PROJEKT LOGICZNY

Temat projektu: mpk-database

Author:
Barbara BEREK
Marcin KĘSY
Piotr SALA

19 stycznia 2010

Sformułowanie zadania projektowego

a) Zaprojektowanie bazy danych, która będzie mogła przechowywać informacje dotyczące rozkładów jazdy komunikacji publicznej w Krakowie. Baza danych ma umożliwić wyszukiwanie połączeń między przystankami i ma być dostosowana do wykorzystania w projektach mpk-gis, oraz mpk-planner.

b) Zaimplementowanie modułu importującego dane z internetu do utworzonej bazy danych.

Analiza stan wyjściowego

Zarówno bazę danych jak i moduł importujący zamierzamy zaimplementować od podstaw nie wzorując się na dostępnych już na rynku podobnych aplikacjach. Jedną ze stron, które udostępniają aktualne rozkłady jazdy jest http://mpk.rozklady.pl/ i dla tej strony w pierwszej kolejności zaimplementujemy moduł importujący, a w miarę możliwości technicznych, oraz czasowych postaramy się o implementację pobierającą informacje ze strony http://rozklady.komunikacja.krakow.pl/. Wybór padł na te serwisy internetowe ponieważ oba są często uaktualniane, oraz zawierają stosunkowo szczegółowe informacje. Import danych będzie polegał na analizowaniu kodu strony w poszukiwaniu wyrażeń regularnych. Dzięki nim moduł będzie w stanie określić położenie interesujących nas informacji i po odpowiedniej konwersji zapisać je do bazy danych.

Analiza wymagań użytkownika

Jako, że celem naszego projektu jest sama baza danych i moduł importujący dane, użytkownikami mają być docelowo osoby które będą wykorzystywały naszą bazę danych w swoich aplikacjach/systemach. Przykładem takiej aplikacji może być Scheduler.

Poniżej przedstawiamy wypunktowane wymagania, względem informacji jakie mają być zawarte w bazie danych biorąc pod uwagę ograniczenia jakie niesie ze sobą import danych z wybranych źródeł internetowych.

- Uwzględnienie pojawiających się autobusów/tramwajów w środku trasy.
- Dodatkowe informacje dotyczące pojedynczych przejazdów (tramwajowych i autobusowych, np. możliwość przewiezienia roweru).
- Umożliwienie rozróżnienia przystanków o tej samej nazwie przez ręczny wpis w jak najmniej inwazyjny i jak najprostszy sposób.
- Komunikaty o zmianach tymczasowych.
- Informacja o rodzaju przejazdu (np. linie autobusowe miejskie przyspieszone).
- Informacja czy dany przystanek jest na żądanie.
- Informacja czy dany przystanek jest aglomeracyjny.
- Informacja na jakich ulicach znajduje się przystanek o danej nazwie.

Po zakończeniu pracy modułu importującego, program umożliwi przetestowanie bazy danych przez:

- Proste wyszukiwanie połączeń.
- Definiowanie zmian tymczasowych.
- Inne metody sprawdzające spójność danych zaimportowanych do bazy.

Określenie scenariuszy użycia

Scenariusze użycia (np. z podziałem na typy użytkowników) będą odnosiły się do potencjalnego systemu wykorzystującego bazę danych będącą efektem naszej pracy i będą one ściśle związane z jego implementacją. System taki jest nam obcy i –z tego faktu- opis i opracowanie scenariuszy użycia nie należą do zadań związanych z naszym projektem. Tym samym w tym dokumencie nie zostały zamieszczone także diagramy STD.

Identyfikacja funkcji

Funkcje które składają się na moduł importujący:

- Import danych o przystankach.
- Import danych o ulicach.
- Import komunikatów.
- Import rozkładów (numery linii, relacje, przejazdy, czasy odjazdów).

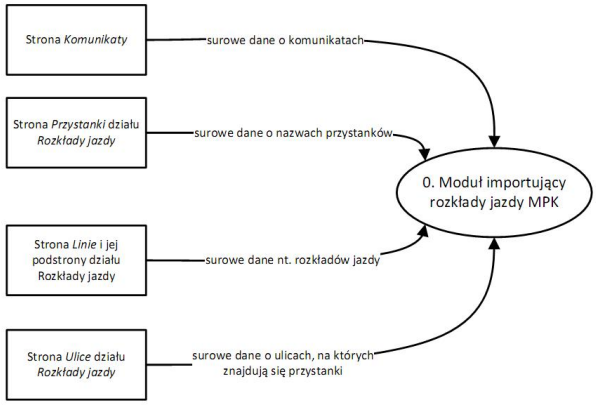
Wybór encji i ich atrybutów

Wybrane przez nas encje oraz ich atrybuty:

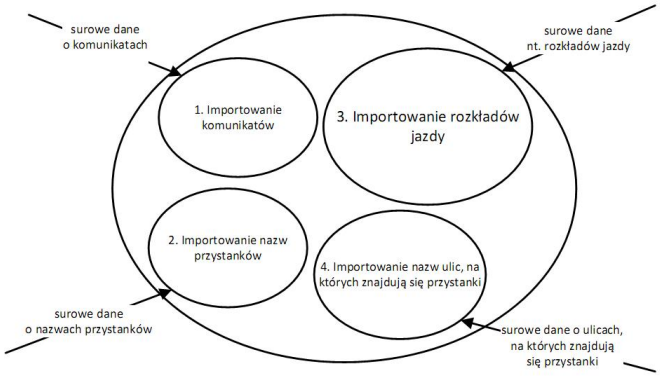
1. Przystanki
   idPrzystanku, nazwa, ogólny_ czas_ przesiadki
2. PodPrzystanki
   idPodPrzystanku, idPrzystanku
3. Ulice
   nazwa, idPodprzystanku
4. Przejazdy
   idPrzejazdu, idRelacji, idPodprzystanku, na_ żądanie
5. Relacje
   idRelacji, numerLinii, idPierwszyPrzystanek, idOstatniPrzystanek
6. CzasyOdjazdow
   idCzasu, idPrzejazdu, idDni, czas
7. Dni
   idDni, tydzien, czyNocny
8. Oznaczenia
   idCzasu, idOpisu
9. OpisyOznaczen
   idOpisu, opis
10. RodzajeLinii
    nazwa, numerLinii
11. Komunikaty
    dataKomunikatu, treść

Wybrane encje oraz ich atrybuty wraz z kluczami zobaczymy na przedstawionym w kolejnym punkcje schemacie ERD.

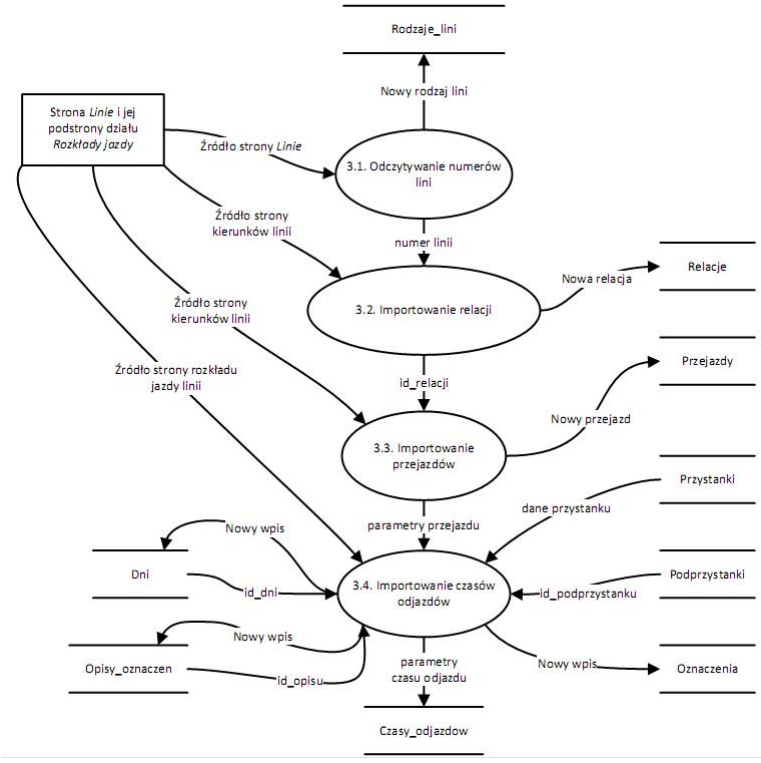
Budowa i analiza diagramu przepływu danych DFD - Data Flow Diagram



Rysunek 1: Diagram kontekstowy

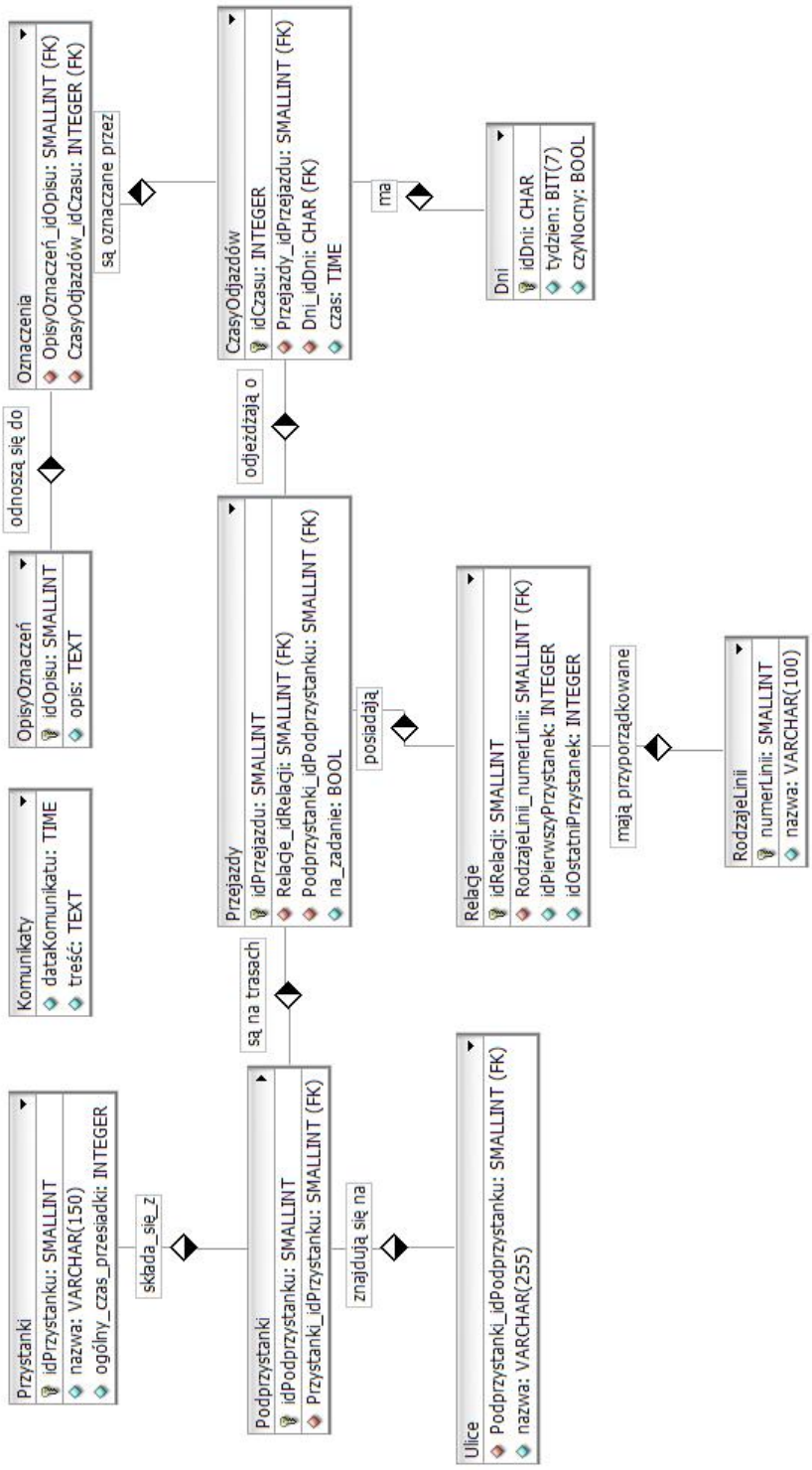


Rysunek 2: Diagram główny Modułu Importującego



Rysunek 3: Diagram dla "Importu rozkładów jazdy"

Projektowanie powiązań (relacji) pomiędzy encjami. Konstrukcja diagramu ERD.



Rysunek 4: Diagram ERD

Projekt bazy w języku SQL

```
CREATE DATABASE mpk
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       LC_COLLATE = 'Polish_Poland.1250'
       LC_CTYPE = 'Polish_Poland.1250'
       CONNECTION LIMIT = -1;

CREATE TABLE komunikaty
(
  datakomunikatu date,
  tresc text
)

CREATE TABLE przystanki
(
  idprzystanku smallint NOT NULL,
  nazwa character varying(150),
  ogolny_czas_przesiadki smallint,
  CONSTRAINT przystanki_pkey1 PRIMARY KEY (idprzystanku),
  CONSTRAINT przystanki_nazwa_key UNIQUE (nazwa)
)

CREATE TABLE podprzystanki
(
  idpodprzystanku integer NOT NULL,
  idprzystanku smallint,
  CONSTRAINT podprzystanki_pkey PRIMARY KEY (idpodprzystanku),
  CONSTRAINT podprzystanki_idprzystanku_fkey FOREIGN KEY (idprzystanku)
      REFERENCES przystanki (idprzystanku) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE ulice
(
  idpodprzystanku smallint,
  nazwa character varying(150),
  CONSTRAINT ulice_idpodprzystanku_fkey FOREIGN KEY (idpodprzystanku)
      REFERENCES podprzystanki (idpodprzystanku) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE rodzajelinii
(
  numerlinii smallint NOT NULL,
  nazwa character varying(100),
  CONSTRAINT rodzajelinii_pkey PRIMARY KEY (numerlinii)
)

CREATE TABLE relacje
(
  idrelacji smallint NOT NULL,
  numerlinii smallint,
```

```
  idpierwszegopodprzystanku smallint,
  idostatniegopodprzystanku smallint,
  CONSTRAINT relacje_pkey PRIMARY KEY (idrelacji),
  CONSTRAINT relacje_idostatniegopodprzystanku_fkey FOREIGN KEY
(idostatniegopodprzystanku)
      REFERENCES podprzystanki (idpodprzystanku) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT relacje_idpierwszegopodprzystanku_fkey FOREIGN KEY
(idpierwszegopodprzystanku)
      REFERENCES podprzystanki (idpodprzystanku) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT relacje_numerlinii_fkey FOREIGN KEY (numerlinii)
      REFERENCES rodzajelinii (numerlinii) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE przejazdy
(
  idprzejazdu smallint NOT NULL,
  idrelacji smallint,
  idpodprzystanku smallint,
  na_zadania boolean,
  CONSTRAINT przejazdy_pkey PRIMARY KEY (idprzejazdu),
  CONSTRAINT przejazdy_idpodprzystanku_fkey FOREIGN KEY (idpodprzystanku)
      REFERENCES podprzystanki (idpodprzystanku) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT przejazdy_idrelacji_fkey FOREIGN KEY (idrelacji)
      REFERENCES relacje (idrelacji) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE czasyodjazdow
(
  idprzejazdu smallint,
  dni character(1),
  czas time without time zone,
  idczasu integer NOT NULL,
  CONSTRAINT czasyodjazdow_pkey PRIMARY KEY (idczasu),
  CONSTRAINT czasyodjazdow_dni_fkey FOREIGN KEY (dni)
      REFERENCES dni (id) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT czasyodjazdow_idprzejazdu_fkey1 FOREIGN KEY (idprzejazdu)
      REFERENCES przejazdy (idprzejazdu) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE dni
(
  "idDni" character(1) NOT NULL,
  tydzien bit(7),
  czynocny boolean,
  CONSTRAINT dni_pkey PRIMARY KEY ("idDni")
)
```

```
CREATE TABLE oznaczenia
(
  idczasu integer,
  opis character varying(200),
  CONSTRAINT oznaczenia_idczasu_fkey FOREIGN KEY (idczasu)
      REFERENCES czasyodjazdow (idczasu) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)

CREATE TABLE opisyoznaczen
(
  idopisu smallint NOT NULL,
  opis text,
  CONSTRAINT opisyoznaczen_pkey PRIMARY KEY (idopisu)
)
```

Słowniki danych

```
znak =[A-Z|a-z|-| |.|/|0-9|]
```

1. **Przystanki**
   idPrzystanku = ** *small integer range*
   nazwa = *nazwa przystanku*{ znak}
   ogólny_ czas_ przesiadki = ** *small integer range*

2. **Podprzystanki**
   idPodprzystanku = ** *small integer range*

3. **Ulice**
   nazwa =*nazwa ulicy*{ znak}

4. **Przejazdy**
   idPrzejazdu = ** *smallinteger range*
   na_ zadanie = *zatrzymanie się autobusu na żądanie (ale tylko na przystankach)*[true|false]

5. **Relacje**
   idRelacji = ***small integer range*
   idPierwszyPrzystanek = *id pierwszego przystanku na trasie danej relacji* *small integer range*
   idOstatniPrzystanek = *id ostatniego przystanku na trasie danej relacji* *small integer range*

6. **RodzajeLinii**
   numerLinii = ** *small integer range*
   nazwa = *opis grupy linii* { znak}

7. **CzasyOdjazdow**
   idCzasu = ** *integer range*
   czas = *czas odjazdu danego przejazdu * *jednostki: TIME*

8. **Dni**
   idDni = ** [a-z]
   tydzien = *Pierwszy bit jest skojarzony z Poniedziałkiem, siódmy, czyli ostatni z Niedzielą(Świętem), 1-dzien jest aktywny, 0-dzien nie jest aktywny* *{ 0-1} *
   czyNocny = *określa czy aktywne bity w atrybucie tydzień odnoszą się do nocy następującej po danym dniu, wówczas przyjmuje wartość 'true'* [true|false]

9. **OpisyOznaczen**
   idOpisu=** *smallint range*
   opis = *opis oznaczenia o danym id* *łańcuch znaków*

10. **Komunikaty**
    dataKomunikatu = *data pojawienia się komunikatu* *jednostki: TIME*
    treść = *treść komunikatu* *łańcuch znaków*

Analiza zależności funkcyjnych i normalizacja tabel

Zależności funkcyjne w tablicach:

- **Przystanki:**
  idPrzystanku –> nazwa
  idPrzystanku –> ogólny_ czas_ przesiadki
  nazwa –> ogólny_ czas_ przesiadki
  nazwa –> idPrzystanku

- **Podprzystanki:**
  idPodprzystanku –> idPrzystanku

- **Ulice:**
  Brak

- **RodzajeLinii:**
  numerLinii –> nazwa

- **Relacje:**
  idRelacji –> numerLinii, idPierwszegoPodPrzystanku, idOstatniegoPodPrzystanku

- **Przejazdy:**
  idPrzejazdu –> idRelacji, idPodprzystanku, na_ żądanie

- **Dni:**
  idDni –> tydzien, czyNocny

- **CzasyOdjazdow:**
  idCzasu –> idPrzejazdu, idDni, czas

- **Oznaczenia:**
  Brak

- **OpisyOznaczen**
  idOpisu –> opis

- **Komunikaty:**
  Brak

a) Pierwsza postać normalna – 1NF

Nasza baza danych spełnia warunki definicji pierwszej postaci normalnej. Możliwe jest stwierdzenie, że każdy z elementów, każdej tablicy naszej bazy danych ma charakter atomiczny, gdyż żaden z nich nie wymaga jakiejkolwiek formy obróbki, czy cięcia w celu poprawnej interpretacji, bądź wydobycia istotnej z praktycznego punktu widzenia informacji z jego treści.

b) Druga postać normalna - 2NF

Wszystkie zdefiniowane klucze dla tablic w naszej bazie danych są jedno-atrybutowe co implikuje wniosek o zgodności z drugą postacią normalną

c) Trzecia postać normalna - 3NF

Relacja jest w trzeciej postaci normalnej tylko wtedy, gdy jest ona w drugiej postaci normalnej i każdy atrybut wtórny jest tylko bezpośrednio zależny od klucza głównego. Nasza baza danych w żadnej ze swoich tabel nie posiada atrybutów niezwiązanych bezpośrednio z kluczem głównym tablicy, a więc nie istnieją zależności tranzytywne.

Zdefiniowanie kwerend dla realizacji funkcji wyspecyfikowanych w projekcie

Przykładowe kwerendy zapisujące informacje w bazie danych:
Kwerenda zapisująca nowy wiersz w tablicy przystanki:
"insert into przystanki values("+ i.ToString() + ", '"+ przystanek + "')"
Kwerenda zapisująca nowy wiersz w tablicy ulice:
"insert into ulice values("+ idPrzystanku + ", '"+ nazwaUlicy + ")"
Sekwencja komend w celu pokazania czasów odjazdów z danego przystanku, danej linii o wybranym kierunku:
*SELECT * FROM przystanki WHERE nazwa='cracovia' a wynik to idPrzystanku np. równe 666*
*SELECT * FROM relacje WHERE numerLinii=144 a wynik to idRelacji np. równe 160*
*SELECT * FROM przejazdy WHERE idRelacji=160 AND idPodPrzystanku=666 a wynik to idPrzejazdu np. równe 2345*
*SELECT * FROM czasyOdjazdow WHERE idPrzejazdu=2345 a wynik to czasy odjazdów linii 144 (o wybranym kierunku) z przystanku Cracovia*