Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Imię i nazwisko
Nr albumu: nralbumu

Intuicyjny język wyszukiwania TQL (Tablets Query Language)

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dra Roberta Dąbrowskiego
Instytut Informatyki

czerwiec 2010

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

Sumerologia jest dziedziną badań nad antycznym językiem Sumerów, w której kluczowym zagadnieniem jest przeszukiwanie dużych zbiorów informacji zapisanych na odnalezionych tabliczkach sumeryjskich.

W pracy przedstawiono definicję przeznaczonego dla sumerologów intuicyjnego języka przeszukiwania zbiorów tabliczek (Tablets Query Language) wraz z jego przykładową implementacją opartą na relacyjnej bazie danych.

Celem tej pracy jest stworzenie języka zapytań intuicyjnego dla sumerologów, stanowiącego znaczące uproszczenie w stosunku do SQL dzięki wprowadzeniu pojęć naturalnych dla rozważanej dziedziny. Jednocześnie TQL nadal pozwala na tworzenie skomplikowanych zapytań wyszukujących, natomiast nie udostępnia funkcji tworzących i modyfikujących bazę. Można go rozszerzać i zmieniać tak, by mógł służyć też do innych zastosowań.

Słowa kluczowe

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.3 Informatyka

Klasyfikacja tematyczna

H. INFORMATION SYSTEMS
H.2. DATABASE MANAGEMENT
H.2.3 Languages

Tytuł pracy w języku angielskim

Intuitive query language TQL (Tablets Query Language)

Spis treści

Wprowadzenie 5

1. Podstawowe pojęcia 7
1.1. Definicje 7

2. Wcześniejsze rozwiązania 9

3. Dziedzina problemu 11

4. Definicja języka TQL 13
4.1. Gramatyka 13
4.1.1. Struktura leksykalna 13
4.1.2. Słowa kluczowe 13
4.1.3. Znaki specjalne 13
4.1.4. Komentarze 13
4.1.5. Struktura syntaktyczna języka 13

5. Implementacja 15
5.1. Trzon 15
5.1.1. Parser 15
5.1.2. Analizator kontekstowy 15
5.1.3. Translator 15
5.1.4. Baza 16
5.1.5. Pliki pomocnicze 17
5.2. Moduły wymienne 17
5.2.1. Baza PostgreSQL 17
5.2.2. Baza XML 19

6. Podsumowanie 21

Bibliografia 23

Wprowadzenie

Sumerolodzy posiadają bazę danych składającą się z prawie 50 tys. tabliczek sumeryjskich w wersji elektronicznej. Potrzebują prostego i intuicyjnego języka służącego do ich wyszukiwania, który jak najmniej będzie ograniczał siłę wyrazu, a jego wykorzystanie będzie powodowało jak najmniejszy narzut czasowy.

Istnieją też inne grupy ludzi potrzebujące podobnego języka (np. językoznawcy). Większość programów ułatwiających tworzenie zapytań jest skomplikowana, daje ograniczone możliwości lub jest przystosowana głównie do przetwarzania danych liczbowych. Tablets Query Language rozwiązuje te problemy: jest prosty i intuicyjny, przystosowany głównie do tekstów, minimalnie zmniejsza siłę wyrazu oraz łatwo go rozbudowywać.

Język TQL jest nakładką na inne języki (m.in. SQL). Dla każdego z nich, w zależności od reprezentacji danych, należy skonstruować translator, którego zadaniem będzie przetłumaczenie zapytania. W ramach niniejszej pracy przedstawione zostaną dwa przykładowe translatory.

Rozdział 1

Podstawowe pojęcia

1.1. Definicje

Rozdział 2

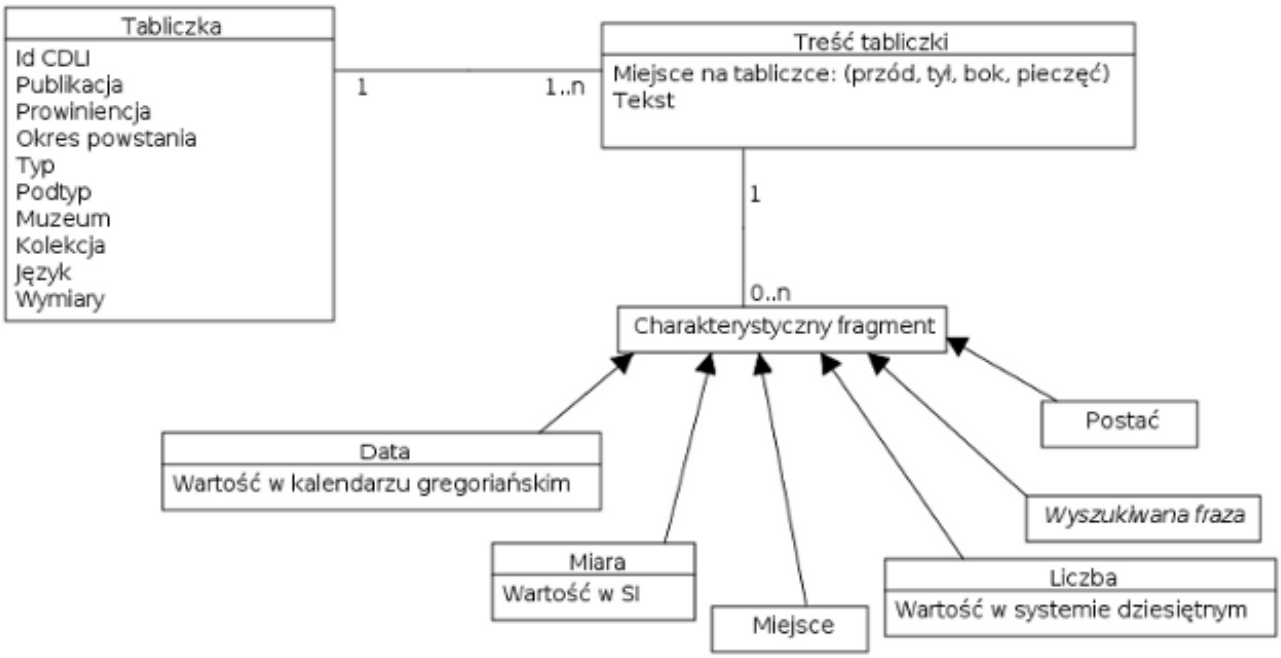
Wcześniejsze rozwiązania

W chwili obecnej nie ma czegoś takiego jak język dostosowany do potrzeb sumerologów. Są strony internetowe oferujące wyszukiwanie, jak np.

- The Cuneiform Digital Library Initiative (http://cdli.ucla.edu) - największa znana nam baza tekstów sumeryjskich, wyszukiwanie po praktycznie wszystkich możliwych parametrach, choć trochę mało wygodne. Brakuje wyjaśnienia jak używać "Advanced search syntax"
- The Electronic Text Corpus of Sumerian Literature (http://etcsl.orinst.ox.ac.uk/) - baza znacznie mniejsza, zawiera głównie teksty literackie. Wyszukiwanie mało rozbudowane.

Rozdział 3

Dziedzina problemu



Głównym pojęciem jest tabliczka. Ma ona swoje metadane i treść. Tabliczka jest rozumiana dwojako - jako fizyczna tabliczka gliniana zapisana klinami lub jako tabliczka w formie cyfrowej zapisana odczytami. Może ona zawierać elementy znaczące takie jak imię jakiejś osoby, liczba, jednostka (np. przy opisywaniu wypłat), miejsce, data, imię bóstwa. Część tych elementów da się przetłumaczyć na współczesny język (np. jednostki przeliczyć na SI, datę na datę liczbową BC). Gliniane tabliczki są zapisywane z różnych stron (od góry, z przodu, z tyłu itp). Poza tym zawierają pieczęcie - fragmenty tekstu po prostu odbijane na tabliczce (coś jak nasza pieczątka).

Odczyty zawarte w cyfrowym zapisie tabliczki są wariantem tłumaczenia z klinów. W cyfrowej wersji nie ma klinów, stąd też możliwe są pomyłki w tłumaczeniach, które ciężko zweryfikować. Są też uszkodzone fragmenty, które zostały cyfrowo zapisane w najróżniejszej formie (np. po niemiecku “Tutaj miałem problem, ale chyba powinno być «xxx»” (-;)

Rozdział 4

Definicja języka TQL

4.1. Gramatyka

4.1.1. Struktura leksykalna

String

Literał ⟨*String*⟩ ma postać "x", gdzie x jest dowolnym ciągiem znaków poza " niepoprzedzonymi \.

Słowo Od Litery

Literał ⟨*Słowo Od Litery*⟩ to ciąg liter, cyfr oraz znaków - _ ', zaczynający się od litery, z wyjątkiem słów kluczowych.

Słowo Od Liczby

Literał ⟨*Słowo Od Liczby*⟩ to ciąg liter, cyfr oraz znaków - _ ', zaczynający się od cyfry.

4.1.2. Słowa kluczowe

```
as      define  in
search
```

4.1.3. Znaki specjalne

```
\n  :    +
/   --   *
(   )
```

4.1.4. Komentarze

W chwili obecnej język nie zawiera komentarzy.

4.1.5. Struktura syntaktyczna języka

Nieterminale są pomiędzy ⟨ a ⟩. Symbole ::= (produkcja), | (lub) i ϵ (pusta reguła) należą do notacji BNF. Wszystkie pozostałe symbole to terminale.

⟨*Zapytanie Złożone*⟩ ::= ⟨*Lista Zapytań*⟩

⟨*Zapytanie*⟩ ::= ⟨*Lista Linii Zapytania*⟩ ⟨*Lista Pustych Linii*⟩
| `define \n` ⟨*Zapytanie*⟩ `as` ⟨*Nazwa*⟩ ⟨*Lista Pustych Linii*⟩
| `search \n` ⟨*Zapytanie*⟩ `in` ⟨*Nazwa*⟩ ⟨*Lista Pustych Linii*⟩
| ⟨*Lista Pustych Linii*⟩

⟨*Linia Zapytania*⟩ ::= ⟨*Identyfikator*⟩ `:` ⟨*Wyrażenie*⟩

⟨*Wyrażenie*⟩ ::= ⟨*Wyrażenie*⟩ + ⟨*Wyrażenie1*⟩
| ⟨*Wyrażenie*⟩ `/` ⟨*Wyrażenie1*⟩
| ⟨*Wyrażenie1*⟩

⟨*Wyrażenie1*⟩ ::= −− ⟨*Wyrażenie1*⟩
| ⟨*Wyrażenie2*⟩

⟨*Wyrażenie2*⟩ ::= ⟨*Tekst*⟩ `*` ⟨*Tekst*⟩
| ⟨*Tekst*⟩ `*`
| `*` ⟨*Tekst*⟩
| ⟨*Tekst*⟩
| `(` ⟨*Wyrażenie*⟩ `)`

⟨*Lista Zapytań*⟩ ::= ⟨*Zapytanie*⟩
| ⟨*Zapytanie*⟩ ⟨*Lista Zapytań*⟩

⟨*Lista Linii Zapytania*⟩ ::= ⟨*Linia Zapytania*⟩ `\n`
| ⟨*Linia Zapytania*⟩ `\n` ⟨*Lista Linii Zapytania*⟩

⟨*Pusta Linia*⟩ ::= `\n`

⟨*Lista Pustych Linii*⟩ ::= ϵ
| ⟨*Pusta Linia*⟩ ⟨*Lista Pustych Linii*⟩

⟨*Tekst*⟩ ::= ⟨*String*⟩
| ⟨*Słowo*⟩

⟨*Słowo*⟩ ::= ⟨*Słowo Od Litery*⟩
| ⟨*Słowo Od Liczby*⟩

⟨*Identyfikator*⟩ ::= ⟨*Słowo Od Litery*⟩

⟨*Nazwa*⟩ ::= ⟨*String*⟩

Rozdział 5

Implementacja

Implementacja składa się z dwóch części - zależnej i niezależnej od struktury danych.

5.1. Trzon

5.1.1. Parser

Parser został utworzony za pomocą narzędzia BNFC. Następnie został zmodyfikowany ręcznie: nazwy stałych oznaczających symbole, dodanie tablicy symboli (stringów), uporządkowanie kodu, zmiana niektórych struktur danych. Na parser składają się następujące pliki:

- Parser.c
- Parser.h
- TQL.y
- TQL.l

5.1.2. Analizator kontekstowy

- sprawdza, czy to co jest po lewej w linii zapytania jest nazwą pola.
- upraszcza zapytania - z zapytania złożonego (wywołanie search in) tworzy jedno zapytanie proste
- wypełnia strukturę danych

Składa się z następujących plików:

- Context.c
- Context.h

5.1.3. Translator

Zadaniem translatora jest przetłumaczenie struktury (drzewa składni abstrakcyjnej) jaka powstała na zapytanie w danym języku. Składa się z następujących plików:

- Translator.c

- Translator.h
- Translator_config.h
- Translator_config.c (implementacja interfejsu z Translator_config.h, zależny od wyboru bazy danych itp)

To jak poszczególne elementy są tłumaczone zależy od pliku Translator_config.c (interfejs jest w Translator_config.h). Plik Translator.c przechodzi całą strukturę i od czasu do czasu wywołuje funkcję z Translator_config.

5.1.4. Baza

Moduł bazy jest odpowiedzialny za wywołanie przetłumaczonego zapytania i przekazanie wyniku w określonej formie - w tym momencie xml. Składa się z następujących plików:

- Database.c
- Database.h
- Database_config.h
- Database_config.c (implementacja interfejsu z Database_conf.h, zależny od wyboru bazy danych itp)

Wywołuje funkcję z Database_config.h, jako parametr podaje treść zapytania, funkcja zwraca wypełnioną strukturę danych Tablets.

```
typedef struct{
    char* id;
    char* id_cdli;
    char* publication;
    char* measurements;
    char* year;
    char* provenience;
    char* period;
    char* genre;
    char* subgenre;
    char* collection;
    char* text;
    Tags *tags; //miejsca gdzie w tekście są wyniki wyszukiwania
} Tablet;

typedef struct{
    int size;
    Tablet *tabs;
} Tablets;
```

Następnie tłumaczy otrzymaną strukturę na xml-a.

5.1.5. Pliki pomocnicze

Tablica symboli (stringów):

- symbols.c
- symbols.h

Obsługa błędów:

- Err.c
- Err.h

Definicja struktur danych:

- Absyn.c
- Absyn.h

5.2. Moduły wymienne

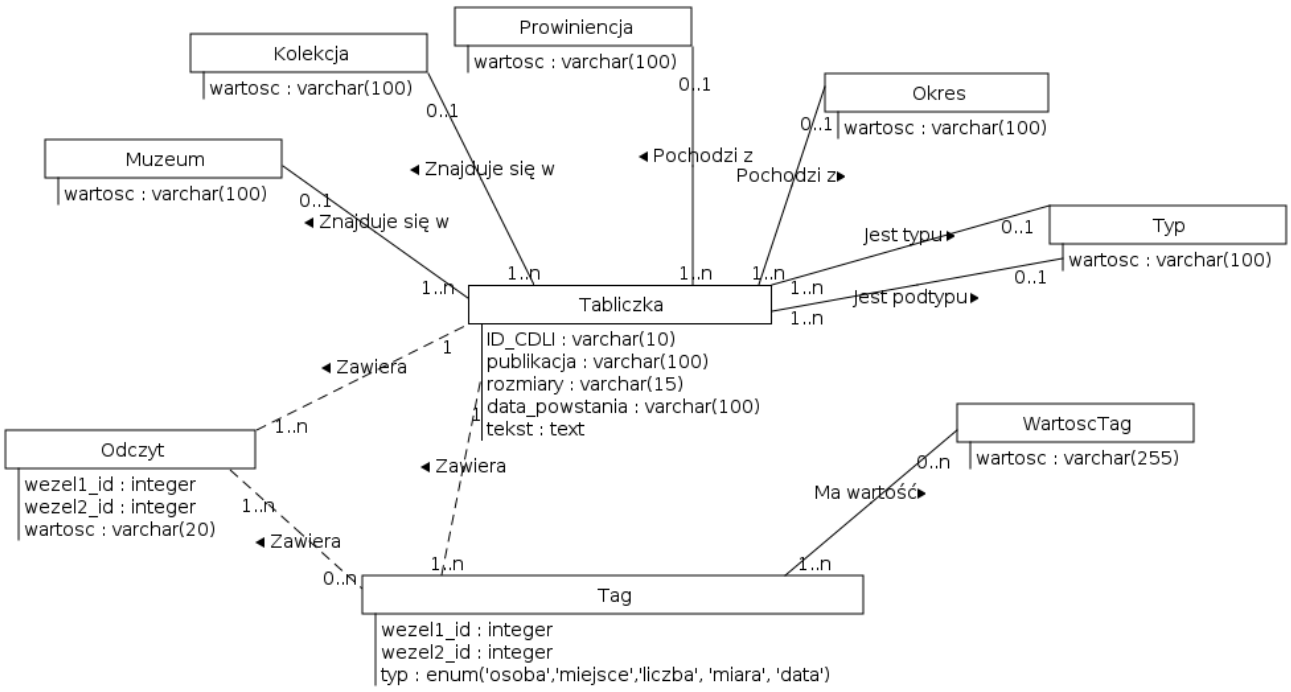
Pliki zależne od wyboru konkretnej bazy danych to:

- Translator_config.c
- Database_config.c

Ich interfejs jest wspólny dla wszystkich baz danych.

5.2.1. Baza PostgreSQL

Diagram encji



Translator_config

Dostaje poszczególne fragmenty drzewa struktury zapytania i tłumaczy je na SQL. Przetłumaczone fragmenty zbiera do buforów (select, from, where), które następnie odpowiednio skleja. Każde proste zapytanie jest tłumaczone na jednego selecta; jak jest kilka prostych zapytań to są sklejane UNION.

Tłumaczenie konstrukcji

Konstrukcja	**Tłumaczenie na SQL**
provenience: wartosc	**From** xxx **Where** p.value LIKE 'wartosc'
publication: wartosc	**From** **Where** t.publication LIKE 'wartosc'
period: wartosc	**From** **Where** pd.value LIKE 'wartosc'
year: wartosc	**From** **Where** t.origin_date LIKE 'wartosc'
genre: wartosc	**From** **Where** g1.value LIKE 'wartosc' OR g2.value LIKE 'wartosc'
cdli_id: wartosc	**From** **Where** t.cdli_id LIKE 'wartosc'
text: wartosc	**From** **Where**

Database_config

Odpowiada za wywołanie zapytania na konkretnej bazie. Korzysta z pliku database.conf, który zawiera dane dostępu do bazy. Korzysta z biblioteki libpq-fe.h do postgresa. Zwrócony wynik zapisuje do struktury Tablets.

5.2.2. Baza XML

Rozdział 6

Podsumowanie

Jesteśmy fajne (-;

Bibliografia