Politechnika Lubelska

Wydział Elektrotechniki i Informatyki

Katedra Informatyki

POLITECHNIKA
LUBELSKA

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

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.

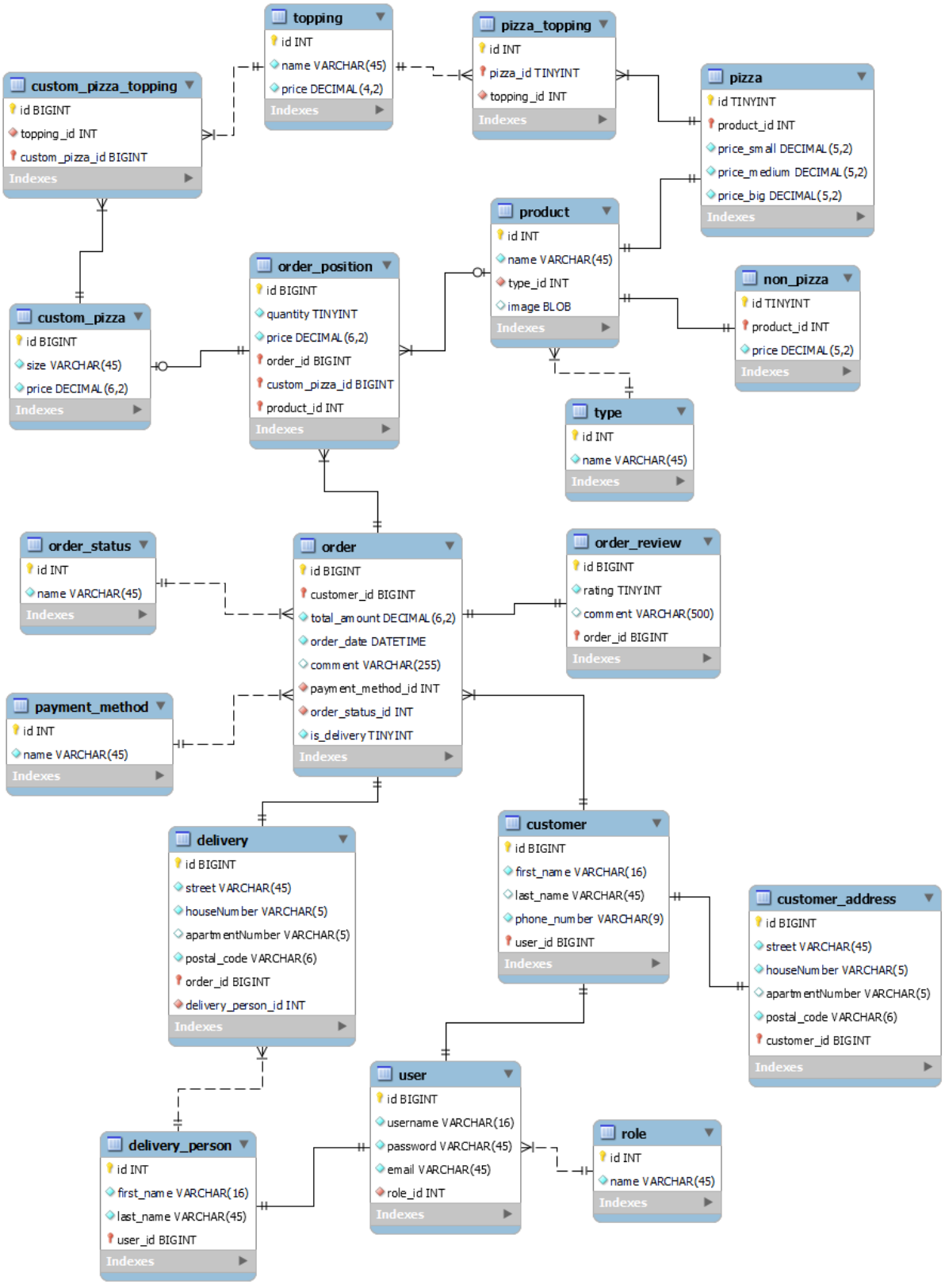
Zamówienia są kluczowym elementem bazy danych pizzerii. Informacje dotyczące zamówienia obejmują identyfikator klienta składającego zamówienie, datę zamówienia, kwotę do zapłaty, stan zamówienia (np. przyjęte, w trakcie realizacji, dostarczone), metodę płatności (np. przelew, blik, gotówka, karta) oraz 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.

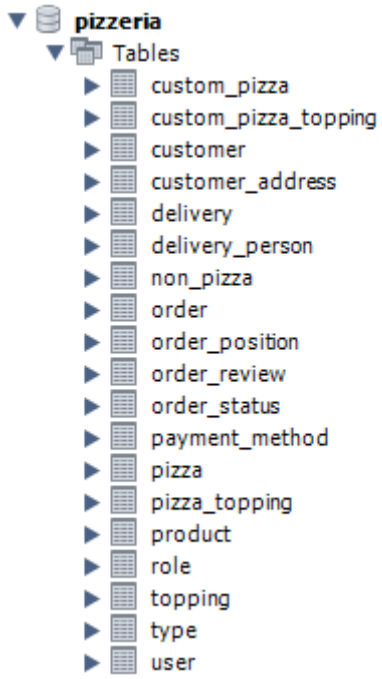
Baza danych pizzerii będzie wykorzystywała klucze główne i klucze obce w celu zapewnienia integralności danych oraz efektywności zapytań. Na przykład, klucz obcy jest wykorzystywany do powiązania zamówień z klientami, a także do powiązania zamówień z produktami w zamówieniach wielopozycyjnych.

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

- non_pizza – tabela przechowująca informacje o dostępnych produktach, które nie są pizzami

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

- 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

```
CREATE TABLE IF NOT EXISTS `pizzeria`.`user` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(16) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `role_id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_user_role1_idx` (`role_id` ASC),
  CONSTRAINT `fk_user_role1`
    FOREIGN KEY (`role_id`)
    REFERENCES `pizzeria`.`role` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
```

	id	username	password	email	role_id
*	NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE pizzeria.user
ADD age TINYINT;
```

	id	username	password	email	role_id	age
*	NULL	NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE pizzeria.user
DROP COLUMN age;
```

	id	username	password	email	role_id
*	NULL	NULL	NULL	NULL	NULL

Tabela order

```
CREATE TABLE IF NOT EXISTS `pizzeria`.`order` (
  `id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `customer_id` BIGINT UNSIGNED NOT NULL,
  `total_amount` DECIMAL(6,2) UNSIGNED NOT NULL,
  `order_date` DATETIME NOT NULL,
  `comment` VARCHAR(255) NULL DEFAULT NULL,
  `payment_method_id` INT UNSIGNED NOT NULL,
  `order_status_id` INT UNSIGNED NOT NULL,
  `is_delivery` TINYINT NOT NULL,
  PRIMARY KEY (`id`, `customer_id`),
  INDEX `fk_orders_customers1_idx` (`customer_id` ASC),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC),
  INDEX `fk_orders_payment_method1_idx` (`payment_method_id` ASC),
  INDEX `fk_order_order_status1_idx` (`order_status_id` ASC),
  CONSTRAINT `fk_order_customer`
    FOREIGN KEY (`customer_id`)
    REFERENCES `pizzeria`.`customer` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_order_payment_method`
    FOREIGN KEY (`payment_method_id`)
    REFERENCES `pizzeria`.`payment_method` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_order_order_status1`
    FOREIGN KEY (`order_status_id`)
    REFERENCES `pizzeria`.`order_status` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
```

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.

```
INSERT INTO `pizzeria`.`role` (name)
VALUES ('admin'), ('klient'), ('dostawca'), ('kelner'), ('pizzerman');
```

id	name
1	admin
3	dostawca
4	kelner
2	klient
5	pizzerman
NULL	NULL

```
INSERT INTO `pizzeria`.`type` (name)
VALUES ('pizza'), ('napój'), ('sos');
```

id	name
1	pizza
2	napój
3	sos
NULL	NULL

```
INSERT INTO `pizzeria`.`payment_method` (name)
VALUES ('karta kredytowa'), ('przelew bankowy'), ('paypal'), ('blik');
```

id	name
4	blik
1	karta kredytowa
3	paypal
2	przelew bankowy
NULL	NULL

```
INSERT INTO `pizzeria`.`order_status` (name)
VALUES ('przyjęte'), ('w trakcie realizacji'), ('w dostawie'), ('dostarczone');
```

id	name
1	przyjęte
2	w trakcie realizacji
3	w dostawie
4	dostarczone
NULL	NULL

```
INSERT INTO `pizzeria`.`topping` (name, price)
VALUES ('salami', 2), ('szynka', 2), ('pieczarki', 2), ('papryka', 2),
('cebula', 2), ('kurczak', 2), ('ananas', 2), ('pomidor', 2);
```

id	name	price
1	salami	2.00
2	szynka	2.00
3	pieczarki	2.00
4	papryka	2.00
5	cebula	2.00
6	kurczak	2.00
7	ananas	2.00
8	pomidor	2.00
NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`product` (name, type_id)
VALUES ('pizza margherita', 1), ('pizza z szynką i pieczarkami', 1),
('pizza hawajska', 1), ('Pepsi 0,5l', 2);
```

id	name	type_id	image
1	pizza margherita	1	NULL
2	pizza z szynką i pieczarkami	1	NULL
3	pizza hawajska	1	NULL
4	Pepsi 0,5l	2	NULL
NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`pizza` (product_id, price_small, price_medium, price_big)
VALUES (1, 15.90, 25.90, 35.90), (2, 19.90, 29.90, 39.90), (3, 19.90, 29.90, 39.90);
```

id	product_id	price_small	price_medium	price_big
1	1	15.90	25.90	35.90
2	2	19.90	29.90	39.90
3	3	19.90	29.90	39.90
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`pizza_topping` (pizza_id, topping_id)
VALUES (2, 2), (2, 3), (3, 2), (3, 7);
```

id	pizza_id	topping_id
1	2	2
2	3	2
3	2	3
4	3	7
NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`user`(`username`,`password`,`email`,`role_id`)
VALUES ('jakub', '1234', 'jakub@gmail.com', 2),
('mateusz', '4321', 'mateusz@gmail.com', 3),
('janusz', '4321', 'janusz@gmail.com', 1),
('norbert', '12345', 'norbert@gmail.com', 2);
```

id	username	password	email	role_id
1	jakub	1234	jakub@gmail.com	2
2	mateusz	4321	mateusz@gmail.com	3
3	janusz	4321	janusz@gmail.com	1
4	norbert	12345	norbert@gmail.com	2
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`customer`(`first_name`,`last_name`,`phone_number`,`user_id`)
VALUES ('Jakub', 'Dudek', '123456789', 1),
('Norbert', '', '123456789', 4);
```

id	first_name	last_name	phone_number	user_id
1	Jakub	Dudek	123456789	1
2	Norbert		123456789	4
NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`delivery_person`(`first_name`,`last_name`,`user_id`)
VALUES ('Mateusz', 'Nowak', 2);
```

id	first_name	last_name	user_id
1	Mateusz	Nowak	2
NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`customer_address` (street, houseNumber, postal_code, customer_id)
VALUES('Nadbystrzycka', '39', '20-136', 1);
```

id	street	houseNumber	apartmentNumber	postal_code	customer_id
1	Nadbystrzycka	39	NULL	20-136	1
NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`order` (`customer_id`, `total_amount`, `order_date`,
`payment_method_id`, `order_status_id`, `is_delivery`)
VALUES(1, 35.90, now(), 1, 1, 1);
```

id	customer_id	total_amount	order_date	comment	payment_method_id	order_status_id	is_delivery
1	1	35.90	2023-06-11 07:57:17	NULL	1	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`order_position` (`quantity`, `price`, `order_id`, `product_id`)
VALUES(1, 35.90, 1, 1);
```

id	quantity	price	order_id	custom_pizza_id	product_id
1	1	35.90	1	NULL	1
NULL	NULL	NULL	NULL	NULL	NULL

```
INSERT INTO `pizzeria`.`delivery` (`street`, `houseNumber`, `postal_code`, `order_id`)
VALUES('Nadbystrzycka', '39', '20-136', 1);
```

id	street	houseNumber	apartmentNumber	postal_code	order_id	delivery_person_id
1	Nadbystrzycka	39	NULL	20-136	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Zapytanie, które przypisuje dostawcę do danego zamówienia.

```
UPDATE `pizzeria`.`delivery`
SET `delivery_person_id` = 1
WHERE `id` = 1 AND `order_id` = 1;
```

id	street	houseNumber	apartmentNumber	postal_code	order_id	delivery_person_id
1	Nadbystrzycka	39	NULL	20-136	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

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

```
DELETE user
FROM `pizzeria`.`user`
WHERE user.id = 4;
```

id	first_name	last_name	phone_number	user_id
1	Jakub	Dudek	123456789	1
NULL	NULL	NULL	NULL	NULL

id	username	password	email	role_id
1	jakub	1234	jakub@gmail.com	2
2	mateusz	4321	mateusz@gmail.com	3
3	janusz	4321	janusz@gmail.com	1
NULL	NULL	NULL	NULL	NULL

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.

```
SELECT pr.name AS pizza_name, SUM(op.quantity) AS ordered_total_quantity
FROM order_position op
JOIN product pr ON op.product_id = pr.id
GROUP BY op.product_id
ORDER BY ordered_total_quantity DESC;
```

	pizza_name	ordered_total_quantity
▶	pizza margherita	3
	pizza hawajska	2

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.

```
SELECT c.first_name, c.last_name, COUNT(o.id) AS order_count
FROM customer c
JOIN `order` o ON c.id = o.customer_id
GROUP BY c.id
ORDER BY order_count DESC;
```

	first_name	last_name	order_count
▶	Norbert	NULL	3
	Jakub	Dudek	1

DCL (Data Control Language)

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

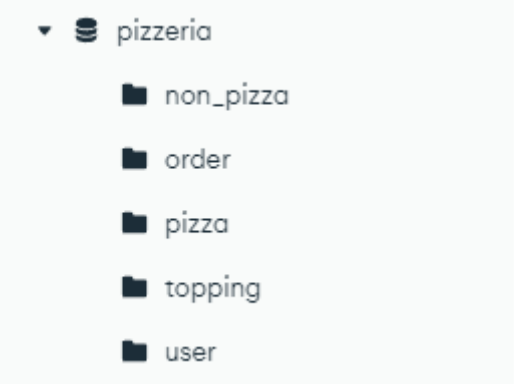
```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';
GRANT ALL PRIVILEGES ON pizzeria.* TO 'admin'@'localhost';

CREATE USER 'user'@'localhost' IDENTIFIED BY 'pass';
GRANT INSERT, UPDATE, SELECT ON pizzeria.* TO 'user'@'localhost';
```

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.

```
> db.non_pizza.insertOne({
    "name": "pepsi 0.5l",
    "price": 8
  });
```

```
_id: ObjectId('648f5f82411f4da9ca7f1330')
name: "pepsi 0.5l"
price: 8
```

```
> db.pizza.insertMany([
  {
    "name": "Margherita",
    "price_small": 15.90,
    "price_medium": 25.90,
    "price_big": 35.90,
    "toppings": []
  },
  {
    "name": "Hawajska",
    "price_small": 19.90,
    "price_medium": 29.90,
    "price_big": 39.90,
    "toppings": ["ananas", "szynka"]
  },
  {
    "name": "Z szynką i grzybami",
    "price_small": 15.90,
    "price_medium": 29.90,
    "price_big": 39.90,
    "toppings": ["szynka", "pieczarki"]
  }
])
```

```
_id: ObjectId('648f5ef0411f4da9ca7f132d')
name: "Margherita"
price_small: 15.9
price_medium: 25.9
price_big: 35.9
▾ toppings: Array
```

```
_id: ObjectId('648f5ef0411f4da9ca7f132e')
name: "Hawajska"
price_small: 19.9
price_medium: 29.9
price_big: 39.9
▾ toppings: Array
    0: "ananas"
    1: "szynka"
```

```
_id: ObjectId('648f5ef0411f4da9ca7f132f')
name: "Z szynką i grzybami"
price_small: 15.9
price_medium: 29.9
price_big: 39.9
▾ toppings: Array
    0: "szynka"
    1: "pieczarki"
```

```
> db.topping.insertMany([
    { "name": "salami", "price": 2 },
    { "name": "szynka", "price": 2 },
    { "name": "pieczarki", "price": 2 },
    { "name": "cebula", "price": 2 },
    { "name": "kurczak", "price": 2 },
    { "name": "ananas", "price": 2 },
    { "name": "pomidor", "price": 2 }
  ])
```

```
_id: ObjectId('648f5d7c38488a7613b793b4')
name: "salami"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793b5')
name: "szynka"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793b6')
name: "pieczarki"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793b7')
name: "cebula"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793b8')
name: "kurczak"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793b9')
name: "ananas"
price: 2
```

```
_id: ObjectId('648f5d7c38488a7613b793ba')
name: "pomidor"
price: 2
```

```
> db.user.insertMany([
    {
      "username": "jakub",
      "password": "1234",
      "email": "jakub@gmail.com",
      "role": "klient",
      "customer": {
        "first_name": "Jakub",
        "last_name": "Dudek",
        "phone_number": "123456789",
        "address": {
          "street": "Nabystrzycka",
          "house_number": "39",
          "apartment_number": "",
          "postal_code": "20-136"
        }
      }
    },
    {
      "username": "norbert",
      "password": "4321",
      "email": "norbert@gmail.com",
      "role": "klient",
      "customer": {
        "first_name": "Norbert",
        "last_name": "",
        "phone_number": "123456789",
        "address": {
          "street": "Nabystrzycka",
          "house_number": "39",
          "apartment_number": "",
          "postal_code": "20-136"
        }
      }
    },
    {
      "username": "mateusz",
      "password": "4321",
      "email": "mateusz@gmail.com",
      "role": "dostawca",
      "delivery_person": {
        "first_name": "Mateusz",
        "last_name": "Nowak"
      }
    }
])
```

```
_id: ObjectId('648f604b411f4da9ca7f1331')
username: "jakub"
password: "1234"
email: "jakub@gmail.com"
role: "klient"
▾ customer: Object
    first_name: "Jakub"
    last_name: "Dudek"
    phone_number: "123456789"
  ▾ address: Object
      street: "Nabystrzycka"
      house_number: "39"
      apartment_number: ""
      postal_code: "20-136"
```

```
_id: ObjectId('648f604b411f4da9ca7f1332')
username: "norbert"
password: "4321"
email: "norbert@gmail.com"
role: "klient"
▾ customer: Object
    first_name: "Norbert"
    last_name: ""
    phone_number: "123456789"
  ▾ address: Object
      street: "Nabystrzycka"
      house_number: "39"
      apartment_number: ""
      postal_code: "20-136"
```

```
_id: ObjectId('648f604b411f4da9ca7f1333')
username: "mateusz"
password: "4321"
email: "mateusz@gmail.com"
role: "dostawca"
▾ delivery_person: Object
    first_name: "Mateusz"
    last_name: "Nowak"
```

```
> db.order.insertMany([
    {
      "user_id": ObjectId("648f604b411f4da9ca7f1331"),
      "payment_method": "karta kredytowa",
      "order_date": "2023-05-29T15:30:00Z",
      "total_amount": 167.4,
      "status": "przyjęte",
      "is_delivery": true,
      "items": [
        { "name": "pizza hawajska", "size": "medium", "quantity": 2, "price": 29.9 },
        { "name": "pizza margherita", "size": "medium", "quantity": 3, "price": 25.9 },
        { "name": "pizza z szynką i pieczarkami", "size": "medium", "quantity": 1, "price": 29.9 }
      ]
    },
    {
      "user_id": ObjectId("648f604b411f4da9ca7f1332"),
      "payment_method": "karta kredytowa",
      "order_date": "2023-05-30T10:30:00Z",
      "total_amount": 71.8,
      "status": "przyjęte",
      "is_delivery": true,
      "items": [
        { "name": "pizza margherita", "size": "big", "quantity": 2, "price": 39.9 }
      ]
    },
    {
      "user_id": ObjectId("648f604b411f4da9ca7f1332"),
      "payment_method": "gotówka",
      "order_date": "2023-05-31T12:45:00Z",
      "total_amount": 39.9,
      "status": "przyjęte",
      "is_delivery": true,
      "items": [
        { "name": "pizza z szynką i pieczarkami", "size": "big", "quantity": 1, "price": 39.9 }
      ]
    },
    {
      "payment_method": "karta kredytowa",
      "order_date": "2023-06-01T15:20:00Z",
      "total_amount": 79.6,
      "status": "przyjęte",
      "is_delivery": false,
      "items": [
        { "name": "pizza hawajska", "size": "small", "quantity": 4, "price": 19.9 }
      ]
    }
  ]);
```

```
_id: ObjectId('648f6af6411f4da9ca7f1338')
user_id: ObjectId('648f604b411f4da9ca7f1331')
payment_method: "karta kredytowa"
order_date: "2023-05-29T15:30:00Z"
total_amount: 167.4
status: "przyjęte"
is_delivery: true
items: Array
  0: Object
      name: "pizza hawajska"
      size: "medium"
      quantity: 2
      price: 29.9
  1: Object
      name: "pizza margherita"
      size: "medium"
      quantity: 3
      price: 25.9
  2: Object
      name: "pizza z szynką i pieczarkami"
      size: "medium"
      quantity: 1
      price: 29.9
```

```
_id: ObjectId('648f6af6411f4da9ca7f1339')
user_id: ObjectId('648f604b411f4da9ca7f1332')
payment_method: "karta kredytowa"
order_date: "2023-05-30T10:30:00Z"
total_amount: 71.8
status: "przyjęte"
is_delivery: true
items: Array
  0: Object
      name: "pizza margherita"
      size: "big"
      quantity: 2
      price: 39.9
```

```
_id: ObjectId('648f6af6411f4da9ca7f133a')
user_id: ObjectId('648f604b411f4da9ca7f1332')
payment_method: "gotówka"
order_date: "2023-05-31T12:45:00Z"
total_amount: 39.9
status: "przyjęte"
is_delivery: true
items: Array
  0: Object
      name: "pizza z szynką i pieczarkami"
      size: "big"
      quantity: 1
      price: 39.9
```

```
_id: ObjectId('648f6af6411f4da9ca7f133b')
payment_method: "karta kredytowa"
order_date: "2023-06-01T15:20:00Z"
total_amount: 79.6
status: "przyjęte"
is_delivery: false
items: Array
  0: Object
      name: "pizza hawajska"
      size: "small"
      quantity: 4
      price: 19.9
```

Update

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

```
> db.order.updateOne(
    { "_id": ObjectId("648f6af6411f4da9ca7f1338") },
    {
      $set: {
        "status": "w drodze",
        "delivery_person_id": ObjectId("648f604b411f4da9ca7f1333"),
        "delivery_address": {
          "street": "Kwiatowa",
          "house_number": "5",
          "apartment_number": "",
          "postal_code": "20-136"
        }
      }
    }
  );
```

```
_id: ObjectId('648f6af6411f4da9ca7f1338')
user_id: ObjectId('648f604b411f4da9ca7f1331')
payment_method: "karta kredytowa"
order_date: "2023-05-29T15:30:00Z"
total_amount: 167.4
status: "w drodze"
is_delivery: true
▸ items: Array
▾ delivery_address: Object
    street: "Kwiatowa"
    house_number: "5"
    apartment_number: ""
    postal_code: "20-136"
  delivery_person_id: ObjectId('648f604b411f4da9ca7f1333')
```

Zapytanie, które dodaje opinie do zamówienia.

```
> db.order.updateOne(
    {
       _id: ObjectId("648f6af6411f4da9ca7f1338")
    },
    {
       $set: {
          review: {
             rating: 5,
             comment: "Bardzo smaczna pizza!"
          }
       }
    }
 );
```

```
_id: ObjectId('648f6af6411f4da9ca7f1338')
user_id: ObjectId('648f604b411f4da9ca7f1331')
payment_method: "karta kredytowa"
order_date: "2023-05-29T15:30:00Z"
total_amount: 167.4
status: "dostarczone"
is_delivery: true
▸ items: Array
▸ delivery_address: Object
delivery_person_id: ObjectId('648f604b411f4da9ca7f1333')
▾ review: Object
    rating: 5
    comment: "Bardzo smaczna pizza!"
```

Aggregate

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

```
> db.order.aggregate([
    { $unwind: "$items" },
    {
      $group: {
        _id: "$items.name",
        totalQuantity: {
    $sum: "$items.quantity" }
      }
    },
    {
      $project: {
        pizzaName: "$_id",
        totalQuantity: 1,
        _id: 0
      }
    },
    { $sort: { totalQuantity: -1 } }
  ]);
```

```
< {
    totalQuantity: 7,
    pizzaName: 'pizza hawajska'
  }
  {
    totalQuantity: 6,
    pizzaName: 'pizza margherita'
  }
  {
    totalQuantity: 4,
    pizzaName: 'pizza z szynką i pieczarkami'
  }
```

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

```
> db.order.aggregate([
    {
      $group: {
        _id: { $ifNull: ["$user_id", "No User"] },
        orderCount: { $sum: 1 }
      }
    },
    {
      $project: {
        userId: {
          $cond: {
            if: { $eq: ["$_id", "No User"] },
            then: null,
            else: "$_id"
          }
        },
        orderCount: 1,
        _id: 0
      }
    },
    { $sort: { orderCount: -1 } }
  ]);
```

```
< {
    orderCount: 2,
    userId: ObjectId("648f604b411f4da9ca7f1332")
  }
  {
    orderCount: 1,
    userId: ObjectId("648f604b411f4da9ca7f1331")
  }
  {
    orderCount: 1,
    userId: null
  }
```

Delete

Zapytanie, które usuwa wszystkie zamówienia usera o podanym id.

```
> db.order.deleteMany({ user_id: ObjectId('648f604b411f4da9ca7f1332') });
< {
    acknowledged: true,
    deletedCount: 2
  }
```

```
_id: ObjectId('648f6af6411f4da9ca7f1338')
user_id: ObjectId('648f604b411f4da9ca7f1331')
payment_method: "karta kredytowa"
order_date: "2023-05-29T15:30:00Z"
total_amount: 167.4
status: "dostarczone"
is_delivery: true
▸ items: Array
▸ delivery_address: Object
delivery_person_id: ObjectId('648f604b411f4da9ca7f1333')
▸ review: Object
```

```
_id: ObjectId('648f6af6411f4da9ca7f133b')
payment_method: "karta kredytowa"
order_date: "2023-06-01T15:20:00Z"
total_amount: 79.6
status: "przyjęte"
is_delivery: false
▸ items: Array
```

Find

Zapytanie, które wyświetla zamówienia bez user_id.

```
> db.order.find({ user_id: { $exists: false } });
< {
    _id: ObjectId("648f6af6411f4da9ca7f133b"),
    payment_method: 'karta kredytowa',
    order_date: '2023-06-01T15:20:00Z',
    total_amount: 79.6,
    status: 'przyjęte',
    is_delivery: false,
    items: [
      {
        name: 'pizza hawajska',
        size: 'small',
        quantity: 4,
        price: 19.9
      }
    ]
  }
```

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.

```
> db.createUser({
    user: "admin",
    pwd: "admin123",
    roles: [{ role: "dbOwner", db: "pizzeria" }]
  });
< { ok: 1 }
```

```
> db.createUser({
    user: "user",
    pwd: "user123",
    roles: [{ role: "read", db: "pizzeria" }]
  });
< { ok: 1 }
```

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.