***Opis transponowania bazy relacyjnej na model nierelacyjny***

Kroki do wykonania:

## **1. Analiza struktury danych:**

### Identyfikacja tabel w bazie relacyjnej i ich zależności

Pierwszym krokiem w procesie transponowania bazy relacyjnej na model nierelacyjny jest analiza istniejącej struktury relacyjnej bazy danych. Zidentyfikowane tabele w oparciu o plik .sql to:

* **instruktorzy** – zawiera dane personalne i kontaktowe instruktorów.
* **kursanci** – przechowuje dane kursantów, takie jak imiona, nazwiska, daty urodzenia.
* **szkolenia** – opisuje kursy, w tym ich nazwę, cenę oraz liczbę godzin.
* **pojazdy** – informacje o pojazdach używanych podczas szkolenia.
* **plac** – baza danych opisująca obiekty treningowe oraz ich rezerwacje.
* **jazdy** – rejestruje sesje jazd, łącząc informacje o kursantach, instruktorach i pojazdach.
* **rezerwacje\_plac** – przechowuje informacje o rezerwacjach obiektów treningowych.
* **kursanci\_szkolenia** – tabela pośrednia, łącząca kursantów z kursami.
* **uprawnienia** – przechowuje dane dotyczące kategorii uprawnień posiadanych przez instruktorów.

### Zidentyfikowanie relacji pomiędzy tabelami

1. **instruktorzy → jazdy (1:N)**
   * Jeden instruktor może brać udział w wielu jazdach.
   * Klucz główny: id\_instruktora w instruktorzy.
   * Klucz obcy: id\_instruktora w jazdy.
2. **kursanci → jazdy (1:N)**
   * Jeden kursant może uczestniczyć w wielu jazdach.
   * Klucz główny: id\_kursanta w kursanci.
   * Klucz obcy: id\_kursanta w jazdy.
3. **kursanci ↔ szkolenia (N:M)**
   * Wielu kursantów może uczestniczyć w wielu szkoleniach.
   * Tabela pośrednia: kursanci\_szkolenia zawiera id\_kursanta i id\_kursu.
4. **szkolenia → kursanci\_szkolenia (1:N)**
   * Jedno szkolenie może być powiązane z wieloma zapisami w tabeli kursanci\_szkolenia.
   * Klucz główny: id\_kursu w szkolenia.
   * Klucz obcy: id\_kursu w kursanci\_szkolenia.
5. **kursanci → kursanci\_szkolenia (1:N)**
   * Jeden kursant może być powiązany z wieloma zapisami w tabeli kursanci\_szkolenia.
   * Klucz główny: id\_kursanta w kursanci.
   * Klucz obcy: id\_kursanta w kursanci\_szkolenia.
6. **plac → rezerwacje\_plac (1:N)**
   * Jeden tor na placu może być zarezerwowany wiele razy.
   * Klucz główny: id\_placu w plac.
   * Klucz obcy: id\_placu w rezerwacje\_plac.
7. **rezerwacje\_plac → jazdy (1:1)**
   * Jedna rezerwacja placu może być powiązana z jedną jazdą.
   * Klucz główny: id\_rezerwacji w rezerwacje\_plac.
   * Klucz obcy: id\_rezerwacji w jazdy.
8. **pojazdy → jazdy (1:N)**
   * Jeden pojazd może być użyty na jazdach.
   * Klucz główny: id\_pojazdu w pojazdy.
   * Klucz obcy: id\_pojazdu w jazdy.
9. **instruktorzy → uprawnienia (1:N)**
   * Jeden instruktor może posiadać wiele uprawnień.
   * Klucz główny: id\_instruktora w instruktorzy.
   * Klucz obcy: id\_instruktora w uprawnienia.

### Kluczowe pola dla operacji CRUD

Podczas projektowania struktury nierelacyjnej należy zidentyfikować kluczowe pola, takie jak:

* Klucze główne (np. id\_instruktora, id\_kursanta, id\_kursu).
* Klucze obce (np. id\_kursu w kursanci\_szkolenia).
* Pola często wyszukiwane.

## **2. Definicja kolekcji:**

### Grupowanie danych

Na podstawie analizy struktury relacyjnej zdecydowano, które dane powinny być przechowywane razem.

Dane dotyczące instruktorów i ich uprawnień mogą zostać zagnieżdżone w jednym dokumencie.

Informacje o kursantach i zapisanych kursach mogą zostać przechowywane razem.

Plac manewrowy i rezerwacje powinny być połączone w ramach jednego dokumentu.

### Mapowanie tabel na kolekcje

Mapowanie tabel na kolekcje to proces przekształcania struktury relacyjnej bazy danych na strukturę wykorzystywaną w bazach danych NoSQL. Nie mamy sztywnej struktury tabel jak w relacyjnych bazach danych, a dane są przechowywane w innych formach, takich jak dokumenty, klucze-wartości, kolumny lub grafy.

1. **Tabela instruktorzy:**

Utworzenie kolekcji *Instruktorzy*. Dane o uprawnieniach (np. kategoriach prawa jazdy) mogą zostać zagnieżdżone jako tablica. Poniżej przykład kodu w JSON odpowiedzialnego za zapisania obiektu .

{

"\_id": 1,

"imie": "Jan",

"nazwisko": "Kowalski",

"telefon": "123456789",

"email": "jan.kowalski@example.com",

"uprawnienia": ["B", "C"]

}

1. **Tabela kursanci:**

Kolekcja *Kursanci* z tablicą zapisanych kursów. Kod w JSON:

{

"\_id": 1,

"imie": "Anna",

"nazwisko": "Nowak",

"kursy": [

{

"id\_kursu": 2,

"nazwa": "Kategoria C",

"godziny": 20

}

]

}

1. **Tabela szkolenia:**

Kolekcja *Szkolenia* z listą uczestników. Kod w JSON:

{

"\_id": 2,

"nazwa": "Kategoria C",

"cena": 2000,

"uczestnicy": [

{ "id\_kursanta": 1, "imie": "Anna", "nazwisko": "Nowak" }

]

}

1. **Tabela pojazdy:**

Kolekcja *Pojazdy*, w której każdy dokument opisuje jeden pojazd. Kod w JSON:

{

"\_id": 1,

"marka": "Toyota",

"model": "Yaris",

"rok": 2018,

"typ": "osobowy"

}

1. **Tabela plac i rezerwacje\_plac:**

Kolekcja *Plac* z tablicą rezerwacji. Kod w JSON:

{

"\_id": 1,

"nazwa": "Plac A",

"rezerwacje": [

{ "id\_rezerwacji": 1, "data": "2025-01-20", "godzina": "10:00" }

]

}

1. **Tabela jazdy:**

Kolekcja *Jazdy* z referencjami do instruktorów, kursantów i pojazdów. Kod w JSON:

{

"\_id": 1,

"data": "2025-01-20",

"godzina": "12:00",

"id\_instruktora": 1,

"id\_kursanta": 1,

"id\_pojazdu": 1,

"id\_rezerwacji": 1

}

## 3. Projektowanie struktury nierelacyjnej:

##### Eksport danych

Dane z relacyjnej bazy danych są eksportowane w formacie JSON przy użyciu zapytań SQL. Każda tabela jest eksportowana do oddzielnego pliku JSON, np.:

SELECT json\_agg(row\_to\_json(instruktorzy)) FROM instruktorzy;

##### Zagnieżdżanie danych

* **Relacje 1:1:** Dane zagnieżdżone w jednym dokumencie.
* **Relacje 1:N:** Dane podrzędne jako tablice.
* **Relacje N:M:** Osobne kolekcje lub referencje.

## 4. Normalizacja danych:

##### Eliminacja redundancji

W modelu nierelacyjnym nadmiarowość danych jest kontrolowana przez utrzymywanie wspólnych informacji w osobnych kolekcjach:

* Dane pojazdów, które mogą być używane w wielu jazdach, są przechowywane w osobnej kolekcji *Pojazdy*.
* Podobnie dane instruktorów, które mogą być powiązane z wieloma jazdami lub uprawnieniami, pozostają w niezależnej kolekcji *Instruktorzy*.

Przykładowo, tabela jazdy w modelu relacyjnym odnosi się do tabel instruktorzy i pojazdy poprzez klucze obce. W modelu nierelacyjnym, zamiast kluczy obcych, stosowane są referencje w postaci identyfikatorów \_id. Kod JSON:

{

"\_id": "123",

"data": "2025-01-20",

"id\_instruktora": "456",

"id\_pojazdu": "789"

}

### Zamiana kluczy obcych na referencje

Wszystkie klucze obce w relacyjnym modelu danych (np. id\_instruktora, id\_pojazdu) są zastępowane referencjami do dokumentów w innych kolekcjach. Dzięki temu model nierelacyjny może zachować integralność danych bez nadmiernej redundancji.

## 5. Decyzja o zagnieżdżaniu dokumentów:

### Kryteria zagnieżdżania danych

Decyzja o zagnieżdżaniu zależy od kilku czynników:

* Częstotliwość używania danych razem: Dane, które są zawsze pobierane w tym samym zapytaniu, powinny być zagnieżdżone.
* Rozmiar danych: Zagnieżdżanie dużych struktur danych może prowadzić do problemów z wydajnością przy operacjach zapisu.

### Zagnieżdżenie w kolekcjach:

1. **Instruktorzy**: Dane o uprawnieniach instruktorów są zagnieżdżane w ich dokumentach. Kod JSON:

{

"\_id": "456",

"imie": "Jan",

"nazwisko": "Kowalski",

"telefon": "123456789",

"email": "jan.kowalski@example.com",

"uprawnienia": ["B", "C"]

}

1. **Plac**: Rezerwacje są przechowywane jako tablica w dokumencie placu. Kod JSON:

{

"\_id": "1",

"nazwa": "Plac A",

"rezerwacje": [

{ "id\_rezerwacji": "100", "data": "2025-01-20", "godzina": "10:00" },

{ "id\_rezerwacji": "101", "data": "2025-01-21", "godzina": "11:00" }

]

}

### Dane pozostawione bez zagnieżdżenia:

1. **Jazdy**: Sesje jazd są przechowywane jako oddzielna kolekcja z referencjami do instruktorów, kursantów i pojazdów. Kod JSON:

{

"\_id": "200",

"data": "2025-01-20",

"id\_instruktora": "456",

"id\_kursanta": "789",

"id\_pojazdu": "123"

}

1. KursanciSzkolenia: Relacja wiele-do-wielu pomiędzy kursantami a szkoleniami jest reprezentowana jako osobna kolekcja. Kod JSON:

{

"\_id": "300",

"id\_kursanta": "789",

"id\_kursu": "101"

}

## **6. Tworzenie kluczowych kolekcji:**

### Główne kolekcje:

1. **Instruktorzy:** Przechowują dane personalne instruktorów oraz ich uprawnienia.
2. **Kursanci:** Dane o kursantach, ich danych kontaktowych i zapisanych kursach.
3. **Szkolenia:** Informacje o kursach, takie jak nazwa, cena, liczba godzin oraz uczestnicy.
4. **Pojazdy:** Informacje o pojazdach używanych podczas szkolenia.
5. **Plac:** Dane o placach treningowych oraz ich rezerwacjach.

### Relacyjne kolekcje:

1. **KursanciSzkolenia:** Relacja kursant-szkolenie.
2. **Jazdy:** Sesje jazd z odniesieniami do instruktorów, kursantów i pojazdów.

### Indeksy:

Dla optymalizacji wyszukiwań należy dodać indeksy na polach które są często wykorzystywane.

## **7. Optymalizacja dostępu do danych:**

##### Minimalizacja operacji odczytu/zapisu

Dzięki zagnieżdżeniu danych takich jak uprawnienia w kolekcji Instruktorzy lub rezerwacje w kolekcji Plac, operacje CRUD mogą być wykonane na jednym dokumencie bez potrzeby wykonywania wielu zapytań.

##### Indeksy dla pól wyszukiwanych:

Indeksy zwiększają wydajność zapytań. Przykładowo, indeks na polu *data* w kolekcji *Jazdy* przyspiesza wyszukiwanie sesji jazd dla danego dnia.

## **8. Przeniesienie danych:**

##### Eksport danych z SQL

Dane są eksportowane z bazy relacyjnej do formatu JSON. Kod SQL:

COPY (SELECT row\_to\_json(t) FROM pojazdy t) TO '/path/to/export/pojazdy.json';

##### Transformacja do nowego formatu:

Po eksportowaniu dane są przekształcane w odpowiednie struktury JSON, np. grupując rezerwacje z placami lub uprawnienia z instruktorami.

##### Import do MongoDB:

Przekształcone dane są importowane do MongoDB za pomocą narzędzi takich jak *mongoimport*. Kod bash:

mongoimport --db osk --collection pojazdy --file pojazdy.json

## 9. Weryfikacja danych i operacji CRUD:

Sprawdzenie integralności danych:

Po imporcie należy upewnić się, że dane zostały zaimportowane poprawnie i zachowują powiązania między kolekcjami.

##### Testowanie operacji CRUD:

* + Pobranie wszystkich jazd dla danego kursanta.
  + Aktualizacja danych instruktora, np. dodanie nowego uprawnienia.
  + Usunięcie rezerwacji z konkretnego placu.

## Wnioski:

Model nierelacyjny oferuje większą elastyczność i skalowalność dzięki zagnieżdżaniu danych oraz referencjom, co ułatwia przechowywanie powiązanych informacji i umożliwia efektywne skalowanie bazy danych. Zagnieżdżanie danych w jednym dokumencie pozwala na optymalizację operacji CRUD, eliminując konieczność wielokrotnych zapytań. Struktura danych jest prostsza i bardziej intuicyjna, co ułatwia zarządzanie informacjami. Model nierelacyjny kontroluje redundancję danych, przechowując je w osobnych kolekcjach, a integralność danych jest zachowywana dzięki referencjom między dokumentami.

Dodatkowo, odpowiednie indeksowanie, jak np. na polu daty w kolekcji Jazdy, poprawia wydajność zapytań. Proces transformacji danych z modelu relacyjnego do nierelacyjnego, umożliwia lepsze dopasowanie bazy do wymagań współczesnych aplikacji. Przechowywanie danych w jednym dokumencie w ramach kolekcji zmniejsza ryzyko niespójnych danych.