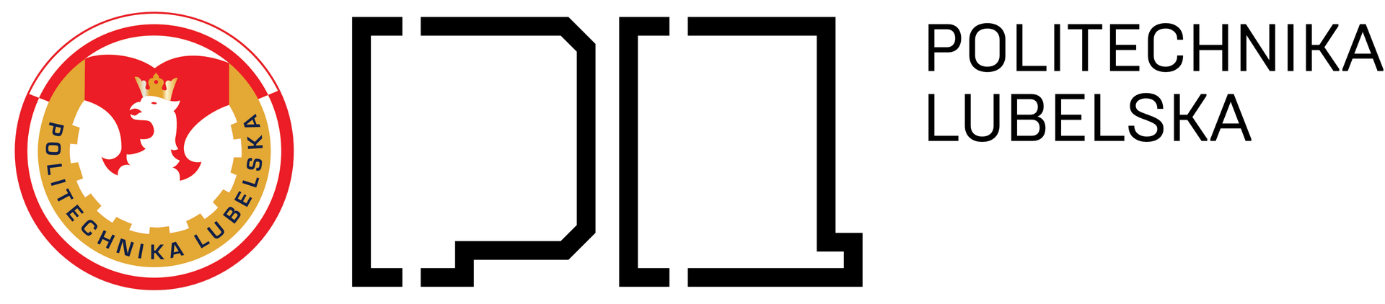
**Politechnika Lubelska**

**Wydział Elektrotechniki i Informatyki**

**Katedra Informatyki**



**Temat projektu:**

**System zamówień dla pizzerii**

**Laboratorium:** Zarządzanie bazami SQL i NoSQL

**Grupa:** IO 6.3

**Termin zajęć:** Poniedziałek 10:30

**Autorzy projektu:**

1. Jakub Dudek - 95391

**Tryb studiów**: stacjonarne

**Rok**: III

Spis treści

[Opis 3](#_Toc138023400)

[Baza MySQL 4](#_Toc138023401)

[Diagram ERD 4](#_Toc138023402)

[Tabele 5](#_Toc138023403)

[Relacje 9](#_Toc138023404)

[Zapytania 10](#_Toc138023405)

[DDL (Data Definition Language) 10](#_Toc138023406)

[DML (Data Manipulation Language) 12](#_Toc138023407)

[DQL (Data Query Language) 16](#_Toc138023408)

[DCL (Data Control Language) 17](#_Toc138023409)

[Wnioski 17](#_Toc138023410)

[MongoDB 18](#_Toc138023411)

[Kolekcje 18](#_Toc138023412)

[Zapytania 18](#_Toc138023413)

[Insert 18](#_Toc138023414)

[Update 23](#_Toc138023415)

[Aggregate 25](#_Toc138023416)

[Delete 26](#_Toc138023417)

[Find 27](#_Toc138023418)

[Create user 27](#_Toc138023419)

[Wnioski 28](#_Toc138023420)

# Opis

Baza danych dla pizzerii została stworzona w celu umożliwienia skutecznego zarządzania procesem zamówień i dostaw, zapewnienia kontroli nad dostępnymi rodzajami pizzy oraz składnikami, ułatwienia gromadzenie danych kontaktowych klientów oraz udostępnienia narzędzi administracyjnych dla nadzoru nad systemem. Pizzeria będzie obsługiwać zarówno dostawy, jak i zamówienia na miejscu.

System ten będzie wykorzystywany przez różne osoby, takie jak klienci, personel obsługujący zamówienia, dostawcy, pracownicy kuchni oraz administratorzy. Klienci będą mieli możliwość tworzenia i edytowania swoich profili, przeglądania menu, składania zamówień i śledzenia ich statusu. Personel obsługujący zamówienia będzie korzystał z systemu w celu przyjmowania zamówień, monitorowania ich statusu i zarządzania dostawami. Dostawcy będą mogli przeglądać zamówienia przypisane do nich oraz aktualizować informacje o dostawach. Pracownicy kuchni będą korzystali z systemu do przygotowywania zamówień. Administratorzy będą mieli dostęp do wszelakich danych i narzędzi, co pozwali im na analizę np. jakie pizzy sprzedają się najlepiej lub jakie składniki najczęściej wybierają klienci.

W bazie danych pizzerii będą przechowywane informacje o użytkownikach, którzy mają dostęp do systemu. Użytkownicy mają różne poziomy uprawnień, co umożliwia im wykonywanie odpowiednich operacji w systemie. Są także gromadzone dane o zamówieniach wraz z adresem dostawy, dostępne składniki i produkty oraz pizze stworzone przez klientów.

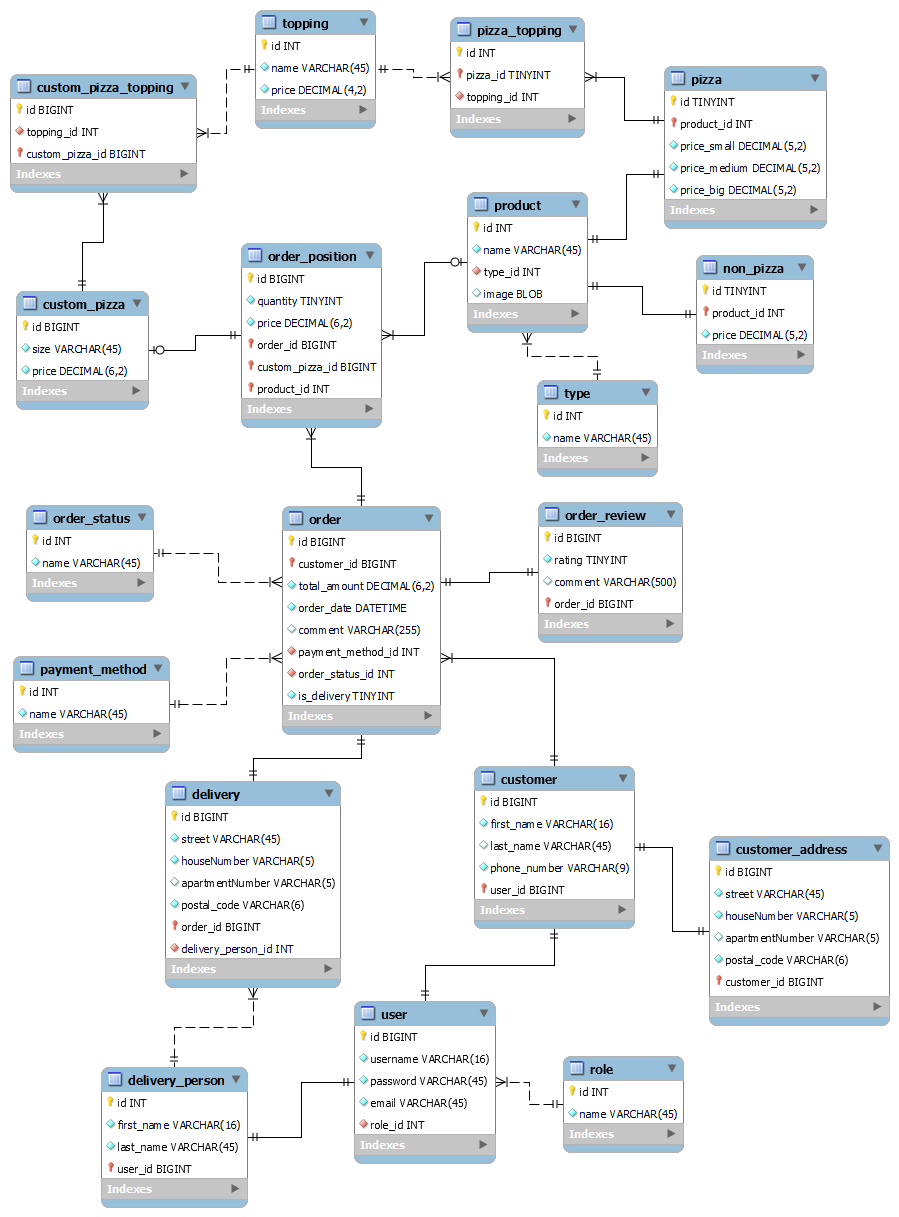
Zamówienia są kluczowym elementem bazy danych pizzerii. Informacje dotyczące zamówienia obejmują identyfikator klienta składającego zamówienie, datę zamówienia, kwotę do zapłaty, stan zamówienia (np. przyjęte, w trakcie realizacji, dostarczone), metodę płatności (np. przelew, blik, gotówka, karta) oraz czy jest to zamówienie z dostawą czy na miejscu. Dodatkowo przy zamówieniach z dostawą zawierają informacje o adresie dostawy oraz dane dostawcy. Przy zamówieniach online jest także możliwość dodania komentarza i wystawienia opinii po odebraniu zamówienia.

Menu składa się z różnych produktów oferowanych przez pizzerię. Każdy produkt ma swoją nazwę i cenę oraz może mieć także obrazek. Dzięki temu użytkownicy mogą przeglądać menu i składać zamówienia na wybrane produkty. Przy produktach, które nie są pizzą (np. napoje, sosy) jest tylko jedna cena, natomiast przy gotowych pizzach są trzy ceny zależnie od wielkości. Klienci mają także możliwość tworzenia własnej pizzy, wybierając dostępne składniki. Cena pizzy zależy od wybranej wielkości i ilości składników.

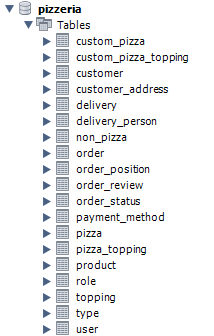
Baza danych pizzerii będzie wykorzystywała klucze główne i klucze obce w celu zapewnienia integralności danych oraz efektywności zapytań. Na przykład, klucz obcy jest wykorzystywany do powiązania zamówień z klientami, a także do powiązania zamówień z produktami w zamówieniach wielopozycyjnych.

# Baza MySQL

## Diagram ERD



## Tabele



* user – tabela przechowująca informacje o użytkownikach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| username | VARCHAR(16) | NOT NULL | UNIQUE |
| password | VARCHAR(45) | NOT NULL | NOT UNIQUE |
| email | VARCHAR(45) | NOT NULL | UNIQUE |
| role\_id (FK) | INT | NOT NULL | NOT UNIQUE |

* delivery\_person – tabela przechowująca informacje o dostawcach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| first\_name | VARCHAR(16) | NOT NULL | NOT UNIQUE |
| last\_name | VARCHAR(45) | NOT NULL | NOT UNIQUE |
| user\_id (FK) | BIGINT | NOT NULL | UNIQUE |

* role – tabela przechowująca informacje o dostępnych rolach użytkownika

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| name | VARCHAR(45) | NOT NULL | UNIQUE |

* customer – tabela przechowująca informacje o klientach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| first\_name | VARCHAR(16) | NOT NULL | NOT UNIQUE |
| last\_name | VARCHAR(45) | NULL | NOT UNIQUE |
| phone\_number | VARCHAR(9) | NOT NULL | NOT UNIQUE |
| user\_id (FK) | BIGINT | NOT NULL | UNIQUE |

* customer\_address – tabela przechowująca informacje o adresach klientów

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| street | VARCHAR(45) | NOT NULL | NOT UNIQUE |
| houseNumber | VARCHAR(5) | NOT NULL | NOT UNIQUE |
| apartmentNumber | VARCHAR(5) | NULL | NOT UNIQUE |
| postal\_code | VARCHAR(6) | NOT NULL | NOT UNIQUE |
| customer\_id | BIGINT | NOT NULL | UNIQUE |

* order – tabela przechowująca informacje o zamówieniach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| customer\_id | BIGINT | NOT NULL | NOT UNIQUE |
| total\_amount | DECIMAL(6,2) | NOT NULL | NOT UNIQUE |
| order\_date | DATETIME | NOT NULL | NOT UNIQUE |
| comment | VARCHAR(255) | NULL | NOT UNIQUE |
| payment\_method\_id | INT | NOT NULL | NOT UNIQUE |
| order\_status\_id | INT | NOT NULL | NOT UNIQUE |
| is\_delivery | TINYINT | NOT NULL | NOT UNIQUE |

* delivery – tabela przechowująca informacje o dostawie zamówienia

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| street | VARCHAR(45) | NOT NULL | NOT UNIQUE |
| houseNumber | VARCHAR(5) | NOT NULL | NOT UNIQUE |
| apartmentNumber | VARCHAR(5) | NULL | NOT UNIQUE |
| postal\_code | VARCHAR(6) | NOT NULL | NOT UNIQUE |
| order\_id (FK) | BIGINT | NOT NULL | UNIQUE |
| delivery\_person\_id (FK) | INT | NOT NULL | NOT UNIQUE |

* order\_status – tabela przechowująca informacje o dostępnych statusach zamówienia

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| name | VARCHAR(45) | NOT NULL | UNIQUE |

* payment\_method – tabela przechowująca informacje o dostępnych metodach płatności

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| name | VARCHAR(45) | NOT NULL | UNIQUE |

* order\_review – tabela przechowująca informacje o wystawionych opiniach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| rating | TINYINT | NOT NULL | NOT UNIQUE |
| comment | VARCHAR(500) | NULL | NOT UNIQUE |
| order\_id (FK) | BIGINT | NOT NULL | UNIQUE |

* order\_position – tabela przechowująca informacje o pozycjach zamówienia

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| quantity | TINYINT | NOT NULL | NOT UNIQUE |
| price | DECIMAL(6,2) | NOT NULL | NOT UNIQUE |
| order\_id (FK) | BIGINT | NOT NULL | NOT UNIQUE |
| custom\_pizza\_id (FK) | BIGINT | NULL | UNIQUE |
| product\_id (FK) | INT | NULL | NOT UNIQUE |

* product – tabela przechowująca informacje o dostępnych produktach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| name | VARCHAR(45) | NOT NULL | UNIQUE |
| type\_id (FK) | TINYINT | NOT NULL | NOT UNIQUE |
| image | BLOB | NULL | NOT UNIQUE |

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | TINYINT | NOT NULL | UNIQUE |
| product\_id (FK) | INT | NOT NULL | UNIQUE |
| price | DECIMAL(5,2) | NOT NULL | NOT UNIQUE |

* non\_pizza – tabela przechowująca informacje o dostępnych produktach, które nie są pizzami
* pizza – tabela przechowująca informacje o dostępnych rodzajach pizzy

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | TINYINT | NOT NULL | UNIQUE |
| product\_id (FK) | INT | NOT NULL | UNIQUE |
| price\_small | DECIMAL(5,2) | NOT NULL | NOT UNIQUE |
| price\_medium | DECIMAL(5,2) | NOT NULL | NOT UNIQUE |
| price\_big | DECIMAL(5,2) | NOT NULL | NOT UNIQUE |

* custom\_pizza – tabela przechowująca informacje o pizzach stworzonych przez klientów

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| size | INT | NOT NULL | NOT UNIQUE |
| price | DECIMAL(6,2) | NOT NULL | NOT UNIQUE |

* topping – tabela przechowująca informacje o dostępnych składnikach

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| name | VARCHAR(45) | NOT NULL | UNIQUE |
| price | DECIMAL(4,2) | NOT NULL | NOT UNIQUE |

* pizza\_topping – tabela przechowująca informacje o składnikach danych pizz

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | INT | NOT NULL | UNIQUE |
| pizza\_id (FK) | TINYINT | NOT NULL | NOT UNIQUE |
| topping\_id (FK) | INT | NOT NULL | NOT UNIQUE |

* custom\_pizza\_topping – tabela przechowująca informacje o składnikach użytych do stworzenia danej pizzy przez klientów

|  |  |  |  |
| --- | --- | --- | --- |
| Nazwa pola | Typ pola | NULL/NOT NULL | UNIQUE/NOT UNIQUE |
| id (PK) | BIGINT | NOT NULL | UNIQUE |
| topping\_id (FK) | INT | NOT NULL | NOT UNIQUE |
| custom\_pizza\_id (FK) | BIGINT | NOT NULL | NOT UNIQUE |

## Relacje

* jeden do jednego:
  + user i customer – jeden użytkownik to jeden klient
  + user i delivery\_person – jeden użytkownik to jeden dostawca
  + customer i customer\_address – jeden klient ma jeden adres, a dany adres jest tylko do jednego klienta
  + order i delivery – jedno zamówienie może mieć tylko jedną dostawę, a jedna dostawa jest do jednego zamówienia
  + order i order\_review – jedno zamówienie może mieć tylko jedną opinie, a jedna opinia jest tylko do jednego zamówienia
  + product i pizza – jeden produkt to jedna pizza
  + product i non\_pizza – jeden produkt to jeden produkt, który nie jest pizzą
  + order\_postition i custom\_pizza – jedna pozycja zamówienia może mieć tylko jedną pizzę stworzoną przez klienta, a jedna pizza należy do jednej pozycji zamówienia
* jeden do wielu:
  + user i role - jeden użytkownik ma jedną rolę, a jedna rolę może mieć wiele użytkowników
  + customer i order – jeden klient może mieć wiele zamówień, ale jedno zamówienie jest tylko jednego klienta
  + order i payment\_method – jedno zamówienie ma tylko jeden sposób płatności, ale jeden sposób płatności może być w wielu zamówieniach
  + order i order\_status – jedno zamówienie ma jeden status zamówienia, ale jeden status może być w wielu zamówieniach
  + order i order\_position – jedno zamówienie może mieć wiele pozycji, ale jedna pozycja może mieć być tylko w jednym zamówieniu
  + order\_position i product – jedna pozycja zamówienia to jeden produkt z menu, a jeden produkt z menu może być w wielu pozycjach
  + product i type – jeden produkt ma jeden typ, a jeden typ może mieć wiele produktów
  + delivery i delivery\_person – jedna dostawa ma jednego dostawcę, a jeden dostawca może mieć wiele dostaw
* wiele do wielu:
  + custom\_pizza i topping – jedna pizza stworzona przez klienta może składać się z wielu dodatków, a jeden dodatek może być użyty w wielu pizzach. Tabela łącząca custom\_pizza\_topping.
  + pizza i topping – jedna pizza może składać się z wielu dodatków, a jeden dodatek może być użyty w wielu pizzach. Tabela łącząca pizza\_topping.
  + order i product – jedno zamówienie może składać się z wielu produktów, a jeden produkt może być w wielu zamówieniach. Tabela łącząca order\_position.

## Zapytania

### DDL (Data Definition Language)

Tabela user

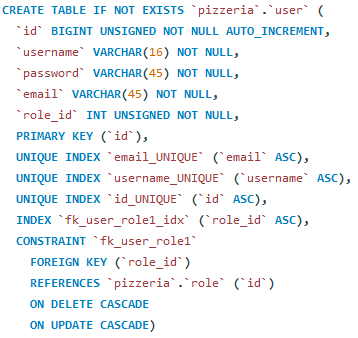




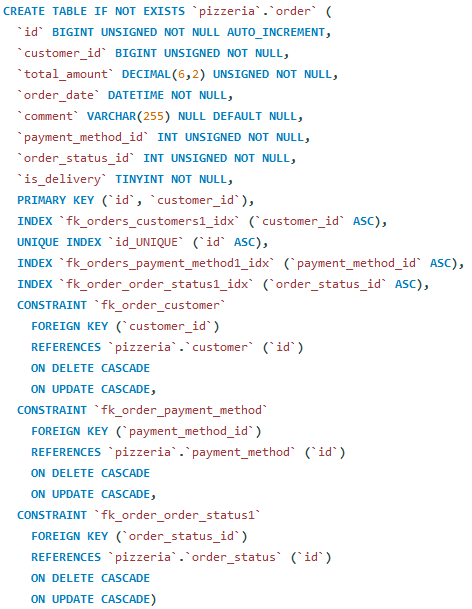








Tabela order



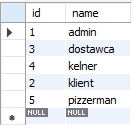
Opis pól:

* id: Pole, służące jako unikalny identyfikator zamówienia. Ma automatyczne inkrementowanie, co oznacza, że przy dodawaniu nowego zamówienia wartość tego pola będzie automatycznie zwiększana.
* customer\_id: Pole, które przechowuje identyfikator klienta powiązanego z zamówieniem. Jest to klucz obcy, który odnosi się do tabeli „customer”.
* total\_amount: Pole, które przechowuje całkowitą kwotę zamówienia. Ma określony format decimalny z maksymalnie 6 cyfr, z czego 2 cyfry po przecinku.
* order\_date: Pole, które przechowuje datę i czas złożenia zamówienia.
* comment: Pole typu, które przechowuje komentarz lub uwagi dotyczące zamówienia. Może przyjmować wartość NULL lub być domyślnie ustawione na NULL.
* payment\_method\_id: Pole typu, które przechowuje identyfikator metody płatności powiązanej z zamówieniem. Jest to klucz obcy, który odnosi się do tabeli „payment\_method”.
* order\_status\_id: Pole, które przechowuje identyfikator statusu zamówienia. Jest to klucz obcy, który odnosi się do tabeli „order\_status”.
* is\_delivery: Pole, które określa, czy zamówienie jest dostarczane (1) czy na miejscu (0).
* PRIMARY KEY: Określa, że kombinacja pól "id" i "customer\_id" jest kluczem głównym tabeli, co zapewnia unikalność rekordów.
* INDEX: Definiuje indeksy dla niektórych pól tabeli, takie jak "customer\_id", "payment\_method\_id" i "order\_status\_id". Indeksy przyspieszają wyszukiwanie danych w tych kolumnach.
* FOREIGN KEY: Określa klucze obce dla pól "customer\_id", "payment\_method\_id" i "order\_status\_id", które odnoszą się do odpowiednich tabel w bazie danych. Użycie ON DELETE CASCADE i ON UPDATE CASCADE umożliwia automatyczne usuwanie lub aktualizację powiązanych rekordów w innych tabelach, zapewniając spójność danych.

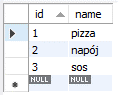
### DML (Data Manipulation Language)

Zapytania, które dodają dane testowe.

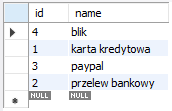




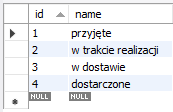


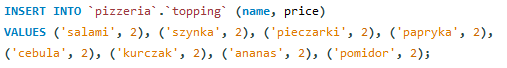


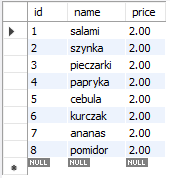


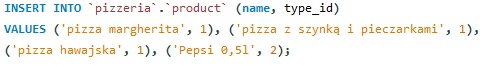


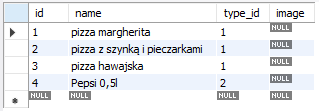




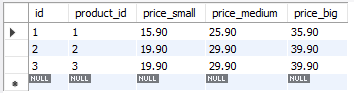




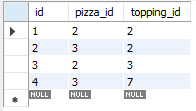


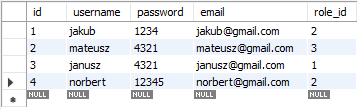


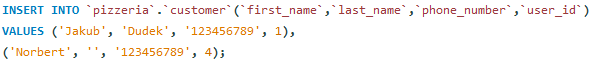


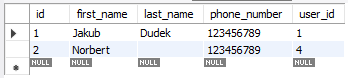










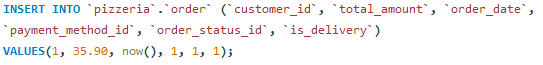










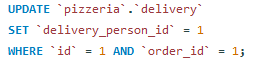






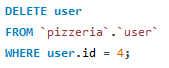


Zapytanie, które przypisuje dostawcę do danego zamówienia.

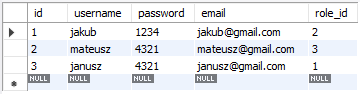




Zapytanie, które usuwanie usera z tabeli user, a poprzez „on delete cascade” również z tabeli customer.

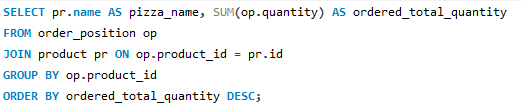




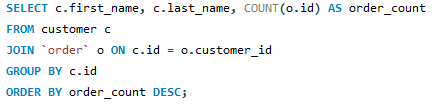


### DQL (Data Query Language)

Zapytanie, które wykonuje agregację danych z tabeli order\_position i product w celu uzyskania sumy zamówionych ilości dla każdej pizzy. W rezultacie otrzymuje się listę pizz wraz z ich zamówionymi ilościami, posortowanymi malejąco według ilości.

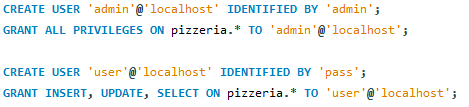


Zapytanie, które wykonuje agregację danych z tabeli customer i order w celu uzyskania liczby zamówień dla każdego klienta. W rezultacie otrzymuje się listę klientów wraz z liczbą zamówień, posortowaną malejąco według liczby zamówień.



### DCL (Data Control Language)

Zapytania, które tworzą dwóch użytkowników. Admin ma wszystkie uprawnienia, a user insert, update i select.

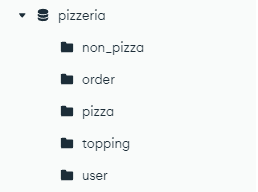


## Wnioski

Stworzona baza danych zawiera tabele reprezentujące podstawowe encje związane z zamówieniami w pizzerii. Umożliwia efektywne zarządzanie zamówieniami, klientami, produktami i innymi aspektami działalności pizzerii. Zdefiniowano w niej odpowiednie klucze główne i indeksy, aby zapewnić unikalność danych oraz umożliwić efektywne wyszukiwanie i łączenie danych. W bazie danych zastosowano relacje pomiędzy tabelami za pomocą kluczy obcych, aby zapewnić integralność danych i zachować spójność między różnymi encjami. Baza danych została skonstruowana w sposób umożliwiający łatwe rozszerzanie i modyfikowanie systemu zamówień dla pizzerii w przyszłości.

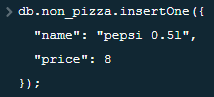
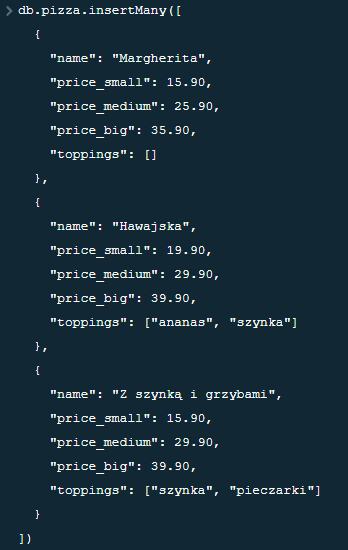
# MongoDB

## Kolekcje

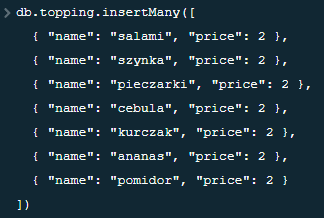


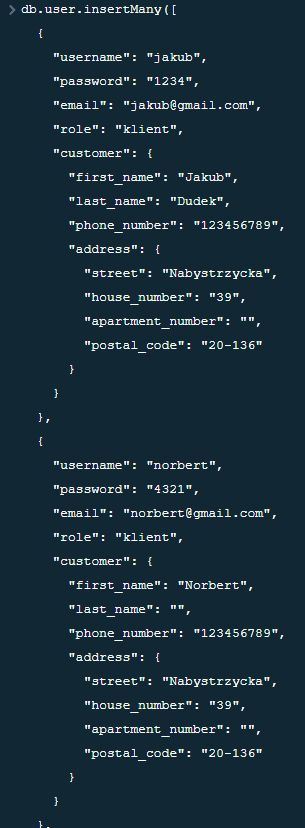
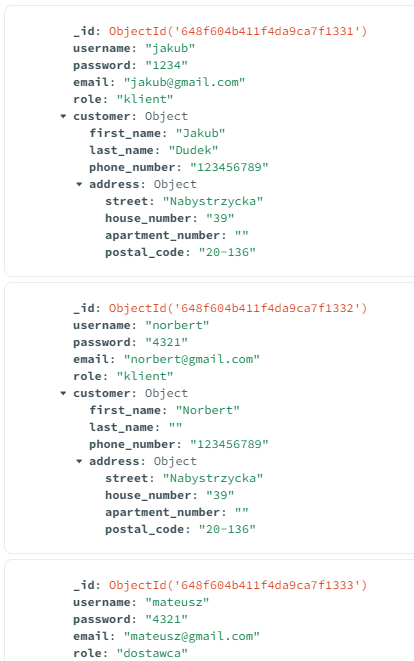
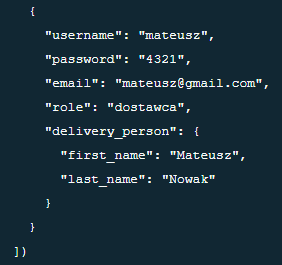
## Zapytania

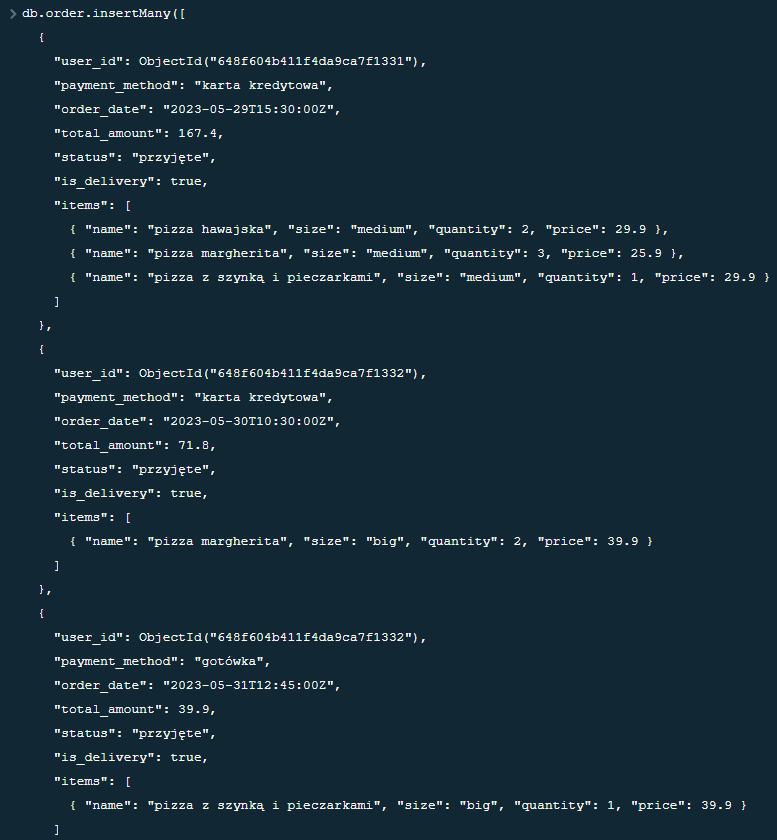
### Insert

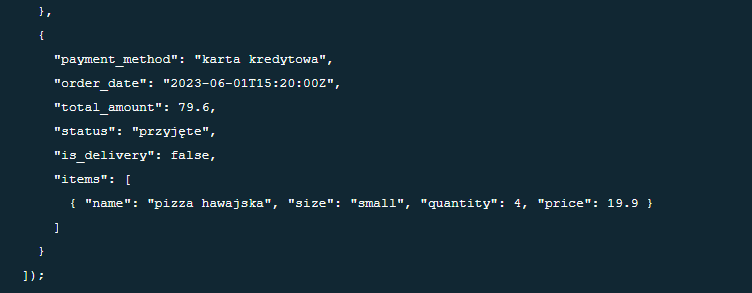
Zapytania, które dodają dane testowe.

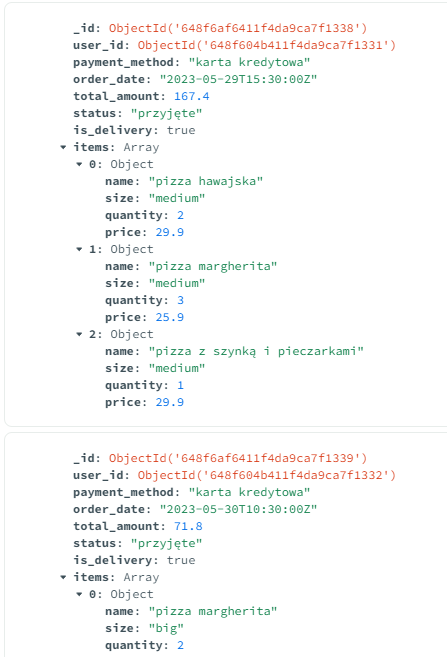








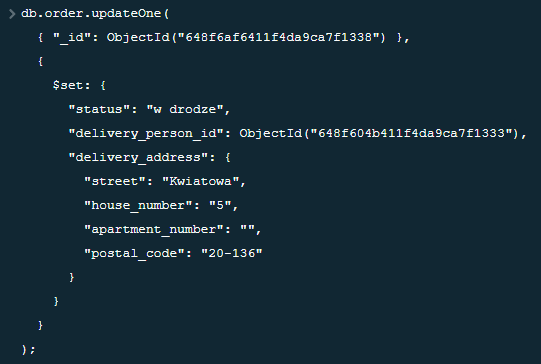






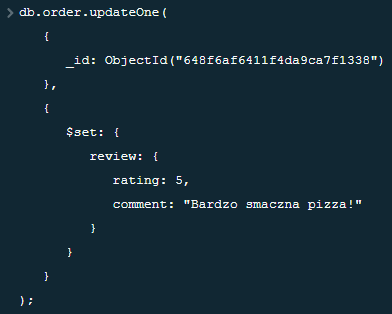
### Update

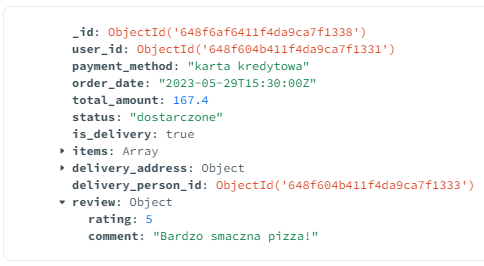
Zapytanie, które zmienia status zamówienia na „w drodze” oraz dodaje dostawcę i adres dostawy.



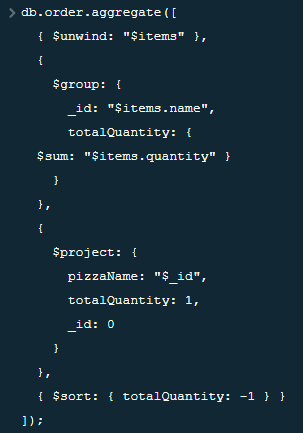
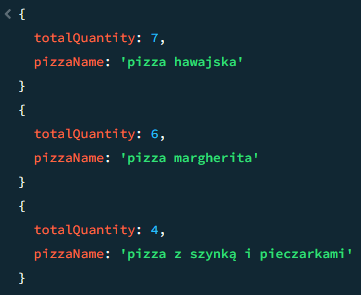


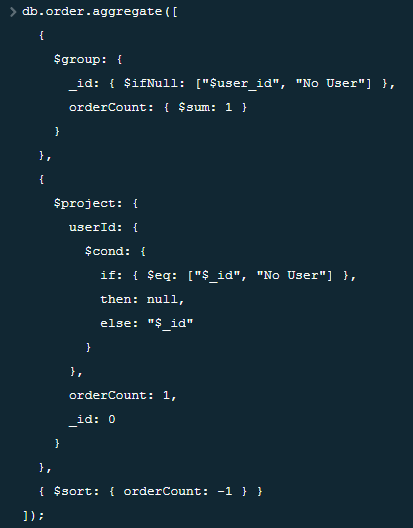
Zapytanie, które dodaje opinie do zamówienia.





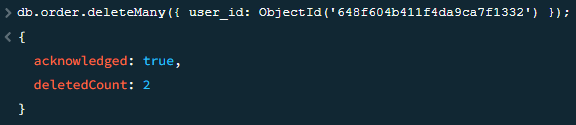
### Aggregate

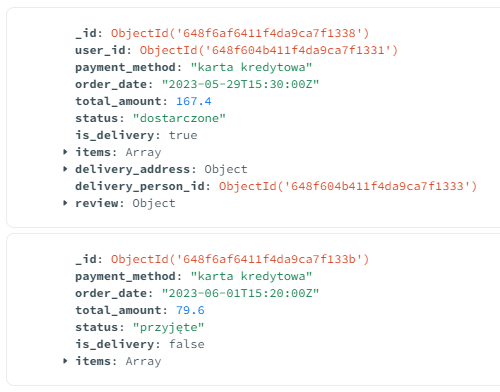
Zapytanie, które zwraca listę nazw pizz i sumaryczną ilość zamówionych sztuk, posortowaną malejąco według ilości zamówień.

Zapytanie, które zwraca listę user\_id i sumaryczną ilość zamówień, posortowaną malejąco według liczby zamówień. Wartość null dla userId oznacza zamówienia na miejscu.

### Delete

Zapytanie, które usuwa wszystkie zamówienia usera o podanym id.



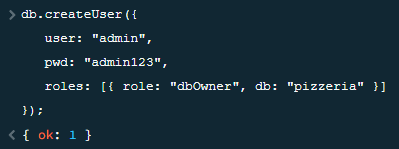


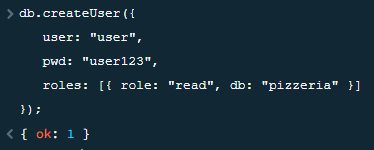
### Find

Zapytanie, które wyświetla zamówienia bez user\_id.

### Create user

Zapytania tworzące dwóch użytkowników. Admin dostał rolę „dbOwner”, która daje pełną kontrolę nad bazą danych. Natomiast user dostał rolę „read”, która daje tylko możliwość odczytu danych z kolekcji.





## Wnioski

Stworzona baza danych zawiera pięć głównych kolekcji: pizza, order, topping, user i non\_pizza. W bazie danych SQL schemat jest z góry określony, a każda tabela ma określone kolumny i typy danych, natomiast w MongoDB schemat jest elastyczny, co oznacza, że różne dokumenty w tej samej kolekcji mogą mieć różne struktury i pola. Można dodawać i usuwać pola w dowolnym momencie bez konieczności dostosowywania schematu. Brak tutaj relacji czy kluczy obcych. Zamiast tego można tworzyć odniesienia (referencje) między dokumentami, które można przechowywać jako pola w dokumentach. Nierelacyjna baza danych jest bardziej skalowalna w porównaniu do tradycyjnych baz SQL.