POLITECHNIKA ŚWIĘTOKRZYSKA

Wydział Elektrotechniki, Automatyki i Informatyki

Technologie Obiektowe

Obiektowe bazy danych

Dulemba Artur nr albumu: 86674

Kierunek: Informatyka
Studia niestacjonarne, semestr II
Grupa: 1IZ21A

Kielce 2022

Spis treści

1. Wstęp

Baza danych (angielskie database) jest to rodzaj komputerowego zbioru, można powiedzieć magazyn danych o określonej budowie. Baza danych jest modelowym ujęciem fragmentu rzeczywistości będącego przedmiotem zainteresowania np. osób, instytucji, organizacji, firm, zakładów itp., reprezentującym fakty dotyczące tej rzeczywistości, w formie umożliwiającej ich przetwarzanie. Istotne obiekty danego przedmiotu określa się jak encje lub klasy. Klasą lub encją w rejestrze samochodów są poszczególne samochody, a także ich właściciele lub użytkownicy. Projekt bazy danych określa jej strukturę (część intensjonalną) i zawartość (część ekstensjonalną). Dane przechowywane w bazie są trwałe, co nie oznacza, że nie ulegają zmianom. W każdej chwili baza danych znajduje się w określonym stanie. Operacje powodujące zmianę stanu bazy danych nazywają się transakcjami.[1]

Obiektowe bazy danych są propozycją uniwersalnego i rozszerzalnego modelu danych dla systemów baz danych. W momencie, w którym się pojawiły miały one zastąpić relacyjny model danych, jako lepiej przystosowane do nowych dziedzin zastosowań systemów baz. Na początku lat 80-90, zaczęto próby zastosowania systemu obiektowych baz danych w nowych dziedzinach, np. systemy wspomagania projektowania, systemy multimedialne lub systemy informacji przestrzennej. Charakterystyka zastosowań takiej obiektowej bazy jest bardzo odmienna od relacyjnej. Składowane oraz przetwarzane danych są złożone strukturalnie. Typowe dla tego rodzaju baz są hierarchicznie złożone struktury danych, a także liczne i intensywnie przetwarzane powiązania między informacjami. Powiązania te mają złożoną semantykę: agregacji, referencji czy też kompozycji. Informacje, które są przetwarzane w nowych dziedzinach zastosowań to np. dokumenty tekstowe, animacje, obrazy czy też dane wielowymiarowe. Integracja aplikacji baz danych pisanych za pomocą języków obiektowych z relacyjnymi bazami danych była trudna i nienaturalna, ponieważ system typów bazy danych i system typów aplikacji są całkowicie odmienne.

2. Omówienie obsługiwanych standardów (np. ODMG, JDO, LINQ, XQuery)

Historycznie pierwszym rozwiązaniem była budowa nowych systemów baz danych od początku na bazie obiektowych języków programowania. Polega to na rozszerzeniu funkcjonalności obiektów tworzonych i przetwarzanych przez języki obiektowe o własności typowe dla danych przechowywanych w systemach baz danych: trwałości, współdzielenia, odtwarzania spójnego stanu po awarii, synchronizacji dostępu, wydajnego przetwarzania dużych zbiorów obiektów, itp. Wynikiem tej strategii są obiektowe bazy danych, które posiadają jednorodny obiektowy model. Standard dla modelu danych tej klasy systemów został opracowany przez organizację „Object Database Management Group” i od nazwy tej grupy jest nazwany „ODMG 3.0”.

Kontynuatorem tej strategii jest rozwiązanie, w którym funkcjonalność obiektowego modelu danych jest implementowana przez dodatkową warstwę programową budowaną na systemie bazy danych o dowolnym modelu danych. Standardem związanym z tą strategią jest ”Java Data Objects” - JDO. [2]

---

1 Płoski Z; Słownik Encyklopedyczny – Informatyka,. wyd. Europa,. Wrocław 1999

2 Tomasz Koszlajda; Obiektowy model danych; Wykład 4: Obiektowe bazy danych – 1. Obiektowy model danych: https://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_systemy_baz_danych

Natomiast LINQ to SQL to składnik programu .NET Framework w wersji 3.5, który zapewnia zarządzanie danymi relacyjnymi jako obiektami. Dane relacyjne są wyświetlane jako kolekcja tabel dwuwymiarowych (relacje lub pliki płaskie), gdzie wspólne kolumny wiążą tabele ze sobą.

W LINQ to SQL model danych w relacyjnej bazy danych jest mapowany na model obiektów wyrażony w danym języku programowania. Po uruchomieniu instrukcji w aplikacji LINQ to SQL wysyła je do bazy danych w celu wykonania. Gdy baza danych zwraca wyniki, LINQ to SQL przekształca je z powrotem na obiekty.[3]

XQuery służy do wydobywania informacji z dokumentów XML. W swoich założeniach ma być tym, czym SQL jest dla relacyjnych baz danych. Zastosowanie XQuery często związane jest z bazami danych. Warto zauważyć, że dokument XML nie jest koniecznie plikiem tekstowym, który jest zapisany na dysku ani strumieniem znaków reprezentujących dokument. Często też struktura dokumentu XML może istnieć tylko w warstwie logicznej, a w innej musi znajdować się zoptymalizowane źródło danych, np. relacyjna lub relacyjno-obiektowa baza danych.

Za pomocą XQuery można odpytywać dane z bazy. Nie jest to efektywniejsze od SQL, ale gdyby w tabelach bazy danych były zapisane fragmenty XML to wtedy XQuery pozwala na przezroczysty dostęp do danych. Wymaga to również wsparcia SZBD i takowe wsparcie jest już oferowane w głównych systemach tj. Oracle, IBM, Microsoft. XQuery oferuję również możliwość definiowania własnych funkcji, których następnie można używać w wyrażeniach, lub także w innych funkcjach. Funkcje zaś powinny być zadeklarowane w prologu, to znaczy przed ciałem zapytania, ale obok innych wcześniejszych deklaracji dokumentu XQuery.[4]

3. Opis/instalacja ObjectDB i IDE

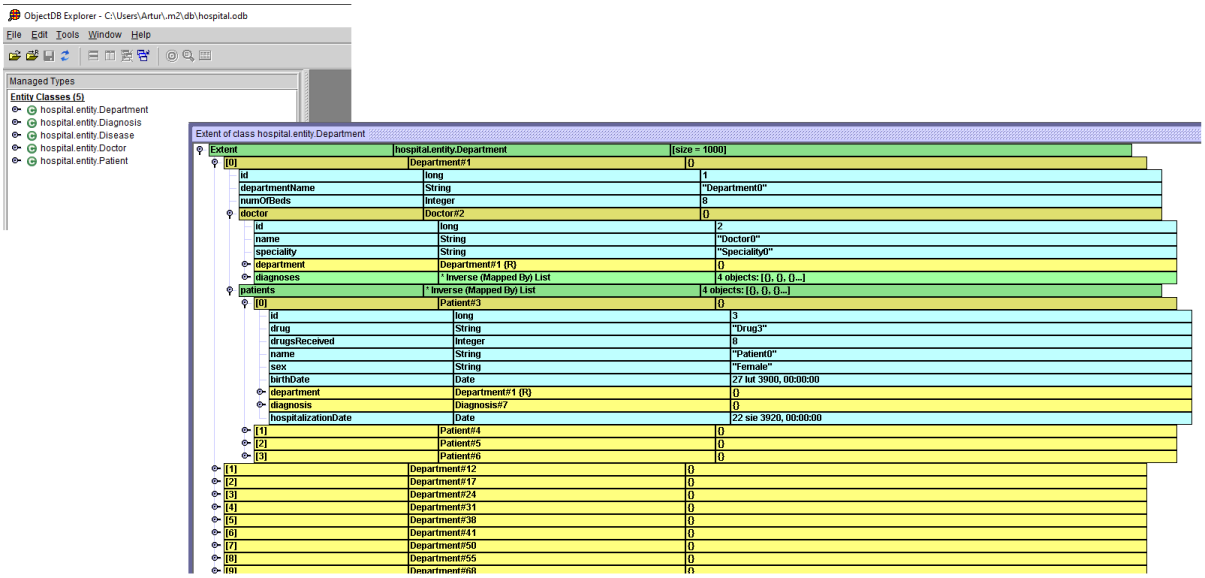
ObjectDB to rozbudowany obiektowy system zarządzania bazą danych (ODBMS). Jest poręczny, niezawodny, łatwy w obsłudze, a także niezwykle szybki. ObjectDB zapewnia wszystkie standardowe usługi zarządzania bazą danych tj. przechowywanie i pobieranie, transakcje, zarządzanie blokadami, przetwarzanie zapytań itp., ale w sposób ułatwiający tworzenie i przyspieszanie aplikacji.

- 100% czysty obiektowy system zarządzania bazami danych Java (ODBMS).
- Brak zastrzeżonego API - zarządzane tylko przez standardowe API Javy (JPA 2/JDO 2).
- Niezwykle szybki - szybszy niż jakikolwiek inny produkt JPA / JDO.
- Nadaje się do plików baz danych od kilobajtów do terabajtów.
- Obsługuje zarówno tryb klient-serwer, jak i tryb osadzony.
- Pojedynczy plik JAR bez zależności zewnętrznych.
- Baza danych przechowywana jest jako pojedynczy plik.
- Zaawansowane możliwości zapytań i indeksowania.
- Skuteczny w mocno obciążonych środowiskach wielu użytkowników.
- Można go łatwo osadzić w aplikacjach dowolnego typu i rozmiaru.

3https://docs.microsoft.com/pl-pl/dotnet/framework/data/adonet/sql/linq/
4 https://www.mimuw.edu.pl/~czarnik/zajecia/xml06/lab07/lab07.html

- Rozwiązanie przetestowane z Tomcat, Jetty, GlassFish, JBoss i Spring.[5]

W celu zainstalowania ObjectDB należy udać się na stronę: https://www.objectdb.com/download i kliknąć w przycisk nazywający się:ObjectDB Development Kit 2.8.7. Po chwili plik powinien się pobrać. Aby zacząć z niej korzystać należy uruchomić server-b.exe oraz explorer-b.exe, pokaże się okno do zarządzania bazami danych. Producent dołącza dwie testowe bazy: point oraz world, także można poznać jak to działa.



Rys. 1 Połączony zrzut ekranu z menu i wygląd tabeli Department

Moim IDE wykorzystywanym w tym projekcie będzie IntelliJ IDEA Community Edition.

IntelliJ IDEA Community Edition to w pełni darmowa edycja komercyjnego środowiska programistycznego (IDE), pozwalającego tworzyć aplikacje w takich językach jak Java, Groovy itp. InteliJ rozwijany jest od 2001 roku, a swoją popularność zawdzięcza m.in. wbudowanym narzędziom do refaktoringu, tzn do procesu modyfikacji istniejącego oprogramowania z równoczesnym zachowaniem przyjętych wzorców oraz ogólnej jakości kodu.

IDE oferuje programiście wszystkie narzędzia i funkcje, dające możliwość wygodnego pisania kodu, a także późniejszej rozbudowy. Znajdziemy wśród nich inteligentny edytor kodu, z opcją kolorowania składni i autouzupełniania, narzędzia do refaktoryzacji, inspekcji kodu, wbudowany edytor XML oraz mechanizmy do kontroli wersji, obsługujące systemy np. CVS, Subversion, Mercurial i Git, a także wizualne kreatory interfejsów graficznych opartych na bibliotece Swing. Środowisko to wspiera także takie języki programowania jak Java i Groovy, a dodatkowe wtyczki wprowadzają wsparcie dla m.in. Scala i Clojure. Ponadto posiada ono zintegrowane frameworki JUnit i TestNG, pozwalające przeprowadzać kompleksowe testy wytwarzanego oprogramowania.[6]

Producent stworzył również docelowy tutorial dla użytkownika IntelliJ-a, który można pobrać pod adresem: https://www.objectdb.com/tutorial/jpa/intellij/maven . Krok po kroku instruuję co należy kliknąć i przeprowadza przez cały proces importowania projektu do IDE.

5 https://www.objectdb.com/database/overview
6 https://www.dobreprogramy.pl/intellij-idea-community-edition,program,windows,6628622615124097

4. Tematyka bazy danych

Baza danych opiera się na tematyce szpitala. W bazie jest 5 tabel: Department, Diagnosis, Disease, Doctor i Patient.

Tabela Department zawiera takie dane jak: id, departmentName, numOfBeds i klucze obce: doctor i patients.

Tabela Diagnosis: id, diagnosis, diagnosisDate i klucze obce: disease, doctor.

Tabela Disease: id, name i klucz obcy: diagnoses.

Tabela Doctor: id, name, speciality i klucz obcy: department i diagnoses.

Tabela Patient: id, drug, drugsReceived, name, sex, birthDate, hospitalizationDate i obce: department, diagnosis.

5. Zapisywanie, aktualizacja i usuwanie obiektów

Zapisywanie do bazy danych odbywa się na zasadzie generowania danych w pętli w IDE. Aktualnie jest ustawione, aby w tabeli Department, Disease, Doctor było generowanie 1000 rekordów. W pozostałych danych jest więcej, ponieważ baza generuję liczbę pacjentów, którzy mogą być na oddziale np. w jednym Department jest jeden pacjent, a w drugim tych pacjentów może być ich nawet do 9, aby jak najbardziej oddać realizm.

Aktualizacja i usuwanie obecnie znajdujących się w bazie danych może się również odbywać w ObjectDB Explorer.

Po wczytaniu bazy i po jej odczycie, w pasku u góry należy wybrać przycisk Edit -> New Entity Objects, później eksplorator pokazuje możliwe tabele do dodania danych, po jej wybraniu należy kliknąć Create and Persist i tworzy nowy wpis.

Aktualizacja:

Aby zedytować daną w tabeli, należy na nią dwukrotnie kliknąć i wpisać nową informację.

Usuwanie:

Usuwanie polega na zaznaczeniu interesującego użytkownika całego rekordu i przyciśnięciu przycisku Delete. Wtedy krotka zmieni zawartość na NULL, natomiast jak zostanie wybrany cały atrybut to zostanie on w całości usunięty.

6. Wczytywanie danych do bazy z IDE

Aby przeglądać dane wygenerowane w IDE należy połączyć ją z OdjectDB. Do tego celu, należało utworzyć nowy interface EntityManagerFactory, do połączenia z bazą danych hospital.odb. Do tego dochodzi EntityManager, który zapewnia zarządzanie cyklem życia encji, tzn. odpowiada za pobieranie, zapisanie czy usunięcie rekordów. W tym projekcie Helper to tam gdzie ma miejsce generowanie danych, później dochodzi do zamknięcia połączeń strumieni danych.

```
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("$objectdb/db/hospital.odb");

EntityManager em = emf.createEntityManager();

Helper.generateDateBase(em);
```

```
em.close();
emf.close();
```

7. Dziedziczenie, polimorfizm, hermetyzacja, abstrakcja (abstrakcyjność)

Dziedziczenie można rozumieć jako klasę pochodną, która dziedziczy funkcjonalności i implementację klasy, która jest bazowa. W klasie pochodnej jest możliwość dodania nowej funkcjonalności lub zredefiniować funkcjonalność odziedziczoną. W obiektowych bazach danych można zdefiniować nowe klasy, czy interfejsy na podstawie zdefiniowanych już wcześniej klas/ interfejsów. Model ODMG rozróżnia dwa typy powiązań między klasami lub interfejsami: związek relacji podtypu i dziedziczenia. W językach obiektowych są one traktowane jako tożsame, jednakowe. Związek relacji podtypu może odnosić się do klas, czy interfejsów, w praktyce oznacza to dziedziczenie przez typ pochodny funkcjonalności typu bazowego. Klasy i interfejsy połączone związkiem podtypu tworzą sieć powiązań o topologii grafu acyklicznego skierowanego. Kryje się pod tym to, że pojedyncza klasa lub interfejs może dziedziczyć funkcjonalności po wielu klasach lub interfejsach. Związek dziedziczenia łączy tylko klasy. Oznacza to, że dziedziczy zarówno funkcjonalności i implementacje. Obejmuje również semantykę relacji podtypu. Klasy połączone tym związkiem tworzą ze sobą sieć powiązań o topologii hierarchii. To znaczy, że pojedyncza podklasa może dziedziczyć zarówno funkcjonalność jak i implementację po dokładnie jednej nadklasie. Dziedziczona funkcjonalność może być poszerzona w stosunku do typu bazowego. Dziedziczona implementacja może być rozszerzana lub przesłaniana - definicje nowego kodu dla odziedziczonych metod, który zastąpi stary kod.

O obiektach, które są składowane w rozszerzeniach klas i które mają klasy pochodne mówi się, że są polimorficzne, odróżniają się cechami i metodami. Na przykład wystąpienie klasy Figura ma określony atrybut „Typ” służący do przechowywania danych o typie figury i metodę „Powierzchnia”, która służy do wyznaczania powierzchni figur. Z kolei wystąpienia klasy pochodnej „Wielokąt” oprócz odziedziczonego atrybutu „Typ” mają dodatkowo wielowartościowy atrybut „Krawędzie”. Odziedziczona po klasie „Figura” metoda „Powierzchnia” jest w klasie „Wielokąt” redefiniowana, ponieważ sposób wyznaczania powierzchni wielokątów jest inny niż uniwersalnych figur. Rozszerzenie klasy „Figura” obejmuje zarówno obiekty klasy „Figura”, jak również różniące się od nich obiekty, które są wystąpieniami klasy „Wielokąt”. Na poziomie składni języków obiektowych polimorfizm obiektów wyraża się w występowaniu zmiennych i podstawień polimorficznych. Przez zmienną polimorficzną rozumie się taką zmienną, pod którą są podstawiane obiekty, które są wystąpieniami różnych klas, ale też które występują w relacji podtypu z typem zmiennej polimorficznej. W podanym przykładzie, zmienną polimorficzną jest zmienna „f”. Mimo, że typem zmiennej „f” jest klasa „Figura”, to przypisywane są jej obiekty innych klas. Najpierw pod zmienną „f” jest podstawiony obiekt typu „Koło”, a później obiekt typu „Wielokąt”. Następnie przez zmienną „f” do przypisanych jej obiektów są przesyłane komunikaty o nazwie „Powierzchnia”. Właściwy kod metody wyznaczania powierzchni jest ustalany dynamicznie na podstawie typu obiektu przypisanego w danym momencie zmiennej „f”. W przykładzie najpierw jest to kod metody „Powierzchnia” zdefiniowanej w klasie „Koło”, następnie kod metody „Powierzchnia” klasy „Wielokąt”. Szczególnym przypadkiem zmiennych polimorficznych

są nazwy rozszerzeń klas, które mają klasy pochodne. Nazwa rozszerzenia klasy bazowej pozwala przetwarzać obiekty należące do rozszerzeń wszystkich klas pochodnych. [7]

Hermetyzacja polega na grupowaniu elementów składowych w obrębie wybranego obiektu i umożliwieniu manipulowania nim jako całością. Hermetyzacja łączy się też z ukrywaniem pewnych informacji dotyczących struktury i implementacji tego co jest wewnątrz. Jest podstawową techniką abstrakcji, to znaczy ukrycia wszelkich szczegółów danego przedmiotu, które na danym etapie rozpatrywania: analizy, projektowania, programowania, nie stanowią istotnej charakterystyki. Pojęcie hermetyzacji, jako jedna z zasad inżynierii oprogramowania, zostało zapoczątkowane w 1975 roku przez D. Parnasa. Możliwym jest wyróżnienie dwóch koncepcji: hermetyzacja ortodoksyjna lub hermetyzacja ortogonalna. Pierwsza z nich, jest dość popularnym stereotypem w obiektowości. Ten rodzaj hermetyzacji został zaimplementowany np. w języku Smalltalk. W tym podejściu, wszystkie operacje, które można wykonać na obiekcie, są określone przez metody do których są przypisane. Natomiast bezpośredni dostęp do atrybutów obiektów jest niemożliwy. Drugim rodzajem jest hermetyzacja ortogonalna, zaimplementowana między innymi w języku C++. W tym przypadku dowolny atrybut i metoda obiektu mogą być niedostępne z zewnątrz, lub publiczne.[8]

8. Porównanie Obiektowych baz danych z Relacyjnymi bazami danych

Relacyjne bazy danych oferują schemat bazy danych ograniczony zbiórem predefiniowanych typów danych dla atrybutów relacji. Przykładem mogą być systemowe typy: np. varchar, int i float, date. Typy danych, które są niedostępne w relacyjnym modelu danych muszą być zaimplementowane poza systemem bazy danych. Kod obsługujący semantykę musi być zaimplementowany i powielany we wszystkich aplikacjach bazy danych. Obiektowe bazy danych umożliwiają projektantom definiowanie nowych typów danych. Definiowanie własnego typu danych obejmuje definicję struktury i funkcjonalności. Definicje typów są składowane w systemie bazy danych. Sposób korzystania z typów danych zdefiniowanych przez użytkownika niczym się nie różni od wykorzystania typów systemowych. Dzięki temu obiektowy model danych jest rozszerzalny. Można go elastycznie dostosować do dowolnej dziedziny zastosowania wymagającej unikatowych, specjalnych typów danych. Rolę typów danych definiowanych przez użytkownika pełnią w obiektowych bazach danych klasy i interfejsy. Definicja interfejsu jest opisem funkcjonalności nowego typu danych, czyli opisem operacji dostępnych dla danego typu. Definicja klasy obejmuje dodatkowo implementację typu danych: to jest wewnętrzną strukturę danych służącą do przechowywania stanu wystąpień typu danych oraz kod operacji zdefiniowanych dla danego typu danych.[9]

---

7 Tomasz Koszlajda; Obiektowy model danych; Wykład 4: Obiektowe bazy danych – 1. Obiektowy model danych: https://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_systemy_baz_danych

8Paweł Józwik, Maciej Mazur; Obiektowe bazy danych – przegląd i analiza rozwiązań; http://www.kapitanat.pl/odb.pdf

9 Tomasz Koszlajda; Obiektowy model danych; Wykład 4: Obiektowe bazy danych – 1. Obiektowy model danych: https://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_systemy_baz_danych

	Relacyjna baza danych	Obiektowa baza danych
Zarządzanie danymi	Dane są przetrzymywane jako encje w sposób tabelaryczny	Przetrzymywane są jako obiekty
Złożoność danych	Przetrzymuje proste dane	Przetrzymuje ogromne i złożone dane
Grupowanie	Typ encji odnosi się do kolekcji tych encji, które dzielą się podobną definicję.	Klasa opisuje grupę obiektów, które ma relację, zachowania, a także podobne właściwości.
Przetrzymywanie danych	Składuje tylko dane	Składuje dane i funkcję wykonujące na plikach.
Obiektowość	Dane są niezależne od aplikacji uruchamianej przez program	Jest wymagane, aby zaimplementować dane.
Klucze	Klucz główny identyfikuje obiekt w bazie	Identyfikator obiektu (OID) jest jednoznaczną nazwą dla każdego typu obiektu czy encji.

Źródło: https://www.tutorialspoint.com/difference-between-rdbms-and-oodbms#

9. POSTGRES

PostgreSQL to jedno z najbardziej popularnych systemów zarządzania obiektowo-relacyjną bazą danych i jednym z niewielu systemów oferujących obiektowo-relacyjne podejście do baz danych.

Obiektowo-relacyjna baza danych jest połączeniem relacyjnej bazy danych z obiektową bazą danych. Podobnie jak MySQL, PostgreSQL, zwany również Postgresem, pozwala na przetrzymywanie danych w postaci tabel opisanych relacjami oraz na manipulacje danych za pomocą języka zapytań. Dodatkowo, integruje on obiektowe podejście do baz danych z relacyjnym pozwalając na utrzymanie narzędzi znanych z relacyjnych baz danych ale poszerzonych o możliwości związane z obiektowością: obiekty, klasy i dziedziczenie. Podobnie jak w programowaniu obiektowym możemy stworzyć klasę oraz jej obiekty i klasę dziedziczącą po tej klasie. Różnicą jest, że obiektowo-relacyjna baza danych pozwala na używanie strukturalnego języka zapytań znanego z relacyjnych baz danych do efektywnego i szybkiego przeszukiwania danych niedostępnego w obiektowych bazach danych.

Warto zauważyć, ze Postgres jest jednym z niewielu systemów zarządzania bazami danych które oferują relacyjny model bazy danych łącznie z obiektowym dając najlepsze rozwiązania z obu modeli. Jest też pod licencją typu open source to jest nie tylko darmowy ale również pozwala na dowolną modyfikacje kodu źródłowego dając możliwość dopasowania go do swoich wymagań. Będąc nieustannie usprawniany od prawie 20 lat, PostgreSQL jest jednym z najbardziej niezawodnych i funkcjonalnych systemów zarządzania bazą danych.[10]

10 https://vavatech.pl/technologie/bazy-danych/postgresql

10. ORM - Hibernate

ORM – czyli mapowanie obiektowo-relacyjne, jest to sposób odwzorowania obiektowej architektury systemu informatycznego na bazę danych o charakterze relacyjnym. Taka implementacja odwzorowania jest stosowana w przypadku, gdy tworzony system oparty jest na obiektach, a system bazy danych pracuje na relacjach.[11]

Hibernate to jeden z najbardziej popularnych bibliotek, która służy do mapowania obiektowo-relacyjnego w języku programowania Java. Jest on wykorzystywany do translacji danych z relacyjną bazą danych a obiektami. Bazuje on na języku XML i opiera się na licencji Open Source. Polepsza on produktywność działań w bazie danych poprzez minimalizację i buforowanie liczby pytań. Hibernate wyróżnia się najbogatszym API z obecnie dostępnych na rynku.[12]

Aby Hibernate współpracował z omawianym projektem, wymagane jest, aby w każdej utworzonej tabeli znalazł się odnośnik, np. dla tabeli departments powinno znaleźć się:

```
@Table(name = "departments")
```

Natomiast w tabeli bezpośrednio działającej z Hibernatem musiało znaleźć się, otwarcie sesji do wymiany danych i wpisanie ich bezpośrednio do Postgresa.

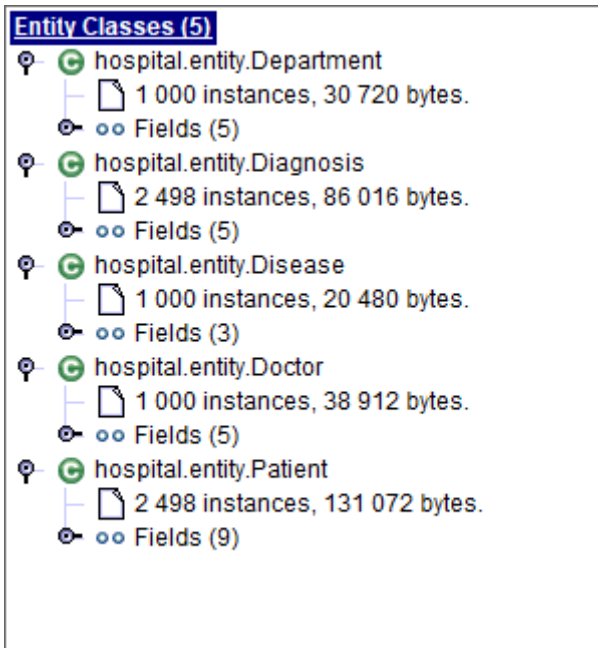
```
Session session = HibernateConnector.getInstance().getSession();

Helper.generateRelationalDatabase(session);

System.out.println("Session created!");

session.close();
```

11. Porównanie

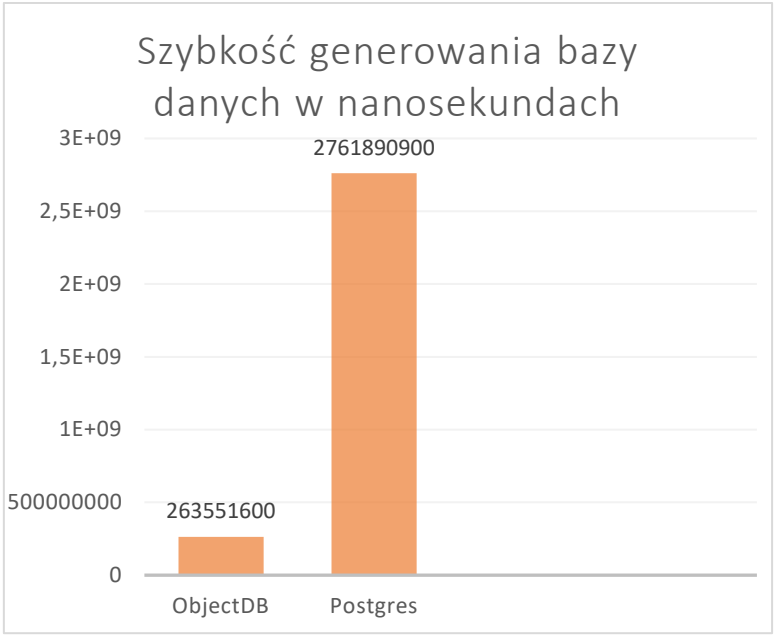
Table name	Tuples inserted	Tuples updated	Tuples deleted	Tuples HOT updated	Live tuples	Dead tuples
departments	1000	1000	0	0	1000	1000
diagnoses	2520	2520	0	0	2520	2520
diseases	1000	0	0	0	1000	0
doctors	1000	0	0	0	1000	0
patients	2520	2520	0	0	2520	2520

Rys. 2 Informacje na temat tabel w pgAdmin 4.

11https://pl.wikipedia.org/wiki/Mapowanie_obiektowo-relacyjne
[12]https://vavatech.pl/technologie/frameworki/hibernate

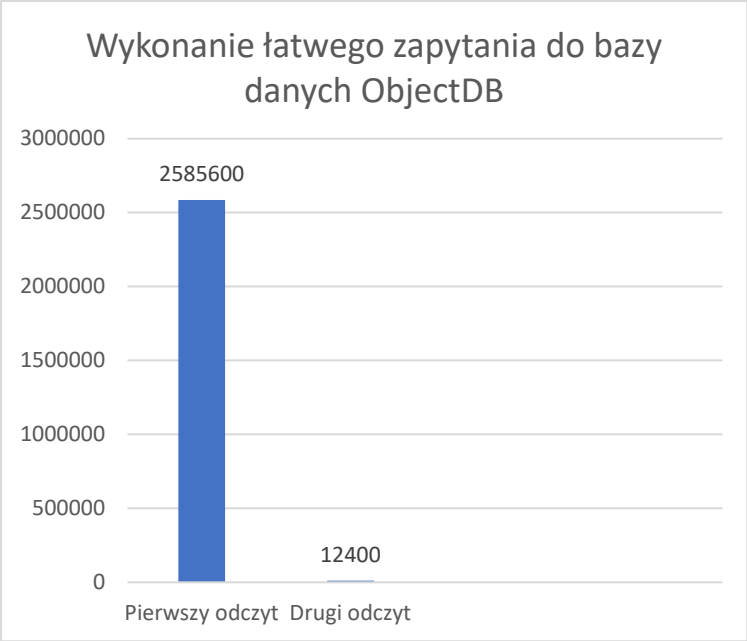


Rys. 3 Informację o tabelach w ObjectDB Explorer.



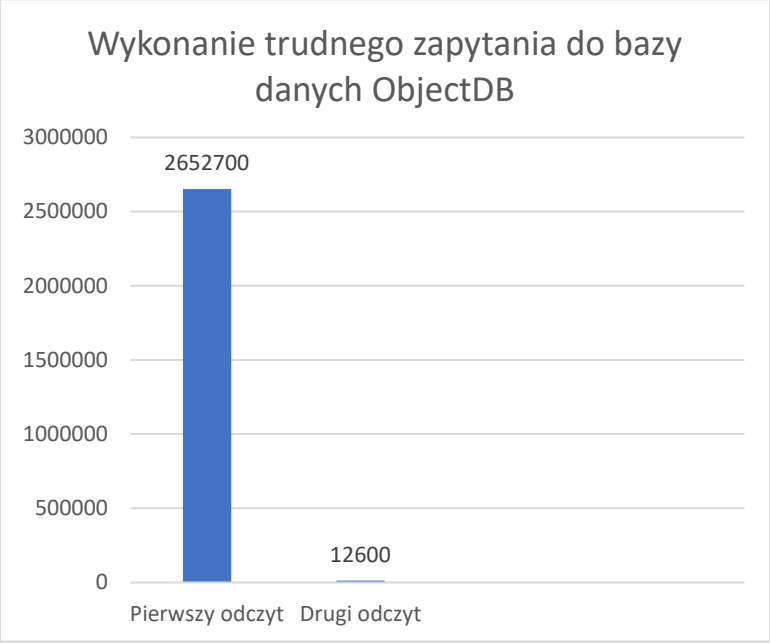
Rys. 4. Wykres zawiera informację o szybkości generowania bazy danych, czas podany w nanosekundach.

Wygenerowanie bazy danych w Postgres zajęło około 2,7 sekund. Natomiast ObjectDB zajęło to niespełna 0,3 sekundy.



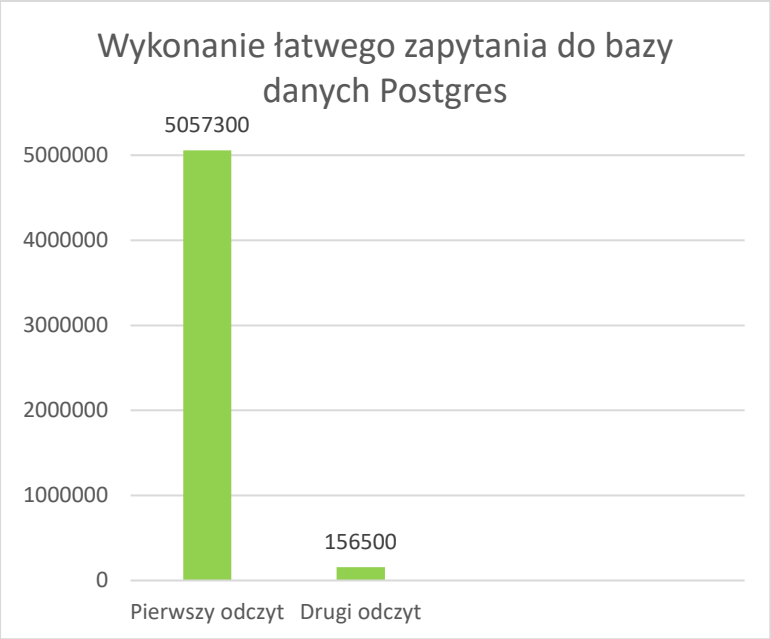
Rys. 5. Wykres zawiera porównanie odczytu zapytania z dysku i z pamięci podręcznej dla łatwego zapytania.

Wykonanie „łatwego" polecenia w OdbjectDB: SELECT d FROM Department d WHERE d.id = 101" - w pierwszym odczycie zajęło 2585600 nanosekund, natomiast już drugie uruchomienie wykonało się w niespełna 12400 ns.



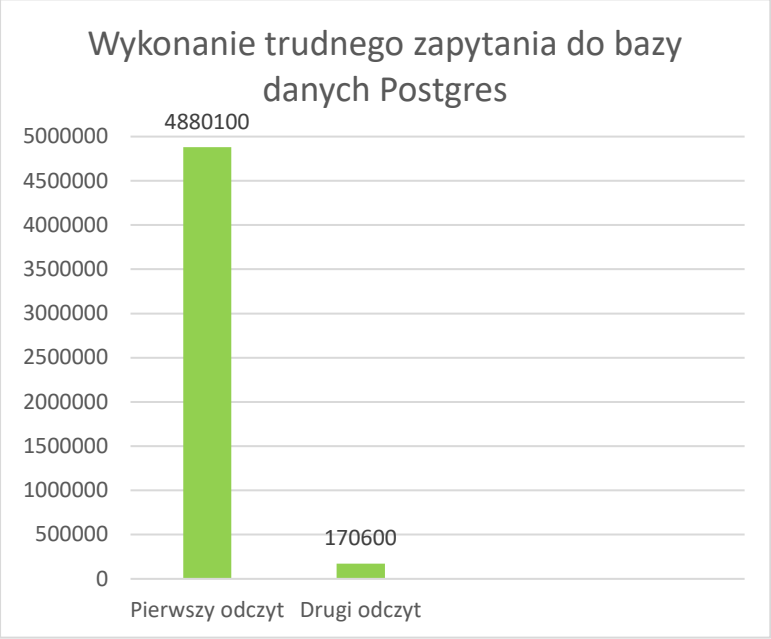
Rys.6 Wykres zawiera porównanie odczytu zapytania z dysku i z pamięci podręcznej dla trudnego zapytania.

Złożone zapytanie do ObjectDB: SELECT d,p FROM Department d INNER JOIN d.patients p WHERE p.drugsReceived = 4 AND p.sex = 'Female'. Dłużej wykonuje się pierwszy odczyt – 2652700 ns, natomiast drugi to już 12600 ns.



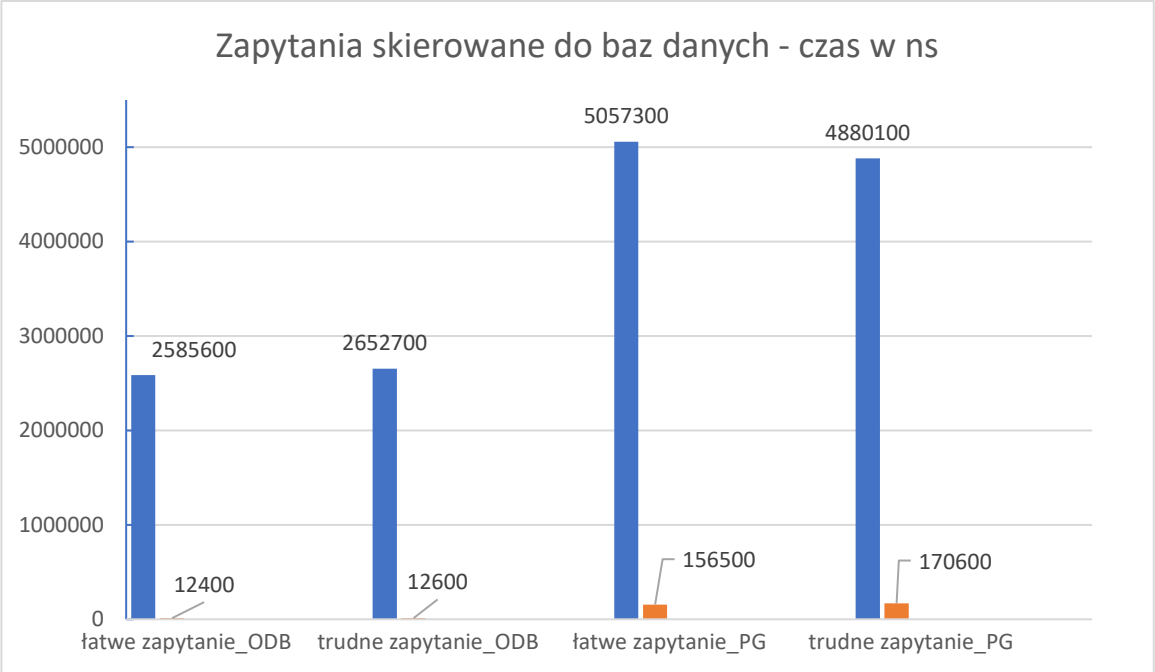
Rys. 7. Wykres zawiera porównanie odczytu zapytania z dysku i z pamięci podręcznej dla łatwego zapytania.

Pierwszy odczyt ze mało skomplikowanym zapytaniem do bazy Postgres: SELECT * FROM Departments WHERE id = 106, zajęło 5057300 nanosekund, natomiast drugi odczyt wyniósł już tylko 156500.



Rys. 8. Prezentacja czasu zajmującego wykonanie złożonego zapytania do bazy Postgres.

Zapytanie: SELECT Departments.id, Departments.departmentname, Departments.numofbeds, Departments.doctor_id, Patients.id as asd, Patients.birthdate, Patients.drug, Patients.drugsreceived, Patients.hospitalizationdate, Patients.name, Patients.sex, Patients.department_id FROM Departments INNER JOIN Patients ON Departments.id = Patients.department_id WHERE Patients.drugsreceived = 4 AND Patients.sex = 'Female', w pierwszym uruchomieniu zajęło 4880100 nanosekund, drugi odczyt już tylko 170600.



Rys. 9 Przedstawienie zebranych zapytań do baz danych.

Wszystkie powyższe wyniki zostały zebrane do jednego wykresu pokazującego jak wyglądają wyniki zapytań, które zostały do niej skierowane. Gołym okiem widać, ze o wiele lepsze są czasy dostępu dla bazy obiektowej, aniżeli relacyjnej. OdjectDB łatwiej sobie radzi z każdym zapytaniem, bo pomijany jest proces mapowania relacyjnego.