

**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 3](#_Toc101634643)

[2. Omówienie obsługiwanych standardów (np. ODMG, JDO, LINQ, XQuery) 3](#_Toc101634644)

[3. Instalacja i konfiguracja OBD 4](#_Toc101634645)

[4. Zapisywanie, aktualizacja i usuwanie obiektów 4](#_Toc101634646)

[5. Metody pobierania obiektów 4](#_Toc101634647)

[6. Dziedziczenie, polimorfizm, hermetyzacja, abstrakcja (abstrakcyjność) 4](#_Toc101634648)

[7. Porównanie z relacyjnymi bazami danych 4](#_Toc101634649)

# Wstęp

Baza danych (angielskie database) jest to rodzaj komputerowego zbioru kartotek, magazyn danych o określonej budowie. Baza danych jest modelowym ujęciem fragmentu rzeczywistości będącego przedmiotem zainteresowania (universe of discours) osób, instytucji, organizacji, firm, zakładów itp., reprezentującym fakty dotyczące tej rzeczywistości w formie umożliwiającej ich przetwarzanie w komputerze. Istotne obiekty danego przedmiotu zainteresowania 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 noszą nazwę transakcji.[[1]](#footnote-1)

Obiektowe bazy danych są propozycją nowego, uniwersalnego i rozszerzalnego modelu danych dla systemów baz danych. W momencie ich pojawienia się miały zastąpić relacyjny model danych, jako lepiej przystosowane do nowych dziedzin zastosowań systemów baz danych. Na początku lat osiemdziesiątych, rozpoczęły się próby stosowania systemów baz danych w nowych dziedzinach, takich jak, systemy wspomagania projektowania, systemy informacji przestrzennej lub systemy multimedialne. Charakterystyka tej klasy zastosowań jest diametralnie odmienna od relacyjnej. Przetwarzane i składowane dane są złożone strukturalnie. Typowe są hierarchicznie złożone struktury danych oraz liczne i intensywnie przetwarzane powiązania między danymi. Powiązanie te mają złożoną semantykę: referencji, agregacji lub kompozycji. Informacje, które są przetwarzane w nowych dziedzinach zastosowań to długie dokumenty tekstowe, obrazy, animacje, 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. Dlatego też postanowiono iść w te rozwiązanie.

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

Historycznie pierwszym rozwiązaniem była budowa systemów baz danych od nowa na bazie obiektowych języków programowania. Rozwiązanie to polega na rozszerzeniu funkcjonalności obiektów tworzonych i przetwarzanych przez języki obiektowe o własności typowe dla danych przechowywanych w systemach baz danych, to jest: trwałości, współdzielenia, synchronizacji dostępu, odtwarzania spójnego stanu po awarii, wydajnego przetwarzania dużych zbiorów obiektów, itp. Wynikiem tej strategii są "czysto" obiektowe bazy danych posiadające jednorodny obiektowy model danych. 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]](#footnote-2)

LINQ to SQL to składnik programu .NET Framework w wersji 3.5, który zapewnia infrastrukturę czasu uruchomieniowego do zarządzania danymi relacyjną jako obiektami.

**Uwaga**

Dane relacyjne są wyświetlane jako kolekcja tabel dwuwymiarowych *(relacje* lub pliki *płaskie), gdzie* wspólne kolumny wiążą tabele ze sobą. Aby efektywnie LINQ to SQL, musisz znać podstawowe zasady relacyjnych baz danych.

W LINQ to SQL model danych relacyjnej bazy danych jest mapowany na model obiektów wyrażony w języku programowania dewelopera. Po wykonaniu aplikacji LINQ to SQL na SQL zapytań zintegrowanych z językiem w modelu obiektów i wysyła je do bazy danych w celu wykonania. Gdy baza danych zwraca wyniki, LINQ to SQL je z powrotem na obiekty, z których można pracować we własnym języku programowania.

Deweloperzy korzystający Visual Studio zwykle używają interfejsu Object Relational Designer, który udostępnia interfejs użytkownika do implementowania wielu funkcji LINQ to SQL.

W dokumentacji dołączonej do tej wersji programu LINQ to SQL opisano podstawowe bloki konstrukcyjne, procesy i techniki potrzebne do tworzenia LINQ to SQL aplikacji. Możesz również wyszukać Microsoft Docs problemów i wziąć udział w [forum LINQ](https://social.msdn.microsoft.com/forums/home?forum=linqtosql), na którym można szczegółowo omawiać bardziej złożone tematy z ekspertami. Na koniec w [LINQ to SQL. .NET Language-Integrated Query for Relational Data white paper](https://docs.microsoft.com/pl-pl/previous-versions/dotnet/articles/bb425822(v=msdn.10)) details LINQ to SQL technology (Zapytanie o dane relacyjne dla danych relacyjnych) wraz z Visual Basic przykładami kodu w języku C#.[[3]](#footnote-3)

XQuery jest językiem służącym do „odpytywania” dokumentów XML, wydobywania informacji z dokumentów XML. W swoich założeniach ma być tym dla XML, czym SQL jest dla relacyjnych baz danych. Zastosowania XQuery często wiążą się właśnie z bazami danych.

Warto tutaj zauważyć, że dokument XML to niekoniecznie plik tekstowy zapisany na dysku ani nawet strumień znaków „w biegu” reprezentujących dokument. Struktura dokumentu XML może istnieć tylko w warstwie logicznej, a pod nią znajdować się zoptymalizowane źródło danych, np. relacyjna lub relacyjno-obiektowa baza danych.

Dzięki temu za pomocą XQuery można odpytywać się o dane z bazy danych. Wątpliwe by było to efektywniejsze od SQL, ale gdyby w tabelach bazy danych zapisane były fragmenty XML, XQuery pozwala na przezroczysty dostęp do danych wewnątrz fragmentów XML. Oczywiście wymaga to wsparcia w SZBD i takie wsparcie już występuje w głównych komercyjnych systemach (Oracle, IBM, Microsoft). W XQuery istnieje możliwość definiowania własnych funkcji, których następnie można używać w wyrażeniach (także w innych funkcjach). Funkcje powinny być zadeklarowane w prologu dokumentu XQuery (przed ciałem zapytania, obok innych deklaracji).[[4]](#footnote-4)

# Instalacja i konfiguracja OBD

# Zapisywanie, aktualizacja i usuwanie obiektów

# Metody pobierania obiektów

# Dziedziczenie, polimorfizm, hermetyzacja, abstrakcja (abstrakcyjność)

Dziedziczenie - Klasy mogą być specjalizowane przez mechanizm dziedziczenia. Klasa pochodna dziedziczy funkcjonalność i implementację klasy bazowej. W klasie pochodnej można dodać nową funkcjonalność lub redefiniować funkcjonalność odziedziczoną. W obiektowych bazach danych można definiować nowe klasy i interfejsy wywodząc je ze zdefiniowanych wcześniej klas lub interfejsów. Model ODMG rozróżnia dwa typy powiązań między klasami lub interfejsami: związek relacji podtypu i związek dziedziczenia. W językach obiektowych te dwa związki są traktowane jako tożsame. Związek relacji podtypu może dotyczyć klas i interfejsów. Oznacza on dziedziczenie przez typ pochodny funkcjonalności typu bazowego. Klasy i interfejsy połączone związkiem podtypu tworzą sieć powiązań o topologii grafu acyklicznego skierowanego. Oznacza to, że pojedyncza klas lub interfejs może dziedziczyć funkcjonalność po wielu klasach lub interfejsach. Związek dziedziczenia łączy jedynie klasy. Oznacza on dziedzicznie zarówno funkcjonalności jak i implementacji. Związek dziedziczenia obejmuje semantykę relacji podtypu. Klasy połączone związkiem dziedziczenia tworzą sieć powiązań o topologii hierarchii. Oznacza to, że pojedyncza podklasa może dziedziczyć zarówno funkcjonalność jak i implementację po dokładnie jednej nadklasie. Dziedziczona funkcjonalność może być rozszerzona w stosunku do typu bazowego. Dziedziczona implementacja może być rozszerzana lub przesłaniana. Przez przesłaniane rozumie się definicje nowego kodu dla odziedziczonych metod, który zastąpi stary kod.

Polimo - O obiektach składowanych w rozszerzeniach klas, które mają klasy pochodne mówi się, że są polimorficzne, bo mają różne cechy i metody. Na przykład wystąpienia klasy Figura mają określony atrybut „Typ” służący do przechowywania informacji o typie figury oraz metodę „Powierzchnia” służącą 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 polimorficznych i podstawień polimorficznych. Przez zmienną polimorficzną rozumie się taką zmienną, pod którą są podstawiane obiekty, które są wystąpieniami różnych klas, ale 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 pokazanym 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.

Abstrakcja - Relacyjne bazy danych oferują projektantowi schematu bazy danych ograniczony zbiór prostych, predefiniowanych typów danych dla atrybutów relacji. Przykładem mogą być systemowe typy tekstowe: varchar i char, numeryczne: int i float oraz temporalne: date. Typy danych niedostępne w relacyjnym modelu danych muszą być implementowane poza systemem bazy danych. Kod obsługujący ich semantykę musi być implementowany 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 typu. Definicje typów są składowane w systemie bazy danych. Sposób korzystania z typów danych zdefiniowanych przez użytkownika nie różni się niczym od korzystania z typów systemowych. Dzięki temu obiektowy model danych jest rozszerzalny. Można go elastycznie dopasowywać do dowolnej dziedziny zastosowania wymagającej unikalnych, specyficznych 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. [[5]](#footnote-5)

Hermetyzacja (ang. encapsulation) polega na grupowaniu elementów składowych w obrębie jednej bryły i umożliwieniu manipulowania tą bryłą jako całością. Hermetyzacja wiąże się z ukrywaniem pewnych informacji dotyczących struktury i implementacji wnętrza tej bryły. Hermetyzacja jest podstawową techniką abstrakcji, czyli ukrycia wszelkich szczegółów danego przedmiotu lub bytu programistycznego, które na danym etapie rozpatrywania (analizy, projektowania, programowania) nie stanowią jego istotnej charakterystyki. Pojęcie hermetyzacji, jako jedna z zasad inżynierii oprogramowania, zostało sformułowane przez D. Parnasa w roku 1975. Można wyróżnić dwie koncepcje hermetyzacji: hermetyzacja ortodoksyjna oraz hermetyzacja ortogonalna. Pierwsza z nich, hermetyzacja ortodoksyjna, jest dość popularnym stereotypem w obiektowości. Ten rodzaj hermetyzacji został zaimplementowany między innymi w języku Smalltalk. W tym podejściu wszelkie operacje, jakie można wykonać na obiekcie, są określone przez metody do niego przypisane (znajdujące się w jego klasie i nadklasach). Bezpośredni dostęp do atrybutów obiektu jest niemożliwy. Drugim rodzajem jest hermetyzacja ortogonalna, zaimplementowana między innymi w językach C++ i Eiffel. W tym przypadku dowolny atrybut i metoda obiektu mogą być prywatne (czyli niedostępne z zewnątrz), bądź też publiczne.[[6]](#footnote-6)

# Porównanie z relacyjnymi bazami danych

https://www.predictiveanalyticstoday.com/top-object-databases/

1. Płoski Z; Słownik Encyklopedyczny – Informatyka,. wyd. Europa,. Wrocław 1999 [↑](#footnote-ref-1)
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 [↑](#footnote-ref-2)
3. https://docs.microsoft.com/pl-pl/dotnet/framework/data/adonet/sql/linq/ [↑](#footnote-ref-3)
4. https://www.mimuw.edu.pl/~czarnik/zajecia/xml06/lab07/lab07.html [↑](#footnote-ref-4)
5. 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 [↑](#footnote-ref-5)
6. Paweł Józwik, Maciej Mazur; Obiektowe bazy danych – przegląd i analiza rozwiązań; http://www.kapitanat.pl/odb.pdf [↑](#footnote-ref-6)