POLITECHNIKA WARSZAWSKA

Wydział Elektroniki i Technik Informacyjnych

Instytut Telekomunikacji

**PRACA DYPLOMOWA**

**MAGISTERSKA**

**Rafał Józwiak**

**Mechanizmy zwiększania wydajności nowoczesnych analitycznych baz danych na potrzeby rynku IT – studium przypadków i rozwiązań**

Opiekun naukowy:

dr inż. Marcin Kowalczyk

..........................................

ocena pracy

..........................................

data i podpis Przewodniczącego

Komisji Egzaminacyjnej

Warszawa, wrzesień 2016

# Wstęp

W dzisiejszych czasach zaobserwowano tendencje polegającą na gromadzeniu oraz przetwarzaniu coraz większych ilości danych. Każde większe przedsiębiorstwo posiada swoje własne bazy pozwalające na szybkie znalezienie niezbędnych informacji. Operatorzy telekomunikacyjni także należą do tej grupy ze względu na powiększające się grupy klientów korzystających z ich usług.

Oprócz zwiększenia się ilości przetwarzanych danych zmienia się także architektura sprzętowa, zgodnie z prawem Moore’a, co pozwala na zwiększenie efektywności korzystania z baz. Nowy sprzęt umożliwia znacznie szybszy dostęp do przechowywanych danych, dzięki zwiększeniu pojemności łączy, oraz mocy obliczeniowej jednostek centralnych. Dzięki temu możliwe jest szybsze przetwarzanie zapytań oraz generowanie odpowiedzi.

Wraz ze wzrostem ilości informacji, zwiększa się także zapotrzebowanie na coraz to efektywniejsze mechanizmy umożliwiające szybkie przetwarzanie danych. Kolejnym argumentem za wprowadzaniem nowych metod jest zmieniająca się architektura. Istnieją bowiem rozwiązania, które nie mogły zostać wprowadzone ze względu na problemy wynikające z niedostatecznej mocy obliczeniowej serwerów w przeszłości, natomiast teraz nie ma żadnych przeszkód w ich stosowaniu.

Mając na uwadze taką sytuację powstał pomysł zbadania dostępnych rozwiązań pod kątem możliwości zwiększenia optymalizacji analitycznych baz danych. Istnieją bowiem przesłanki na podstawie, których można wybrać najoptymalniejsze rozwiązanie dla baz danych statystycznego operatora telekomunikacyjnego.

# Plan pracy

1. Synteza literatury.
   1. Zebranie oraz przegląd literatury pod kątem potencjalnych rozwiązań możliwych do użycia w niniejszej pracy
   2. Synteza rozwiązań na podstawie zebranych materiałów
2. Stworzenie środowiska CRM przykładowego operatora telekomunikacyjnego.:
   1. Utworzenie operatorskiej bazy danych
   2. Stworzenie aplikacji serwerowej zarządzającej bazą
   3. Utworzenie cienkiego klienta umożliwiającego podłączenie do aplikacji serwerowej
3. Testy używanych rozwiązań pod kątem wydajności działania.
   1. Porównanie kolumnowego systemu zarządzania bazą danych z systemem wierszowym.
   2. Sprawdzenie różnych systemów indeksowania bazy.
   3. Testy innych rozwiązań, znalezionych podczas powstawania pracy.
4. Praca nad dokumentem, pracy dyplomowej.
   1. Rozpoczęcie prac nad dokumentem.
   2. Stworzenie pierwszej wersji dokumentu pracy dyplomowej.
   3. Wprowadzanie niezbędnych poprawek – maj.
   4. Utworzenie ostatecznej wersji dokumentu pracy.

# Zrealizowane cele

W bieżącym semestrze dokonano analizy rozwiązań na podstawie dostępnej literatury pod względem optymalizacji. Podczas badań skupiono się przede wszystkim na rozwiązaniach dotyczących miejsca, oraz sposobu przechowywania danych.

Podczas poszukiwań znaleziono dwa rozwiązania, które ze względu na wzrost możliwości współczesnych komputerów mogą znacznie zwiększyć efektywność wykorzystania baz danych. Rozwiązaniami tymi są in-memory, oraz bazy danych zorientowane kolumnowo.

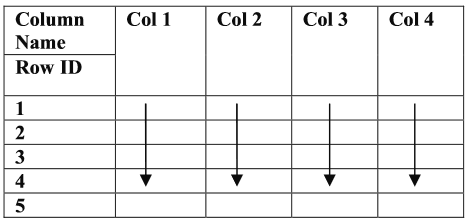
Bazy danych in-memory stosują wykorzystują pamięć operacyjną do przechowywania danych pozwalając na uniknięcie operacji I/O spowolniające odczyt lub zapis danych w bazie, natomiast bazy danych zorientowane kolumnowo zmieniają podejście do sposobu przechowywania danych, znacznie przyspieszając odczyt np. w hurtowniach danych.

Kolejnym celem wykonanym w bieżącym semestrze było znalezienie oraz instalacja odpowiednich narzędzi umożliwiających implementacje systemu CRM przykładowego operatora telekomunikacyjnego. Projekt ten składa się z bazy danych, oraz aplikacji umożliwiającej zarządzanie danym systemem. Jako narzędzia wybrano Google Web Toolkit (GWT) wraz z frameworkiem Spring do budowy aplikacji, oraz Oracle w wersjach 11 i 12 do budowy bazy danych. Zdecydowano się na wybór dwóch wersji systemu baz danych ze względu na potrzebę przetestowania dwóch różnych sposobów przechowywania danych w bazie.

# Podstawowe pojęcia

## Kolumnowe bazy danych

Kolumnową bazą danych nazywamy bazę, w której zastosowany jest odmienny model przechowywania danych. W odróżnieniu od standardowego podejścia, w którym to dane przechowywane są wierszami zawierającymi wszystkie składowe atrybuty danego rekordu, bazy te przechowują dane w kolumnach będących wartościami jednego atrybutu dla wielu rekordów (Rysunek 4.1.).



Rysunek 4.1. – Wzorzec ułożenia danych w kolumnowych bazach danych [1]

Kolumnowy typ przechowywania danych jest szczególnie istotny w dziedzinie hurtowni danych. W związku z tym, że w hurtowniach danych istnieje potrzeba przetwarzania ogromnych ilości danych, to odpowiednią wydajność systemu osiągnąć można stosując właściwe podejście do zarządzania danymi.

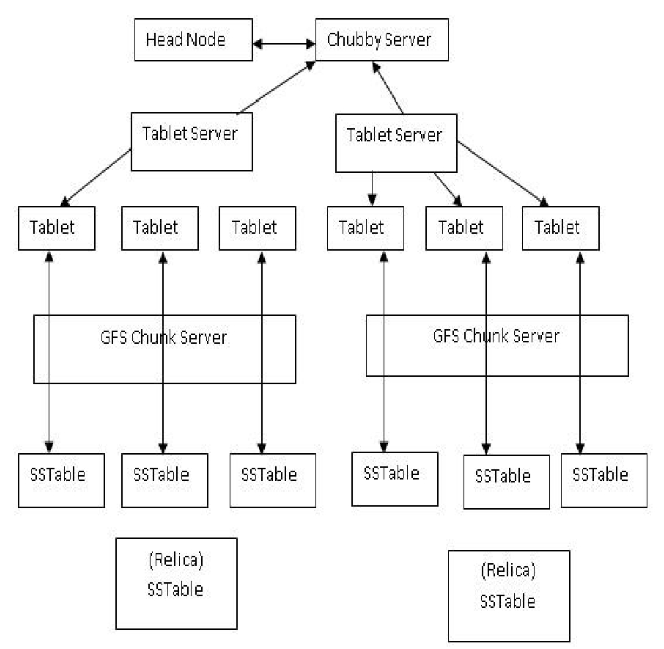
### Architektura systemu

W celu opisania architektury kolumnowych baz danych zdecydowano się na wykorzystanie modelu bazy danych BigTable stworzonej i udostępnianej przez firmę Google w ramach Google App Engine. Podstawowym formatem plików jest SSTable [2]. Pliki te są przygotowane tak aby w czasie dostępu do danych wystarczył pojedynczy odczyt pliku z przestrzeni dyskowej. Raz utworzony plik nie jest w żaden sposób modyfikowany. W przypadku dodawania danych tworzony jest nowy plik SSTable, natomiast stary zostaje usunięty przez Garbage-Collector.

BigTable posiada trzy różne typy serwerów : Master, Tablet i Lock. Serwery typu Master przypisują jednostki Tablet do serwerów typu Tablet. Oprócz tego równoważą obciążenia serwerów tablet, wykrywają utratę połączenia z nimi, znajdują nowe serwery, a także uruchamiają Garbage Collector. Ważnym faktem jest to że dane klienta w żaden sposób nie przechodzą przez Master tj. Klient nie komunikuje się z Masterem. Master utrzymuje komunikację jedynie z Tabletami by upewnić się, że podtrzymują powierzoną im komunikację.

Każdy serwer typu Tablet zarządza pomiędzy 10 a 1000 obiektami tablet, o rozmiarze 100-200 MB. Każdy tablet zawiera wszystkie informacje na temat grup wierszy. Nowo stworzona tabela składa się na jeden obiekt tablet. Serwery Tablet przetwarzają żądania odczytu i zapisu fla obiektów tablet.

Serwery Lock zwane Chubby pozwalają na koordynacje wielu tysięcy węzłów BigTable. Serwer ten sam w sobie jest aplikacją klastrową zaprojektowaną tak, by utrzymywać ruch sieciowy na niskim poziomie. Usługa udostępniana przez Chuby ma za zadanie monitoroważ każdy serwer Tablet, który to w momencie uruchomienia tworzy odpowiednie powiązanie do unikalnego pliku usługi serwera Chubby.



Rysunek 4.2. – Architektura BigTable [2]

### Korzyści i ograniczenia

Wykorzystanie kolumnowych baz danych przynosi następujące korzyści oraz ogramiczenia.

Korzyści:

* Systemy zorientowane kolumnowo są efektywne, gdy potrzebne jest wykonanie agregacji na danych z wielu rzędów w przypadku wykorzystania mniejszego podzbioru kolumn danego wiersza. Jest to spowodowane wczytywaniem mniejszej ilości danych niż w przypadku pobierania całych wierszy
* Systemy zorientowane kolumnowo są efektywne, gdy istnieje potrzeba dostarczania nowej wartości danej kolumny dla wszystkich wierszy na raz. Jest to spowodowane tym, że istnieje możliwość znacznie szybszego zapisu danych, po czym tak stworzona kolumna zastępuje starą bez potrzeby poruszania pozostałych kolumn danego rekordu.
* Lepsza wydajność analityczna, spowodowana faktem, że cała kolumna posiada ten sam typ danych i jest umieszczona w jednym miejscu.
* Szybkie łączenia i agregacje – Proste obliczanie wyników funkcji agregujących, co jest krytyczne dla hurtowni danych
* Bral wymagania umieszczania obok siebie kolumn o różnych typach danych – Kolumnowa alokacja danych w wielu jednostkach przetwarzania i przechowywania pozwala na równoległy dostęp i agregację danych zwiększając ogólną wydajność zapytań.
* Możliwość kompresji – Ułatwia znaczną poprawę kompresji, co może powodować dodatkowe zmniejszenie pamięci przy jednoczesnym utrzymaniu wysokiej wydajności.
* Przydatne w dziedzinie hurtowni danych – Hurtownie danych wymagają skomplikowanych przekształceń i ogromnych agregacji danych dużej ilości informacji w nich zawartych.
* Kolumnowe podejście jest szczególnie przydatne dla obciążeń analitycznych, co jest spowodowane tym, że w dużej mierze aplikacje analityczne są ukierunkowane na atrybuty a nie na rekordy. W związku z tym istnieje mała liczba kolumn w tabeli, które mogą być interesujące dla niektórych zapytań. To pozwala na wykorzystanie modelu, w którym następuje odczyt tylko wymaganych kolumn natomiast cała reszta tabeli może być ignorowana.
* Czytanie tylko niezbędnych kolumn charakteryzuje bardziej przyjazny dla pamięci podręcznej wzorzec I/O (wejścia, wyjścia) zarówno dla dyskowych baz danych jak i in-memory. W przypadku dyskowych baz danych wymagane jest znalezienie pierwszego pola w danej kolumnie. Od tego momentu dane przesyłane są w dużych blokach.

Ograniczenia:

* Główną wadą kolumnowych baz danych jest to, że zapewniają mniej optymalne warunki dla importu, eksportu, oraz zgłoszeń typu „bulk”, niż wierszowe bazy danych w przypadku przeprowadzania procesów zorientowanych rekordowo, które są wymaganą częścią każdej hurtowni danych.
* Wady kolumnowych baz są szczególnie widoczne w procesie wstawiania i aktualizacji rekordów. Wymagają one czytania każdej kolumny w rekordzie, określania gdzie nowa wartość należy w tym zestawie, wkładania jej w odpowiednie miejsce, a następnie budowania wymaganych informacji łączących, aby móc je odzyskać i umieścić prawidłowo w oryginalnym formacie rekordu.
* Wolny czas wczytywania – Słaba wydajność w przypadku udziału setek lub więcej gigabajtów danych.
* Przyrost obciążeń może być problematyczny wydajnościowo.
* Ograniczenia struktury – Wykorzystanie techniki ograniczającej możliwość rozrostu systemu w przyszłości.
* Skalowalność – Mogą występować problemy przy prostych, niedużych bazach

## Bazy in-memory

Baza danych in-memory to system zarządzania bazą danych, który opiera się przede wszystkim na głównej pamięci do przechowywania danych komputerowych. Jest to system, który jest przeciwieństwem do systemów on-disk wykorzystujących przestrzeń dyskową. Zazwyczaj bazy danych in-memory są szybsze niż dyskowe ze względu na to, że wewnętrzne algorytmy optymalizacji są prostsze i wykonują mniej instrukcji procesora. Dostęp do danych w pamięci eliminuje czas wyszukiwania podczas badania danych, co zapewnia lepszą i bardziej przewidywalną wydajność niż dostęp do dysku.

W zastosowaniach, w których czas reakcji jest krytyczny, np sieci urządzeń telekomunikacyjnych, bazy danych in-memory są często stosowane. IMDB zyskały bardzo duże znaczenie, zwłaszcza w przestrzeni analitycznej, dzięki taniejącym pamięciom RAM.

Wraz z wprowadzeniem technologii NVDIMM, bazy danych in-memory są w stanie pracować na pełnych obrotach i utrzymywać dane nawet w przypadku awarii zasilania.

W swojej najprostszej postaci, bazy in-memory przechowują na dane w nietrwałej pamięci operacyjnej. Urządzenia te tracą wszystkie przechowywane dane, gdy urządzenie traci zasilanie lub zostanie zrestartowane. W przypadku, IMDB można powiedzieć, że brakuje wsparcia dla ACID (atomowość, spójność, izolacja, trwałość). Jednak systemy IMDB oparte na pamięci nietrwałej mogą i często obsługują trzy właściwości ACID: atomowość, spójność i izolację. Od czasu trwałych pamięci operacyjnych NVDIMM spełniona została także czwarta właściwość ACID.

# GWT

Google Web Toolkit (GWT), jest open sourceowym zbiorem narzędzi, który pozwala programistom na tworzenie i utrzymanie złożonych aplikacji z javascriptowym frontendem w Javie. Oprócz kilku rodzimych bibliotek, wszystko jest w javie, która może być zbudowana na dowolnej platformie z wbudowanym GWT Ant.

Używając GWT deweloperzy mogą rozwijać i debugować aplikacje Ajax w języku Java z wykorzystaniem narzędzi programistycznych Java. Po wdrożeniu aplikacji, cross-kompilator GWT przekłada aplikację Java do samodzielnych plików JavaScript, które ewentualnie są ukrywane i głęboko zoptymalizowane. W razie potrzeby można również osadzić JavaScript bezpośrednio w kodzie Java

GWT nie kręci się tylko wokół programowania interfejsu użytkownika; jest ogólnym zestawem narzędzi do budowania jakiejkolwiek funkcjonalności z wysoką wydajności po stronie klienta funkcjonalności.

Aplikacje GWT można uruchomić w dwóch trybach:

* Tryb rozwoju (tryb dawniej Hosted). Aplikacja jest uruchamiana jako kod bajtowy Javy w Java Virtual Machine (JVM). Tryb ten jest zazwyczaj wykorzystywany do rozwoju, wspierania hot swapping kodu i debugowania.
* Tryb produkcji (dawniej tryb internetowy): aplikacja jest uruchamiana w czystym JavaScript i HTML, kompilowany ze źródeł Javy. Tryb ten jest zazwyczaj używany do wdrożenia.

Główne składniki GWT to:

* GWT Kompilator Java do JavaScript - Tłumaczy Javę do JavaScript.
* GWT Development mode - Umożliwia programistom uruchomienie i wykonywanie aplikacji GWT w trybie rozwoju (aplikacja działa w Javie w JVM bez kompilowania kodu JavaScript).
* JRE emulation library - Implementacje powszechnie stosowanych klas JavaScript w standardowej bibliotece klas Javy (tak jak większość rodzajów pakietów java.lang oraz pakiet java.util).
* GWT Web UI class library - Zestaw niestandardowych interfejsów i klas do tworzenia widgetów.

Cechy:

* Dynamiczne komponenty wielokrotnego użytku UI - programiści mogą używać wstępnie zaprojektowanych klas do wdrożenia czasochłonnych mechanizmów (w przypadku nie wykorzystywania tych komponentów), takich jak przeciąganie elementów lub wyświetlanie zaawansowane struktury drzewa
* Prosty mechanizm RPC
* Zarządzanie historią przeglądarki
* Wsparcie dla debugowania aplikacji Java
* Integracja testów jednostkowych
* Wsparcie w zakresie internacjonalizacji i lokalizacji
* HTML wsparcie dla canvas (z zastrzeżeniem zmian API)
* Deweloperzy mogą mieszać czysty JavaScript w kodzie wraz z Javą przy wykorzystaniu JavaScript Native Interface (JSNI).
* Wsparcie przy użyciu interfejsów API Google w aplikacji GWT (początkowo, wsparcie dla Google Gears)
* Open-source
* Programiści mogą projektować i rozwijać swojej aplikacje w czysto obiektowym podejściu, ponieważ są one tworzone przy użyciu języka Java (zamiast JavaScript). Znane błędy z JavaScript, takie jak literówki i niedopasowania typów, zostają wyłowione w czasie kompilacji.
* Liczba bibliotek jakie są dostępne dla GWT, Google i osób trzecich.

# Hibernate

Hibernate ORM (Hibernate w skrócie) jest biblioteką mapowania obiektowo-relacyjnego dla języka Java, dostarczającą framework dla odwzorowania modelu domeny obiektowej do tradycyjnej relacyjnej bazy danych. Hibernate rozwiązuje problemy obiektowo-relacyjnego niedopasowania, zastępując bezpośrednie bazy danych obiektami wysokiego szczebla.

Hibernate jest darmowym oprogramowaniem, które jest rozpowszechniane na licencji GNU Lesser General Public License.

Podstawową cechą hibernate jest mapowanie klas Javy do tabel bazy danych (a z klas Java na typy danych SQL). Hibernate zapewnia również narzędzia dla kolejek danych i pobierania. Generuje połączenia SQL i zwalnia programistę z ręcznej konwersji obiektu. Aplikacje korzystające z Hibernate są przenośne dla obsługiwanych baz danych SQL z niewielkim narzutem wydajności.

Mapowanie klas Javy do tabel bazy danych odbywa się za pośrednictwem konfiguracji w pliku XML lub za pomocą adnotacji Java. Podczas korzystania z pliku XML, Hibernate może wygenerować kod źródłowy szkieletu dla klas utrwalania. Nie jest to konieczne, kiedy stosowane są adnotacje. Hibernate może skorzystać z pliku XML lub adnotacji do utrzymania schematu bazy danych.

Zapewnione są usługi umożliwiające organizację relacji jeden-do-wielu i wiele-do-wielu pomiędzy klasami. Oprócz tego do zarządzania powiązaniami między obiektami, Hibernate może także zarządzać powiązaniami refleksyjnymi, gdzie obiekt ma relację jeden-do-wielu z innymi instancjami własnego typu.

Hibernate obsługuje mapowanie typów wartości niestandardowych. Sprawia to, że możliwe są następujące scenariusze:

* Nadpisanie domyślnego typu SQL, który Hibernate wybiera podczas mapowania kolumny.
* Mapowanie Java Enum do kolumn, jakby były zwykłymi zmiennymi.
* Mapowanie jednej wartości składowej klasy do wielu kolumnach.

Hibernate zapewnia inspirowane SQL’em język o nazwie Hibernate Query Language (HQL), który pozwala tworzenie zapytań o obiekty danych Hibernate. Zapytania (Kryteria) są jako obiektowa alternatywa do HQL. Zapytanie kryteria służy do modyfikowania obiektów i zapewnia ograniczenie dla obiektów.

# Oracle Database

Oracle Database (powszechnie określane jako Oracle RDBMS lub po prostu jako Oracle) to obiektowo-relacyjny system zarządzania bazą danych produkowany i sprzedawany przez firmę Oracle Corporation.

## Struktury fizyczne i logiczne

System bazy danych Oracle zidentyfikowany przez alfanumeryczny identyfikator systemu lub SID zawiera przynajmniej jedną instancję aplikacji, wraz z przestrzenią danych. Instancja – identyfikowana stale przez numer instancji (lub id aktywacji: SYS.V\_$DATABASE.ACTIVATION#) - zawiera zestaw procesów systemu operacyjnego i struktur pamięci, które współdziałają z przestrzenią danych. Typowe procesy obejmują PMON (monitor procesów) i SMON (monitor systemu). Dokumentacja Oracle może odnosić się do aktywnej instancji bazy danych jako "wspólna sfera pamięci".

Użytkownicy baz danych Oracle odnoszą się do struktury pamięci po stronie serwera jako SGA (System Global Area). SGA zazwyczaj przechowuje informacje pamięci podręcznej takie jak bufory danych, polecenia SQL i dane użytkownika. Oprócz przechowywania, baza składa się z internetowych dzienników powtórzeń, które posiadają historię transakcji. Procesy mogą z kolei archiwizować internetowe dzienniki przerabiając je na logi archiwalne, które zapewniają podstawę (jeśli to konieczne) do odzyskiwania danych.

Oracle DBMS może przechowywać i wykonać procedury i funkcje w sobie.

## Pliki na dysku

Przede wszystkim pliki na dysku stanowią jedną z następujących struktur:

* Dane i indeksy plików - Pliki te zapewniają fizyczne przechowywanie danych, które mogą składać się z danych słownikowych (związanych z systemem tabel), danych użytkownika lub danych indeksowych. Pliki te mogą być zarządzane ręcznie lub przez same Oracle ("Oracle-managed files"). Należy pamiętać, że datafile musi należeć dokładnie do jednej tabeli, natomiast tabela może składać się z wielu plików danych.
* Pliki redo dziennika, zawierające wszystkie zmiany w bazie danych, wykorzystywane w celu odzyskania danych po awarii instancji. Należy pamiętać, że w bazie często zapisuje się te pliki wiele razy, w celu dodatkowego zabezpieczenia w przypadku awarii dysku.
* Pliki undo - Te specjalne pliki danych, które mogą zawierać tylko informacje, pomocne przy odzysku, cofaniu i spójności odczytu.
* Archiwalne pliki dziennika - Te pliki, kopie plików redo dziennika, są zwykle przechowywane w różnych miejscach. Są konieczne (na przykład) przy wprowadzaniu zmian w gotowości bazy lub podczas odzyskiwania po awarii nośników. Jest to możliwe do archiwizacji w wielu miejscach.
* Pliki temp - Te specjalne pliki danych służą wyłącznie do przechowywania danych tymczasowych (stosowane na przykład w przypadku dużych zbiorów lub globalnych tabel tymczasowych)
* Plik sterowania, niezbędny do uruchomienia bazy danych. "Plik binarny, który rejestruje fizyczną strukturę bazy danych, zawiera nazwy i lokalizacje plików dziennika redo, znacznik czasu utworzenia bazy danych, aktualny sekwencyjny numer protokołu, itd."

Na poziomie fizycznym, pliki danych zawierają jeden lub kilka bloków danych, w którym rozmiar bloku może wahać się pomiędzy plikami danych.

Pliki danych mogą zajmować wstępnie przyznane miejsca w systemie plików serwera komputerowego, wykorzystywać bezpośrednio niezajmowane miejsce dysku, lub istnieją w ASM woluminów logicznych.

# Planowane działania i cele

Po zebraniu rozwiązań planowane jest stworzenie systemu zarządzania klientami (CRM) przykładowego operatora telekomunikacyjnego. Na taki system składa się:

* Cienki klient – Część kliencka umożliwiająca dostęp do systemu z poziomu większości przeglądarek.
* Część serwerowa – Część umożliwiająca zarządzanie systemem po stronie serwera.
* Baza danych – Część zawierająca wszelkie dane niezbędne operatorowi telekomunikacyjnemu.

Po wykonaniu następującego systemu planowane jest przeprowadzenie testów sprawdzających mechanizmy wydajności zaimplementowane dla każdego z badanych systemów. Testy te obejmują:

* Sprawdzenie struktur dodawanych przez system
* Sprawdzenie indeksowania w systemie
* Przekształcenie modeli
* Przeładowanie do pamięci
* Zwiększenie obszaru SGA
* Optymalizację raportów.

# Bibliografia

1. *Comparative Study of Row and Column Oriented Database* - Vandana Bhagat, Arpita Gopal
2. *Choosing Right Database System: Row or Column-Store* - Anuradha S. Kanade, Dr. Arpita Gopal
3. *Prisma/db: A parallel main memory relational DBMS. IEEE Transactions on Knowledge and Data Engineering* - P. M. G. Apers, C. A. V. D. Berg, J. Flokstra, P. W. P. J. Grefen, M. L. Kersten, and A. N. Wilschut