Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Bazy danych 2

Sprawozdanie z projektu

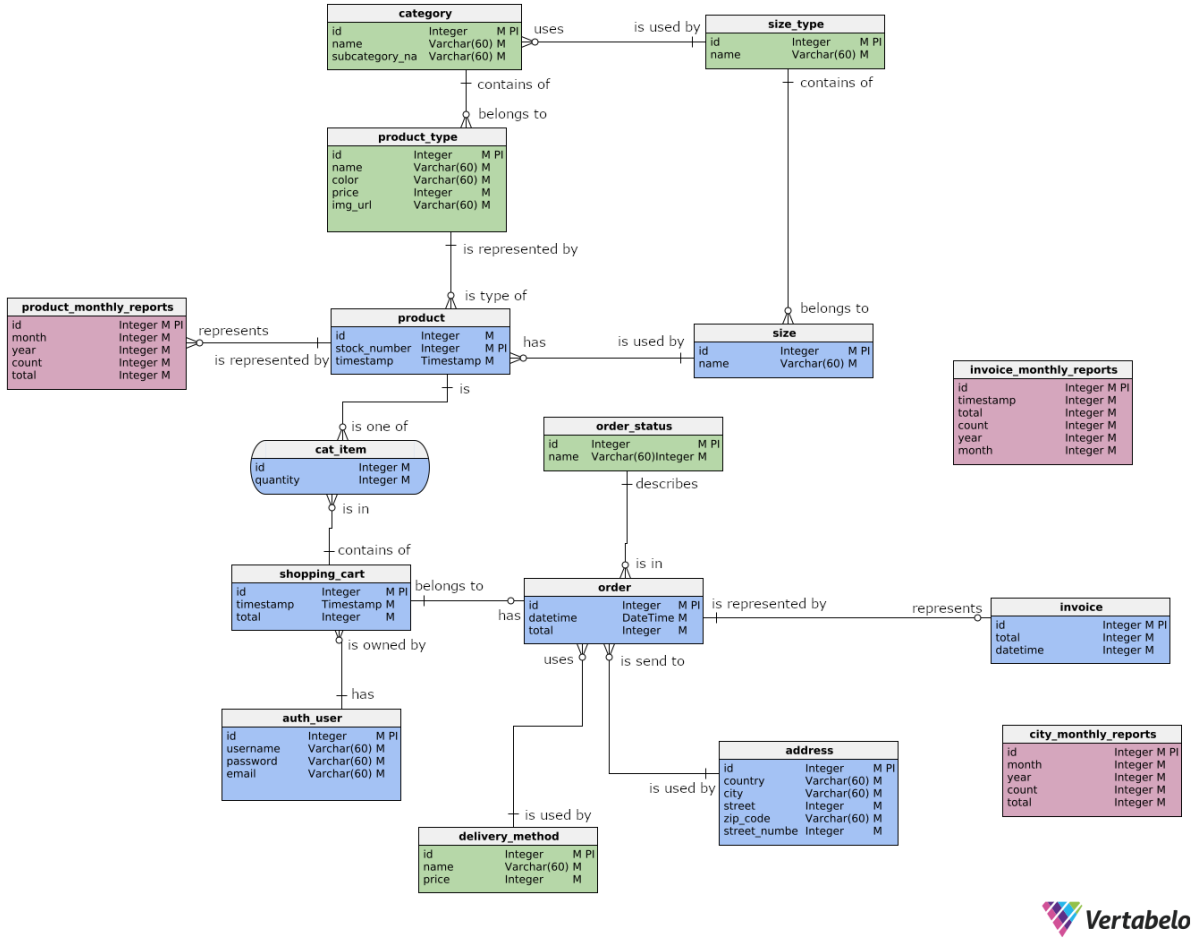
Paulina Dąbrowska, Natalia Iwańska, Mateusz Krajewski,
Julia Macuga, Filip Szczygielski

Warszawa, 2024

Spis treści

1. Model pojęciowy

Na rysunku 1.1 został przedstawiony model pojęciowy bazy danych. Encje oznaczone kolorem zielonym reprezentują metamodel bazy, czyli zawierające dane predefiniowane. Encje oznaczone kolorem niebieskim reprezentują encje części operacyjnej bazy danych, czyli zawierające dane wprowadzane w trakcie używania aplikacji i w trakcie trwania procesów zakupów i zamawiania. Encje oznaczone kolorem czerwonym reprezentują część analityczno-raportową aplikacji, czyli zawierają dane wspomagające analizę procesów i funkcjonowania aplikacji.



Rys. 1.1. Diagram modelu ER bazy danych

1.1. Encje metamodelu

Category - kategoria produktów

— id - identyfikator kategorii
— name - nazwa kategorii
— subcategory name - nazwa podkategorii

Size - rozmiar produktu

— id - identyfikator rozmiaru
— name - nazwa rozmiaru (np. S, M, 34, 38, 40)

Size Type - typ rozmiaru produktu

— id - identyfikator typu rozmiaru
— name - nazwa typu rozmiaru (np. 35-45, XS-XXL)

Product Type - typ produktu

— id - identyfikator typu produktu
— name - nazwa produktu
— color - kolor produktu
— price - cena produktu
— img_url - adres URL obrazka produktu

Product - konkretny produkt o konkretnym rozmiarze

— id - identyfikator produktu
— stock_number - liczba dostępnych sztuk
— timestamp - znacznik czasu

Cart Item - pozycja w koszyku

— id - identyfikator pozycji w koszyku
— quantity - ilość

Shopping Cart - koszyk

— id - identyfikator koszyka
— timestamp - znacznik czasu
— total - suma wartości koszyka

Auth User - użytkownik

— id - identyfikator użytkownika
— username - nazwa użytkownika
— password - wartość funkcji skrótu hasła
— email - adres e-mail

Order - zamówienie

— id - identyfikator zamówienia
— total - suma wartości zamówienia
— datetime - znacznik czasu

Delivery Method - metoda dostawy

— id - identyfikator metody dostawy
— name - nazwa metody dostawy
— price - cena dostawy

Order Status - status zamówienia

— id - identyfikator statusu zamówienia

— name - nazwa statusu zamówienia

Address - adres zamówienia

— id - identyfikator adresu
— street - ulica
— city - miasto
— country - kraj
— zip_code - kod pocztowy

Invoice - faktura

— id - identyfikator faktury
— total - suma wartości faktury
— datetime - znacznik czasu

City Monthly Reports - raport miesięczny miasta

— id - identyfikator raportu
— city - miasto
— month - miesiąc
— year - rok
— count - liczba zamówień
— total - suma wartości zamówień

Invoice Monthly Reports - raport miesięczny faktur

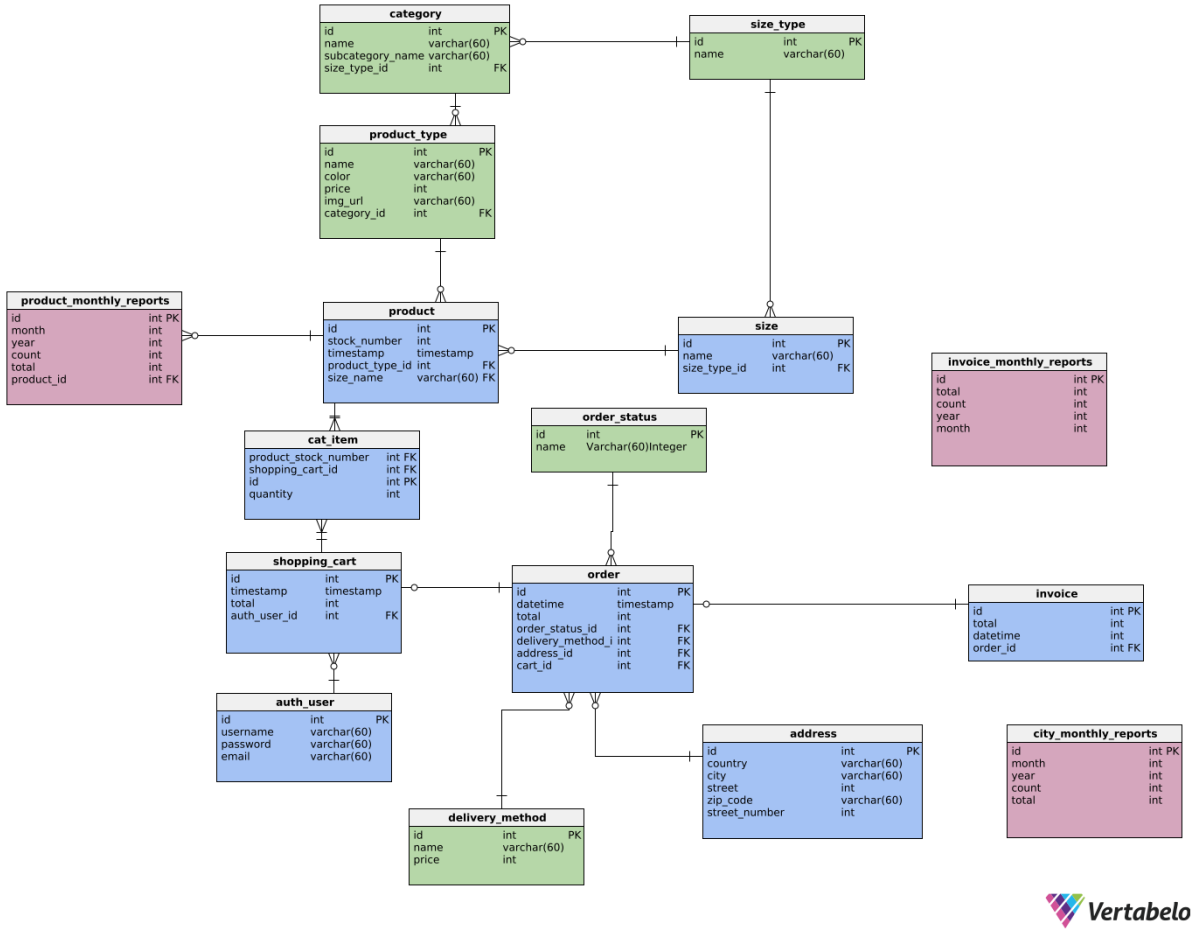
— id - identyfikator raportu
— month - miesiąc
— year - rok
— count - liczba faktur
— total - suma wartości faktur

Product Monthly Reports - raport miesięczny produktów

— id - identyfikator raportu
— product_item_id - identyfikator produktu
— month - miesiąc
— year - rok
— count - liczba zamówień
— total - suma wartości zamówień

2. Model relacyjny

Na podstawie modelu ER został opracowany model relacyjny bazy danych przedstawiony na rysunku 2.1



Rys. 2.1. Diagram modelu relacyjnego bazy danych

3. Dokumentacja użytkowa

3.1. Wprowadzenie

Aplikacja to internetowy sklep odzieżowy, umożliwiający użytkownikom przeglądanie produktów, dodawanie ich do koszyka oraz składanie zamówień. Zarejestrowani użytkownicy mogą również przeglądać historię swoich zamówień.

3.2. Rejestracja i logowanie

Rejestracja: Aby zarejestrować się w aplikacji, użytkownik musi przejść do strony rejestracji ("Register" w górnym pasku nawigacyjnym), wypełnić formularz rejestracyjny podając unikalny login, hasło oraz adres e-mail, a następnie zatwierdzić formularz. Po pomyślnej rejestracji, użytkownik zostanie automatycznie przekierowany do strony logowania.

Logowanie: Aby zalogować się do aplikacji, użytkownik musi przejść do strony logowania ("Login" w górnym pasku nawigacyjnym), wprowadzić swój login i hasło, a następnie zatwierdzić formularz. Po pomyślnym zalogowaniu, użytkownik zostanie przekierowany na stronę główną.

3.3. Nawigacja po aplikacji

Strona główna: Strona główna zawiera przegląd produktów wraz z filtrowaniem (przez kategorię, kolor oraz rozmiar) oraz menu nawigacyjne umożliwiające dostęp do różnych sekcji aplikacji.

Strona produktu: Użytkownik może obejrzeć szczegóły produktu prze kliknięcie "Buy now". Strona produktu umożliwia wybranie dostępnego rozmiaru produktu i ilości a następnie dodanie produktów do koszyka.

Koszyk: Po kliknięciu "Shopping cart" na górnym pasku nawigacyjnym pojawia się strona koszyka zawierająca listę produktów dodanych przez użytkownika oraz ich całkowity koszt. Umożliwia usuwanie produktów oraz przejście do składania zamówienia.

Składanie zamówienia: Strona zamówienia pozwala użytkownikowi na wybór adresu dostawy, metody dostawy oraz finalizację zakupu.

Przeglądanie zamówień: Po kliknięciu "Your Orders" na górnym pasku nawigacyjnym użytkownik może przeglądać historię swoich zamówień, sprawdzać szczegóły zamówień oraz statusy.

3.4. Dodawanie produktów do koszyka

1. Przejdź do strony produktu, który chcesz dodać do koszyka, klikając "Buy now".
2. Wybierz odpowiedni rozmiar oraz ilość produktu.
3. Kliknij przycisk "Add to cart".
4. Produkt zostanie dodany do koszyka, a użytkownik będzie mógł kontynuować zakupy lub przejść do koszyka, aby sfinalizować zakup.

3.5. Składanie zamówienia

1. Przejdź do strony koszyka, klikając "Shopping cart" w menu nawigacyjnym.
2. Zweryfikuj produkty w koszyku, ich ilości oraz ceny.
3. Kliknij przycisk "Proceed to checkout", aby przejść do strony zamówienia.
4. Wybierz zapisany adres dostawy lub wprowadź nowy adres.
5. Wybierz metodę dostawy spośród dostępnych opcji.
6. Kliknij przycisk "Submit Order", aby sfinalizować zakup.
7. Jeśli wszystkie produkty są dostępne, zamówienie zostanie złożone, użytkownikowi wyświetli się potwierdzenie zamówienia oraz zostanie utworzona nowa faktura. Koszyk znów stanie się pusty

3.6. Przeglądanie zamówień

1. Przejdź do sekcji "Your Orders" w menu nawigacyjnym.
2. Zobacz listę swoich zamówień, wraz z datami, kwotami, statusami, listami produktów, adresami dostawy oraz metodami dostawy.

4. Część analityczna aplikacji

W ramach części analitycznej naszej aplikacji została opracowana podstrona skierowana dla administratora, na której można oglądać zbiorcze statystyki aplikacji. Dostępne są trzy rodzaje statystyk:

— statystyki z faktur,
— statystyki z produktów,
— statystyki z miast.

Statystyki z faktur

W zakładce statystyki z faktur, zamieszczony jest roczny histogram prezentujący odniesione zyski w poszczególnych miesiącach danego roku.

Dane do wykresów pobierane są z tabeli **invoice_monthly_reports**, która zawiera w sobie raporty dotyczące miesięcznych zysków ze sprzedaży.

Statystyki z produktów

W zakładce statystyki z produktów, zamieszczone są dwa typy wykresów: wykres przedstawiający 10 najczęściej kupowanych produktów w danym miesiącu oraz wykres przedstawiający procentowy udział poszczególnych kategorii w zakupach w danym miesiącu.

Dane do wykresów pobierane są z tabeli **product_monthly_reports**, która zawiera w sobie raporty dotyczące jakie produkty zostały zakupione w danym miesiącu.

Statystyki z miast

W zakładce statystyki z miast, zamieszczony jest histogram prezentujący 10 miast, w których najczęściej były robione zamówienia w danym miesiącu.

Dane do wykresów pobierane są z tabeli **city_monthly_reports**, która zawiera w sobie raporty dotyczące ile zamówień wypłynęło z danego miasta w danym miesiącu.

Podstrona ma na celu umożliwić administratorowi monitorowanie i analizę funkcjonowania sklepu - np. analizę na jakie produkty jest największy popyt, jakie produkty i kategorie cieszą się największym zainteresowaniem, czy w jakich miastach sklep ma powodzenie.

5. Dobór technologii

Podczas realizacji projektu kluczowym aspektem było dokładne dobranie technologii, które umożliwią skuteczną implementację oraz efektywne zarządzanie bazą danych i aplikacją. Po przeprowadzeniu analizy różnych opcji, zdecydowano się na wykorzystanie PostgreSQL, Pythona, Flaska, SQLAlchemy oraz Dockera. Każdy z tych wyborów był podyktowany specyficznymi wymaganiami projektowymi oraz dostępnymi kompetencjami w zespole.

PostgreSQL

PostgreSQL został wybrany jako system zarządzania relacyjnymi bazami danych z uwagi na jego niezawodność i skalowalność, kluczowe dla aplikacji obsługującej zarówno transakcje operacyjne, jak i generowanie raportów analitycznych. Dodatkowo oferuje zaawansowane funkcje, takie jak procedury składowane i triggery, które są niezbędne do implementacji złożonej logiki biznesowej w bazie danych.

Python

Python został wybrany jako główny język programowania ze względu na prostotę, czytelność kodu oraz bogaty ekosystem bibliotek, co umożliwiło szybki rozwój aplikacji i integrację z różnymi komponentami.

Flask

Do stworzenia aplikacji webowej użyto Flaska, lekkiego mikroframeworka dla Pythona, znakomicie integrującego się z SQLAlchemy i pozwalającego na elastyczne dodawanie i dostosowywanie funkcjonalności.

SQLAlchemy

Biblioteka SQLAlchemy została użyta jako narzędzie do mapowania obiektowo-relacyjnego (ORM), umożliwiające bardziej intuicyjne zarządzanie danymi w systemie. Pozwoliło na operowanie na bazie danych za pomocą obiektów Pythona, co znacząco zwiększyło czytelność kodu oraz ułatwiło manipulację danymi. ORM SQLAlchemy zapewnia także zaawansowane możliwości konfiguracji i optymalizacji zapytań, co było kluczowe dla zapewnienia wysokiej wydajności naszej aplikacji.

Docker

Docker został wykorzystany do zarządzania środowiskiem projektowym poprzez konteneryzację aplikacji. Zapewnił on spójność konfiguracji aplikacji poprzez zdefiniowanie wszystkich jej komponentów (takich jak kod, zależności, ustawienia) w plikach konfiguracyjnych. Umożliwiło to łatwe replikowanie i zarządzanie środowiskiem, co było kluczowe dla efektywnego rozwoju i wdrożenia aplikacji.

6. Elementy funkcjonalne bazy

W ramach projektu projektowania systemu bazodanowego, kluczowym aspektem jest odpowiednie zaprojektowanie funkcjonalności bazy danych. Elementy te obejmują zarówno strukturalne aspekty bazy, jak i procedury operacyjne, które mają być realizowane bezpośrednio na poziomie bazy danych. Poniżej przedstawiono ich opis.

Procedury składowane

W celu zaimplementowania części logiki aplikacyjnej bezpośrednio w bazie danych wykorzystano procedury składowane zaprojektowane do obsługi raportowania miesięcznego.

1. Procedura "city_reports" wywoływana jest w momencie zaksięgowania nowego zamówienia odpowiada za aktualizację miesięcznych raportów dla miast w zależności od dostarczonych parametrów: nazwy miasta, miesiąca, roku oraz całkowitej wartości zamówienia. Jeśli nie istnieje wcześniejszy raport dla danego miesiąca i roku, procedura dokonuje jego wstawienia; w przeciwnym razie aktualizuje istniejący raport, zwiększając licznik wykonanych transakcji i ich całkowitą wartość.
2. Procedura "invoice_reports" wywoływana analogicznie w momencie wystawienia nowej faktury odpowiedzialna jest za generowanie miesięcznych raportów dotyczących faktur na podstawie podanego miesiąca, roku oraz ich łącznej kwoty. Podobnie jak w przypadku "city_reports", procedura sprawdza istnienie wcześniejszego raportu i wstawia nowy lub aktualizuje istniejący.
3. Procedura "product_reports" obsługuje generowanie raportów miesięcznych dla produktów na podstawie identyfikatora produktu, miesiąca, roku oraz liczby sprzedanych produktów. Dodatkowo, procedura pobiera typ produktu i jego cenę, aby obliczyć całkowitą wartość sprzedaży. Podobnie jak w poprzednich przypadkach, procedura wywoływana w momencie złożenia nowego zamówienia aktualizuje istniejące raporty lub wstawia nowe, w zależności od istniejących danych.

Triggery

W projekcie zdecydowano się na zastosowanie triggera samodzielnie tworzącego faktury w momencie zaksięgowania nowego zamówienia. Zamiast standardowej implementacji bezpośrednio w bazie danych został on zaimplementowany przy pomocy tzw. "eventów" dostępnych w bibliotece SQLAlchemy. Zdecydowano się na taki sposób implementacji z kilku powodów:

1. Chociaż PostgreSQL posiada swoje mechanizmy triggerów, eventy SQLAlchemy są częścią aplikacji Pythonowej, co ułatwia zarządzanie logiką biznesową i integrację z innymi częściami systemu. To podejście pozwala na elastyczne dostosowanie aplikacji bez konieczności modyfikowania struktury samej bazy danych.
2. Kod aplikacyjny napisany w SQLAlchemy jest łatwiejszy do zarządzania i debugowania niż skrypty SQL używane w triggerach. Eventy SQLAlchemy są integralną częścią aplikacji, co ułatwia zrozumienie i utrzymanie przez zespół programistów.

3. Eventy w SQLAlchemy mogą być testowane w izolacji od bazy danych, co przyspiesza procesy testowania i zapewnienia jakości. Można łatwo tworzyć jednostkowe testy dla logiki eventów, co jest trudniejsze w przypadku triggerów bazodanowych.
4. Eventy SQLAlchemy pozwalają na łatwe modyfikowanie i rozbudowywanie logiki aplikacji bez potrzeby zmiany struktury baz danych.

7. Optymalizacja

Optymalizacja bazy danych jest kluczowym elementem w zapewnieniu efektywności, wydajności i skalowalności aplikacji. W ramach naszego projektu przeprowadziliśmy szereg działań optymalizacyjnych, koncentrując się na denormalizacji danych oraz dodawaniu indeksów w celu poprawy wydajności zapytań. Poniżej przedstawiam opis tych działań:

7.1. Denormalizacja danych

Denormalizacja danych polega na celowym dodawaniu redundancji danych do bazy w celu zwiększenia wydajności odczytu kosztem utrzymywania spójności danych. W naszym projekcie zdecydowaliśmy się na denormalizację, która umożliwiła szybszy dostęp do informacji związanych z przeglądaniem zamówienia. W wersji przed denormalizacją w celu uzyskania potrzebnych danych trzeba było wykonać 4 operacje *join*, które mocno osłabiają wydajność bazy. Pomijając skomplikowane operacje złończeń dostęp do danych był szybszy i wygodniejszy.

7.2. Dodawanie indeksów

Indeksy są kluczowymi strukturami w bazach danych, które przyspieszają wyszukiwanie danych poprzez organizację i sortowanie zawartości tabeli. W naszym projekcie dodaliśmy indeksy kolumnowe do kluczowych kolumn w celu zwiększenia wydajności zapytań. Skupiliśmy się na kolumnach, które często są filtrowane lub sortowane w zapytaniach. Na przykład, dodaliśmy indeksy do kolumn takich jak $size-type$ w tabeli $Size$, $product-category$ w tabeli $ProductType$, czy $product-subcategory$ itp. Dzięki temu zapytania wykorzystujące te kolumny stał← się bardziej efektywne i szybsze.

7.3. Perspektywa zmaterializowana

Rozważaliśmy również implementację perspektyw zmaterializowanych dla pewnych zapytań lub widoków w bazie danych. Perspektywy zmaterializowane mogą być użyteczne w sytuacjach, gdzie potrzebujemy często wykonywać złożone zapytania, które mogą być czasochłonne lub wymagających wielu operacji złączeń. Idealnie sprawdzają się do tworzenia raportów i analiz.

8. Testy

W ramach procesu tworzenia aplikacji webowej oraz bazy danych, kluczowym etapem jest przeprowadzenie testów jednostkowych. Testy te mają na celu weryfikację poprawności funkcjonowania poszczególnych komponentów aplikacji oraz zapewnienie, że baza danych działa zgodnie z oczekiwaniami.

W przypadku funkcjonalności aplikacji przetestowaliśmy zarządzanie koszykiem: dodawanie, usuwanie i sprawdzanie produktów.

Testowanie bazy danych jest kluczowe dla zapewnienia integralności danych i efektywności operacji zapisu oraz odczytu. Nasze testy jednostkowe koncentrują się na tworzeniu i modyfikacji schematu bazy danych:

— Testujemy proces tworzenia tabel, indeksów oraz relacji między nimi.
— Sprawdzamy, czy modyfikacje schematu (np. dodawanie nowych kolumn) są wykonywane poprawnie.