MINISTERSTWO ŚRODOWISKA

Sfinansowano ze środków Narodowego Funduszu Ochrony Środowiska i Gospodarki Wodnej

TRAPPA

Szkolenie
Język SQL w bazie PostgreSQL

Część teotyczna

Prowadzący

Michał Bednarczyk

Sfinansowano ze środków Narodowego Funduszu Ochrony Środowiska i Gospodarki Wodnej

Wprowadzenie

Wprowadzenie do relacyjnych baz danych i bazy PostgreSQL/PostGIS

Czym jest System Zarządzania Bazą Danych

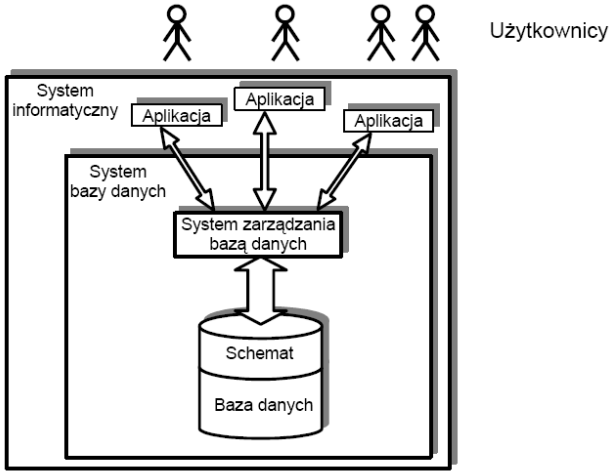
Baza danych

- Duża, uporządkowana kolekcja danych

- *Model - reprezentacja danych świata rzeczywistego*
  - *Encje* (np. Studenci, Zajęcia)
  - *Relacje* (np. Marysia uczęszcza na zajęcia z baz danych)

System Zarządzania Bazą Danych (DBMS) to oprogramowanie zaprojektowane do przechowywania i zarządzania bazą danych

Terminologia



Charakterystyka baz danych (1)

1. Trwałość danych
   – Długi czas życia – kilka, kilkadziesiąt, kilkaset lat
   – Niezależność od działania aplikacji
2. Rozmiar wolumenu danych
   – Dane nie mieszczą się w pamięci operacyjnej – wymagana pamięć zewnętrzna
   – Danych jest zbyt dużo dla ich liniowego przeglądania przez użytkowników

Charakterystyka baz danych (2)

3. Złożoność danych
   – Złożoność strukturalna i złożoność zależności pomiędzy danymi
   – Złożoność semantyczna
   – Ograniczenia integralnościowe

Wymagania (1)

- Spójność bazy danych
- Efektywne przetwarzanie danych
- Poprawne modelowanie świata rzeczywistego
- Autoryzacja dostępu do danych
- Współbieżność dostępu do danych
- Metadane

Wymagania (2)

1. Spójność bazy danych
   - Poprawność danych z punktu widzenia przyjętych kryteriów
   - wierne odzwierciedlenie danych rzeczywistych
   - spełnienie ograniczeń nałożonych przez użytkowników

Wymagania (3)

- Spójność bazy danych cd.
  – Odporność na anomalie będące wynikiem współbieżności dostępu do baz danych
  – Odporność na błędy, awarie i inne anormalne sytuacje wynikające z zawodności środowiska sprzętowo-programowego
  – Odporność na błędy użytkowników

Wymagania (4)

2. Efektywne przetwarzanie danych
   – Efektywne metody dostępu do danych
   – Optymalizacja metod dostępu do danych
   – Niezależność aplikacji od fizycznych metod dostępu

Wymagania (5)

3. Poprawne modelowanie świata rzeczywistego
   – Wspomaganie procesu projektowania i utrzymania bazy danych
   – Różne poziomy modelowania danych – Transformacje między modelami danych
4. Autoryzacja dostępu do danych
   – użytkownicy z hasłami dostępu
   – użytkownicy i ich uprawnienia

Wymagania (6)

5. Współbieżność dostępu do danych
   – równoczesny dostęp do tych samych danych przez wielu użytkowników
   – konflikt odczyt-zapis, zapis-zapis
6. Metadane
   – dane o danych, strukturach dostępu, użytkownikach i ich prawach

Technologia baz danych (1)

1. Fizyczne struktury danych i metody dostępu
   – Pliki uporządkowane, haszowe, zgrupowane,indeksy drzewiaste i bitmapowe
   – Metoda połowienia binarnego, haszowanie statyczne i dynamiczne, metody połączenia, sortowanie, grupowanie
   – Składniowe i kosztowe metody optymalizacji dostępu
   – Fizyczna niezależność danych

Technologia baz danych (2)

2. Przetwarzanie transakcyjne (spójność baz danych)
   – Dostęp do bazy danych za pomocą transakcji o własnościach ACID
   – Metody synchronizacji transakcji (2PL, znaczniki czasowe, wielowersyjność danych)
   – Metody odtwarzania spójności bazy danych (plik logu, odtwarzanie i wycofywanie operacji, Write Ahead Log, punkty kontrolne)
   – Archiwizacja bazy danych i odtwarzanie po awarii

Technologia baz danych (3)

3. Modele danych
   – Modele pojęciowe (model związków-encji, UML)
   – Modele logiczne (relacyjny, obiektowy, obiektowo-relacyjny, semistrukturalny, hierarchiczny, sieciowy)
4. Narzędzia programistyczne
   – Języki budowy aplikacji
   – Narzędzia modelowania i projektowania
   – Metodyki projektowania

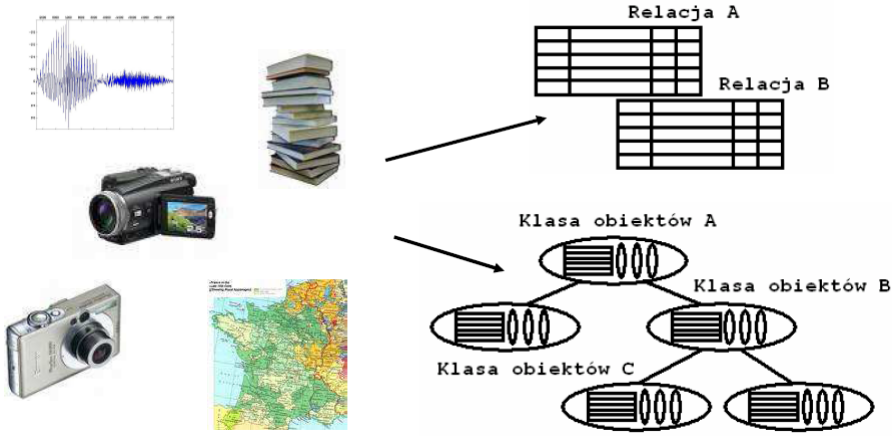
System zarządzania bazą danych (SZBD)

- Oprogramowanie zarządzające całą bazą danych
- Funkcjonalność
  - Język bazy danych - tworzenie, definiowanie, wyszukiwanie i pielęgnacja danych w bazie danych
  - Struktury danych - efektywne składowanie i przetwarzanie dużych wolumenów danych
  - Optymalizacja dostępu do danych
  - Współbieżny dostęp do danych
  - Zapewnienie bezpieczeństwa danych zagrożonego awaryjnością środowiska sprzętowo-programowego
  - Autoryzacja dostępu do danych
  - Wielość interfejsów dostępu do bazy danych

Model danych (1)

Obiekty świata rzeczywistego

Obiekty modelu danych



Model danych (2)

1. Struktury danych

JAN	NOWAK	47
TADEUSZ	KOWALSKI	34
MACIEJ	NOWAK	26
JANINA	RZEPA	19
KUBA	TARZAN	31
JÓZEF	MALINIAK	29
JAN	NOWAK	56

Model danych (3)

2. Operacje (operatory modelu danych)
3. Ograniczenia integralnościowe

Nazwa projektu	Budżet	Data rozpoczęcia	Data zakończenia
Indeksy w BD	500 000,-	1.07.2002	30.06.2005
Magazyny danyc	700 000,-	1.09.2002	31.08.2001

Prosty, działający przykład

- Rozważmy system zarządzania zajęciami (SZZ):
  - Studenci
  - Kursy
  - Wykładowcy

  } *Encje*

  - Kto na co uczęszcza
  - Kto czego uczy

  } *Relacje*

Modele danych - jak korzystamy

- Model danych to zbiór pojęć, zasad i relacji opisujących te dane
  - Obecnie, najpowszechniej używanym modelem jest realcyjny model danych
- **Schemat** jest opisem okrelonego zestawu danych w oparciu o **konkretny model danych**

Modelowanie systemu zarządzania zajęciami

- *Logical Schema*
  - Studenci(sid: *string*, imie: *string*, srednia: *float*)
  - Kursy(cid: *string*, cname: *string*, punkty: *int*)
  - Grupy(sid: *string,* cid*: string, ocena: string*)

Relacje

sid	Imie	Srednia
101	Bob	3.2
123	Mary	3.8

Studenci

cid	cname	punkty
564	564-2	4
308	417	2

Kursy

sid	cid	Ocena
123	564	A

Grupy

Modelowanie systemu zarządzania zajęciami

- *Logical Schema*
  - Studenci(sid: *string*, imie: *string*, srednia: *float*)
  - Kursy(cid: *string*, cname: *string*, punkty: *int*)
  - Grupy(sid: *string,* cid*: string, ocena: string*)

Powiązane klucze

sid	Imie	Srednia
101	Bob	3.2
123	Mary	3.8

Studenci

cid	cname	punkty
564	564-2	4
308	417	2

Kursy

sid	cid	Ocena
123	564	A

Grupy

Inne rodzaje schematów…

- *Schemat fizyczny*: opisuje układ danych
  - Relacje (tabele) jako pliki nieuporządkowane
  - Część danych posortowana (indeksy)
- *Shemat logiczny:* Poprzedni slajd
- *Schemat zewnętrzny*: (View)
  - Kurs_info(cid: *string*, grupa: *integer*)
  - Dane pozyskane z innych tabel, jednej lub wielu

Administratorzy

Aplikacje

Przykładowa baza danych - dlaczego stosujemy więcej niż jedną tabelę?

Pracownicy

IdPrac	Nazwisko	Etat	Szef	DataZat	Płaca	Premia	IdZesp
7340	Kowalski	referent	7548	17.12.90	1800		20
7341	Nowak	asystent	7340	20.02.91	1600	300	30
7342	Tarzan	asystent	7340	22.02.91	1500	500	20
7544	Colargol	kierownik	7600	2.04.91	2500	100	20
7548	Król	księgowy	7600	28.09.91	3500		10
7600	Dziuba	dyrektor		17.10.89	5000		10
7880	Buba	referent	7544	23.05.95	2100		30
7900	Misiek	kierownik	7600	29.10.01	2700		30

Etaty

Nazwa	PłacaMin	PłacaMax
dyrektor	4000	9999
kierownik	2000	5000
referent	1500	2500
asystent	1100	1800
księgowy	2500	4500

Zespoły

IdZesp	Nazwa	Adres
10	Administracyjny	Poznań
20	Sprzedaży	Poznań
30	Reklamy	Gniezno
40	Badań	Oborniki

Niezależność danych

Koncepcja: aplikacje nie muszą się "zastanawiać" jaka jest fizyczna struktura danych i w jakiej formie są przechowywane

Logiczna niezależność danych: ochrona przed zmianami w logicznej strukturze danych

Np. nie powinno zaistnieć pytanie: czy można dodać nową encję lub atrybut bez pisania aplikacji od nowa?

Fizyczna niezależność danych: ochrona przed zmianami w fizycznej strukturze danych

Np. nie powinno zaistnieć pytanie: na którym dysku dane są przechowywane? Czy dane są indeksowane?

Jeden z najważniejszych powodów aby stosować SZBD

Wielodostęp

- Załóżmy, że nasza aplikacja serwuje dane dla tysięcy użytkowników, jakie stoją przed nią wyzwania**?**
  - Bezpieczeństwo: różni użytkownicy, różne role

Nie jest to przedmiotem kursu, lecz jest niezmiernie istotne

  - Wydajność: należy zapewnić dostęp równoległy

Dostęp do dysku jest wolny, SZDB maskuje opóźnienie poprzez przekazanie procesorowi zadań do równoległego przetwarzania.

  - Spójność: równoległy dostęp może powodować problemy z aktualizacją danych

SZBD pozwala użytkownikom tworzyć programy, jakby każdy z nich był **jedynym użytkownikiem** bazy

Transakcje

- Podstawowe założenie **transakcji (TXN)**: **atomowa (niepodzielna)** sekwencja operacji na bazie danych (odczyt/zapis)

Atomowość (niepodzielność): operacja wykonuje się w całości lub wcale.

Przed

Konto	Saldo
k10	20,000
k20	15,000

Przelew $3k z a10 do a20:
1. Odejmij $3k z a10
2. Dodaj $3k do a20

Po

Konto	Saldo
k10	17,000
k20	18,000

W jakich stanach **atomowość** jest zachowana??

- Błąd przed krokiem 1,
- Po 1 ale przed 2,
- Po 2.

BD zawsze zachowuje atomowość!

Transakcje

- Podstawowe założenie **transakcji (TXN)**: **atomowa** sekwencja operacji na bazie danych (odczyt/zapis)
  - Jeżeli użytkownik anuluje transakcję, powinno być tak, jakby nic się nie stało!
- Transakcje pozostawiają bazę danych w stanie **spójnym**
  - Użytkownicy mogą tworzyć ograniczenia integralnościowe (integrity constraints) np.”każde zajęcie przypisane są do dokładnie jednej sali”

Zauważmy jednak, że SZBD nie “rozumie” prawdziwego znaczenia ograniczeń - dbanie o spójność ciągle jest po stronie użytkownika.

Atomowość: operacja wykonuje się w całości lub wcale.

Spójność: Operacja skutkuje takim stanem bazy danych, który jest zgodny ze wszystkimi ograniczeniami integralnościowymi

Zarządzanie równoległymi transakcjami

- SZBD musi zapewnić, że wykonanie zbioru $\{T_1,\ldots,T_n\}$ transakcji, jest równoważne wykonaniu równoległemu
- Jeden ze sposobów aby to osiągnąć: **Blokowanie (Locking)**
  - Przed operacją odczytu lub zapisu, transakcja żąda od SZBD blokady (lock), która utrzymywana jest do końca transakcji
- **Kluczowa koncepcja:** jeżeli T_i zapisuje do elementu x oraz T_j chce odczytać z x, wtedy T_i, T_j są w **konflikcie**. Rozwiązanie poprzez blokowanie:
  - Tylko jedna z nich uzyskuje blokadę
  - Druga jest zablokowana (czeka) na zakończenie pierwszej

Wszystkimi operacjami związanymi z przetwarzaniem transakcji zarządza SZBD

Zapewnienie atomowości i trwałości

- SZBD zapewnia atomowość, nawet w przypadku niepowodzenia transakcji!
- Jednym ze sposobów aby to osiągnąć jest tzw. **Write-ahead logging (WAL)**
- **Kluczowe założenie:** Utrzymywać plik logu z informacją o każdym wykonanym zapisie.
  - W przypadku niepowodzenia, transakcje niedokończone (wykonane częściowo) są wycofywane na podstawie zapisu w pliku log.

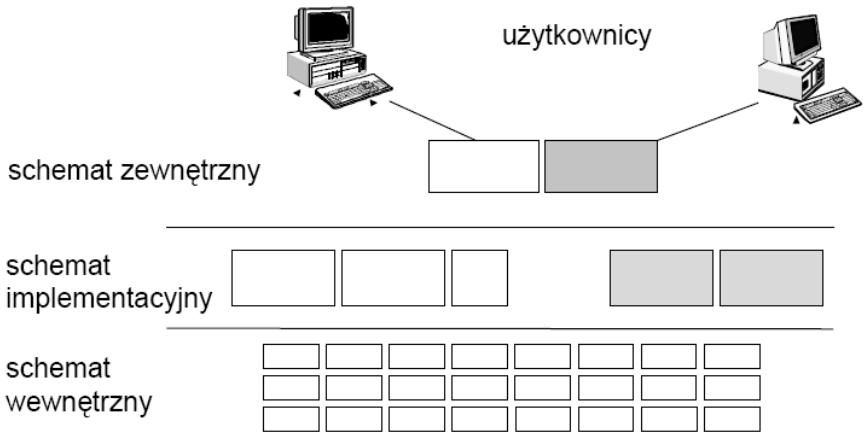
Write-ahead Logging (WAL): Zanim operacja zostanie sfinalizowana, informacja o niej zapisywana jest do pliku log na dysku.

Zakładamy, że plik log przechowywany jest na stabilnym nośniku.

Zadaniami związanymi z atomowością transakcji zarządza SZBD

Architektura systemu bazy danych

- 3-warstwowa architektura wg standardu ANSI/SPARC



Interakcja z bazą danych (1)

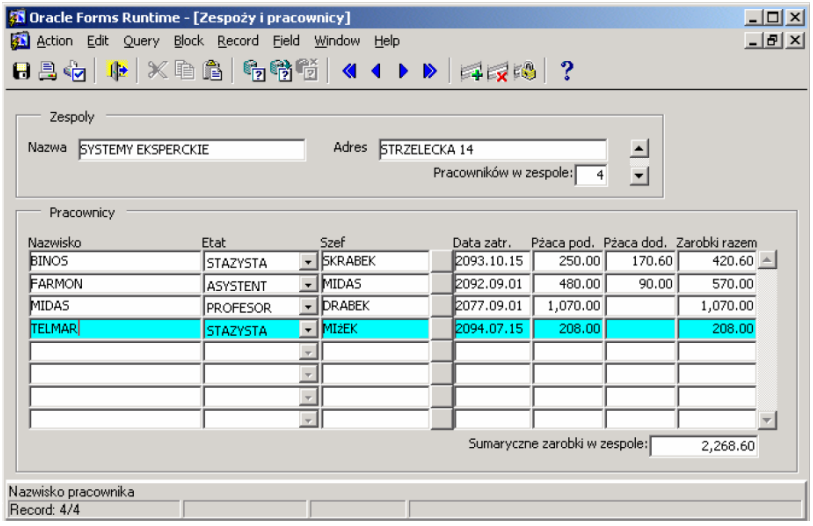
Język SQL
- jedyny sposób interakcji z bazą danych
- język deklaratywny
- ustandaryzowany
  - producenci systemów komercyjnych i niekomercyjnych starają się implementować ten standard

```
SELECT nazwisko, etat, płaca
FROM pracownicy
WHERE idzesp=30 AND etat='kierownik'
```

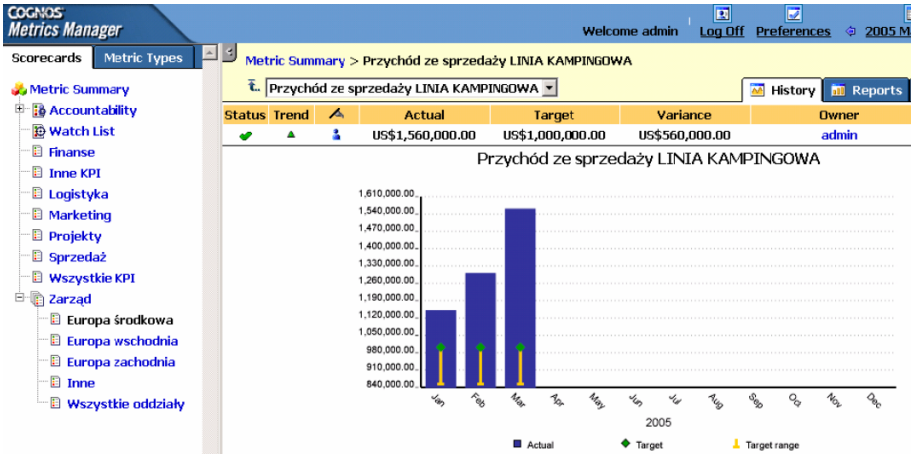
Interakcja z bazą danych (2)

Aplikacje
- formularze
  - elektroniczne formularze z polami, listami, elementami wyboru
  - umożliwiają wstawianie, modyfikowanie, usuwanie, wyszukiwanie danych
- raporty
  - umożliwiają prezentowanie zawartości bazy danych
- teksty
- wykresy
- grafika

Formularz - przykład



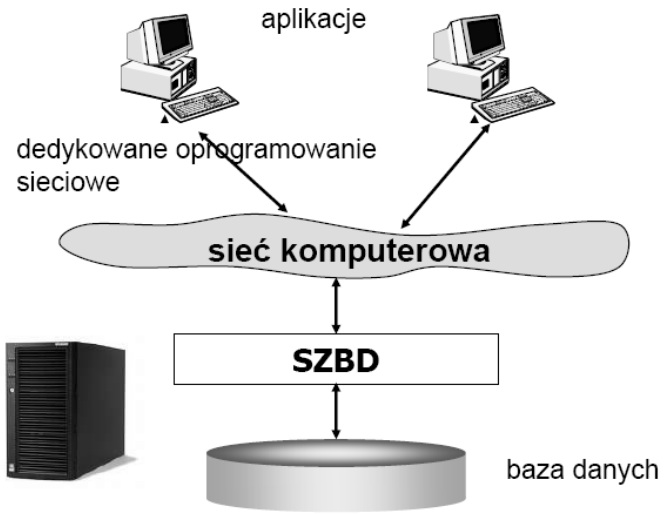
Raport - przykład



Technologie implementacyjne aplikacji

- Języki 3GL
  - np. C, C++, Visual Basic, Visual C++
  - biblioteki umożliwiające zagnieżdżanie poleceń SQL w kodzie
- Języki 4GL
  - np. SAS 4GL, Oracle Forms
  - umoliwiają bezpośrednie umieszczanie poleceń SQL w kodzie aplikacji i bezpośrednią obsługę wyników poleceń SQL
- Java, PHP, Perl
  - stosowane w aplikacjach web'owych pracujących w architekturze 3-warstwowej

Architektura komunikacyjna – klient-serwer



Architektura komunikacyjna – 3 warstwowa

Podział systemów baz danych (1)

- Kryteria podziału
  - wykorzystywany model danych
  - liczba węzłów / baz danych
  - cel stosowania

- **Model danych**
- relacyjny
- obiektowy
- obiektowo-relacyjny
- semistrukturalny (XML)
- hierarchiczny
- sieciowy

- **Liczba węzłów / baz danych**
- bazy scentralizowane
- bazy rozproszone

Podział systemów baz danych (2)

- Cel stosowania
  - przetwarzanie transakcyjne (On-Line Transaction Processing - OLTP)
    - wszelkiego rodzaju systemy ewidencyjne
  - przetwarzanie analityczne (On-Line Analytical Processing - OLAP)
    - hurtownie danych
  - wspomaganie projektowania (Computer Aided Design - CAD)
    - konstrukcje, budynki, urządzenia
  - systemy informacji geograficznej (Geographical Information Systems - GIS)
  - wytwarzanie oprogramowania (Computer Aided Software Engineering - CASE)

Dostępne SZBD (1) - przykłady

- Komercyjne
  - Oracle
    - wersje 9i, 10g, 11g, 12c, 18c
  - IBM
    - DB2 UDB
    - Informix(R) Dynamic Server
  - Microsoft
    - SQL Server2000, SQL Server2005
  - Sybase
    - Adaptive Server Enterprise, Adaptive Server Anywhere

Dostępne SZBD (2) - przykłady

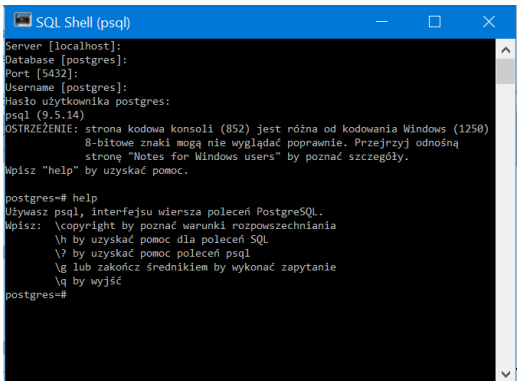
- Niekomercyjne
  - MySQL
  - PostgreSQL
  - FireBird

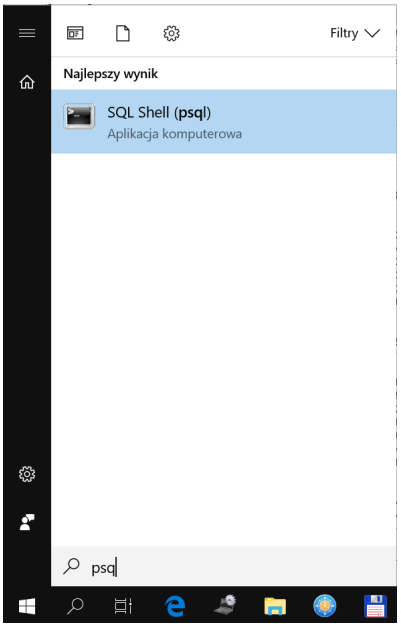
PostgreSQL/PostGIS

- **PostgreSQL** - często nazywany także Postgres to, obok MySQL i Firebird, jeden z trzech najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych. Początkowo opracowywany na Uniwersytecie Kalifornijskim w Berkeley i opublikowany pod nazwą Ingres. W miarę rozwoju i zwiększania funkcjonalności, baza danych otrzymała nazwy Postgres95 i ostatecznie PostgreSQL, aby upamiętnić pierwowzór oraz zaznaczyć zgodność ze standardem SQL. Aktualnie baza implementuje większość standardu SQL:2011
- **PostGIS** – rozszerzenie relacyjno-obiektowej bazy danych PostgreSQL, dodające możliwość zapisywania danych geograficznych wprost do bazy danych zgodnie ze specyfikacją OpenGIS Simple Features dla profilu SQL.

Praca z klientem bazy danych (PgAdmin, psql)

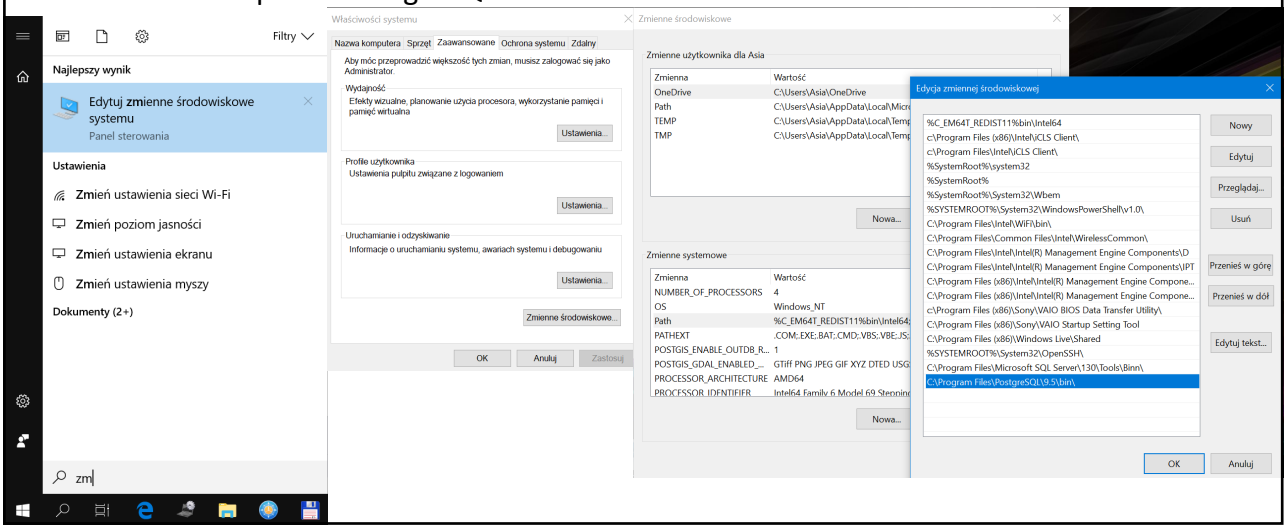
Klient w konsoli - psql



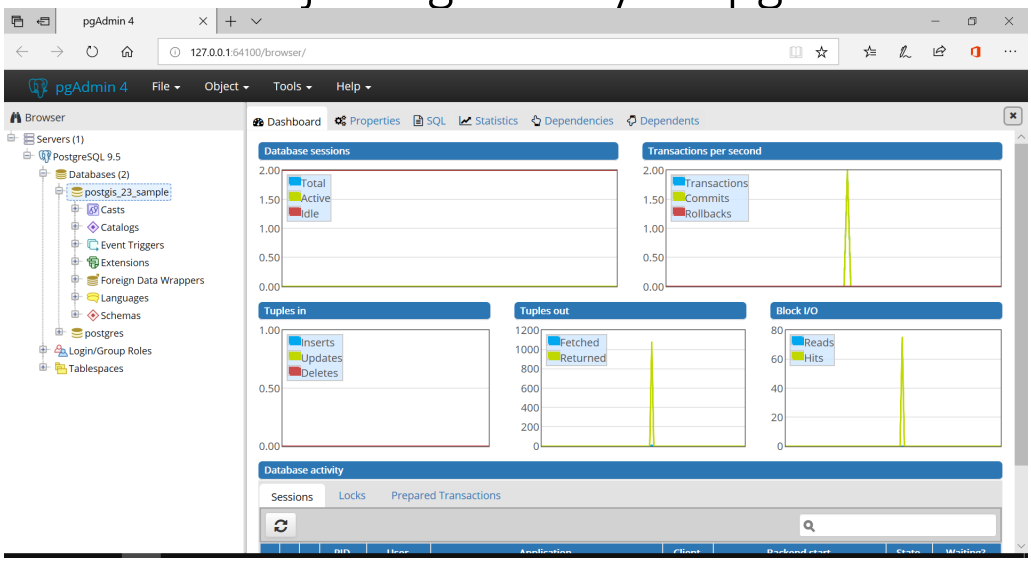


Klient w konsoli systemowej

- psql można uruchomić z poziomu powłoki systemu,
- uprzednio należy upewnić się, że smienna systemowa PATH ma podaną wartość ścieżki do plików PostgreSQL'a



Klient z interfejsem graficznym - pgAdmin



Wstęp do projektowania relacyjnej bazy danych

Normalizacja
Zależności funkcyjne

Zależność funkcyjna (1)

Def: Dane są zbiory atrybutów A iB
Piszemy A → B lub mówimy A ***determinuje funkcyjnie*** B (B zależy funkcyjnie od A) jeżeli, dla dowolnej pary krotek t_1 i t_2:

$$t_1[A] = t_2[A] \text{ implikue } t_1[B] = t_2[B]$$

A → B nazywamy <u>zależnością funkcyjną</u>

A→B oznacza, że
"z równości krotek na atrybutach ze zbioru A wynika równość tych krotek na atrybutach ze zbioru B."

Zależność funkcyjna (2)

W teorii relacyjnych baz danych, zależność funkcyjna (FD) stanowi ograniczenie (constraint) pomiędzy dwoma zbiorami atrybutów relacji w bazie danych.

Lub - innymi słowy, zależność funkcyjna jest ograniczeniem, opisującym związek pomiędzy atrybutami relacji.

Można też powiedzieć, że:

Zależność FD: X → Y oznacza, że wartości zbioru Y są determinowane przez wartości zbioru X. Dwie krotki podzielające tę samą wartość ze zbioru X, będą posiadały odpowiednio te same wartośći w zbiorze Y.

Przykład (1)

- Przykład ilustruje koncepcję zależności funkcjonalnej..
- Poniżej mamy model dziedziny przedmiotowej, w której **studenci uczęszczają na więcej niż jedne zajęcia**. Na każdych zajęciach **mają przypisanego asytenta nauczania**. Przyjmijmy również, że **każdy student jest na jakimś semestrze**, oraz jest **identyfikowany poprzez** wartość **ID**.

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

Przykład (2)

- Zauważamy, że w przypadku, gdy dwa wiersze w poniższej tabeli mają tą samą wartość **StudentID**, mają jednocześnie tę samą wartość **Semester.** Ten prosty fakt może być wyrażony jako zależcość funkcyjna:
- **StudentID → Semester** (StudentID determinuje funkcyjnie Semenster)

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

Przykład (3)

Zauważmy, że jeżeli dodany zostanie wiersz, w którym ten sam student będzie miał przypisaną inną wartość semestru, niż wynikająca z zależności funkcyjnej, wtedy zależność funkcjonalna przestaje istnieć.

To oznacza, że FD jest implikowana przez dane, ponieważ jest możliwe wprowadzenia takich wartości, które unieważnią tę zależność funkcyjną.

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

Przykład (4)

W niniejszym przykładzie można zidentyfikować jeszcze inne, nietrywialne zależności funkcyjne, np.:

{StudentID, Lecture} → TA

{StudentID, Lecture} → {TA, Semester}

Oznacza to, że podzbiór **{StudentID, Lecture}** jest tzw. nadkluczem relacji.

StudentID	Semester	Lecture	TA
1234	6	Numerical Methods	John
1221	4	Numerical Methods	Smith
1234	6	Visual Computing	Bob
1201	2	Numerical Methods	Peter
1201	2	Physics II	Simon

Normalizacja
Postacie normalne

Normalizacja

- Normalizacja jest procesem, w którym schematy realcji, posiadające pewne niepożądane właściwości, są dekomponowane do mniejszych schematów, posiadających pożądane właściwości.
- W procesie normalizacji powinny zostać zachowane trzy warunki:

1. Nie zostanie utracony żaden z atrybutów
2. Nie zostaną utracone dane
3. Wszystkie funkcyjne zależności zostaną zachowane

Klucze i nadklucze

Nadklucz jest zbiorem atrybutów $A_1, \ldots, A_n$ takim, że dla dowolnego innego atrybutu B w R, mamy zależność funkcyjną $\{A_1, \ldots, A_n\} \rightarrow B$

Inaczej: wszystkie atrybuty są funkcyjnie determinowane przez nadklucz

Klucz definiujemy jako minimalny **nadklucz**

To ozna że, żaden podzbiór utworzony z klucza nie jest jednocześnie nadkluczem (usunięcie jakiegoś atrybutu z klucza sprawia, że przestaje być nadkluczem).

Klucze potencjalne i atrybuty

- Klucze potencjalne (candidate keys)
  - Klucz główny
  - Pozostałe klucze
- Klucz obcy
- Atrybuty
  - atrybut główny - wchodzący w skład klucza
  - atrybut niegłówny - nie wchodzący w skład klucza
- Klucze proste - jednoatrybutowe
- Klucze złożone - wieloatrybutowe

Pierwsza postać normalna (**1**st **N**ormal **F**orm)

- Relacja jest w pierwszej postaci normalnej, gdy:
  - opisuje jeden obiekt (jeden rodzaj elementu, zjawiska)
  - wartości atrybutów są atomowe (niepodzielne)
  - nie zawiera kolekcji (powtarzających się grup informacji)
  - kolejność wierszy nie jest istotna

Płeć	Imię
Męska	Jan, Piotr, Zenon
Żeńska	Anna, Maria, Zofia

Przed normalizacją

Płeć	Imię
Męska	Jan
Męska	Piotr
Męska	Zenon
Żeńska	Anna
Żeńska	Maria
Żeńska	Zofia

Po normalizacji do 1NF

W **1NF** musi być:
1. Zdefiniowany klucz
2. Wszystkie atrybuty nie należące do klucza muszą być w zależności funkcyjnej od klucza

Druga postać normalna (2NF)

- Relacja jest w drugiej postaci normalnej wtedy i tylko wtedy, gdy jest w I postaci normalnej i żadna kolumna niekluczowa nie jest częściowo funkcyjnie zależna od jakiegokolwiek klucza potencjalnego.

Przed normalizacją

Imię	Nazwisko	Płeć	Stanowisko	Stawka za godzinę
Antoni	Anonim	Męska	Stolarz	10 zł
Natalia	Niewiadoma	Żeńska	Sekretarka	20 zł
Alina	Enigma	Żeńska	Sekretarka	20 zł

Mamy tu **klucz potencjalny: {Imię,Nazwisko}**
i zależność funkcyjną: **{Imię,Nazwisko} → {Stanowisko, Stawka za godzinę}**
Ale też **{Imię} → {Płeć}**
Co oznacza, że atrybut **{Płeć}** jest funkcyjnie zależny od części klucza potencjalnego (od jednego z jego atrybutów).

Druga postać normalna (2NF)

- Po normalizacji do 2NF

Imię	Nazwisko	Stanowisko	Stawka za godzinę
Antoni	Anonim	Stolarz	10 zł
Natalia	Niewiadoma	Sekretarka	20 zł
Alina	Enigma	Sekretarka	20 zł

Imię	Płeć
Antoni	Męska
Natalia	Żeńska
Alina	Żeńska

W tym przypadku - każdy atrybut nie należący do klucza jest funkcjonalnie zależny od całego klucza potencjalnego (od wszystkich jego atrybutów)

Trzecia postać normalna (3NF)

- Relacja jest w trzeciej postaci normalnej wtedy i tylko wtedy, gdy jest w II postaci normalnej i żaden atrybut niekluczowy nie jest zależny funkcyjnie od innych atrybutów niekluczowych.

Przed normalizacją

Imię	Nazwisko	Stanowisko	Stawka za godzinę
Antoni	Anonim	Stolarz	10 zł
Natalia	Niewiadoma	Sekretarka	20 zł
Alina	Enigma	Sekretarka	20 zł

Klucz potencjalny: **{imię, Nazwisko}**
Oba niekluczowe atrybuty: {Stanowisko} oraz {Stawka za godzinę} są funkcyjnie zależne od klucza potencjalnego:
{imię, Nazwisko} → {Stanowisko, Stawka za godzinę}
Ale jeżeli założymy, że ludzie na tym samym stanowisku **{Stanowisko}** zarabiają tyle samo **{Stawka za godzinę}** wtedy mamy zależność: **{Stanowisko} → {Stawka za godzinę}** - która jest zależnością częściową od klucza potencjalnego i prowadzi do powstawania anomalii (redundancja, anomale przy aktualizacji itp.)

Trzecia postać normalna (3NF)

- Możemy znormalizować do 3NF:

Imię	Nazwisko	Stanowisko
Antoni	Anonim	Stolarz
Natalia	Niewiadoma	Sekretarka
Alina	Enigma	Sekretarka

Stanowisko	Stawka
Stolarz	10 zł
Sekretarka	20 zł

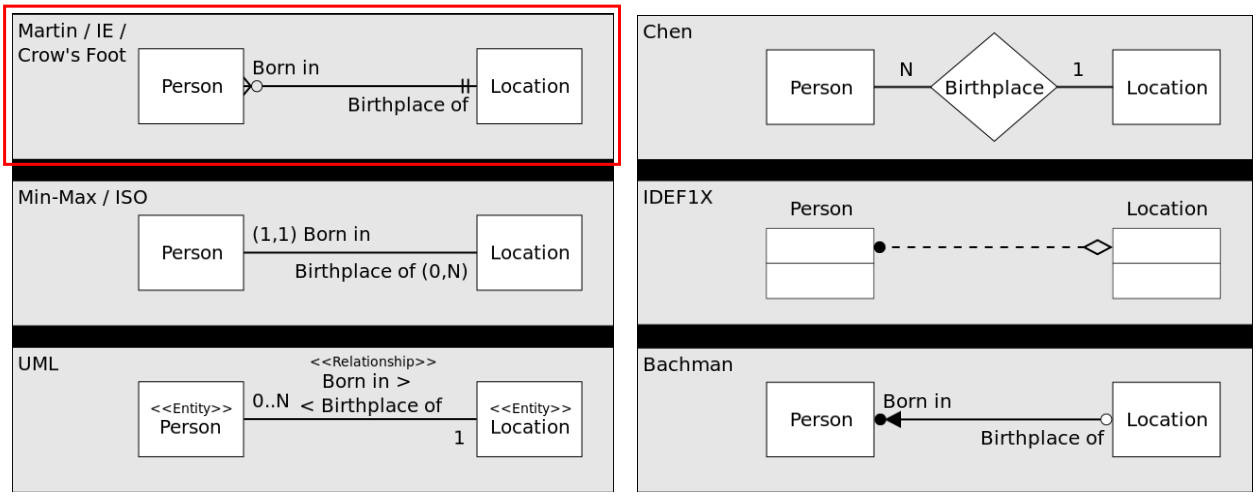
Postać normalna Boyce'a-Codd'a
(Boyce-Codd Normal Form - BCNF lub 3.5NF)

- W tej postaci zależności funkcyjne muszą mieć następującą postać:

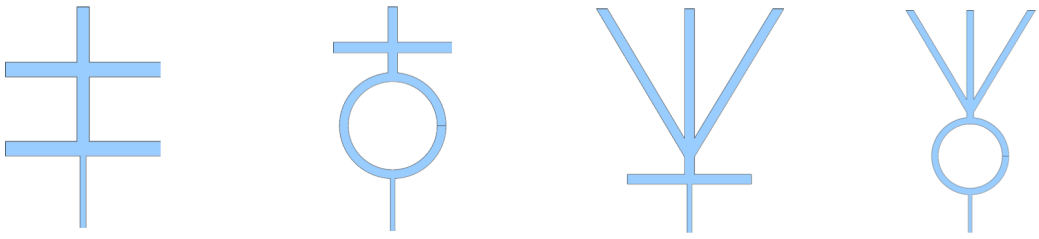
jeśli **X → A** i atrybut **A** nie jest zawarty w **X**, wtedy **X** jest kluczem lub zawiera klucz.

Diagramy związków encji

Różne notacje ERD

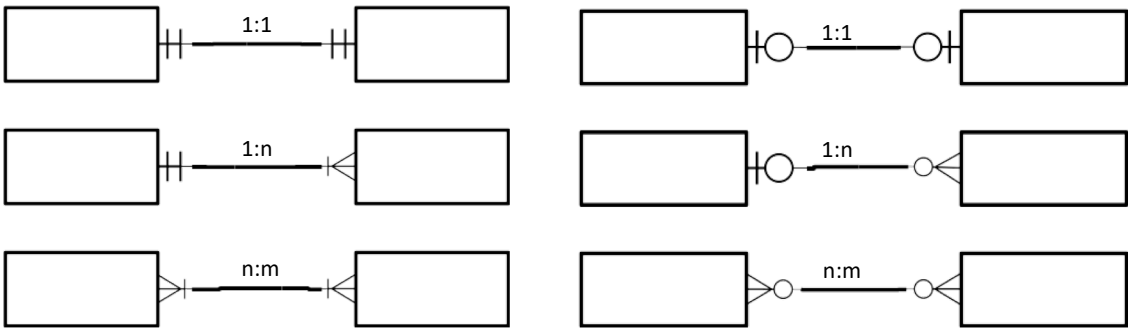


Łączniki crow's foot

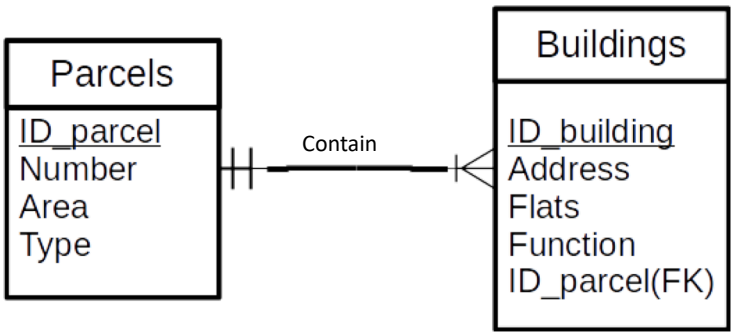


Jeden i tylko jeden | Zero lub jeden | Jeden lub wiele | Zero lub wiele

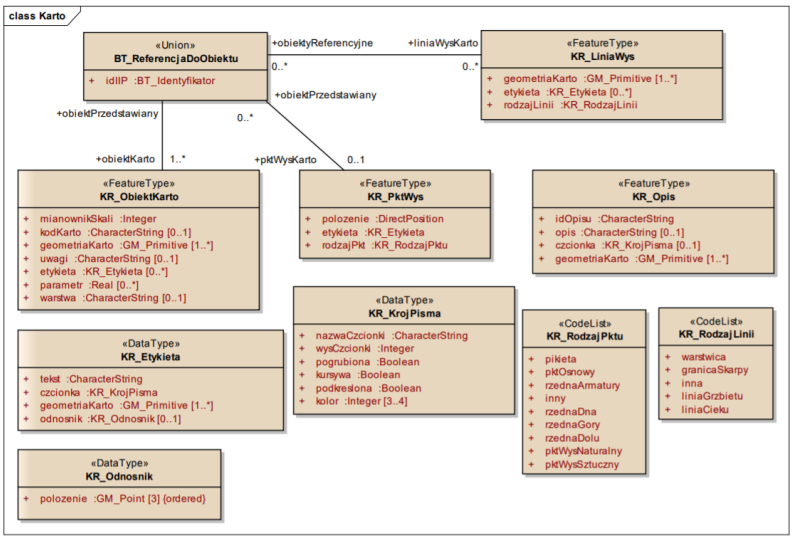
Przykładowe relacje



Encje, attrybuty, relacje



UML w modelowaniu sruktury bazy danych



Typy danych

(znakowe, numeryczne, geometryczne, daty i czasu oraz inne, konwersja typów)

Podział zasadniczy

- Dane liczbowe
- Dane tekstowe
- Dane binarne

Typy danych w Postgresql

Nazwa	Alias	Opis
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [(n)]		fixed-length bit string
bit varying [(n)]	varbit	variable-length bit string
boolean	bool	logical Boolean (true/false)
box		rectangular box in the plane
bytea		binary data ("byte array")
character varying [(n)]	varchar [(n)]	variable-length character string
character [(n)]	char [(n)]	fixed-length character string
cidr		IPv4 or IPv6 network address
circle		circle in the plane
date		calendar date (year, month, day)
double precision	float8	double precision floating-point number
inet		IPv4 or IPv6 host address

Typy danych w Postgresql

Nazwa	Alias	Opis
integer	int, int4	signed four-byte integer
interval [(p)]		time span
line		infinite line in the plane
lseg		line segment in the plane
macaddr		MAC address
money		currency amount
numeric [(p, s)]	decimal [(p, s)]	exact numeric of selectable precision
path		geometric path in the plane
point		geometric point in the plane
polygon		closed geometric path in the plane
real	float4	single precision floating-point number
smallint	int2	signed two-byte integer
serial	serial4	autoincrementing four-byte integer
text		variable-length character string

Typy danych w Postgresql

Nazwa	Alias	Opis
time [(p)] [without time zone]		time of day
time [(p)] with time zone	timetz	time of day, including time zone
timestamp [(p)] [without time zone]		date and time
timestamp [(p)] with time zone	timestamptz	date and time, including time zone
tsquery		text search query
tsvector		text search document
txid_snapshot		user-level transaction ID snapshot
uuid		universally unique identifier
xml		XML data

Typy geometryczne - postGIS

- W systemie postgres z rozszerzeniem postGIS zaimplementowano obsługę typów danych przestrzennych zgodnie z OGC (Open GIS - WKB i WKT). Przykłady:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT((0 0),(1 2))
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))

PostGIS - dane geograficzne wektorowe i rastrowe

- PostGIS umożliwia przechowywanie wektorowych danych geograficznych z uwzględnieneim układu odniesień przestrzennych, również zgodnie z OGC
- Jako dane geograficzne nie mogą być zapisane krzywe (curves), siatki (TIN), ani powierzchnie (POLYHEDRALSURFACE). Pozostałe typy geometryczne mogą być użyte.
- Tabela zawierająca geometrię z określonym SRID:

```
CREATE TABLE tabelageom(gid serial PRIMARY KEY, geog geography(POINT,4326) );
```

- Tabela zawierająca dane rastrowe:

```
CREATE TABLE mojraster(rid serial primary key, rast raster);
```

Konwersja typów

- CAST(wartość_konwertowana AS typ_danych)

```
CAST(HireDate as varchar(10))
```

- CONVERT (typ_danych, wartość_konwertowana, opcjonalnie_styl)

```
CONVERT(varchar(100),TotalDue,2)
```

Ćwiczenie

Zaprojektujmy przykładową bazę danych z wykorzystaniem notacji ERD

MINISTERSTWO ŚRODOWISKA

Sfinansowano ze środków Narodowego Funduszu Ochrony Środowiska i Gospodarki Wodnej

TRAPPA

Szkolenie
Język SQL w bazie PostgreSQL

Dziękuję za uwagę

Sfinansowano ze środków Narodowego Funduszu Ochrony Środowiska i Gospodarki Wodnej