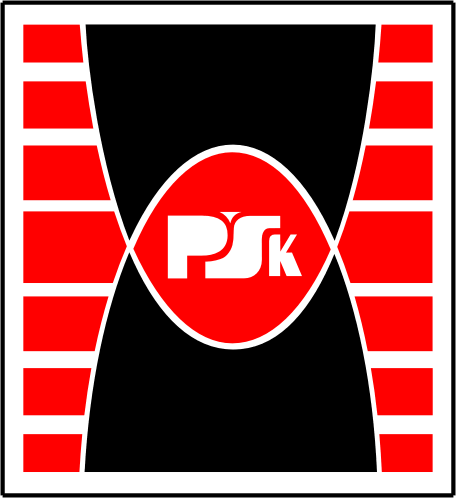
****

**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-2)

Obiektowe bazy danych są propozycją nowego, 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 danych. Na początku lat 80-dziesiątych, rozpoczęły się próby zastosowania systemu obiektowych baz danych w nowych dziedzinach, np. systemy wspomagania projektowania, systemy multimedialne lub systemy informacji przestrzennej. Charakterystyka tych zastosowań jest bardzo odmienna od relacyjnej. Składowane oraz przetwarzanedanych 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 lub też kompozycji. Informacje, które są przetwarzane w nowych dziedzinach zastosowań to 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.

# 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: współdzielenia, trwałości, synchronizacji dostępu, odtwarzania spójnego stanu po awarii, wydajnego przetwarzania dużych zbiorów obiektów, itp. Wynikiem tej strategii są obiektowe bazy danych, które posiadają 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-3)

Natomiast LINQ to SQL to składnik programu .NET Framework w wersji 3.5, który zapewnia infrastrukturę czasu uruchomieniowego do zarządzania 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 relacyjnej bazy danych jest mapowany na model obiektów wyrażony w języku programowania. Po wykonaniu aplikacji LINQ to SQL na SQL zapytań zintegrowanych z językiem w modelu obiektów wysyła je do bazy danych w celu wykonania. Gdy baza danych zwraca wyniki, LINQ to SQL przekształca je z powrotem na obiekty. W dokumentacji dołączonej do tego standardu opisano podstawowe bloki konstrukcyjne, procesy, a także techniki potrzebne do tworzenia aplikacji. [[3]](#footnote-4)

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

Za pomocą XQuery można odpytywać dane z bazy danych. Nie jest to efektywniejsze od SQL, ale gdyby w tabelach bazy danych były zapisane fragmenty XML, XQuery pozwala na przezroczysty dostęp do danych, które są wewnątrz fragmentów XML. Wymaga to również wsparcia w SZBD i takowe wsparcie jest już oferowane w głównych komercyjnych systemach tj. Oracle, IBM, Microsoft. W XQuery oferuję możliwość definiowania też 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]](#footnote-5)

# 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.
* Rozwiązanie przetestowane z Tomcat, Jetty, GlassFish, JBoss i Spring.[[5]](#footnote-6)

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, points oraz world?, także można poznać jak to działa.

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. Rozwijane jest ono od 2001 roku, a swoją popularność zawdzięcza m.in. wbudowanym narzędziom do refaktoringu, czyli procesu modyfikacji istniejącego oprogramowania z jednoczesnym zachowaniem przyjętych wzorców oraz ogólnej jakości kodu.

IDE oferuje programiście wszystkie niezbędne narzędzia i funkcje, dające możliwość wygodnego pisania kodu, a także jego późniejszej rozbudowy. Znajdziemy wśród nich inteligentny edytor kodu wraz z opcją kolorowania składni i autouzupełniania, narzędzia do refaktoryzacji i inspekcji kodu, wbudowany edytor XML oraz mechanizmy do kontroli wersji, obsługujące systemy CVS, Subversion, Mercurial i Git, a także wizualne kreatory interfejsów graficznych opartych na bibliotece Swing. Środowisko wspiera 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. Najnowsza wersja IntelliJ IDEA Community Edition oferuje pełne wsparcie dla Javy 7 oraz pakietu SDK dla systemu Android.[[6]](#footnote-7)

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.

# Zapisywanie, aktualizacja i usuwanie obiektów

# Metody pobierania obiektów

# Dziedziczenie, polimorfizm, hermetyzacja, abstrakcja (abstrakcyjność)

Dziedziczenie można rozumieć jako klasa pochodną, która dziedziczy funkcjonalność i implementację klasy bazowej. W klasie pochodnej można dodać nową funkcjonalność lub zredefiniować funkcjonalność odziedziczoną. W obiektowych bazach danych można definiować nowe klasy i interfejsy na podstawie zdefiniowanych już 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, jednakowe. Związek relacji podtypu może dotyczyć klas, czy interfejsów, 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 i wyłączenie klasy. Co za tym idzie, dziedziczy zarówno funkcjonalności jak i implementacje. Związek dziedziczenia obejmuje semantykę relacji podtypu. Klasy połączone związkiem dziedziczenia tworzą ze sobą 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 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, bo mają różne cechy i metody. Na przykład wystąpienie klasy Figura ma określony atrybut „Typ” służący do przechowywania danych o typie figury oraz 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]](#footnote-8)

Hermetyzacja polega na grupowaniu elementów składowych w obrębie jednego obiektu i umożliwieniu manipulowania nim jako całością. Hermetyzacja wiąże się też z ukrywaniem pewnych informacji dotyczących struktury i implementacji zawartości. Jest podstawową techniką abstrakcji, czyli ukrycia wszelkich szczegółów danego przedmiotu lub bytu programistycznego, które na danym etapie rozpatrywania (analizy, projektowania, programowania) nie stanowią istotnej charakterystyki. Pojęcie hermetyzacji, jako jedna z zasad inżynierii oprogramowania, zostało sformułowane w roku 1975 przez D. Parnasa. Możliwe jest wyróżnienie dwóch koncepcji hermetyzacji: hermetyzacja ortodoksyjna lub hermetyzacja ortogonalna. Pierwsza z nich, jest dość popularnym stereotypem w obiektowości. Ten rodzaj hermetyzacji został zaimplementowany między innymi w języku Smalltalk. W tym podejściu wszelkie operacje, które można wykonać na obiekcie, są określone przez metody do których są przypisane - znajdują się w jego klasie i nad klasach. Natomiast bezpośredni dostęp do atrybutów obiektów 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), lub publiczne.[[8]](#footnote-9)

# Porównanie z relacyjnymi bazami danych

Pod pojęciem abstrakcji, kryję się. 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.[[9]](#footnote-10)

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

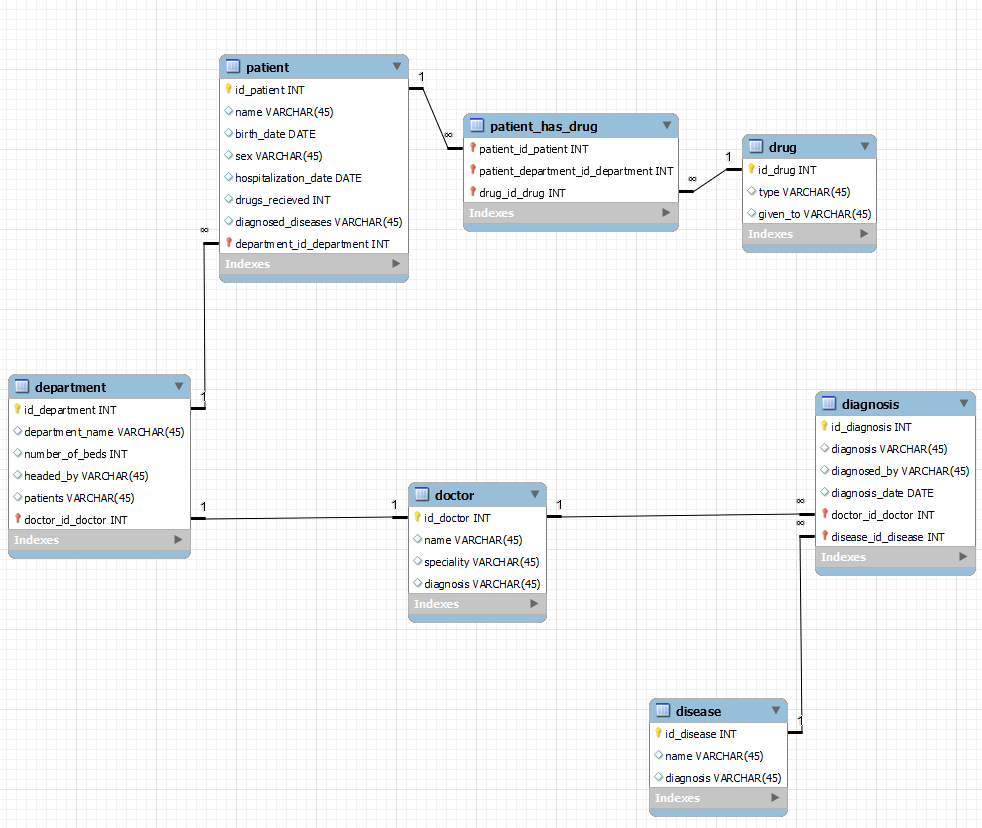
Opis relacyjnej bazy danych – MySQL Workbench

MySQL Workbench to narzędzie do zarządzania i modelowania baz danych MySQL. Za jego pomocą można edytować konfigurację serwera i jego komponentów, a także zaprojektować i stworzyć schematy (wizualne reprezentacje tabel, widoków itp.) nowych baz danych, wykonać dokumentację istniejących oraz zapewnić wsparcie przy procesach migracji do MySQL.

Główne cechy narzędzia to oparcie o silnik graficzny OpenGL, wsparcie dla procesów reverse-engineeringu i synchronizacji baz danych, możliwość generowania skryptów SQL, przeglądowy tryb pracy w którym cały model bazy jest prezentowany w jednym przekrojowym widoku, wsparcie dla projektowania baz na poziomach koncepcyjnym, logicznym i fizycznym, rozszerzalna architektura, eksport modelu jako skryptu SQL typu CREATE, import i eksport modeli DBDesigner4, wizualna reprezentacja tabel, widoków, procedur wbudowanych oraz funkcji oraz pełne wsparcie dla możliwości MySQL 5.

Aplikacja łączy możliwości nierozwijanych już narzędzi dostępnych niegdyś w ramach MySQL GUI Tools. W MySQL Workbench znajdziemy więc funkcjonalność między innymi MySQL Administrator i MySQL Query Browser. Pomimo tego, że w niektórych wersjach zdarzają się problemy ze stabilnością aplikacji, to jednak nadal jest to najlepsze narzędzie dla administratora, jak i programisty MySQL, pozycja obowiązkowa dla każdego środowiska wyposażonego w serwer MySQL.

https://www.dobreprogramy.pl/mysql-workbench,program,windows,6628677367437441

Tematyka bazy danych

Zdecydowałem się na temat Szpitala.

Tabla patient – pacjent zawiera informację o jego imieniu i nazwisku, dacie urodzenia, płci, kiedy został przyjęty do szpitala, ilości otrzymanych leków, o jego diagnozie i na jakim oddziale aktualnie przebywa.

Tabela department – oddział, w niej znajdziemy informacje nt. Nazwy oddziału, liczby łóżek, kto zarządza danym oddziałem, wraz z liczbą pacjentów – zmienić na INT i ID doktora, który pracuje na tym oddziale.

1. Płoski Z; Słownik Encyklopedyczny – Informatyka,. wyd. Europa,. Wrocław 1999 [↑](#footnote-ref-2)
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-3)
3. https://docs.microsoft.com/pl-pl/dotnet/framework/data/adonet/sql/linq/ [↑](#footnote-ref-4)
4. https://www.mimuw.edu.pl/~czarnik/zajecia/xml06/lab07/lab07.html [↑](#footnote-ref-5)
5. https://www.objectdb.com/database/overview [↑](#footnote-ref-6)
6. https://www.dobreprogramy.pl/intellij-idea-community-edition,program,windows,6628622615124097 [↑](#footnote-ref-7)
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 [↑](#footnote-ref-8)
8. Paweł Józwik, Maciej Mazur; Obiektowe bazy danych – przegląd i analiza rozwiązań; http://www.kapitanat.pl/odb.pdf [↑](#footnote-ref-9)
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 [↑](#footnote-ref-10)