np. JSON), który może zawierać wiele kolejnych par klucz-wartość.
* Grafowe (graph database) – gdzie węzły przechowują dane, a relacje są przedstawiane w postaci krawędzi. Jest to elastyczna struktura, do której bardzo łatwo można dodawać kolejne węzły, a zapytania o konkretne dane są szybkie. Ich istotną wadą jest trudność w przeszukiwaniu całej bazy danych. Wykorzystywane są w bazach danych, które mają reprezentować sieci danych: np. sieć połączeń pomiędzy grupą ludzi, struktury internetowe.
* Szeroko-kolumnowe (wide-column database) – są podobne strukturalnie do relacyjnych baz danych jednak zachowują elastyczność nierelacyjnych. Dane są przechowywane w postaci kolumn, gdzie dostęp do przechowywanych wartości jest za pomocą klucza, o który można myśleć jak o indeksie wiersza w relacyjnej bazie danych. Cechą szczególną takiej bazy danych jest, że poszczególne wiersze w tabeli nie muszą mieć takich samych typów danych. Oznacza to że zamiast wypełniać brakujące w kolumnie dane wartościami NULL, po prostu nie tworzy się pary klucz-wartość w danej kolumnie. Kolumny mogą być przechowywane na różnych urządzeniach. Warto też dodać że szeroko-kolumnowe bazy danych w odróżnieniu do kolumnowych (columnar database) wspierają przechowywanie na dysku zgrupowanych pod jednym kluczem kolumn. Zapytania o dane w takim przypadku są szybkie, natomiast transakcje grupujące różne wiersze czy nawet tabele są bardzo powolne ze względu na ilość kolumn, które nie są normalizowane tak jak w relacyjnych bazach danych. Takie struktury są często wykorzystywane np. w DataScience czy ML.

### Przegląd nierelacyjnych baz danych

* key-value database:
  + Berkeley DB – szybka, dojrzała baza danych z bardzo dobrą dokumentacją i binding-ami dla wielu języków (C, C++, Java, C#, itd.). Dobrze skalowalna, open-source i prosta. Elastyczna poprzez nie wspieranie schema-y (co może być również wadą, gdyż nie można nakładać ograniczeń na dane). Wspierana przez Oracle.
  + Riak KV – rozproszona baza danych, która w swej architekturze nie posiada głównego serwera (masterless). Gwarantuje opóźnioną synchronizację danych (eventually consistent). Głównym celem autorów przy projektowaniu tej bazy danych jest utrzymanie dostępu do bazy danych oraz jej poprawność nawet w przypadku poważnych awarii sprzętu czy sieci internetowej. Z racji na swoje zastosowanie nie jest polecany do mniejszych baz danych (mniejszych niż 5 serwerów na klaster). Ponad to nadaje się do prostych zapytań klucz-wartość i jeśli charakter bazy danych zakłada duże, grupujące wiele danych zapytania (np. SELECT \* FROM table) to wówczas odradza się jego zastosowanie.
  + Dynamo DB – baza danych zaprojektowana z myślą o dużych zapytaniach o ścisłych ograniczeniach na przetworzenie zapytania. Gwarantuje przewidywalne opóźnienie do pewnego rozmiaru zapytania. Dość ograniczony zakres dostępnych zapytań, nie wspiera łączenia tabel a zatem również skomplikowanych zapytań. Ma ścisły limit 400KB co do wielkości elementów w tabeli i nie może on być zwiększony. Można też dodać, że jest częścią usługi AWS, w której posiada cechę że jest rozliczany pod względem liczby zapytań, a nie godzin hostowania.
  + Redis – bardzo szybka baza danych działająca w pamięci RAM. Często stosowana obok innych baz danych w roli cache-u. Łatwa w instalacji i użyciu.
* Document database:
  + MongoDB – replikująca baza danych pozwalająca na rozbudowę klasterów wertykalnie (poprzez zwiększanie zasobów), ale również horyzontalnie poprzez partycjonowanie (partitioning, również: sharding). Zapytania od klientów do różnych node-ów są oddelegowywane przez klaster. Jest to dość skomplikowana baza danych o wielu mechanizmach i konfigurowalnych ustawieniach, którą wykorzystuje się głównie w sporych, rozproszonych systemach.
* Graph database:
  + Neo4J – bardzo trudna do shardowania baza danych co oznacza, że cały dataset musi znaleźć się na jednym serwerze. Pociąga to za sobą fakt że, aby powiększać pojemność na dane to trzeba rozbudowywać serwery wertykalnie. Posiada odgórne ograniczenie na liczbę węzłów. Architektura master-slave może prowadzić do sporych opóźnień w przypadku nagłego wzrostu zapisów/odczytów. Brak zaawansowanych mechanizmów indeksowania. Do zalet można jednak zaliczyć bardzo szybki czas odczytu (jak na bazę grafową). Zaimplementowana w Javie pociąga za sobą użycie GC, który może powodować pauzy w systemie w trakcie zwalniania pamięci. Zapytania OLTP pozwalają na przeszukiwanie podzbioru grafu.
  + Giraph – wyspecjalizowana baza danych do zrównoleglonego przetwarzania relacji pomiędzy węzłami. Nakłada przez to wiele ograniczeń co do zapytań. Wykorzystuje zapytania typu OLAP i stąd nadaje się tylko do systemów, w których zapytania są przetwarzane na wszystkich węzłach. Znajduje zastosowanie głównie w ML.
* Wide-column:
  + Cassandra – baza danych w architekturze P2P. Wykorzystuje JVM co skutkuje wysokim zużyciem pamięci. Nie wspiera Joinów. Zoptymalizowana pod szybszy zapis niż odczyt, wprowadza redundancję danych. Nie spełnia ACID tzn. 1. w obrębie jednego zapytania, część operacji może się udać, a część nie. 2. Transakcje nie przestrzegają wszystkich ograniczeń bazy danych. 3. Część zapytań nie może być zrównoleglona (bo nie wszystkie są izolowane). 4. Baza danych nie została zaprojektowana z myślą o modyfikowaniu danych. Istotną cechą CassandryDB jest że wewnętrznie operacje Write, Read, Update i Delete są tym samym, czyli dopisaniem „młodszego” rekordu do już istniejącego, bez jego faktycznego usunięcia. Prowadzi to do sytuacji, w której zapytania o często modyfikowalne dane muszą na samym końcu jeszcze zweryfikować, która wersja danej zmiennej jest najnowsza.
  + HBase -

### Wybrana nierelacyjna baza danych

DynamoDB albo MongoBD. Raczej Mongo – TODO: Napisać czemu

## Weryfikacja przyjętych założeń i ograniczeń