NIERELACYJNE ROZWIĄZANIA BAZODANOWE

Wykład 3

Agenda

- Przegląd typów nierelacyjnych baz danych
- Praktyczne aspekty NoSQL
- Integracja NoSQL w języku Python

Wykorzystane źródła

- https://microsoft.com
- https://apache.com
- https://mongodb.com
- https://python.org
- https://scylladb.com

4

Rodzaje nierelacyjnych baz danych

Nierelacyjna baza danych

Nie używa schematu tabelarycznego wierszy i kolumn, który można znaleźć w większości tradycyjnych systemów baz danych

Używają modelu przechowywania zoptymalizowanego w zakresie konkretnych wymagań dla typu przechowywanych danych

Rodzaje baz danych NoSQL

- Dokumentowe
- Kolumnowe
- Klucz-wartość
- Grafowe
- Szeregów czasowych
- Obiektowe
- Zewnętrznych indeksów

Rodzaje baz NoSQL

Dokumentowe

Podstawowa charakterystyka baz dokumentowych (1/3)

Przechowywanie danych w dokumentach:

- Dane są przechowywane w formatach takich jak JSON, BSON lub XML.
- Każdy dokument jest samodzielną jednostką, która reprezentuje rekord.

Elastyczność schematu:

- Dokumenty mogą mieć różne pola i struktury, co umożliwia elastyczne schematy.
- Nie jest wymagany żaden wstępnie zdefiniowany schemat.

Hierarchiczna reprezentacja danych:

- Dokumenty mogą zawierać zagnieżdżone pola i tablice, co ułatwia reprezentację złożonych hierarchii danych.

Dostęp do klucza głównego:

- Każdy dokument jest identyfikowany za pomocą unikalnego klucza, zwykle używanego do szybkiego pobierania danych.

Wysoka skalowalność:

- Obsługiwane jest skalowanie poziome, co umożliwia dystrybucję danych na wielu serwerach.

Obsługa indeksowania:

- Zapewnia wydajne indeksowanie pól w dokumentach, umożliwiając szybkie wykonywanie zapytań.

Podstawowa charakterystyka baz dokumentowych (2/3)

Osadzone dokumenty:

- Powiązane dane można osadzić w pojedynczym dokumencie, aby zmniejszyć liczbę połączeń i poprawić wydajność odczytu.

Zaprojektowany dla danych niestrukturalnych i semi-strukturalnych:

- Nadaje się do przypadków użycia z nieregularnymi lub ewoluującymi modelami danych.

Elastyczność zapytań:

- Obsługuje rozbudowane zapytania przy użyciu pól i zagnieżdżonych obiektów, bez konieczności zastosowania języka SQL.

Systemy rozproszone i wysoka dostępność:

- Często zaprojektowany do pracy w architekturach rozproszonych z wbudowaną replikacją i tolerancją błędów.

Podstawowa charakterystyka baz dokumentowych (3/3)

Zoptymalizowany pod kątem wydajności odczytu i zapisu:

- Idealny dla aplikacji wymagających wysokiej przepustowości zarówno dla operacji odczytu, jak i zapisu.

Integracja z logiką aplikacji:

- Dokumenty ściśle odzwierciedlają struktury obiektów w językach programowania, zmniejszając potrzebę złożonego mapowania obiektowo-relacyjnego (ORM).

Spójność:

- Wiele baz danych dokumentów zapewnia spójność zamiast transakcji ACID, chociaż niektóre obsługują ACID na poziomie dokumentu.

Obsługa analityki:

- Niektóre bazy danych dokumentów oferują integrację z narzędziami analitycznymi lub zapewniają wbudowane możliwości analityczne.

Dokumentowa baza danych

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Przykłady:
MongoDB
Couchbase
ElasticSearch
Amazon DocumentDB
Amazon OpenSearch
RavenDB

Źródło: microsoft.com

Rodzaje baz NoSQL

Kolumnowe

Kolumnowe bazy danych (1/4)

Organizacja danych według kolumn:

- Dane są przechowywane w kolumnach, a nie w wierszach, co optymalizuje operacje odczytu i zapisu w określonych kolumnach.
- Kolumny są grupowane w „rodziny kolumn”, które reprezentują powiązane dane.

Wydajne w przypadku zapytań analitycznych:

- Idealne do zapytań obejmujących agregacje lub zapytania dotyczące określonych kolumn, takich jak analiza danych i Business Intelligence.

Rozproszone przechowywanie:

- Wartości zerowe lub puste nie są przechowywane, co czyni je wydajnymi pod względem pamięci w przypadku rozproszonych zestawów danych.

Skalowalność pozioma:

- Zaprojektowane dla systemów rozproszonych, umożliwiające bezproblemowe skalowanie poziome w węzłach.

Kolumnowe bazy danych (2/4)

Wysoka przepustowość zapisu i odczytu:

- Zoptymalizowane pod kątem operacji zapisu i odczytu na dużą skalę, z szybkim dostępem do danych w kolumnach.

Elastyczny schemat:

- Nowe kolumny można dodawać dynamicznie bez wpływu na istniejące wiersze lub kolumny.

Lokalizacja danych:

- Kolumny w obrębie rodziny kolumn (column-family) są przechowywane razem na dysku, co poprawia wydajność dostępu dla zapytań ukierunkowanych na powiązane kolumny.

Kolumnowe bazy danych (3/4)

Partycjonowanie i dystrybucja:

- Dane są partycjonowane i dystrybuowane w węzłach, często na podstawie kluczy wierszy, aby zapewnić równoważenie obciążenia i dostępność.

Spójność:

- Zwykle cechuje się modelem spójności ostatecznej, chociaż niektóre bazy danych oferują regulowane poziomy spójności.

Zoptymalizowany pod kątem zapisu:

- Często wykorzystuje techniki, takie jak drzewa scalania o strukturze logarytmicznej (drzewa LSM – Log Structured Merge Tree), aby wydajnie obsługiwać duże obciążenia zapisu danych.

Kompresja kolumnowa:

- Przechowywanie zorientowane na kolumny umożliwia zaawansowane techniki kompresji, zmniejszając wymagania dotyczące pamięci masowej.

Kolumnowe bazy danych (4/4)

Obsługuje dane szeregów czasowych:

- Dobrze nadaje się do szeregów czasowych lub danych dziennika ze względu na możliwość wydajnego przechowywania danych sekwencyjnych.

Dostosowywalne modele danych:

- Umożliwia elastyczne projekty, w których powiązane dane są logicznie grupowane w rodziny kolumn, ale nadal można do nich uzyskiwać dostęp niezależnie.

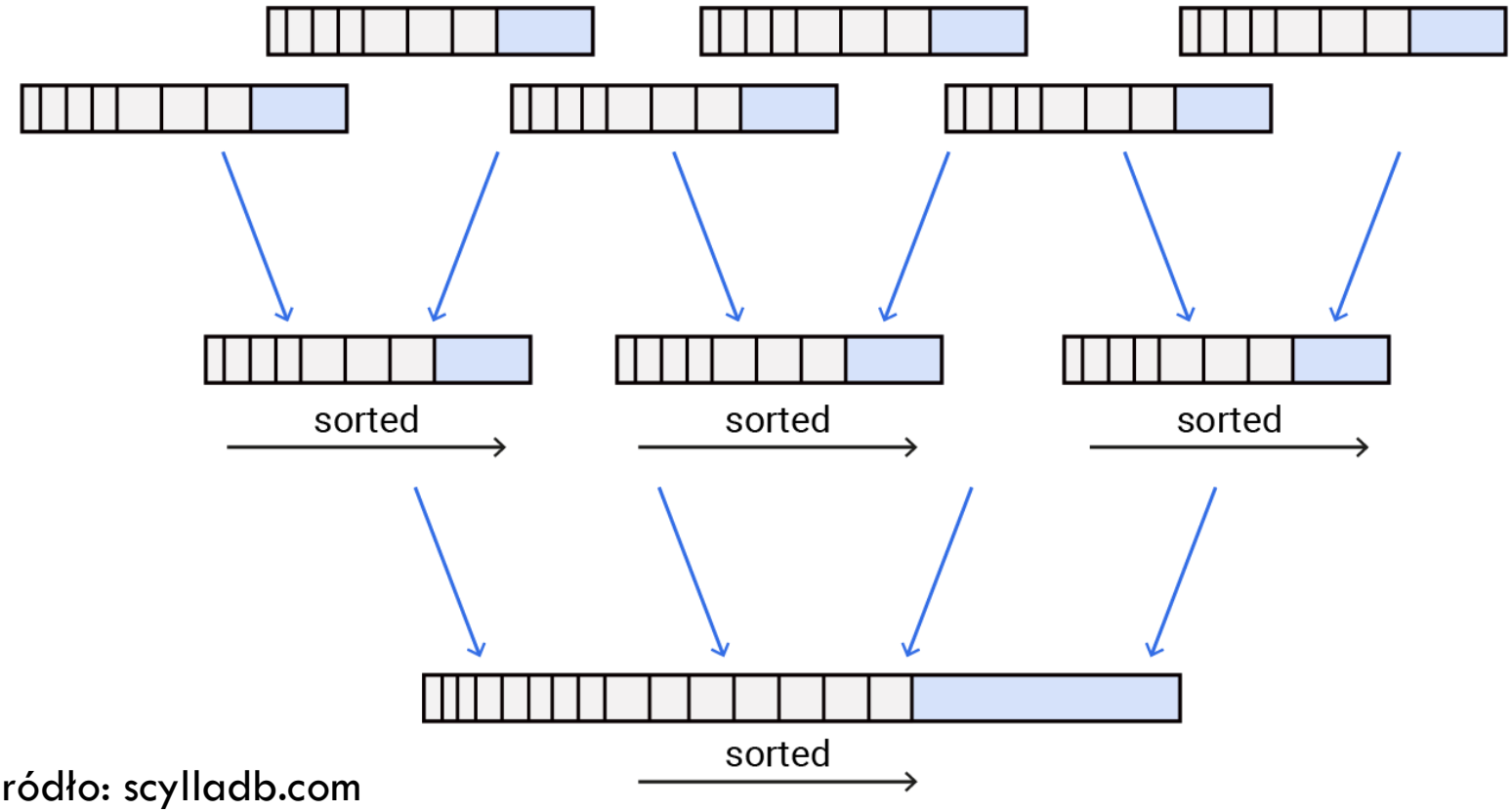
Rozproszona tolerancja błędów:

- Zapewnia tolerancję błędów poprzez replikację między węzłami, gwarantując wysoką dostępność.

Zoptymalizowany pod kątem operacji agregacji:

- Doskonały do operacji takich jak SUM, COUNT lub AVG na dużych zestawach danych dzięki przechowywaniu kolumnowemu.

Drzewo Log Structured Merge Tree



Źródło: scylladb.com

Kolumnowa baza danych

CustomerID	Column Family: Identity
001	**First name:** Mu Bae **Last name:** Min
002	**First name:** Francisco **Last name:** Vila Nova **Suffix:** Jr.
003	**First name:** Lena **Last name:** Adamcyz **Title:** Dr.

CustomerID	Column Family: Contact Info
001	**Phone number:** 555-0100 **Email:** someone@example.com
002	**Email:** vilanova@contoso.com
003	**Phone number:** 555-0120

Przykłady:
Apache Cassandra
Hbase
ScyllaDB
Amazon Keyspaces
Google Bigtable

Źródło: microsoft.com

Rodzaje baz NoSQL

Klucz-wartość

Bazy NoSQL typu klucz-wartość (1/3)

Prosty model danych:

- Dane są przechowywane jako pary klucz-wartość, gdzie klucz jest unikalny, a wartość to powiązane dane.
- Wartości mogą być ciągami znaków, JSON, blob (binary large object) lub dowolnym obiektem binarnym.

Bez schematu:

- Brak wstępnie zdefiniowanego schematu; struktura wartości jest nieznana dla bazy danych.

Wysoka wydajność:

- Zoptymalizowane pod kątem ultraszybkich operacji odczytu i zapisu, co czyni je idealnymi do aplikacji o niskim opóźnieniu.

Skalowalność:

- Łatwa skalowalność pozioma poprzez dystrybucję danych między węzłami przy użyciu strategii partycjonowania (np. spójne haszowanie).

Bazy NoSQL typu klucz-wartość (2/3)

Elastyczne przechowywanie wartości:

- Wartości mogą przechowywać dowolny typ danych, od prostych po złożone zagnieżdżone obiekty.

Efektywne dla prostych wzorców dostępu:

- Najlepiej nadaje się do scenariuszy, w których dostęp odbywa się głównie poprzez wyszukiwania oparte na kluczach (np. buforowanie).

Wysoka dostępność i tolerancja błędów:

- Często obsługuje replikację i rozproszone klastrowanie w celu zapewnienia tolerancji błędów i dostępności danych.

Brak relacji lub połączeń:

- Nie obsługuje natywnie relacji między danymi ani złożonych zapytań, takich jak łączenie zbiorów danych.

Bazy NoSQL typu klucz-wartość (3/3)

Spójność:

- Wiele magazynów klucz/wartość priorytetowo traktuje dostępność i partycjonowanie.

Fragmentacja i partycjonowanie danych:

- Dane są partycjonowane i dystrybuowane między węzłami, zwykle na podstawie skrótu klucza.

Dostosowywalne poziomy spójności:

- Niektóre systemy umożliwiają dostrajane poziomy spójności, pozwalając użytkownikom na zachowanie równowagi między wydajnością a spójnością.

Specyficzne dla przypadku użycia:

- Zaprojektowane dla określonych przypadków użycia, takich jak buforowanie (np. Redis), zarządzanie konfiguracją lub zarządzanie sesjami.

Minimalne możliwości zapytań:

- Zwykle brakuje zaawansowanych funkcji zapytań, takich jak filtrowanie, grupowanie lub agregacje.

Baza danych typu klucz-wartość

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

Przykłady:
Redis
Amazon DynamoDB
Riak KV
Memcached
Aerospike

Źródło: microsoft.com

Rodzaje baz NoSQL

Grafowe

Grafowe bazy danych (1/4)

Sposób zapisu danych:

- Dane są reprezentowane jako węzły (jednostki) i krawędzie (relacje) w strukturze grafu.
- Węzły przechowują atrybuty danych (właściwości), podczas gdy krawędzie definiują połączenia między węzłami.

Elastyczność schematu:

- Nie wymaga stałego schematu; węzły i krawędzie mogą mieć różne właściwości.

Zoptymalizowany pod kątem relacji:

- Zaprojektowany do wydajnego zarządzania i wykonywania zapytań dotyczących złożonych relacji między danymi, co czyni go idealnym do połączonych danych.

Natywne przetwarzanie grafu:

- Obsługuje operacje specyficzne dla grafu, takie jak przejścia, najkrótsza ścieżka i wyznaczanie centrum grafu.

Grafowe bazy danych (2/4)

Zapytania za pomocą przejść:

- Zapytania często obejmują przechodzenie przez graf, rozpoczynając od węzła startowego i eksplorując połączone węzły i krawędzie.

Rozbudowane języki zapytań:

- Używają specjalistycznych języków zapytań, takich jak Cypher (Neo4j), Gremlin lub SPARQL, aby uzyskać intuicyjne zapytania dotyczące grafu.

Efektywne dla głębokich połączeń:

- Wysoce zoptymalizowane pod kątem zapytań obejmujących głębokie relacje, takie jak przejścia wieloskokowe.

Grafowe bazy danych (3/4)

Transakcje ACID:

- Wiele baz danych grafowych obsługuje właściwości ACID, zapewniając integralność i spójność danych dla operacji transakcyjnych.

Lokalizacja danych:

- Przechowuje powiązane węzły i krawędzie blisko siebie, poprawiając wydajność przejścia.

Skalowalność:

- Obsługuje skalowanie pionowe i poziome, chociaż skalowanie poziome może być złożone w przypadku niektórych obciążeń.

Dynamiczne relacje:

- Nowe węzły i relacje można dodawać w każdym momencie bez zmiany struktury bazy danych.

Grafowe bazy danych (4/4)

Przyjazne dla wizualizacji:

- Struktura grafu jest z natury łatwa do wizualizacji, co pomaga w zrozumieniu i analizowaniu relacji danych.

Zaawansowana analityka:

- Zapewnia możliwości uruchamiania algorytmów grafowych, takich jak PageRank, wykrywanie społeczności i klastrowanie.

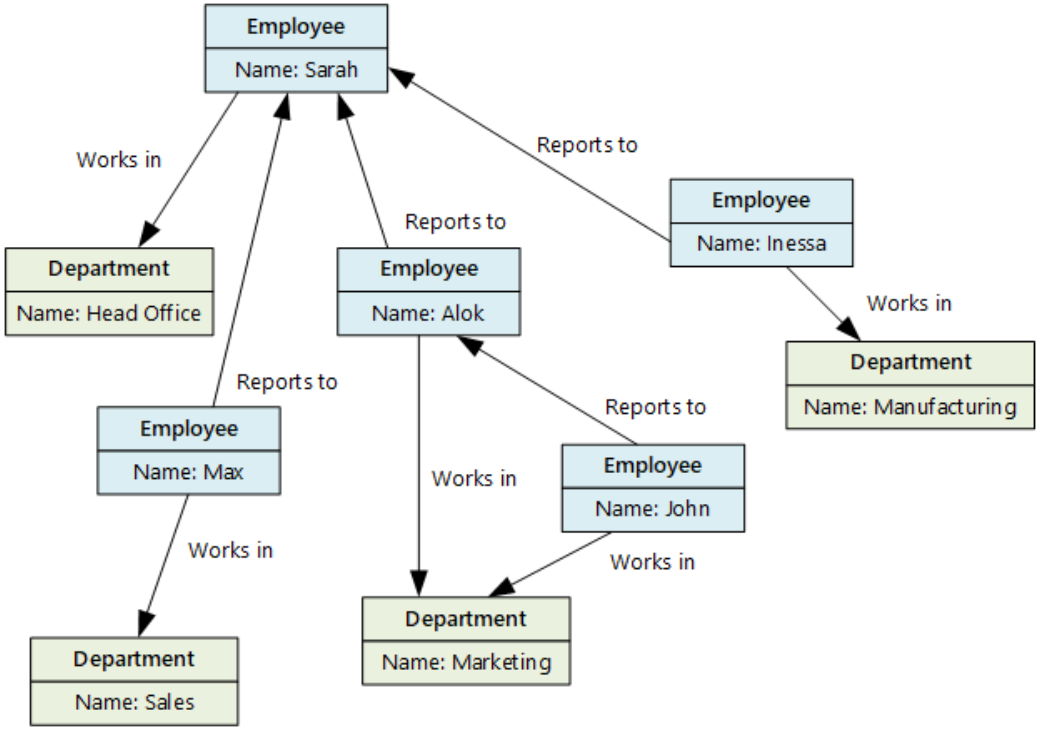
Skupione na przypadku użycia:

- Idealne dla aplikacji takich jak sieci społecznościowe, systemy rekomendacji, wykrywanie oszustw, grafy wiedzy i zarządzanie siecią.

Spójność i tolerancja błędów:

- Często projektowane z mechanizmami replikacji i tolerancji błędów w celu zapewnienia wysokiej dostępności.

Grafowe bazy NoSQL



Źródło: microsoft.com

Przykłady:
Neo4j
Amazon Neptune
ArangoDB
OrientDB
JanusGraph
TigerGraph

Rodzaje baz NoSQL

Szeregów czasowych

Bazy NoSQL typu szeregów czasowych (1/3)

Zoptymalizowany pod kątem danych z sygnaturą czasową:

- Zaprojektowany do obsługi danych powiązanych ze specyficznymi sygnaturami czasowymi, takimi jak metryki, zdarzenia lub dzienniki.

Wydajne przechowywanie danych sekwencyjnych:

- Używa specjalistycznych formatów przechowywania, takich jak struktury danych kolumnowych lub skompresowanych, aby wydajnie przechowywać dane uporządkowane czasowo.

Elastyczność schematu:

- Często bez schematu lub elastyczny pod względem schematu, dostosowujący się do dynamicznych zmian w polach danych.

Zoptymalizowany pod kątem odczytu:

- Dostrojony do wysokich szybkości pobierania, umożliwiający szybkie przechowywanie dużych ilości danych z sygnaturą czasową.

Wbudowane funkcje czasowe:

- Zapewnia natywne wsparcie dla zapytań opartych na czasie, takich jak agregacja czy próbkowanie.

Zasady przechowywania:

- Umożliwia ustawienie zasad przechowywania danych w celu automatycznego archiwizowania starszych danych.

Kompresja:

- Wykorzystuje techniki, takie jak kodowanie delta lub kompresja LZ4, aby zmniejszyć wymagania dotyczące wielkości magazynu danych.

Bazy NoSQL typu szeregów czasowych (2/3)

Skalowalność:

- Skalowalność pozioma w celu obsługi ogromnych zestawów danych w rozproszonych systemach.

Indeksowanie według czasu:

- Czas jest zazwyczaj głównym indeksem, umożliwiającym wydajne zapytania w zakresach czasu.

Efektywność zapytań i agregacji:

- Zoptymalizowany pod kątem typowych operacji, takich jak średnie, sumy i łączenia szeregów czasowych.

Integracja z narzędziami analitycznymi:

- Zapewnia integrację z narzędziami do wizualizacji i monitorowania danych dla pulpitów nawigacyjnych i raportów w czasie rzeczywistym.

Bazy NoSQL typu szeregów czasowych (3/3)

Odczyty o niskim opóźnieniu:

- Umożliwia wykonywanie zapytań i analiz w czasie rzeczywistym, co jest krytyczne dla systemów monitorowania i powiadamiania.

Obsługa wykrywania anomalii:

- Niektóre bazy danych zawierają funkcje do wykrywania trendów, wartości odstających lub anomalii w danych szeregów czasowych.

Zaprojektowany dla IoT i danych czujników:

- Obsługuje nieregularne próbkowanie, dane o wysokiej częstotliwości i tagowanie metryk z czujników i urządzeń IoT.

Replikacja i tolerancja błędów:

- Obsługuje replikację danych w celu zapewnienia niezawodności i dostępności.

Baza danych NoSQL typu szeregów czasowych

timestamp	deviceid	value
2017-01-05T08:00:00.123	1	90.0
2017-01-05T08:00:01.225	2	75.0
2017-01-05T08:01:01.525	2	78.0

Przykłady:
InfluxDB
TimescaleDB (hybryda – relacyjna oraz NoSQL)
OpenTSDB
Amazon Timestream
Prometheus
Druid

Źródło: microsoft.com

Rodzaje baz NoSQL

Obiektowe

Bazy NoSQL typu obiektowego (1/4)

Model danych zorientowany obiektowo:

- Dane są przechowywane jako obiekty, ściśle powiązane z koncepcjami programowania zorientowanego obiektowo.
- Obiekty obejmują zarówno dane (pola), jak i zachowania (metody).

Schemat:

- Nie wymaga wstępnie zdefiniowanego schematu, co pozwala na elastyczne i dynamiczne struktury obiektów.

Złożona reprezentacja danych:

- Obsługuje złożone typy danych, takie jak zagnieżdżone obiekty, tablice i relacje hierarchiczne, dzięki czemu nadaje się do bogatych modeli danych.

Bazy NoSQL typu obiektowego (2/4)

Bezpośrednie mapowanie na języki programowania:

- Obiekty w bazie danych są mapowane bezpośrednio na obiekty w językach programowania, co zmniejsza potrzebę mapowania obiektowo-relacyjnego (ORM).

Obsługa dziedziczenia i polimorfizmu:

- Umożliwia przechowywanie i wyszukiwanie hierarchii obiektów oraz obsługuje zachowanie polimorficzne.

Transakcje ACID:

- Często obsługuje właściwości ACID dla transakcji, zapewniając spójność i integralność danych.

Wydajne przechowywanie i pobieranie obiektów:

- Obiekty są przechowywane w całości, co może zmniejszyć narzut związany z pobieraniem w porównaniu z bazami danych relacyjnymi.

Bazy NoSQL typu obiektowego (3/4)

Indeksowanie i zapytania:

- Oferuje indeksowanie w celu szybkiego wyszukiwania i obsługuje zapytania dotyczące właściwości i relacji obiektów.

Skalowalność i dystrybucja:

- Może skalować poziomo lub pionowo, w zależności od implementacji, i często obsługuje systemy rozproszone w celu zapewnienia tolerancji błędów.

Integracja z aplikacjami zorientowanymi obiektowo:

- Idealne dla aplikacji opracowanych przy użyciu paradygmatów programowania zorientowanego obiektowo, ponieważ minimalizuje niezgodność w implementacji.

Bazy NoSQL typu obiektowego (4/4)

Trwałe przechowywanie obiektów:

- Umożliwia bezpośrednie przechowywanie obiektów bez konwersji do tabel lub dokumentów.

Wysoka wydajność dla operacji zorientowanych obiektowo:

- Zoptymalizowane pod kątem obciążeń, w których złożone obiekty są często tworzone, aktualizowane lub pobierane.

Różne opcje zapytań:

- Obsługuje zapytania oparte na atrybutach obiektów, relacjach lub metodach.

Baza danych NoSQL typu obiektowego

path	blob	metadata
/delays/2017/06/01/flights.cscv	0XAABBCCDDEEF...	{created: 2017-06-02}
/delays/2017/06/02/flights.cscv	0XAADDCCDDEEF...	{created: 2017-06-03}
/delays/2017/06/03/flights.cscv	0XAEBBDEDDEEF...	{created: 2017-06-03}

Przykłady:
Db4o (obiektowa baza danych dla Java i .NET)
ObjectDB
ZODB (obiektowa baza danych Zope)
GemStone/S (obiektowa baza danych Smalltalk)

Źródło: microsoft.com

Rodzaje baz NoSQL

Zewnętrzne indeksy

Bazy NoSQL typu zewnętrzne indeksy (1/4)

Oddzielenie pamięci masowej i indeksowania:

- Dane są przechowywane w jednym systemie (np. w bazie danych NoSQL), podczas gdy indeksowanie jest zarządzane zewnętrznie przez dedykowany moduł indeksujący.

Zaawansowane możliwości zapytań:

- Umożliwia wyszukiwanie pełnotekstowe, zapytania geoprzestrzenne i złożone możliwości filtrowania, których może brakować w podstawowej bazie danych NoSQL.

Integracja z wyszukiwarkami:

- Często integrowane z systemami takimi jak Apache Solr, Elasticsearch lub OpenSearch w celu indeksowania i wyszukiwania.

Poprawiona wydajność odczytu:

- Zapytania korzystają z wstępnie obliczonych indeksów, co przyspiesza wyszukiwanie i wyszukiwanie.

Bazy NoSQL typu zewnętrzne indeksy (2/4)

Skalowalność:

- Zarówno baza danych, jak i zewnętrzny moduł indeksujący mogą skalować się niezależnie, co umożliwia rozproszone architektury.

Obsługa różnych typów danych:

- Obsługuje dane ustrukturyzowane, częściowo ustrukturyzowane i nieustrukturyzowane przeznaczone do indeksowania.

Dostosowywalne indeksowanie:

- Indeksy można dostosowywać w celu optymalizacji określonych wzorców zapytań, takich jak tokenizacja do wyszukiwania tekstowego lub indeksy oparte na polach.

Narzuty indeksowania:

- Obejmuje dodatkowe narzuty w zakresie pamięci masowej i mocy obliczeniowej w celu utrzymania indeksu zewnętrznego.

Bazy NoSQL typu zewnętrzne indeksy (3/4)

Spójność:

- Indeksy zewnętrzne nie zawsze mogą być idealnie zsynchronizowane z bazą danych główną, co prowadzi do braku spójności.

Złożona synchronizacja:

- Wymaga mechanizmów synchronizujących zmiany między bazą danych główną NoSQL a zewnętrznym systemem indeksowania.

Zaawansowane wsparcie analityczne:

- Niektóre zewnętrzne systemy indeksowania zapewniają możliwości analityczne, takie jak agregacja i zapytania oparte na aspektach.

Bazy NoSQL typu zewnętrzne indeksy (4/4)

Wyszukiwanie geoprzestrzenne i pełnotekstowe:

- Specjalistyczne funkcje zapytań geoprzestrzennych i wyszukiwania pełnotekstowego, często nieobsługiwane natywnie przez główną bazę danych.

Idealne do obciążeń intensywnie odczytujących:

- Zoptymalizowane pod kątem scenariuszy, w których wydajność zapytań ma kluczowe znaczenie, takich jak wyszukiwarki, systemy rekomendacji i pulpity analityczne.

Tolerancja błędów i replikacja:

- Systemy zewnętrzne często obejmują tolerancję błędów i replikację w celu zapewnienia dostępności i trwałości danych.

Baza danych NoSQL typu zewnętrzne indeksy

id	search-document
233358	{"name": "Pacific Crest National Scenic Trail", "county": "San Diego", "elevation":1294, "location": {"type": "Point", "coordinates": [–120.802102,49.00021]}}
801970	{"name": "Lewis and Clark National Historic Trail", "county": "Richland", "elevation":584, "location": {"type": "Point", "coordinates": [–104.8546903,48.1264084]}}
1144102	{"name": "Intake Trail", "county": "Umatilla", "elevation":1076, "location": {"type": "Point", "coordinates": [–118.0468873,45.9981939]}}

Źródło: microsoft.com

Przykłady:
Elasticsearch z MongoDB
Apache Solr z Cassandra
Amazon OpenSearch Service z DynamoDB

Praktyczne aspekty NoSQL

Aspekty związane z NoSQL

Tworzenie bazy danych i kolekcji:

- MongoDB tworzy bazy danych i kolekcje tylko wtedy, gdy dane są wstawiane.

Operacje CRUD:

- insert_one, insert_many: Do dodawania dokumentów.
- find_one, find: Do wykonywania zapytań dotyczących dokumentów.
- update_one, update_many: Do aktualizowania dokumentów.
- delete_one, delete_many: Do usuwania dokumentów.

Filtry zapytań:

- Operatory $gt, $lt, $set do filtrowania i aktualizowania.

Import zbiorczy:

- Łatwe ładowanie danych z pliku JSON.

Usuwanie kolekcji:

- Przydatne do czyszczenia środowisk testowych.

Zarządzanie połączeniami:

- Otwieranie i zamykanie połączeń.

Operatory MongoDB (dokumentacja mongodb.com)

Operator	Meaning	Example	Equivalent to SQL operator
$gt	Greater Than	“score”:{"$gt”:0}	>
$lt	Less Than	“score”:{"$lt”:0}	<
$gte	Greater Than or Equal	“score”:{"$gte”:0}	>=
$lte	Less Than or Equal	“score”:{"$lte”:0}	<=
$all	Array Must Contain All	„skills”:{"$all”:["mongodb”,"python"]}	N/A
$exist	Property Must Exist	“email”:{"$exists”:True}	N/A
$mod	Modulo X Equals Y	“seconds”:{"$mod”:[60,0]}	MOD()
$ne	Not Equals	„seconds”:{"$ne”:60}	!=
$in	In	„skills”:{"$in”:["c”,"c++"]}	IN
$nin	Not in	„skills”:{"$nin”:["php”,"ruby”,"perl"]}	NOT IN
$nor	Nor	“$nor”:[{"language”:"english"},{"country”:"usa"}]	N/A
$or	Or	„$or”:[{"language”:"english"},{"country”:"usa}]	OR
$size	Array Must Be Of Size	“skills”:{"$size”:3}	N/A

Przykład pliku JSON stanowiącego źródło danych

- [
- {"name": "Eve", "age": 30, "city": "Los Angeles"},
  {"name": "Frank", "age": 40, "city": "Seattle"},
  {"name": "Grace", "age": 25, "city": "Boston"}
-]

NoSQL w języku Python

NoSQL w języku Python


```
from pymongo import MongoClient

# Połącz się z serwerem MongoDB
client = MongoClient("mongodb://localhost:27017/")

# Krok 1: Utwórz lub uzyskaj dostęp do bazy danych
db = client["my_database"] # Baza danych jest tworzona

# Krok 2: Utwórz lub uzyskaj dostęp do kolekcji
collection = db["my_collection"] # Kolekcja jest tworzona

# Krok 3: Wstaw pojedynczy dokument
document = {"name": "Alice", "age": 28, "city": "New York"}
result = collection.insert_one(document)
print(f"Wstawiony identyfikator dokumentu: {result.inserted_id}")
```

NoSQL w języku Python


```
# Krok 4: Wstaw wiele dokumentów
documents = [
{"name": "Bob", "age": 34, "city": "San Francisco"},
{"name": "Carol", "age": 22, "city": "Chicago"},
{"name": "David", "age": 45, "city": "Miami"}
]
result = collection.insert_many(documents)
print(f"Identyfikatory wstawionych dokumentów: {result.inserted_ids}")
```

NoSQL w języku Python


```
# Krok 5: Zapytanie do kolekcji
# Znajdź jeden dokument
one_document = collection.find_one({"name": "Alice"})
print("Pojedynczy dokument:", one_document)

# Znajdź wszystkie dokumenty
all_documents = collection.find() # Zwraca kursor
print("Wszystkie dokumenty:")
for doc in all_documents:
print(doc)

# Zapytanie z filtrami
filtered_documents = collection.find({"age": {"$gt": 30}}) # Wiek powyżej 30 lat
print("Przefiltrowane dokumenty:")
for doc in filtered_documents:
print(doc)
```

NoSQL w języku Python

- # Krok 6: Aktualizacja dokumentów
- # Aktualizacja pojedynczego dokumentu
- update_result = collection.update_one({"name": "Alice"}, {"$set": {"age": 29}})
- print(f"Zaktualizowano {update_result.modified_count} dokumentów).")

- # Aktualizacja wielu dokumentów
- update_many_result = collection.update_many({"age": {"$gt": 30}}, {"$set": {"city": "Unknown"}})
- print(f"Zaktualizowano {update_many_result.modified_count} dokumentów).")

NoSQL w języku Python

- # Krok 7: Usuwanie dokumentów
- # Usuwanie pojedynczego dokumentu
- delete_result = collection.delete_one({"name": "Carol"})
- print(f"Usunięto {delete_result.deleted_count} dokumentów).")

- # Usuń wiele dokumentów
- delete_many_result = collection.delete_many({"age": {"$lt": 30}})
- print(f"Usunięto {delete_many_result.deleted_count} dokumentów).")

NoSQL w języku Python

- # Krok 8: Import zbiorczy z pliku JSON
- import json

- # Zakładając, że masz plik „data.json” z listą obiektów JSON
- with open("data.json", "r") as file:
- bulk_data = json.load(file) # Załaduj dane z pliku JSON
- result = collection.insert_many(bulk_data)
- print(f"Wstawiono {len(result.inserted_ids)} dokumentów z JSON plik")

NoSQL w języku Python

- # Krok 9: Usuń kolekcję (opcjonalnie)
- # Spowoduje to usunięcie wszystkich danych w kolekcji
- collection.drop()
- print("Kolekcja została usunięta")

- # Krok 10: Zamknij połączenie MongoDB
- client.close()

Demonstracja

- Python, MS Excel, MongoDB, JSON, strony internetowe.

Pytanie

Rodzaje nierelacyjnych baz danych to:

- grafowe
- tabelaryczne
- obiektowo-relacyjne
- klucz-wartość