Szczegółowe Notatki na Egzamin: Nierelacyjne Rozwiązania Bazodanowe

# 1. Wprowadzenie do Nierelacyjnych Baz Danych

Nierelacyjne bazy danych (NoSQL) są zaprojektowane do przechowywania i przetwarzania dużych, zróżnicowanych danych, które nie pasują do sztywnego schematu relacyjnego. Są często wykorzystywane w systemach rozproszonych i aplikacjach, które muszą obsługiwać duże wolumeny danych w czasie rzeczywistym.

# 2. Typy Nierelacyjnych Baz Danych

Wyróżnia się kilka głównych typów baz NoSQL, w zależności od sposobu organizacji i przechowywania danych:

\*\*1. Bazy dokumentowe:\*\*

Przechowują dane w postaci dokumentów (JSON, BSON, XML). Są elastyczne i pozwalają na przechowywanie złożonych struktur danych bez konieczności sztywnego definiowania schematu.

Przykłady: MongoDB, Couchbase.

\*\*2. Bazy kolumnowe:\*\*

Dane są przechowywane w kolumnach zamiast w wierszach, co optymalizuje operacje odczytu i zapisu dla określonych kolumn. Idealne do zapytań analitycznych.

Przykłady: Cassandra, HBase.

\*\*3. Bazy klucz-wartość:\*\*

Przechowują dane jako pary klucz-wartość. Są bardzo szybkie i skalowalne, ale nie obsługują relacji między danymi.

Przykłady: Redis, DynamoDB.

\*\*4. Bazy grafowe:\*\*

Dane są reprezentowane jako węzły i krawędzie, co pozwala na efektywne zarządzanie złożonymi relacjami między danymi.

Przykłady: Neo4j, Amazon Neptune.

# 3. Modele Dystrybucyjne Baz NoSQL

Bazy NoSQL wspierają różne modele dystrybucji danych:

- \*\*Pojedynczy serwer:\*\* Dane przechowywane na jednej maszynie.

- \*\*Sharding:\*\* Dane dzielone na fragmenty i rozproszone na różnych serwerach.

- \*\*Replikacja master-slave:\*\* Serwer master synchronizuje dane z serwerami podrzędnymi.

- \*\*Replikacja peer-to-peer:\*\* Wszystkie serwery mają równorzędne prawa, co zwiększa dostępność danych.

\*\*Sharding:\*\* Proces dzielenia danych na mniejsze fragmenty (shardy), które są przechowywane na różnych serwerach. Umożliwia równoważenie obciążenia i skalowalność poziomą.

# 4. Operacje CRUD w MongoDB

MongoDB wspiera operacje CRUD (Create, Read, Update, Delete):

1. \*\*Tworzenie dokumentów:\*\* `db.collection.insertOne()` i `db.collection.insertMany()`

2. \*\*Wyszukiwanie dokumentów:\*\* `db.collection.find()` pozwala na filtrowanie danych za pomocą operatorów.

3. \*\*Aktualizowanie dokumentów:\*\* `db.collection.updateOne()` i `db.collection.updateMany()`

4. \*\*Usuwanie dokumentów:\*\* `db.collection.deleteOne()` i `db.collection.deleteMany()`

## Przykłady Operacji CRUD w MongoDB

1. \*\*Wstawienie pojedynczego dokumentu:\*\*  
```javascript  
db.users.insertOne({name: 'Jan', age: 30})  
```

2. \*\*Wyszukanie dokumentów:\*\*  
```javascript  
db.users.find({age: {$gt: 25}})  
```

3. \*\*Aktualizacja dokumentu:\*\*  
```javascript  
db.users.updateOne({name: 'Jan'}, {$set: {age: 31}})  
```

4. \*\*Usunięcie dokumentu:\*\*  
```javascript  
db.users.deleteOne({name: 'Jan'})  
```

# 5. Porównanie SQL i NoSQL

Porównanie operacji SQL i MongoDB:

|  |  |
| --- | --- |
| SQL | NoSQL (MongoDB) |
| CREATE TABLE people (...) | db.createCollection('people') |
| SELECT \* FROM people | db.people.find() |
| UPDATE people SET age=30 WHERE id=1 | db.people.updateOne({id: 1}, {$set: {age: 30}}) |
| DELETE FROM people WHERE id=1 | db.people.deleteOne({id: 1}) |

# 6. Hadoop i Big Data

Hadoop to platforma open source do przetwarzania dużych zbiorów danych. Składa się z:

- \*\*Hadoop Common:\*\* Podstawowe narzędzia i biblioteki.

- \*\*HDFS:\*\* System plików umożliwiający przechowywanie danych w klastrach.

- \*\*MapReduce:\*\* Model programowania do przetwarzania danych równolegle.

# 7. Model Programowania MapReduce

MapReduce składa się z dwóch głównych etapów:

1. \*\*Mapowanie:\*\* Przekształcanie danych wejściowych na pośrednie pary klucz-wartość.

2. \*\*Redukcja:\*\* Grupowanie i agregacja pośrednich wartości dla określonych kluczy.

Przykład liczenia słów z wykorzystaniem MapReduce:

\*\*Etap Map:\*\*  
```javascript  
map(String input\_key, String input\_value):  
 for each word in input\_value:  
 emit (word, '1');  
```

\*\*Etap Reduce:\*\*  
```javascript  
reduce(String output\_key, Iterator intermediate\_values):  
 int result = 0;  
 for each v in intermediate\_values:  
 result += parseInt(v);  
 emit(result);  
```