|  |
| --- |
| POLITECHNIKA ŁÓDZKA  INSTYTUT INFORMATYKI  PRACA DYPLOMOWA INŻYNIERSKA |
| Interaktywne nauczanie bazodanowych języków zapytań |
|  |
| Autor: Promotor: |
| Łukasz Ochmański dr. Inż Krzysztof Myszkorowski  Nr albumu: 183566  Łódź, 8 luty 2015 |

Streszczenie

Informatyka

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej

Praca dyplomowa inżynierska

Interaktywne nauczanie bazodanowych języków zapytań

Łukasz Ochmański

Nr albumu: 183566

Celem pracy jest stworzenie interaktywnego samouczka wspierającego polskich studentów w nauce bazodanowych języków zapytań. Główną zaletą systemu jest łatwość dostępu i brak konieczności instalacji silników bazodanowych typu Oracle database client czy też konfiguracji narzędzi typu SQL Server Management Studio. Niekiedy konfiguracja przerasta umiejętności studentów, a w najlepszych wypadku pochłania godziny lub dni. Tego typu działania nie są głównym celem dydaktycznym przedmiotu o nazwie „Podstawy baz danych”.   
Aby skupić uwagę studentów na nauce języka, postanowiłem stworzyć ten oto serwis.

**Spis treści**

1. Cel projektu 6

2. Opis struktury pracy 6

3. Założenia dotyczące projektu 6

4. Dostęp 6

4.1. Adres 6

4.2. Serwer 6

4.3. CVS kablowanie Poziome 6

4.4. Użyte narzędzia 6

5. Wymagania sprzętowe 7

5.1. Przeglądarka 7

5.2. Strona kodowania 7

5.3. System operacyjny 7

5.4. CPU 7

5.5. Pamięć operacyjna 7

6. Schemat działania 7

6.1. Podział modułów 7

6.2. Sposoby komunikacji 7

6.3. Model trójwarstwowy DAO 7

7. Baza danych 7

7.1. Instancja STUDENT 7

7.2. Instancja ADMINISTRATOR 7

8. Warstwa prezentacji (front-end) 8

8.1. HTML 5 8

8.2. JavaScript 8

8.2.1. jQuery 8

8.3. Websocket 8

8.4. STOMP 8

8.5. JSON 8

8.6. BootStrap 8

8.7. CSS 3 8

9. Warstwa dostępu do danych (back-end) 8

9.1. JVM 8

9.2. Java 8 8

9.3. Lambda 8

9.4. Apache Tomcat 8 8

9.5. Spring framework 8

9.6. Spring MVC 8

9.7. Spring boot 8

9.8. Spring session 8

9.9. H2 embedded 8

9.10. Gradle 8

10. Techniki, wzorce architektoniczne oraz programistyczne 9

10.1. DAO 9

10.2. MVC 9

10.3. Subscribe 9

10.4. Listener 9

10.5. Singleton 9

10.6. Wyrażenia Lambda 9

11. Funkcjonalności 9

11.1. Praca grupowa 9

11.2. Praca indywidualna 9

11.3. Forum 9

11.4. Komunikaty 9

11.5. Materiały naukowe 9

11.6. HELP 9

12. Szczegółowy opis implementacji 9

12.1. Okablowanie Poziome 9

13. Dokumentajca techniczna 9

13.1. API 9

14. Przykłady działania aplikacji oraz dokumentacja użytkownika 10

14.1. Okablowanie Poziome 10

14.2. Praca grupowa 10

15. Bezpieczeństwo 10

15.1. WebSocket Authentication 10

15.2. Uprawnienia do bazy danych 10

15.3. Izolacja 10

15.4. Atomowość 10

15.5. Transakcyjność 10

16. Dostępność i niezawodność 10

16.1. Load Balancing 10

16.2. Hot backup 10

16.3. Incremental backup 10

17. Analiza wydajności 10

17.1. Porównanie WebSocket vs HTTP 10

17.2. H2 vs Oracle 10

17.3. H2 vs MS SQL Server 10

18. Perspektywy rozwoju 11

18.1. Moduł logowania 11

18.1.1. Resetowanie haseł 11

18.1.2. Przypominanie zapomnianych haseł 11

18.1.3. Single Sign-On, Kerberos, LDAP, Active Directory 11

18.1.4. Blokowanie dostępu 11

18.1.5. Czarna lista 11

18.2. Moduł oceniania 11

18.2.1. Wykresy ocen, SVG 11

18.2.2. Rozkład normalny, SVG 11

18.2.3. Statystyka grupy, roku 11

18.2.4. Archiwum ocen 11

18.2.5. Wykrywanie plagiatu 11

18.3. Rozbudowa panelu administracyjnego 11

18.3.1. Automatyczne dodawanie pytań 11

18.3.2. Modyfikacja kont użytkowników 11

18.3.3. Analiza logów 11

18.3.4. Personalizacja wyglądu 11

18.4. Integracja z platformą Moodle 11

18.5. Interaktywność 11

18.5.1. Zapamiętywanie wpisywanych zapytań 11

18.5.2. Podpowiedzi składni w czasie rzeczywistym 11

18.5.3. Podpowiadanie nazw obiektów 11

18.5.4. Analiza stylu i składni 11

18.6. Angielska wersja językowa. 11

19. Wnioski i podsumowanie 12

19.1. Okablowanie Poziome 12

20. Szczegółowy opis wybranej technologii 12

20.1. Okablowanie Poziome 12

# Wstęp

## Historia SQL

Rozwój relacyjnych baz danych, który miał miejsce w latach 70-tych ubiegłego wieku uwarunkował konieczność opracowania języka do manipulacji, wyciągania i obsługi danych w bazach.

Pierwszym oficjalnym językiem relacyjnych baz danych, był SEQUEL (Structured English Query Language), opracowany przez pracowników firmy IBM (Raymond F.Boyce oraz Donald Chamberline). Zaimplementowany w 1973 roku w SYSTEM R – pierwszym silniku bazodanowym opartym o model relacyjny (jednak pierwszy komercyjny system RDBSM to wdrożenie firmy ORACLE w 1979 r.).

Jak sama nazwa wskazuje, SEQUEL to język w domyśle przyjazny dla użytkownika, służący odpytywaniu baz. Jednym z założeń była łatwość tworzenia zapytań, operacji na zbiorach za pomocą słów kluczowych w języku angielskim. Język miał być intuicyjny i prostoty. Te cechy to także założenia samego modelu relacyjnego i chyba właśnie dlatego, systemy baz danych oparte o model relacyjny podbiły świat i są do dziś dominującymi środowiskami bazodanowymi.

Nazwa ewoluowała – SEQUEL, okazała się być nazwą zastrzeżoną przez brytyjską firmę przemysłu lotniczego. Stąd została skrócona do znanej obecnie formy czyli SQL (Structured Query Language).

Najważniejszymi systemami RDBMS (Relational DataBase Management System), w których podstawowym językiem jest SQL, to oczywiście : MS SQL Server, Oracle, DB2, MySQL, PostgreSQL, Sybase.

## Standardy, dialekty

Konkurencyjność rynku spowodowała konieczność ustandaryzowania języka SQL i na szczęście stało się to już w roku 1986, kiedy został opracowany przez ANSI pierwszy standard określany jako SQL:86. Podkreślam, że na szczęście tak szybko, bo choć istnieją istotne różnice np. w nazwach implementowanych funkcji, to ogólne zasady dla relacyjnych baz danych, różnych producentów, są spójne. Ma to znaczenie szczególnie podczas integracji platform i dla nas, pracujących w różnych środowiskach.

Powstały, więc dialekty językowe. Transact-SQL (T-SQL) – historycznie wprowadzony przez Sybase, rozwijany do dziś przez Microsoft w SQL Server. Inne dialekty, mające duże znaczenie na rynku to oczywiście PL/SQL (firmy ORACLE) oraz SQL/PSM (najpopularniejszy silnik relacyjny w serwisach WWW – MySQL).

Pomimo różnic w dialektach, nazwach funkcji, typach danych – istnieje szeroki wspólny mianownik – relacyjny model oparty o teorię zbiorów. Dlatego, jeśli poznasz T-SQL, odnajdziesz się szybko np. w bazie MySQL czy Oracle.

Standardy ANSI są regularnie aktualizowane. Od 1986 roku zostało opublikowanych szereg wersji (aktualnie obowiązująca to ANSI SQL:2011/2011), wprowadzających porządek w nowych funkcjonalnościach. Np. w SQL:2003 zostały wprowadzone standardy związane z obsługą XML.

## Kategorie SQL

Istnieją grupy istotnych akronimów, które powinny być znane każdemu użytkownikowi bazy danych. W zależności od zastosowań są to :

DDL – Data Definition Language – czyli komendy dot. tworzenia, modyfikacji obiektów w bazie np. CREATE TABLE, ALTER VIEW, DROP,  
DML – Data Modification Language (UPDATE, INSERT, DELETE)  
DCL – Data Control Language – kontrola uprawnień (GRANT, DENY, REVOKE)  
TCL – Transaction Control Language – obsługa transakcji np. BEGIN TRANSACTION, COMMIT, ROLLBACK.  
I w końcu DQL – Data Querying Language – czyli polecenie SELECT  
Dotyczy on tylko podzbioru języka SQL – związanego z pisaniem zapytań (kwerend).

# Cel projektu

Celem głównym pracy jest zaprojektowanie i zbudowanie aplikacji internetowej przy użyciu dostępnych technologii typu open source w celu nauczania bazodanowych języków zapytań w standardzie SQL-92. Natomiast celem stworzenia aplikacji jest wsparcie studentów w opanowaniu biegłego posługiwania się językiem SQL, któgo znajomość jest kluczowa w poznawaniu niezbędnej wiedzy teoretycznej z zakresu systemów baz danych.

Użytkownicy, zdobywają kompetencje swobodnego tworzenia dowolnych kwerend SQL i umiejętności korzystania z wiedzy zawartej w bazach. Poznają metodykę i narzędzia do sprawnego przetwarzania dużych ilości danych, wyciągania tylko istotnych informacji

# Opis struktury pracy

## Wstęp

W tym rozdziale jest opisana geneza powstania relacyjnych baz danych i języków zapytań.

## Cel projektu

W tym rozdziale znajduje się charakterystyka ogólny opis prezentowanego projektu.

## Opis struktury pracy

Skrócone zestawienie wszystkich rozdziałów.

## Założenia dotyczące projektu

Założenia typu, wymagana składnia, niedozwolone operacje, możliwości i ograniczenia techniczne, czego można oczekiwać od systemu, a czego nie.

## Dostęp

Ogólne informacje dotyczące lokalizacji aplikacji, sposobów dostępu, kod źródłowego, wymaganych uprawnień.

## Wymagania sprzetowe

Specyfikacja wymagań technicznych, niezbędnych do uruchomienia aplikacji.

## Architektura systemu

Ogólny zarys budowy aplikacji. Model graficzny.

## Warstwa prezentacji (front-end)

Ogólny zarys, zasada działania i funkcje pełniące przez warstwę prezentacji.

## Warstwa dostępu do danych (back-end)

Ogólny zarys, zasada działania i funkcje pełniące przez warstwę dostępu do danych.

## Warstwa danych

Opis silnika bazodanowego, możliwości, trybów pracy, protokołów komunikacji i sposobów integracji z klientami. Utworzone instancje, schematy i użytkownicy.

## Schemat działania

Zasada funkcjonowania i przepływu informacji pomiędzy poszczególnymi modułami.

## Techniki, wzorce architektoniczne, wzorce programistyczne

Opis zastosowanych technik i wzorców projektowych (design patterns).

## Opis wybranych funkcjonalności

Szczegółowe wyjaśnienie zasady działania najważniejszych funkcjonalności systemu.

## Szczegółowy opis implementacji

Analiza kodu źródłowego.

## Dokumentacja techniczna

Dokumentacja dla programistów chcących rozwijać istniejącą aplikację i dla osbób chcących korzystać z dostępnych metod API (Application Programming Inteface).

## Przykłady działania aplikacji

Graficzna prezentacja, pokaz możliwości aplikacji

## Bezpieczeństwo

Poruszenie tematów związanych z separacją danych, uprawnień użytkowników, dostępu do ocen prze niepowołane osoby.

## Dostępność i niezawodność

Tutaj zostaną poruszone kwestie dostępności do systemu w czasie i miejscu. Odporność na ataki, awarie, wirusy, zawieszenia, wyjątki, błędy. Akcja przedsięwzięte w celu uniknięcia problemów.

## Analiza wydajności

Porównanie dostępnych na rynku baz oraz porównanie dostępnych protokołów, które mogą zastąpić isniejące.

## Perspektywy rozwoju

Lista planowanych funkcjonalności

## Wnioski i podsumowanie

Wnioski i podsumowanie.

## Bibliografia

TextText TextText

## XXXXXXXXX

TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText TextText

# Założenia dotyczące projektu

* Aplikacja została stworzona do użytku wszystkich zainteresowanych i może być używana nieodpłatnie przez wszystkich studentów w celach dydaktycznych,
* W poniższej pracy będę używał akronimu INBJZ jako skrót od „Interaktywne nauczanie bazodanowych języków zapytań”.
* INBJZ jest aplikacją typu open source i można korzystać z kodu źródłowego w celach niekomercyjnych.
* Użyte przeze mnie bilbioteki i frameworki H2, Apache Tomcat, Spring są typu Open Source i korzystanie z kodu źródłowego mojej aplikacji zobowiązuje wszystkich jego użytkowników do udostępnienia całego kodu i podania autora na zasadach omówionych w licencji.
* Podzas korzystania z aplikacji należy być ostrożnym, gdyż na pewno istnieje wiele możliwośći „złamania” systemu. Starałem się jednak jak najbardziej taką możliwość ograniczyć i wszystkie operacje wykonywane na głównym schemacie są odwracane przy każdym zapytaniu.
* Aby dokonać trwałych zmian należy stworzyć swój schemat i to powinno zwiększyć ochronę danych.
* Nie ma gwarancji, ani zabezpieczeń, że inny użytkownik nie usunie informacji wprowadzonych przez innego użytkownika.
* Każda skuteczna zmiana w bazie jest logowana (zakończona powodzeniem) i widoczna przez administratora systemu.
* Wszystkie próby (skuteczne i nieskuteczne) udzielenia odpowiedzi są logowane i widoczne przez administratora systemu.
* Dialekt SQL bazy danych H2 powinien być zgodny z dialektem MySQL oraz MS SQL Server.
* Baza wspiera rozróżnianie wielkości znaków, jeśli używa się cudzysłowów.
* Zarówno producent silnika bazodanowego H2 jak i autor aplikacji (Łukasz Ochmański) nie ponoszą odpowiedzialności za problemy spowodowane użytkowaniem aplikacji.
* Wspierane kodowanie to UTF-8. Możliwa jest również obsługa Unicode. Można zatem śmiało wprowadzać polskie znaki.
* Dla poprawnego działania aplikacji zalecane jest używanie najnowszej przeglądarki Firefox.
* Stabilność systemu nie jest gwarantowana. Projekt został stworzony jedynie do domowego użytku.
* W razie jakichkolwiek problemów, należy się kontaktować ze mna poprzez forum na stronie lub e-mail.

# Dostęp

## Adres Wersja beta aplikacji jest dostępna pod adresem: <http://5.135.146.42:8080/> .

## Godziny dostepu

## W planach system będzie dostępny całą dobę. Jednak w przypadku wystąpienia błędów wymagany będzie restart, co spowoduje unieruchomienie aplikacji na nie więcej niż 5 minut.

## Docelowa grupa odbiorców

## W początkowej fazie wdrożenia nie będzie ograniczeń co do osób korzystających z serwisu. Każdy może anonimowo korzystać z INBJZ. Docelowo będą to studenci przedmiotu „Podstawy baz danych” na wydziale Fizyki Technicznej Informatyki i Matematyki Stosowanej Politechniki Łódzkiej kierunku informatyka. W obecnej wersji można się przedstawić poprzez wpisanie swojego identyfikatora w prawym górnym rogu ekranu. Nie ma przeprowadzanej żadnej autentykacji, więc można się podać za dowolną osobę i nie ma z tego tytułu żadnych negatywnych konsekwencji.

## VCS (Version Control System)

## Repozytorium kontroli wersji jest prowadzone na bieżąco w systemie GIT. Można sklonować sobie całe repozytorium poprzez wydanie polecenia:

## git clone https://ochman2000@code.google.com/p/interaktywne-nauczanie-jezykow-bazodanowych/

## Aby jednak brać czynny udział w rozwoju projektu należy mnie o to poprosić. Wtedy udostępnię zasoby.

## Kod źródłowy i użyte narzędzia

## Projekt został napisany w języku Java. Aby go uruchomić należy posiadać Java Development Kit 8. Następnie gotowy projekt można zaimiportować do dowolnego środowiska programistycznego za pomocą systemu budowania (Build Automation Software) Gradle. Gotowy projekt jest skonfigurowany do działania w IntelliJ 14. Kod źródłowy można również podejrzeć za pomocą dowolnej przeglądarki internetowej pod adresem: <https://code.google.com/p/interaktywne-nauczanie-jezykow-bazodanowych/>

# Wymagania sprzętowe

## Przeglądarka

## Wymagana przeglądarka zdolna do obsługi JavaScript, HTML5, CSS3 oraz WebSocket. Strona działa na wszystkich najnowszych przeglądarkach, ale zalecana jest Mozilla Firefox.

## Strona kodowania

## UTF-8

## System operacyjny

## Dowolny system operacyjny z graficznym interfejsem użytkownika i działającą przeglądarką.

## Rozdzielczość ekranu

## Zalecana rodzielczość: 1366x768 pikseli.

## Pamięć operacyjna

## Zdolna uruchomić przeglądarkę Firefox czyli około 512MB

# Architektura systemu

## Sposoby komunikacji

## W aplikacj są używane trzy główne protokoły komunikacji:

## HTTP Poprzez ten protokół serwowane są statyczne strony HTML, pliki CSS oraz pliki z kodem JavaScript do klienta i od klienta. Za pomocą prostych metod GET i POST można uzyskać prawie wszystkie funkcjonalności omawianej aplikacji.

## WebSocket Poprzez ten protokół następuje wymiana danych pomiędzy serwerem, a klientem. Wszystkie zapytania i komendy wpisywane w polu tekstowym na stronie HTML są wysyłane w postaci JSON (JavaScript Object Notation) poprzez tekst lub strumień bajtów, następnie wysyłane do serwera na jeden z dostępnych portów bezpośrednio w niższej warstwie modelu TCP/IP. Zapewnia to najwyższą efyktywność przy jednoczesnym zapewnieniu pewności transmisji.

## JDBC (Java Database Connectivity) komunikuje się ze sterownikiem bazy danych i tłumaczy wszystkie zapytania w postaci tekstowej z poziomu kodu napisanego w języku Java na natywne komendy udostępnione przez producenta bazy danych. Następnie dane są zwracane poprzez sterownik do programu wysokiego poziomu i interpretowane przez programistę, czyli mnie.

## Model trójwarstwowy

## Cała architektura projektu opiera się o klasyczny model trójwarstowy (Three-Tier-Architecture), gdzie moduły systemu zostały oddzielone i uniezależnione od siebie. Każdy z modułów jest samodzielny i może w każdej chwili zostać wymieniony. Poniższy rysynek prezentuje zastosowany model:

## http://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Overview_of_a_three-tier_application_vectorVersion.svg/593px-Overview_of_a_three-tier_application_vectorVersion.svg.pngźródło: http://en.wikipedia.org/wiki/Multitier\_architecture

## Data Access Object (DAO)

# Warstwa prezentacji (front-end)

## HTML 5

## JavaScript

### jQuery

## WebSocket

## STOMP

## JSON

## BootStrap

## CSS 3

Pozi

# Warstwa dostępu do danych (back-end)

## JVM

## Java 8

## JDBC

## Lambda

## Apache Tomcat 8

## Spring framework

## Spring MVC

## Spring boot

## Spring session

## H2 embedded

## Gradle

Pozi

# Warstwa danych

W projekcie użyto bazy danych H2. Technologia została wybrana ponieważ zapewnia najszybszy dostęp do aplikacji, które działają na tej samej instancji wirtualnej maszyny Java JVM

Schemat ten uwzględnia dostęp do Internetu dzięki dwóm modemom różnych usługodawców internetowych (w razie awarii lub braku dostępu do internetu jednego z nich). Modemy połączone są z routerem a pomiędzy nimi umieszczony jest sprzętowy firewall. Następnie router połączony jest z przełącznikiem (switchem) w Centralnym Punkcie Dostępu (CPD).

Proponujemy dwa rodzaje rozwiązań a mianowicie:

## Instancja STUDENT

W pierwszym wariancie konstrukcji sieci Centralny Punkt Dystrybucji podłączony zostanie z piętnastoma switchami (po jednym na każdym piętrze) umieszczonymi w Lokalnych Punktach Dostępu (LPD). Szafa w Centralnym Punkcie Dystrybucji będzie również odgrywała rolę LPD dla stacji roboczych znajdujących się na parterze.

## Instancja ADMINISTRATOR

Drugi wariant zakłada połączenie switcha w CPD ze stacjami roboczymi znajdującymi się na parterze budynku oraz z pięcioma switchami w LPD umieszczonymi na piętrach tak aby swoją pracą obejmowały (każdy z nich) swoje piętro oraz dwa piętra sąsiadujące (jedno z góry oraz jedno z dołu).

# Schemat działania

## Przykładowy workflow

## Weryfikacja odpowiedzi

## Przyznanie punktów

## Błędna odpowiedź

# Techniki, wzorce architektoniczne oraz programistyczne

## DAO

## MVC

## Subscribe

## Listener

## Singleton

## Wyrażenia Lambda

Pozi

# Opis wybranych funkcjonalności

## Praca grupowa

## Praca indywidualna

## Forum DISQUS

## Komunikaty

## Materiały naukowe

## HELP

Pozi

# Szczegółowy opis implementacji

## Okablowanie Poziome

Pozi

# Dokumentajca techniczna

## API

Pozi

# Przykłady działania aplikacji oraz dokumentacja użytkownika

## Okablowanie Poziome

## Praca grupowa

Pozi

# Bezpieczeństwo

## WebSocket Authentication

## Uprawnienia do bazy danych

## Izolacja

## Atomowość

## Transakcyjność

# Dostępność i niezawodność

## Load Balancing

## Hot backup

## Incremental backup

Pozi

# Analiza wydajności

## Porównanie WebSocket vs HTTP

## H2 vs Oracle

## H2 vs MS SQL Server

Pozi

# Perspektywy rozwoju

## Moduł logowania

### Resetowanie haseł

### Przypominanie zapomnianych haseł

### Single Sign-On, Kerberos, LDAP, Active Directory

### Blokowanie dostępu

### Czarna lista

## Moduł oceniania

### Wykresy ocen, SVG

### Rozkład normalny, SVG

### Statystyka grupy, roku

### Archiwum ocen

### Wykrywanie plagiatu

## Rozbudowa panelu administracyjnego

### Automatyczne dodawanie pytań

### Modyfikacja kont użytkowników

### Analiza logów

### Personalizacja wyglądu

## Integracja z platformą Moodle

## Interaktywność

### Zapamiętywanie wpisywanych zapytań

### Podpowiedzi składni w czasie rzeczywistym

### Podpowiadanie nazw obiektów

### Analiza stylu i składni

## Angielska wersja językowa.

# Wnioski i podsumowanie

## Okablowanie Poziome

Pozi

# Bibliografia

[1] Jakub Kasprzak   
Język SQL – historia, standardy   
Styczeń 2013  
URL <http://www.sqlpedia.pl/jezyk-sql-historia-standardy/>

[2] Wikipedia

<http://pl.wikipedia.org/wiki/SQL>

[3] <http://www.h2database.com/html/links.html#projects>

[4] Core J2EE Patterns - Data Access Object

<http://www.oracle.com/technetwork/java/dataaccessobject-138824.html>

[5] Mutli-their-architecture

http://en.wikipedia.org/wiki/Multitier\_architecture