Spis treści

1. Wstęp 3
   1.1. Plan pracy 3
2. Analiza tematu 5
   2.1. Istniejące rozwiązania 5
   2.2. Wymagania 7
   2.3. Język SQL 7
      2.3.1. Elementy języka 8
      2.3.2. Rodzaje instrukcji 10
   2.4. Logika rozmyta 13
3. Przedmiot pracy 14
   3.1. Wykorzystane technologie 15
      3.1.1. PostgreSQL 15
      3.1.2. JavaScript/ECMAScript 15
      3.1.3. TypeScript 15
      3.1.4. Webpack 15
      3.1.5. Electron 15
      3.1.6. React 15
      3.1.7. Redux 15
   3.2. Implementacja operatorów rozmytych 15
   3.3. Architektura systemu 15
4. Badania 17
   4.1. Metodyka badań 17
   4.2. Zbiory danych 17
   4.3. Wyniki 18
5. Podsumowanie 19

Bibliografia i
Spis skrótów i symboli..... ii
Zawartość dołączonej płyty iii
Spis rysunków iv
Spis tabel.....v

1. Wstęp

Celem niniejszej pracy dyplomowej jest realizacja nowego narzędzia dydaktycznego umożliwiającego tworzenie zapytań SQL zawierających elementy rozmyte z wykorzystaniem nowych technologii i zbadanie wydajności takiego rozwiązania. Nowe narzędzie ma bazować na operatorach rozmytych opracowanych przez dr inż. Bożenę Małysiak-Mrozek w rozprawie doktorskiej „Metody aproksymacyjnego wyszukiwania obiektów w bazach danych" oraz narzędzia dydaktycznego FuzzyQ, stworzonego w 2005r. przez mgr Bartosza Dziedzica w jego pracy dyplomowej. 13-letnia aplikacja, wykorzystująca połączenie technologii Flash oraz C#, skompilowana jako samodzielna aplikacja działająca w środowisku Windows, jest na dzień dzisiejszy przestarzała technologicznie. Nowe narzędzie ma wprowadzić funkcjonalność tworzenia zapytań z użyciem podpowiedzi kontekstowych, a także umożliwić automatyczne dodawanie operatorów do bazy. Ponadto, dzięki zastosowaniu wieloplatformowego środowiska Electron, ma być możliwe uruchomienie aplikacji na systemach Linux, Windows oraz macOS.

1.1. Plan pracy

Praca składa się z dziesięciu rozdziałów. W rozdziale pierwszym opisany jest cel, motywacja i plan pracy dyplomowej. W rozdziale drugim są określone wymagania funkcjonalne systemu i usprawnienia

względem FuzzyQ. Rozdział trzeci opisuje język SQL, na którym opiera się system dydaktyczny. W rozdziale czwartym omawiane jest zagadnienie logiki rozmytej i podstawowych operatorów w analogii do logiki boolowskiej. Rozdział piąty przybliża technologie wykorzystane do zbudowania systemu, uzasadniając ich wybór. W rozdziale szóstym opisana jest architektura całego systemu, składającego się z oddzielnie przygotowanych operatorów rozmytych i aplikacji do zarządzania bazami danych ze wsparciem zapytań rozmytych, działającej w środowisku Electron. Rozdział ósmy dotyczy rozwiązań technicznych, takich jak budowa komponentów aplikacji, komunikacja między wątkami, czy dodawanie operatorów do dowolnej bazy relacyjnej. W rozdziale dziewiątym jest opisany interfejs użytkownika. W rozdziale dziesiątym zaprezentowane są badania porównujące szybkość prezentacji danych w środowisku przeglądarkowym jakim jest Electron, a także różnice w wykorzystaniu zasobów między FuzzyQ, a nowym systemem. Rozdział jedenasty zawiera podsumowanie prac nad systemem i możliwości jego rozwoju.

2. Analiza tematu

Obecną wersja systemu dydaktycznego FuzzyQ wyróżnia prosty interfejs, możliwość tworzenia zapytań do bazy i graficzny kreator zapytań. Aplikacja jest jednak przestarzała pod względem technologicznym, gdyż wykorzystuje porzucone środowisko Flash i nie zadziała bezpośrednio na systemie Ubuntu, wykorzystywanym w większości stanowisk laboratoryjnych. FuzzyQ ma także braki pod względem interfejsu, nie spełniając wysokich standardów dzisiejszych narzędzi programistycznych i trendów z zakresu *user experience* wyznaczanych przez firmy JetBrains, Microsoft, czy Google. Za przykład może posłużyć brak kolorowania składni, wsparcia dla wcięć i podpowiedzi kontekstowych w polu edycji zapytania. Nowy system dydaktyczny nie będzie zatem prostym przepisaniem istniejącego narzędzia na inny język programowania, lecz nową implementacją pomysłu. System, z nowymi funkcjonalnościami, zostanie zrealizowany w sposób otwarty na rozszerzanie i zgodny ze współczesnymi standardami projektowania interfejsów.

2.1. Istniejące rozwiązania

Poza aplikacją FuzzyQ stworzoną na Politechnice Śląskiej, na Politechnice Poznańskiej w 2006 roku powstało narzędzie SQLf_j oparte o bazę danych MySQL, dokonujące translacji z języka SQLf (SQL z rozszerzeniami rozmytymi) na SQL przy użyciu tabeli pomocniczych.

Tabela „possibility” zawiera definicje funkcji trapezowych dla danych liczbowych, a tabela „similarity” opisuje podobieństwa między parami ciągów znaków. Obsługiwany jest również minimalny próg przynależności wyników i ograniczenie ich ilości. Jest to podejście, które pozwala na dodawanie definicji i zależności w prosty sposób, lecz całość wymaga korzystania z ograniczonego interfejsu konsolowego, który służy do tłumaczenia poleceń, pozwalającego jedynie na wykonanie instrukcji SELECT.

W sieci obecna jest również strona opisująca język FSQL (Fuzzy SQL) wraz z serwerem FSQL dla nieprecyzyjnej relacyjnej bazy danych (FSQL Server for a Fuzzy Relational Database) bazująca na serwerze Oracle. Rozwiązanie pozwala na definiowanie etykiet dla zakresów danych i definicję progu wyników oraz posiada standardowy zestaw komparatorów rozmytych.

Poza wymienionymi narzędziami nadającymi się wyłącznie do celów akademickich, nie ma rozpowszechnionych i nadających się do wykorzystania w praktyce systemów wyszukiwania z numerycznymi elementami rozmytymi. Istnieją natomiast przykłady pokazujące jak rozszerzyć bazę danych o odpowiednie funkcje i operatory, które można zastosować w medycynie czy ekonomii. Popularnością cieszą się natomiast funkcje do wyszukiwania rozmytego w ciągach znaków. PostgreSQL zawiera implementację metod ułatwiających użytkownikowi wyszukanie tekstu nawet, jeśli został on wpisany z błędem: obliczającą odległość Levenshteina między wyszukiwaną frazą a przeszukiwanymi danymi, *metaphone* – określającą przybliżony zapis fonetyczny. Za przykład może posłużyć również pełnotekstowy silnik wyszukiwania Elasticsearch, mogący przeszukiwać tekst licząc odległość Levenshteina.

2.2. Wymagania

Podstawowym wymaganiem jest współczesny interfejs umożliwiający tworzenie zapytań SQL ze wsparciem dla elementów rozmytych. Przyjmując, że użytkownik systemu zna język SQL na poziomie podstawowym, w odróżnieniu od FuzzyQ, tworzony system nie będzie miał graficznej reprezentacji zapytań. Zamiast tego edytor zapytań SQL ma być wzbogacony o kolorowanie składni oraz podpowiedzi kontekstowe, takie jakie występują we wszystkich popularnych środowiskach programistycznych po naciśnięciu skrótu klawiszowego (np. IntelliJ, VS Code). Podpowiedzi kontekstowe mają zawierać zarówno standardowe elementy języka SQL, jak i rozszerzenie w postaci elementów rozmytych. System ma wspierać tworzenie połączeń z bazami danych i przechowywanie ich. Interfejs aplikacji ma umożliwiać wyświetlanie tabel bazy i swobodne przeglądanie wyników zapytań. System powinien działać na wszystkich popularnych systemach operacyjnych: Windows, Linux (Ubuntu) oraz macOS.

System ma posiadać zintegrowany zestaw operatorów, które będzie można dodać do dowolnej bazy. Operatory w systemie będą bazować na operatorach będących częścią rozprawy doktorskiej dr inż. Bożeny Małysiak-Mrozek, zostaną jednak zrefaktoryzowane i przystosowane do systemu tak, aby można było rozszerzyć o nie dowolną bazę danych. Atutem byłaby możliwość samodzielnego tworzenia nowych operatorów.

2.3. Język SQL

Język SQL został zaproponowany przez Donalda Chamberlina i Raymonda Boyce'a dwa lata po tym, jak Edgar „Ted" Codd zaprezentował

koncepcję relacyjnego modelu danych na sympozjum w 1972 roku. Twórcy SQLa zauważyli, że matematyczną notację w językach Codda – algebrze relacji (*relational algebra*) i rachunku relacji (*relational calculus*) – ciężko pojąć osobie bez wykształcenia matematycznego, a także jej zapis na klawiaturze będzie problematyczny. Tak powstał SEQUEL (*A Structured English Query Language*) - język pozwalający na tworzenie złożonych zapytań do relacyjnych baz danych w czytelny i prosty sposób, a także obejmujący operacje modyfikacji danych i administracji bazą. Po rozpowszechnieniu języka SQL wśród twórców oprogramowania konieczna była standaryzacji. Na przestrzeni lat powstało 9 kolejnych wersji standardu, dodające nowe typy danych, rodzaje dozwolonych operacji i zabezpieczenia. Najnowsza wersja standardu, SQL:2016, została wydana w grudniu 2016 roku, m.in. dodając wsparcie dla dokumentów w notacji JSON, formatowanie i parsowanie daty i czasu oraz rozpoznawanie wzorców w wierszach. Niestety, żadna z istniejących implementacji nie jest zgodna ze standardami opracowanymi 2 lata temu. Nie zawsze oznacza to brakujące funkcjonalności, lecz ich realizację przy użyciu składni niezgodnej ze standardem. Dla przykładu, baza Postgres posiada wsparcie dla obiektów JSON już od wersji 9.2 z roku 2012.

2.3.1. Elementy języka

Język SQL posiada zestaw podstawowych typów danych, które mogą być składowane w tabelach i muszą być podane podczas tworzenia kolumn. Najważniejsze to: `CHARACTER`, `BINARY`, `BOOLEAN`, `INTEGER`, `FLOAT`, `REAL`, `DATE`, `TIME`, `YEAR`, `MONTH`, `DAY`, `HOUR` i `MINUTE`. Ciągi, takie jak `CHARACTER` i `BINARY` przyjmują domyślnie stałą długość, gdzie puste miejsce jest wypełniane znakami spacji w przypadku ciągu znaków, a zerami w przypadku ciągu binarnego. Dzięki opcji `VARYING` można przechowywać ciągi zmiennej długości. Drugim standardowym

elementem języka są operatory: porównania (`=`, `<>`, `<`, `<=`, `=>`, `>`), zawierania w zakresie liczb (`BETWEEN`), zawierania w zbiorze (`IN`), podobieństwa ciągów znaków (`LIKE`), porównania do wartości `NULL`/`TRUE`/`FALSE` (`IS`, `IS NOT`), porównania do innej wartości z uwzględnieniem `NULL` (`IS NOT DISTINCT FROM`) oraz operator przemianowania zwracanej kolumny (`AS`). Podobnie jak w zdecydowanej większości języków, nadmiarowe białe znaki są ignorowane, a komentarze mogą być zawarte wewnątrz znaków `/* */`. Jednoliniowe komentarze rozpoczyna się znakami `--`. Przechodząc do składni języka, można z niej wyodrębnić:

- wyrażenia (ang. *expressions*) – mogą zwracać tabele bądź wartości skalarne, np. `age + 1`.
- predykaty (ang. *predicates*) – warunki, składające się z operatorów i wyrażeń, przeliczane na wartości prawda/fałsz/nieznane z użyciem logiki trójwartościowej; służą do filtrowania wyników zapytań i instrukcji, bądź zmiany przebiegu programu, np. `sex='M'`.
- klauzule (ang. *clauses*) – słowa kluczowe, z których zbudowane są zapytania i instrukcje, np. `SELECT name, last_name`, `WHERE age>20`, `SET value=2`.
- instrukcje (ang. *statements*) – pozwalają na odczyt i zapis danych, transakcjami, połączeniami i przebiegiem wykonania programu, są szerzej opisane w podrozdziale 2.3.2. Instrukcje muszą być oddzielone od siebie średnikiem.
- zapytania (ang. *queries*) – instrukcje zwracające dane na podstawie podanych kryteriów, np. SELECT * FROM employees WHERE salary<3000; inne instrukcje zwracają najczęściej ilość zmodyfikowanych (`UPDATE`) lub usuniętych (`DELETE`) wierszy.

W poniższym przykładzie obrazującym elementy języka SQL, najpierw następuje dodanie wierszy do tabeli ze studentami, a następnie jest wykonane zapytanie zwracające grupy wraz z ilością studentów, którzy ukończyli pierwszy semestr:

```
INSERT INTO students (name, faculty, group, semester)
  VALUES
```

```
    („Krzysztof Miemiec”, „AEI”, „ISMiP”, 3),
    („Jan Kowalski”, „AEI”, „ISMiP”, 1),
    („Adam Nowak”, „AEI”, „BDiS”, 2);

SELECT group, COUNT(*)
  FROM students
  WHERE semester>=2
  GROUP BY group;
```

2.3.2. Rodzaje instrukcji

Z języka SQL można wydzielić kilka podzbiorów zajmujących się osobnymi rodzajami zadań.

- DQL – *Data Query Language* (z ang. język zapytań do danych); zawiera polecenie SELECT zwracające dane z podanej tabeli.
- DML – *Data Manipulation Language* (z ang. język manipulacji danymi); zawiera polecenia INSERT, UPDATE i DELETE, które odpowiednio wstawiają, modyfikują i usuwają dane z tabeli.
- DDL – *Data Definition Language* (z ang. język definiowania danych); składa się z poleceń CREATE, ALTER i DROP (TABLE/INDEX/VIEW), służące do tworzenia, modyfikacji i usuwania tabel, indeksów i widoków.
- DCL – *Data Control Language* (z ang. język kontroli nad danymi); obejmuje polecenia CRAETE USER, GRANT, DENY i REVOKE, pozwalające na nadanie uprawnień, zabronienie wykonania operacji lub odebranie uprawnień.
- TCL – *Transactional Control Language* (z ang. język kontroli nad transakcjami); składa się z poleceń COMMIT i ROLLBACK służących do zatwierdzania i wycofywania zmian z transakcji.

Choć operatory mają zastosowanie w dowolnej instrukcji, najczęściej wykorzystywaną w pracy instrukcją będzie SELECT, w związku z tym zostanie ona opisana szerzej. Przedstawiona niżej składnia zapytania z

użyciem instrukcji SELECT jest oparta o dokumentację bazy PostgreSQL:

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    [ * | expression [ [ AS ] output_name ] [, ...] ]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY grouping_element [, ...] ]
    [ HAVING condition [, ...] ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start [ ROW | ROWS ] ]
```

Klauzula `SELECT` przyjmuje najpierw ciąg wyrażeń do zwrócenia w wyniku. W celu zwrócenia wszystkich kolumn można wykorzystać znak `*`. Domyślnie uzupełniane słowo kluczowe `ALL` powoduje zwrócenie wszystkich wyników zapytania, wliczając duplikaty. Po dodaniu słowa kluczowego `DISTINCT`, duplikaty nie są zwracane w wyniku zapytania, a `DISTINCT ON` usuwa zduplikowane wiersze odpowiadające podanym wyrażeniom.

Następnie obliczany jest wynik elementów z klauzuli `FROM`. Może być to jeden lub więcej elementów, na które składają się: nazwy tabel, zagnieżdżone instrukcje, nazwy funkcji i złączenia (`JOIN`). W przypadku występowania wielu elementów, zawartość ich wierszy jest łączona. Po słowie kluczowym `ON` wymagane jest podanie predykatu określającego warunek złączenia tabel. Klauzula `JOIN` pozwala na złączenie dwóch elementów przyjmowanych przez klauzulę `FROM`. `CROSS` oraz `INNER JOIN` zwracają proste złączenie, takie jakie powstaje przy podaniu wielu elementów do klauzuli `FROM`, ograniczone jednak warunkiem następującym po słowie kluczowym `ON`. Można je zastąpić przy użyciu klauzul `FROM` i `WHERE`. `LEFT OUTER JOIN` zwraca natomiast

złączone elementy obu tabel, wraz z elementami tabeli po lewej stronie, które nie mogły zostać dopasowane przy użyciu warunku i w których brakujące pola zostały zastąpione `NULL`ami. Analogicznie, `RIGHT OUTER JOIN` zwraca złączone elementy obu tabel wraz z niedopasowanymi elementami prawej tabeli, a `FULL OUTER JOIN` zwraca złączone i niedopasowane elementy obu tabel, w których brakujące pola przyjmują wartość `NULL`.

Jeśli zostanie podana klauzula `WHERE`, wynik będzie zawierał jedynie wiersze spełniające podane w niej predykaty, które mogą być połączone operatorami logicznymi `AND` i `OR`.

Kolejna, opcjonalna klauzula `GROUP BY` pozwala na złączenie wielu wierszy, które spełniają warunki grupowania, w pojedyncze wiersze dzielące pola z tymi samymi wartościami. Funkcje agregujące (np. `COUNT`, `AVG`, `SUM`, `MAX`, `MIN`) zostaną wykonane na wierszach grup, tworząc oddzielne wyniki dla każdej z nich. Klauzula `HAVING`, podobnie jak `WHERE`, przyjmuje predykaty sprowadzane do wartości logicznej, jednak nie mogą być użyte kolumny niezgrupowane bądź niebędące funkcjami agregującymi. Klauzula `HAVING` jest aplikowana na zapytania zawierające wiersze zgrupowane, a zatem po zgrupowaniu ich przy użyciu `GROUP BY`. Jeżeli klauzula `GROUP BY` nie występuje w zapytaniu, jest ono automatycznie przekształcane w zapytanie zgrupowane.

Klauzula `ORDER BY` przyjmuje listę wyrażeń w postaci nazw lub liczb porządkowych kolumn wynikowych lub wyrażeń utworzonych z kolumn wejściowych. Liczbę porządkową kolumny można użyć, gdy nazwy kolumn wynikowych powtarzają się, czego można jednak uniknąć stosując operator `AS`. Do każdego wyrażenia można dodać słowo kluczowe `DESC`, aby posortować wyniki malejąco. Domyślnie przyjmowana jest wartość `ASC` – sortowanie rosnące. Dopuszczalne jest również użycie własnych operatorów sortowania. Ostatnim elementem wyrażenia są słowa kluczowe `NULLS LAST` i `NULLS FIRST`. Oznaczają one odpowiednio posortowanie wierszy z wartościami `NULL` jako ostatnie i jako pierwsze. Domyślnie przyjęto, że w przypadku sortowania

rosnącego wartości `NULL` są wartościami ostatnimi, a w przypadku sortowania malejącego, wartości `NULL` znajdują się na początku.
Ostatnimi klauzulami są `LIMIT` oraz `OFFSET`. Ograniczają one wyniki do zadanej liczby wierszy, a także pozwalają wyświetlić je z przesunięciem. W przypadku wykorzystywania tych klauzul ważne jest użycie klauzuli `ORDER BY`, gdyż bez jej podania standard SQL nie gwarantuje zwracania wyników w tej samej kolejności i w rezultacie można otrzymać różne wyniki dla tego samego zapytania.

2.4. Logika rozmyta

TODO

Rozdział zawiera takie elementy, jak:

- analiza tematu,
- wprowadzenie do dziedziny (state of the art) – sformułowanie problemu,
- poszerzone studia literaturowe, przegląd literatury tematu (należy wskazać źródła wszystkich informacji zawartych w pracy),
- opis znanych rozwiązań, algorytmów, osadzenie pracy w kontekście,
- tytuł rozdziału jest często zbliżony do tematu pracy,
- rozdział jest wysycony cytowaniami do literatury: książek [1], artykułów w czasopismach [2] i materiałach konferencyjnych [3].

3. Przedmiot pracy

Do zbudowania systemu dydaktycznego postanowiono wykorzystać jedno z najpopularniejszych w ostatnich latach środowisk do tworzenia aplikacji wieloplatformowych – Electron. Środowisko to jest wykorzystywane m.in. przez komunikatory Skype i Slack, środowisko programistyczne VS Code czy narzędzie do zarządzania repozytoriami z kodem GitKraken. Aplikacja, oparta o frameworki React i Redux, zostanie napisana w języku TypeScript, który jest następnie tłumaczony do języka JavaScript. Całość jest pakowana do skompresowanej i zoptymalizowanej paczki z kodem przy użyciu narzędzia Webpack, a następnie zamknięta w pliku natywnej aplikacji (np. *.exe dla Windows, *.app dla macOS) poprzez wspomniane środowisko Electron. Uzasadnieniem powyższych wyborów jest uniwersalność, prostota działania, a także znajomość technologii przez autora pracy.

Poza samą aplikacją, system dydaktyczny ma zawierać również zestaw funkcji rozmytych stanowiących rozszerzenie dla baz danych. Podobnie jak w poprzednim rozwiązaniu, zostanie użyta baza Postgres, a funkcje pozostaną w języku C, jednak zostaną poddane refaktoryzacji.

3.1. Wykorzystane technologie

3.1.1. PostgreSQL

3.1.2. JavaScript/ECMAScript

3.1.3. TypeScript

3.1.4. Webpack

3.1.5. Electron

3.1.6. React

3.1.7. Redux

3.2. Implementacja operatorów rozmytych

3.3. Architektura systemu

Rozdział zawiera takie elementy, jak:

- rozwiązanie zaproponowane przez dyplomanta,
- analiza teoretyczna rozwiązania,
- uzasadnienie wyboru zastosowanych metod, algorytmów, narzędzi.

4. Badania

Rozdział przedstawia przeprowadzone badania. Jest to zasadnicza część i musi wyraźnie dominować w pracy. Badania i analizę wyników należy przeprowadzić, tak jak jest przyjęte w środowisku naukowym (na przykład korzystanie z danych porównawczych, walidacja krzyżowa, zapewnienie powtarzalności testów itd.).

4.1. Metodyka badań

Rozdział ten zawiera:

- przedstawienie i omówienie zastosowanych algorytmów,
- szczegóły wybranych fragmentów implementacji.

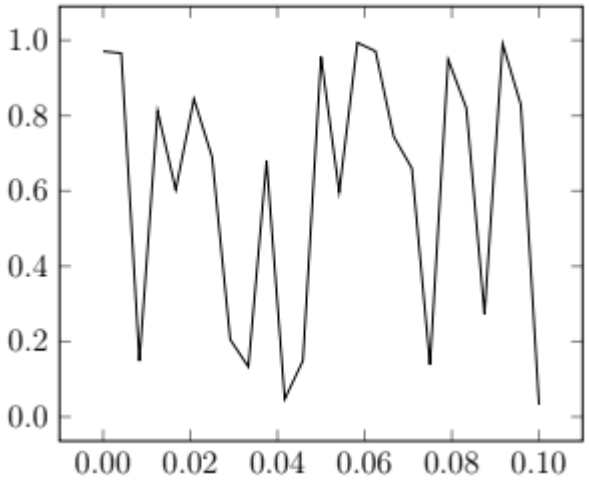
4.2. Zbiory danych

Opis danych wykorzystywanych podczas badań.

4.3. Wyniki

Prezentacja wyników, opracowanie i poszerzona dyskusja wyników, wnioski.

W całym dokumencie powinny znajdować się odniesienia do zawartych w nim ilustracji (Rys. 4.1).



Rys.4.1. *Wykres przebiegu funkcji*

Tekst dokumentu powinien również zawierać odniesienia do tabel (Tabela 4.1).

Tabela 4.1. *Rozmiar czcionek w tytułach rozdziałów*

Poziom 1	24 pt
Poziom 2	20 pt
Poziom 3	16 pt

5. Podsumowanie

otwarcie źródła, rozszerzalność, klient sql

Rozdział ten zawiera:

- syntetyczny opis wykonanych prac,
- wnioski,
- opis możliwości rozwoju, kontynuacji prac, potencjalne nowe kierunki,
- informację, czy cel pracy zrealizowany.

Bibliografia

[1] Imię Nazwisko, Imię Nazwisko. *Tytuł książki*. Wydawnictwo, Warszawa, 2017.

[2] Imię Nazwisko, Imię Nazwisko. Tytuł artykułu w czasopiśmie. *Tytuł czasopisma*, 157(8):1092–1113, 2016.

[3] Imię Nazwisko, Imię Nazwisko, Imię Nazwisko. Tytuł artykułu konferencyjnego. *Nazwa konferencji*, str. 5346–5349, 2006.

Spis skrótów i symboli

DNA	kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)
MVC	model – widok – kontroler (ang. *model–view–controller*)
N	Liczebność zbioru danych

Zawartość dołączonej płyty

Na płycie DVD dołączonej do dokumentacji znajdują się następujące materiały:

- praca w formacie pdf,
- źródła programu,
- zbiory danych użyte w eksperymentach.

Spis rysunków

Spis tabel

SQLf - implementacja w Javie
http://calypso.cs.put.poznan.pl/projects/sqlf_j/
http://calypso.cs.put.poznan.pl/projects/sqlf_j2006/tutorial/

FSQL - Oracle (model GEFRED):
http://www.lcc.uma.es/~ppgg/FSQL.html

Grupowanie:
http://bdas.polsl.pl/BDAS%6017%20-%20Realizacja%20grupowania%20i%20agregacji%20danych%20w%20module%20translacji%20zapytań%20rozmytych%20na%20zapytania%20klasyczne.pdf?Id=575&val=3

więcej o GEFRED:
http://sci-hub.tw/https://doi.org/10.1016/0020-0255(94)90069-8

fuzzy (trapezoid):
http://sci-hub.tw/https://doi.org/10.1016/j.proeng.2011.11.2653

dokumentacja postgresa dot. fuzzy text search:
https://www.postgresql.org/docs/9.1/static/fuzzystrmatch.html

fuzzy queries - artykuł polskiego autorstwa:

http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-90ea0e0a-66bc-4424-8711-f35c1cb4859f
wyszukiwanie fonetyczne tekstu - metaphone:
http://www.informit.com/articles/article.aspx?p=1848528

fuzzy query w elasticu:
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-fuzzy-query.html

historia sqla:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6359709

sql 2016
https://modern-sql.com/blog/2017-06/whats-new-in-sql-2016