

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

LABORATORIUM
Z ROZPROSZONYCH I OBIEKTOWYCH
SYSTEMÓW BAZ DANYCH

**System obsługi internetowego rozkładu jazdy
oparty o rozproszoną bazę danych**

AUTORZY:

Michał Kowalczyk

Indeks: 163368

Piotr Stęplewski

Indeks: 164713

PROWADZĄCY ZAJĘCIA:

Dr inż. Robert Wójcik, W4/I-6

OCENA PRACY:

Wrocław 2011

Spis treści

Spis rysunków	4
1. Wstęp	5
2. Cel i zakres pracy	6
2.1. Cel projektu	6
2.2. Zakres projektu.....	6
3. Replikacja w systemie baz danych MS SQL Server 2008.....	7
3.1. Przegląd typów replikacji	7
3.2. Przegląd programów (pośredników) wykorzystywanych przez MS SQL.....	10
3.3. Zasada działania replikacji transakcyjnej	11
3.4. Konfiguracja serwera poprzez SQL Server Management Studio	13
4. Analiza wymagań.....	19
4.1. Wymagania funkcjonalne	19
4.2. Wymagania niefunkcjonalne	20
4.3. Założenia projektowe	20
5. Projekt systemu	21
5.1. Schemat logiczny i koncepcja działania systemu	21
5.2. Diagramy przypadków użycia	22
5.3. Projekt rozproszonej bazy danych	22
5.3.1. Model koncepcyjny – diagram związków encji	22
5.3.2. Model fizyczny bazy danych w środowisku MS SQL	24
5.4. Projekt aplikacji klienckich	25
5.4.1. Interfejs operatora	25
5.4.2. Interfejs aplikacji klienckiej	26
6. Implementacja elementów systemu	27
6.1. Realizacja konfiguracji bazy danych	27
6.2. Realizacja mechanizmu replikacji.....	27
6.3. Realizacja wybranych funkcjonalności	28
6.3.1. Wyszukiwarka połączeń	28
6.3.2. Aplikacja operatora kolejowego.....	30

6.3.3.	Aplikacja operatora autobusowego	31
6.4.	Testy funkcjonalne	32
6.5.	Testy systemu replikacji	33
6.6.	Wnioski z testów	35
7.	Podsumowanie	36
	Literatura	37

Spis rysunków

Rysunek 1 MS SQL Server 2008 - porównanie edycji	7
Rysunek 2 Replikacja transakcyjna	12
Rysunek 3 Uruchomienie kreatora konfiguracji trybu rozproszonego.....	14
Rysunek 4 Kreator konfiguracji trybu rozproszonego	14
Rysunek 5 Tworzenie nowej publikacji	15
Rysunek 6 Wybór trybu replikacji	15
Rysunek 7 Tworzenie subskrypcji	16
Rysunek 8 Wybór publikatora.....	16
Rysunek 9 Przykładowa konfiguracja MS Management Studio	17
Rysunek 10 Uruchamianie Monitora Replikacji.....	17
Rysunek 11 Replication Monitor.....	18
Rysunek 12 Schemat działania systemu	21
Rysunek 13 Diagram przypadków użycia systemu	22
Rysunek 14 Główne okno aplikacji klienckiej	26
Rysunek 15 Interfejs wyszukiwarki połączeń komunikacyjnych.....	29
Rysunek 16 Aplikacja operatora PKP - widok połączeń	30
Rysunek 17 Test nawigacji po aplikacji klienckiej. Strona z informacjami o projekcie.....	32
Rysunek 18 Test wyszukania połączenia z błędnie wprowadzonymi stacjami	32
Rysunek 19 Test wyszukania połączenia bez podania daty lub godziny odjazdu	33
Rysunek 20 Test pozytywny wyszukiwania połączeń	33
Rysunek 21 Test replikacji przy dodaniu nowego połączenia	34
Rysunek 22 Widok zbiorczej bazy danych po zakończonej replikacji	34

1. Wstęp

System *Internetowego Rozkładu Jazdy* powstał w ramach przedmiotu: rozproszone i obiektowe systemy baz danych. Złożył się on z trzech niezależnych aplikacji webowych oraz rozproszonych baz danych, które obsługiwane są przy pomocy mechanizmu replikacji. Aplikacje zostały wykonane w środowisku .NET przy wykorzystaniu języka ASP.NET, natomiast bazy danych opierały się o rozwiązania bazy MS SQL.

W dalszej części dokumentacji opisane teoretyczne i praktyczne aspekty wykonanej pracy, począwszy od celu i zakresu projektu, przez teoretyczne wprowadzenie do wykorzystanych technologii replikacji baz danych, aż do opisu implementacji aplikacji i jej wyników jej testowania.

2. Cel i zakres pracy

W niniejszym rozdziale zostanie opisany cel stawiany przed projektem oraz zakres pracy, który powinien w jego obrębie zostać wykonany.

2.1. Cel projektu

Temat wykonanego projektu to: „System obsługi internetowego rozkładu jazdy, oparty o rozproszoną bazę danych”. Jego celem było wykorzystanie rozproszonych baz danych do zbudowania aplikacji zestawiającej dane różnego formatu i lokalizacji, a następnie wykorzystującej je do pewnego, określonego celu. Szczególny nacisk położony został na zbadanie mechanizmu replikacji transakcyjnej baz danych.

2.2. Zakres projektu

W zakres projektu wchodziło zaprojektowanie i implementacja aplikacji webowej, która umożliwi dostęp do rozproszonej bazy danych w celu obsługi jednolitego rozkładu jazdy (*Internetowy Rozkład Jazdy*).

Do tego celu zaprojektowane zostały niezależne bazy danych operatorów komunikacyjnych oraz webowe aplikacje operatorskie do ich edycji. W dalszej kolejności powstała zbiorcza baza danych oraz skonfigurowany mechanizm replikacji, który zapewnia automatyczne aktualizowanie baz. Następnym krokiem było stworzenie aplikacji klienckiej, która łączy się ze zbiorczą bazą i wykonuje zapytania o połączenie między danymi miastami w określonej porze, integrując tym samym obie, rozproszone bazy danych.

Ostatnim zadaniem wchodzącym w zakres projektu było przetestowanie całego systemu i wyciągnięcie wniosków dotyczących funkcjonowania tego rodzaju aplikacji.

3. Replikacja w systemie baz danych MS SQL Server 2008

MS SQL Server 2008 udostępnia mechanizmy replikacji (zbiór technologii umożliwiających kopiowanie i rozprowadzanie danych oraz obiektów bazodanowych pomiędzy różnymi bazami danych) na zróżnicowanym poziomie w zależności od rodzaju edycji oprogramowania. Pełną funkcjonalność (wraz z obsługą baz danych Oracle) zapewniają edycje Enterprise oraz Datacenter. Darmowa edycja Express zapewnia jedynie podstawowe mechanizmy replikacji w trybie subskrybenta. Szczegółowe różnice pomiędzy konkretnymi edycjami przedstawia rysunek:

	Datacenter	Enterprise	Standard	Web	Workgroup	Express
Snapshot replication	✓	✓	✓	Subscriber only	✓	Subscriber only
Merge replication	✓	✓	✓	Subscriber only	Restricted	Subscriber only
Transactional replication	✓	✓	✓	Subscriber only	Restricted	Subscriber only
SQL Server change tracking	✓	✓	✓	✓	✓	✓
Publishing data from SQL Server to non SQL Server subscribers	✓	✓	✓			
Publishing data from Oracle to SQL Server	✓	✓				
Peer to Peer replication	✓	✓				

Rysunek 1 MS SQL Server 2008 - porównanie edycji

3.1. Przegląd typów replikacji

Microsoft SQL Server zapewnia trzy typy replikacji:

- replikację transakcyjną (*Transactional replication*)
- replikację scalającą (*Merge replication*)
- replikację migawkową (*Snapshot replication*)

Wybór odpowiedniego typu replikacji jest zależny od wielu czynników, na przykład:

- Fizyczna struktura systemu: ilość oraz lokalizacja komputerów wykorzystywanych przy replikacji, ich rodzaj (czy są to komputery stacjonarne, czy np. urządzenia mobilne)
- Ilość i rodzaj danych do replikacji
- Tryb pracy subskrybenta: pasywny lub z możliwością aktualizacji danych

Ponadto, rodzaj replikacji zależy od zakładanej funkcjonalności, jaką chce osiągnąć projektant systemu. Przykładowe scenariusze zastosowań:

- Poprawa skalowalności i dostępności: utrzymywanie nieprzerwanie aktualizowanych kopii danych umożliwia przeniesienie operacji odczytu danych na wiele serwerów. Nadmiarowość wynikająca z takiego podejścia jest istotna podczas planowanych oraz awaryjnych operacji administracyjnych.
- Hurtownie danych: replikacja wykorzystywana jest do transferu danych pomiędzy serwerami OLTP (*online transaction processing*), a serwerami wspierającymi podejmowanie decyzji.
- **Integracja danych z wielu źródeł:** zbieranie danych z wielu serwisów i prezentowanie ich w ujednoliconej formie.
- Integracja danych z różnych systemów bazodanowych: umożliwienie aplikacjom pobieranie danych z systemów innych niż MS SQL np. z baz danych Oracle.
- Wymiana danych pomiędzy użytkownikami urządzeń mobilnych

Biorąc pod uwagę powyższe czynniki, bazę danych można skonfigurować do pracy z konkretnym typem replikacji:

Replikacja migawkowa:

Ten typ replikacji rozprowadza dane w ustalonych chwilach czasowych bez monitorowania późniejszych zmian tych danych. W trakcie wystąpienia zdarzenia synchronizacji, generowana jest cała migawka i wysłana do subskrybentów. Replikacja migawkowa może być wykorzystywana jako samodzielny mechanizm replikacji, jednak często jest częścią innych typów replikacji, zapewniając wstępną synchronizację danych.

Wykorzystanie samodzielnej replikacji migawkowej jest najbardziej korzystne, gdy system charakteryzuje się następującymi cechami:

- Dane są zmieniane rzadko
- Można zaakceptować, że zreplikowana kopia danych jest nieaktualna przez pewien okres czasu (do następnej migawki)
- Replikacji podlegają niewielkie ilości danych
- Duże zmiany występują w krótkim okresie czasu

Replikacja migawkowa charakteryzuje się mniejszym narzutem pod względem wykorzystania zasobów (po stronie serwera publikującego) niż replikacja transakcyjna, ze względu na brak śledzenia zmian w sposób przyrostowy. Jednak, gdy zbiór danych do replikacji jest bardzo duży, serwer wykorzystujący ten typ replikacji, musi zapewnić odpowiednie zasoby do generacji i zastosowania migawki.

Replikacja transakcyjna:

Replikacja transakcyjna, w typowym przypadku, rozpoczyna się od wykonania migawki początkowej danych oraz obiektów bazodanowych. Po wykonaniu takiej inicjalizacji, wszystkie kolejne zmiany danych oraz modyfikacje struktury bazy zachodzące po stronie Publikatora przekazywane są natychmiastowo (zazwyczaj w czasie rzeczywistym) do Subskrybentów. Zmiany aplikowane Subskrybentom zachodzą w tej samej kolejności w jakiej wystąpiły po stronie Publikatora, co gwarantuje spójność wykonywanych transakcji.

Replikacja transakcyjna wykorzystywana jest głównie w następujących przypadkach:

- Wymagane jest przyrostowe śledzenie zmian w momencie ich występowania
- Wymagane jest małe opóźnienie pomiędzy wystąpieniem zmiany u Publikatora a aktualizacją Subskrybentów
- Aplikacja wymaga dostępu do pośrednich stanów w jakich znajdują się przesyłane dane. Na przykład, podczas kilkukrotnej zmiany danych w wierszu tabeli, replikacja transakcyjna pozwala aplikacji odpowiedzieć (np. poprzez uruchomienie odpowiedniego triggera) na każdą z tych zmian.
- Publikator wykonuje bardzo często operacje wstawiania i aktualizacji danych (*INSERT, UPDATE*)
- Wymagana jest komunikacja z serwerami innych producentów (np. Oracle)

W typowym zastosowaniu, replikacja transakcyjna działa w sposób jednokierunkowy (Subskrybent nie może zmienić danych u Publikatora). MS SQL Server udostępnia jednak specjalny tryb pracy replikacji transakcyjnej, umożliwiający komunikację dwukierunkową.

Replikacja scalająca:

Tak samo jak w przypadku replikacji transakcyjnej, replikacja scalająca zwykle rozpoczyna się od wykonania migawki inicjalizującej. Kolejne zmiany danych lub schematu bazy danych są śledzone za pomocą wyzwalaczy (triggerów). Subskrybent synchronizuje się z Publikatorem poprzez wymianę wszystkich danych, które zmieniły się od czasu ostatniej synchronizacji.

Typowe sytuacje w których stosowana jest replikacja scalająca:

- Wielu Subskrybentów może zaktualizować dane w różnym czasie i rozprowadzić je do Publikatora i innych Subskrybentów.
- Subskrybent otrzymuje dane, wykonuje zmiany w trybie offline, a następnie w późniejszym czasie synchronizuje się z Publikatorem i innymi Subskrybentami.
- Każdy z Subskrybentów wymaga różnej ilości danych
- Mogą wystąpić konflikty i wymagana jest możliwość ich detekcji i naprawy.
- Aplikacja wymaga zmian całej grupy danych, bez znajomości ich stanów pośrednich. Przykładowo, gdy Publikator zaktualizuje jeden wiersz tabeli

kilkukrotnie przed wykonaniem synchronizacji, Subskrybent otrzyma tylko ostateczną wersję danych.

Replikacja skalająca pozwala różnym stronom na autonomiczną pracę, a następnie na późniejsze scalenie aktualizacji w jeden, spójny rezultat. Ponieważ zmiany są wykonywane w kilku „węzłach” systemu, ta sama zmiana musi być zaaplikowana zarówno przez Publikatora jak i u wielu Subskrybentów. Prowadzi to do występowania konfliktów i niejednoznaczności. Replikacja skalająca w MS SQL Server 2008 dostarcza wielu narzędzi do rozwiązywania tego typu problemów.

3.2. Przegląd programów (pośredników) wykorzystywanych przez MS SQL

Mechanizmy replikacji wykorzystują podczas swojej pracy dodatkowe, samodzielne programy pomocnicze, nazywane również „pośrednikami”. Programy te związane są z zadaniami odpowiadającymi za śledzenie zmian i dystrybucję danych. Domyślnie, pośrednicy pracują jako zadania uruchomione przez „SQL Agent Server”. Pośrednicy mogą być uruchomieni również z poziomu linii poleceń lub przez aplikacje korzystające z RMO (Replication Management Objects), na przykład SQL Server Replication Monitor i SQL Server Management Studio.

SQL Server Agent:

SQL Server Agent hostuje i harmonogramuje pracę innych programów pośredniczących. Program ten również steruje i monitoruje operacjami nie związanymi z samym procesem replikacji. Po instalacji serwera, domyślnie SQL Server Agent jest wyłączony i wymaga jawnego określenia trybu jego uruchamiania.

Snapshot Agent:

Program ten wykorzystywany jest zazwyczaj przez wszystkie typy replikacji. Przygotowuje schematy, początkowe struktury danych oraz obiekty bazodanowe, a następnie zapisuje je w postaci migawek. Zarządza również informacjami o synchronizacji wewnątrz bazy danych odpowiedzialnej za dystrybucję.

Log Reader Agent:

Log Reader Agent ma zastosowanie przy replikacji transakcyjnej. Przenosi on (z logów Publikatora) transakcje oznaczone jako rozproszone, do bazy odpowiedzialnej za dystrybucję danych. Każda baza danych publikująca dane z wykorzystaniem replikacji transakcyjnej wykorzystuje własną instancję programu Log Reader Agent.

Distribution Agent:

Pośrednik ten wykorzystywany jest przez replikację migawkową oraz replikację transakcyjną. Aplikuje on początkową migawkę Subskrybentom, oraz przenosi transakcje znajdujące się w bazie dystrybucyjnej. Distribution Agent uruchomiony jest zarówno przez Dystrybutora do wysyłania subskrypcji, jak i przez Subskrybentów do ich pobierania.

Merge Agent:

Merge Agent jest używany przez replikację scalającą. Wykonuje on początkową migawkę do Subskrybenta, oraz przenosi uzgodnione wersje przyrostowych zmian danych. Każda subskrypcja wykorzystująca replikację transakcyjną korzysta z oddzielnej instancji tego programu. Łączy on zarówno Publikatora jak i Subskrybenta oraz aktualizuje obydwie strony replikacji. Merge Agent jest uruchomiony zarówno na Dystrybutorze do wysyłania subskrypcji, jak i na Subskrybencie do ich pobierania. Domyślnie Merge Agent wysyła zmiany od Subskrybenta do Publikanta, a następnie pobiera zmiany w odwrotnym kierunku.

Queue Reader Agent:

Program ten jest wykorzystywany przez replikację transakcyjną z włączoną opcją aktualizacji kolejkowej. Jest on uruchomiony na serwerze Dystrybutora i przenosi zmiany wykonane przez Subskrybenta z powrotem do Publikatora. W przeciwieństwie do programów Distribution Agent oraz Merge Agent, występuje tylko jedna instancja tego pośrednika dla danej rozproszonej bazy danych.

3.3. Zasada działania replikacji transakcyjnej

Replikacja transakcyjna jest zaimplementowana z wykorzystaniem programów pośredniczących: SQL Server Snapshot Agent, Log Reader Agent oraz Distribution Agent.

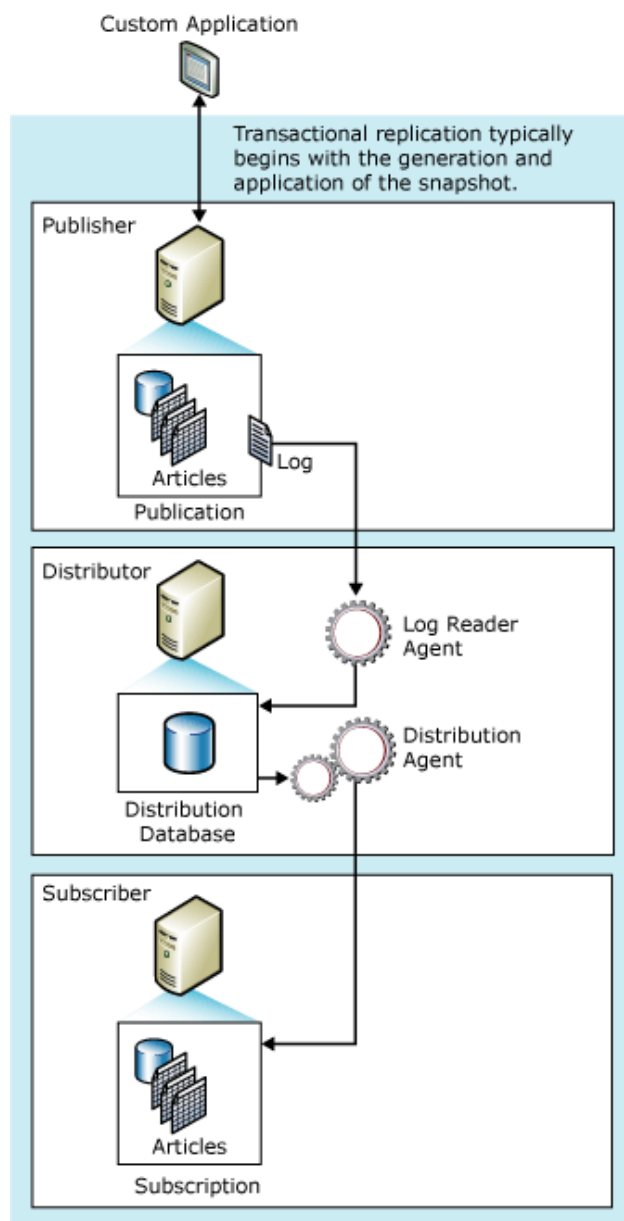
Snapshot Agent przygotowuje pliki z migawkami zawierające schemat oraz dane publikowanych tabel oraz obiektów bazodanowych. Zapisuje te informacje w specjalnym folderze przeznaczonym do przechowywania migawek i zapisuje zadania synchronizacyjne w bazie danych Dystrybutora.

Log Read Agent monitoruje logi transakcji w każdej bazie danych skonfigurowanej do wykonywania replikacji transakcyjnej. Kopiuje transakcje zaznaczone jako rozproszone z logów transakcyjnych do bazy dystrybucyjnej w formie kolejki typu „store-and-forward”. Następnie, Distribution Agent aplikuje początkową migawkę, oraz przenosi transakcje znajdujące się w bazie dystrybucyjnej do Subskrybentów.

Przyrostowe zmiany wykonane po stronie Publikatora przepływają do Subskrybenta zgodnie z harmonogramem Pośrednika Dystrybucji (Distribution Agent). Może on być uruchomiony w trybie pracy ciągłej (aby uzyskać minimalne opóźnienia), lub w trybie okresowych synchronizacji. Ponieważ wszystkie zmiany danych występują

jedynie po stronie Publikatora (w przypadku replikacji transakcyjnej jednokierunkowej) – nie występują konflikty. Ostatecznie, wszyscy Subskrybenci otrzymują te same wartości, które znajdują się u Publikatora.

Podstawowe komponenty replikacji transakcyjnej:



Rysunek 2 Replikacja transakcyjna

Inicjalizacja:

Zanim nowy Subskrybent replikacji transakcyjnej będzie mógł otrzymać przyrostowe zmiany od Publikatora, musi zostać utworzona w nim struktura bazy danych, odpowiadająca subskrybowanym obiektom. Wykonywane jest to automatycznie, zazwyczaj podczas początkowej migawki wykonanej przez pośrednika Snapshot Agent oraz rozprowadzonej i zastosowanej za pomocą pośrednika

Distribution Agent. Początkowy zbiór danych i struktura bazy może zostać również uzyskana w inny sposób, na przykład z kopii zapasowej lub przez oprogramowanie SQL Server Integration Services.

Aplikowanie początkowej migawki obejmuje jedynie te subskrypcje, które są w stanie oczekiwania na migawkę inicjalizującą. Pozostałe subskrypcje (którym dostarczono migawki) są pomijane w procesie inicjalizacji.

W przeciwieństwie do replikacji migawkowej, w której podczas generowania migawek zakładane są blokady na wszystkie publikowane tabele, w replikacji transakcyjnej domyślnym systemem jest współbieżne przetwarzanie migawek. Oznacza to, że użytkownik może wykonywać pracę nieprzerwanie podczas tworzenia migawek inicjalizujących.

Modyfikacja danych:

Za modyfikację danych odpowiedzialny jest Log Reader Agent. Uruchomiony jest on na serwerze dystrybucyjnym i zazwyczaj pracuje w trybie pracy ciągłej. Pośrednik ten, najpierw przegląda (odczytuje) logi transakcji publikatora, oraz identyfikuje wszystkie operacje typu INSERT, UPDATE, DELETE lub inne modyfikacje danych wykonane w transakcjach zaznaczonych jako rozproszone. Następnie, pośrednik ten kopiuje każdy pakiet poleceń do Dystrybutora. Log Read Agent wykorzystuje wewnętrzną procedurę składowaną *sp_replcmd*, aby przejść do kolejnego zbioru poleceń oznaczonych jako rozproszone. Baza dystrybutora wykorzystywana jest następnie jako kolejka typu „store-and-forward” z której zmiany wysyłane są do Subskrybentów. Tylko zatwierdzone transakcje są wysyłane do bazy dystrybucyjnej.

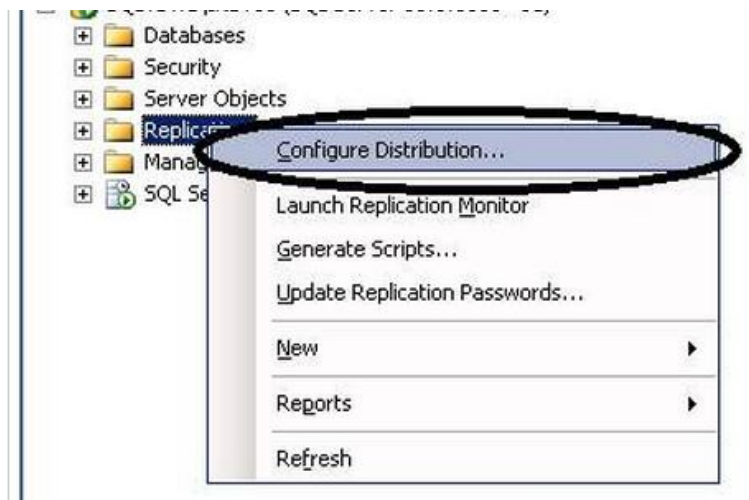
Po poprawnym zapisaniu całego pakietu transakcji rozproszonych do bazy dystrybucyjnej, są one zatwierdzane (commit). Po zatwierdzeniu każdego pakietu poleceń przez Dystrybutora, Log Reader Agent wywołuje procedurę *sp_repldone* aby zaznaczyć w którym miejscu replikacja została zakończona. Ostatecznie, pośrednik zaznacza te wiersze w logu transakcji, które zostały pomyślnie przetworzone. Wiersze wciąż nie przetworzone nie zostaną oczyszczone.

Komendy transakcji są przechowywane w bazie dystrybutora do momentu, aż zostaną pomyślnie rozproszone do wszystkich subskrybentów, lub gdy minie maksymalny ustalony czas wykonania operacji. Subskrybenci otrzymują transakcje w tej samej kolejności, w jakiej były zastosowane po stronie Publikatora.

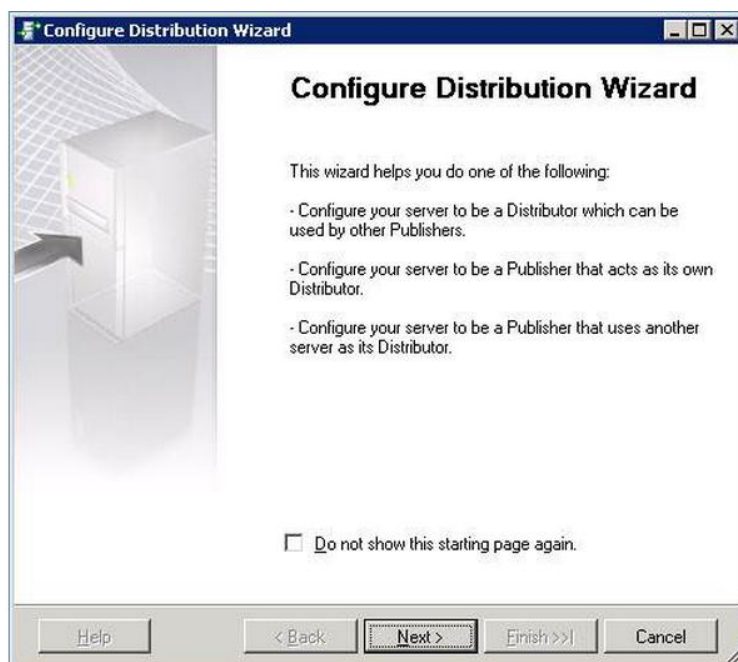
3.4. Konfiguracja serwera poprzez SQL Server Management Studio

SQL Management Studio to zintegrowane środowisko umożliwiające zarządzanie, administrowanie oraz rozwijanie wszystkich komponentów serwera MS SQL. SQL Management Studio udostępnia te możliwości za pomocą graficznych narzędzi oraz licznych skryptów.

Aby skonfigurować serwer do pracy w trybie rozproszonym, należy wykonać kilka podstawowych operacji. Pierwszą czynnością, jest uruchomienie kreatora konfiguracji mechanizmów replikacji:



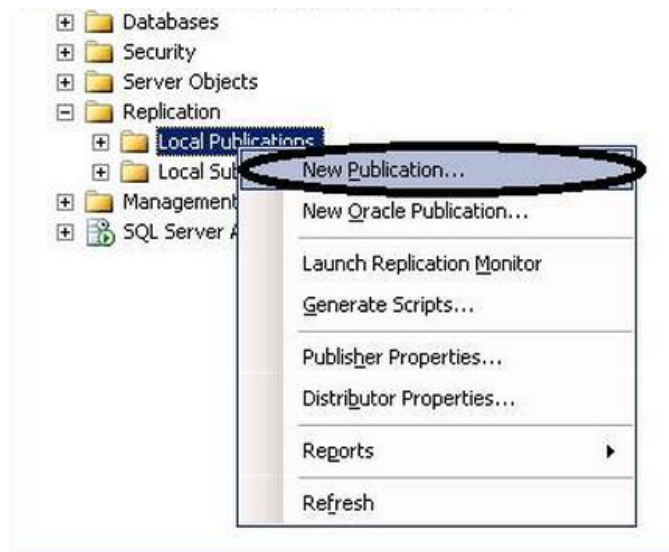
Rysunek 3 Uruchomienie kreatora konfiguracji trybu rozproszonego



Rysunek 4 Kreator konfiguracji trybu rozproszonego

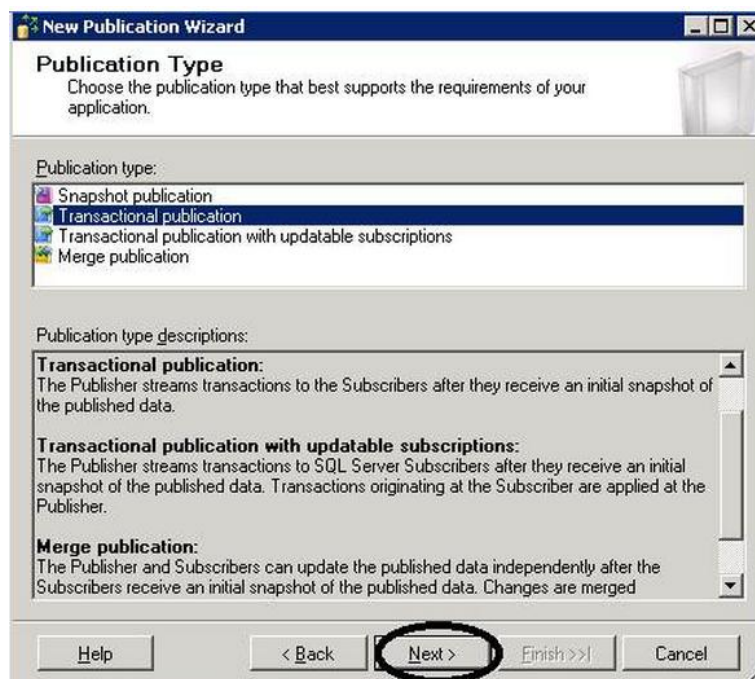
W trakcie pracy kreatora, możemy wskazać odpowiedni folder do przechowywania migawek, wskazać serwer Dystrybutora (w szczególności może być to ten sam serwer na którym działa Publikator) oraz wybrać tryb uruchamiania pośrednia SQL Server Agent.

Następnym krokiem, jest utworzenie nowej publikacji:



Rysunek 5 Tworzenie nowej publikacji

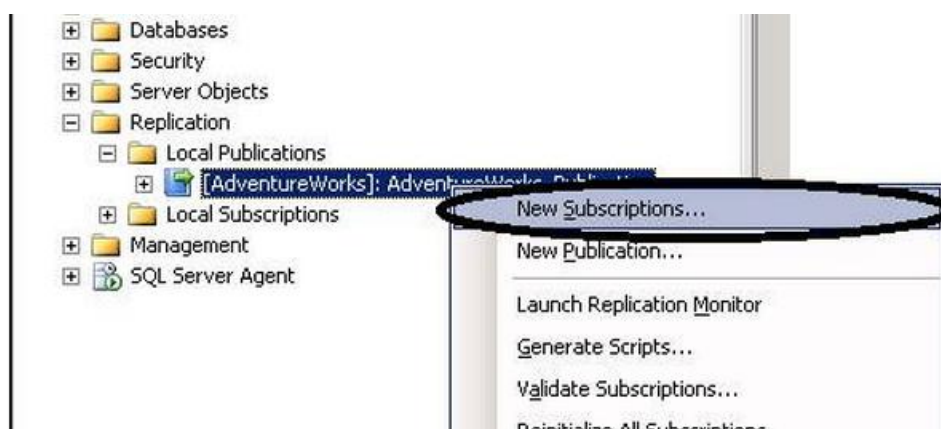
Po kliknięciu w zaznaczoną opcję, uruchomi się kolejny kreator. W trakcie jego działania, możemy m.in. określić rodzaj replikacji:



Rysunek 6 Wybór trybu replikacji

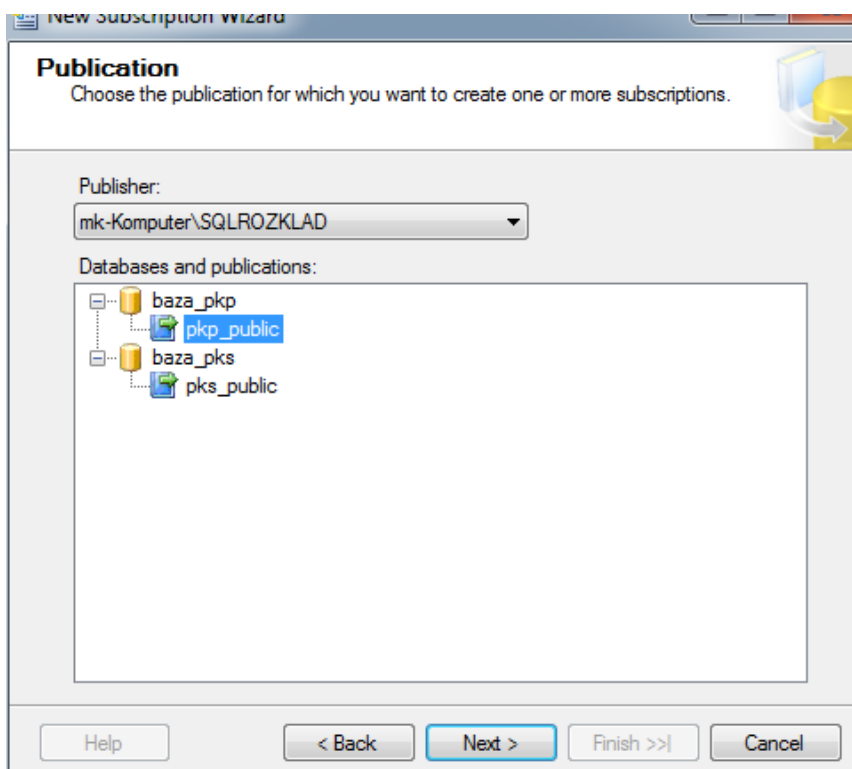
Możemy również wskazać, które dane chcemy publikować (wraz z definicją odpowiednich filtrów), określić tryb pracy pośrednika migawek, oraz wprowadzić dane potrzebne do autoryzacji użytkowników.

Po utworzeniu publikacji, można zacząć konfigurować subskrypcje, które będą z nich korzystały:



Rysunek 7 Tworzenie subskrypcji

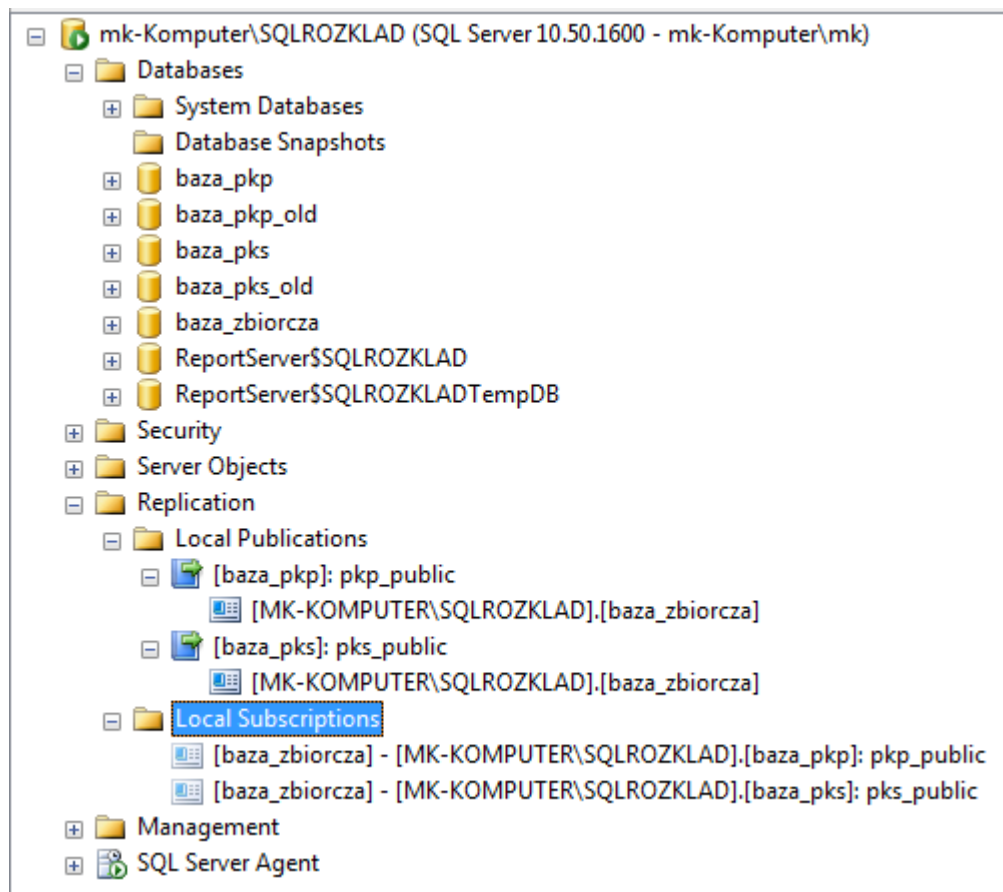
W tym celu ponownie korzystamy z kreatora. Możemy określić bazę danych z której będziemy subskrybować:



Rysunek 8 Wybór publikatora

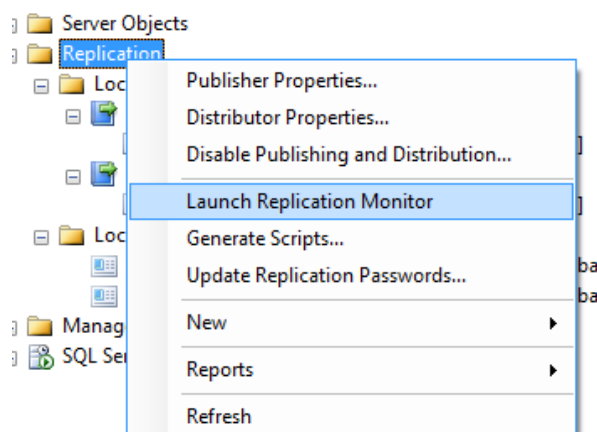
Kreator tworzenia subskrypcji umożliwia również podanie ścieżek dostępu do folderów przeznaczonych na logi, ustawienie odpowiedniego trybu pracy Dystrybutora oraz wybranie metod autoryzacji.

Po przykładowej konfiguracji, eksplorator obiektów MS Management Studio wygląda następująco:



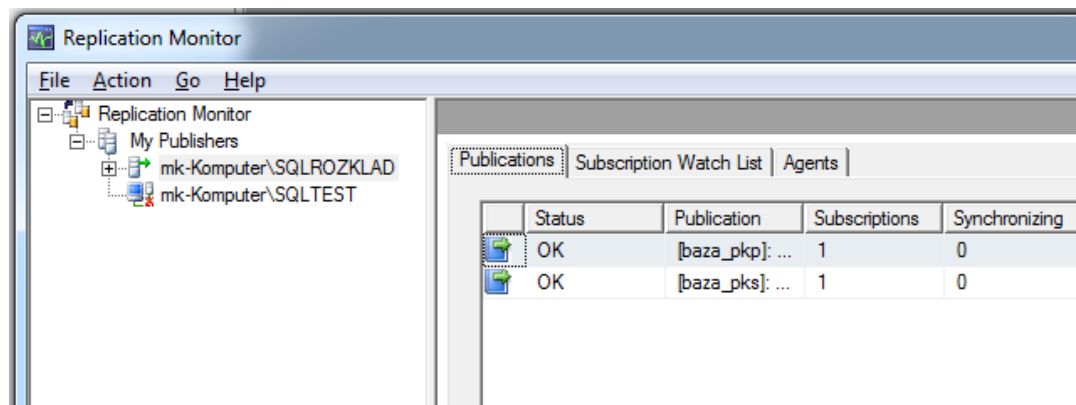
Rysunek 9 Przykładowa konfiguracja MS Management Studio

Ważnym narzędziem udostępnianym wraz z MS Management Studio jest Monitor Replikacji (Replication Monitor):



Rysunek 10 Uruchamianie Monitora Replikacji

Umożliwia on podgląd stanu działania wszystkich replikacji uruchomionych w systemie. Zawiera również informacje o stanie uruchomionych pośredników oraz przeglądarkę logów:



Rysunek 11 Replication Monitor

4. Analiza wymagań

W kolejnych punktach zawarte zostały wymagania stawiane przed projektem, zarówno funkcjonalno, jak i нефункционалне oraz założenia projektowe.

4.1. Wymagania funkcjonalne

Obsługa replikacji baz danych operatorów

- Automatyczne wypełnienie bazy zbiorczej po skonfigurowaniu replikacji
- Aktualizacja bazy zbiorczej po dokonanej zmianie przez aplikację operatorską

Wyszukiwanie połączeń komunikacyjnych

- Wybór stacji początkowej
- Wybór stacji końcowej
- Wybór daty odjazdu
- Wybór godziny odjazdu
- Wyświetlenie wyników wyszukiwania

Aplikacje operatorskie

- Przeglądanie aktualnej zawartości baz danych
- Dodawanie nowych wpisów w tabelach baz danych (połączenia, stacje, legendy, nazwy operatorów)
- Usuwanie istniejących wpisów
- Modyfikacja instniejących wpisów

4.2. Wymagania niefunkcjonalne

Bezpieczeństwo

- Fizyczne zabezpieczenie aplikacji i baz danych na komputerach
- Bezpieczne kanały komunikacji
- Odporność systemu na zewnętrzne zagrożenia, np. ataki DoS

Dostępność

- System musi być dostępny przez całą dobę przez 7 dni w tygodniu
- Dostęp równocześnie do 100 użytkowników
- Maksymalny czas niedostępności w czasie miesiąca = 3h

Wydajność

- Obsługa do 1000 transakcji bazy danych
- Przetwarzanie do 50 000 pozycji

Skalowalność

- System powinien być dobrze przystosowany do swobodnej rozbudowy jego funkcjonalności
- Wydajne równoważenie obciążenia serwerów i baz danych

Platforma

- Oprogramowanie: system Windows, Microsoft Data Access Components (MDAC) 2.6, IIS 5.0
- Sprzęt: procesor Pentium 90MHz lub szybszy, 96MB pamięci RAM

4.3. Założenia projektowe

Projekt do komunikacji z klientem będzie wykorzystywał model klient-serwer. Wszelkie zarządzanie danymi odbywa się po stronie serwerów baz danych, osobny serwer integrujący pobiera na bieżąco (przy każdej edycji baz operatorskich) potrzebne informacje dla aplikacji klienckiej. Dostęp z poziomu użytkownika do baz danych będzie całkowicie transparentny.

System Internetowego Rozkładu Jazdy został wykonany w technologii ASP.NET w środowisku MS Visual Studio przy użyciu baz danych MS SQL. Aplikacje operatorów i aplikacja kliencka posiadają interfejsy webowe, które dostępne są z poziomu przeglądarki internetowej.

5. Projekt systemu

Niniejszy rozdział zawiera opis projektu systemu Internetowego Rozkładu Jazdy, zarówno aplikacji i schematy baz danych, jak i koncepcję działania całego systemu.

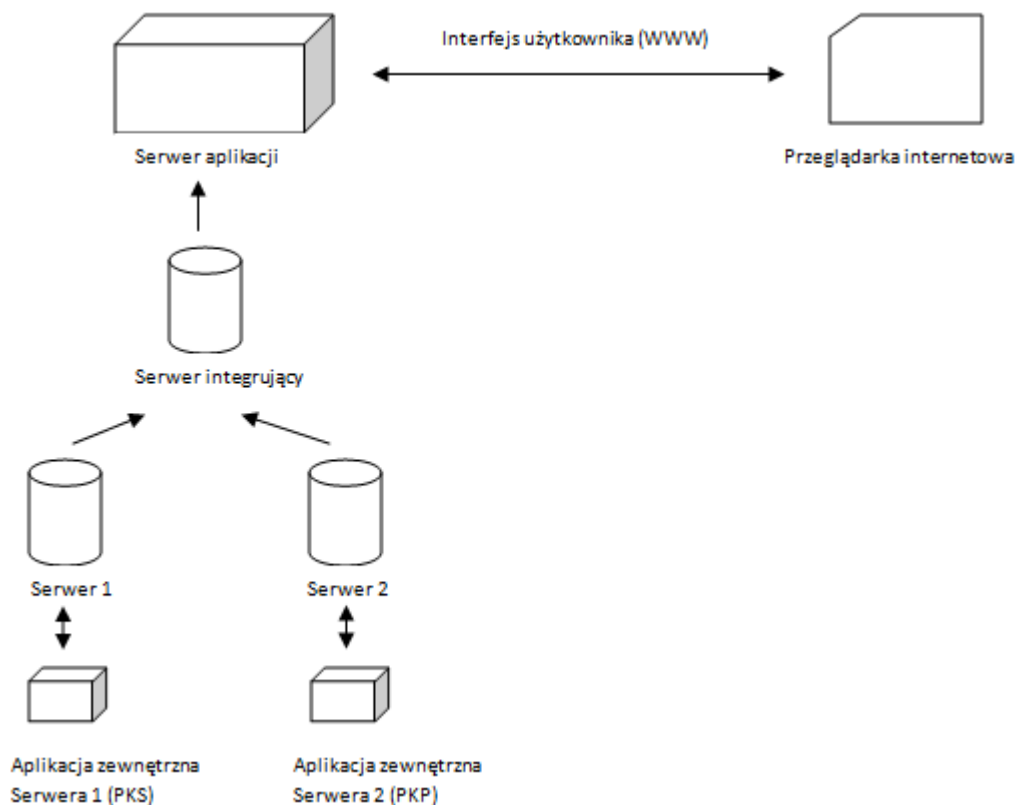
5.1. Schemat logiczny i koncepcja działania systemu

Koncepcja działania systemu opiera się na działaniu dwóch niezależnych, rozproszonych baz danych operatorów kolejowych i autobusowych. Każdy z tych operatorów posiada także niezależne aplikacje webowe, dzięki którym mogą swobodnie edytować dane dotyczące swoich połączeń.

Oprócz baz danych operatorów, istnieje jeszcze jedna – zbiorcza baza danych, która w procesie replikacji transakcyjnej zostaje zaktualizowana tak, że zawiera aktualne informacje o połączeniach wszystkich operatorów. Proces replikacji jest uruchamiany przy pierwszej konfiguracji zbiorczej bazy danych, a następnie przy każdej aktualizacji bazy danych operatorów.

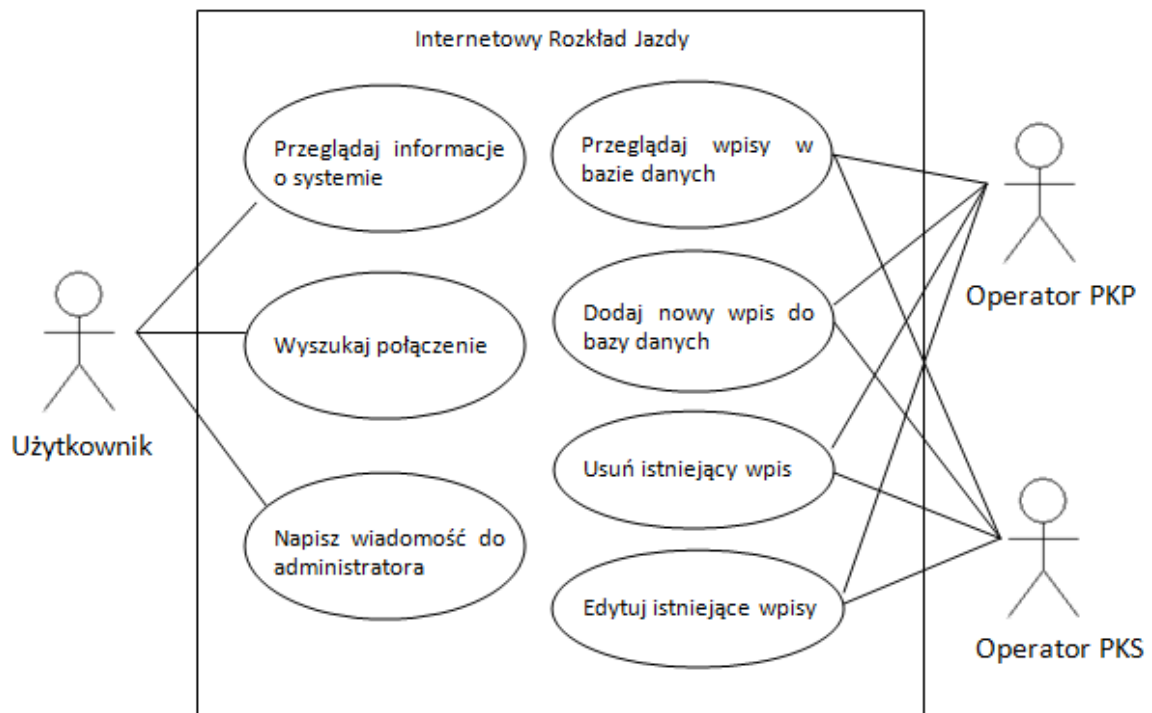
Ostatnim elementem Internetowego Rozkładu Jazdy jest webowa aplikacja kliencka, która ma dostęp do zbiorczej bazy danych. Na jej podstawie system oferuje użytkownikom usługę wyszukiwania połączeń komunikacyjnych integrującą wyniki wyszukiwania wszystkich przewoźników na jednej stronie.

Schemat działania systemu pokazany jest poniżej:



Rysunek 12 Schemat działania systemu

5.2. Diagramy przypadków użycia



Rysunek 13 Diagram przypadków użycia systemu

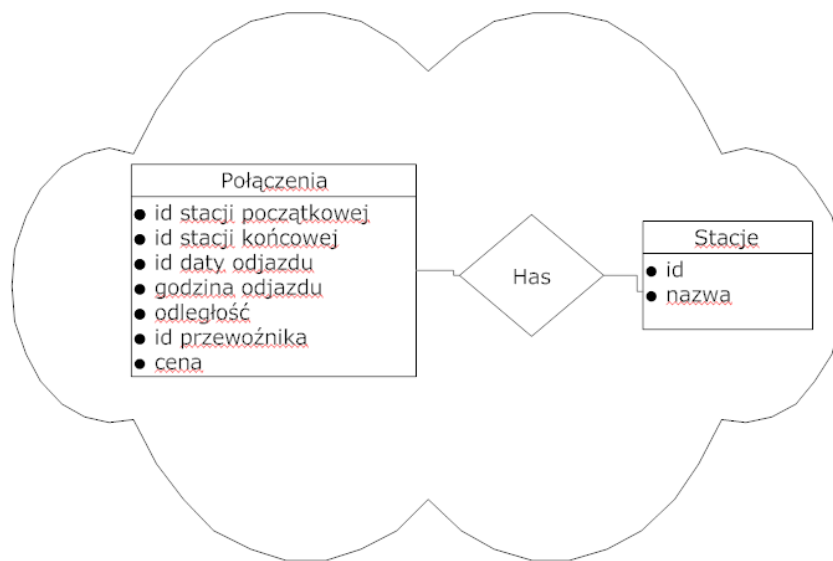
5.3. Projekt rozproszonej bazy danych

W dalszych podpunktach znajduje się opis modeli bazy danych wykorzystanych w całym systemie.

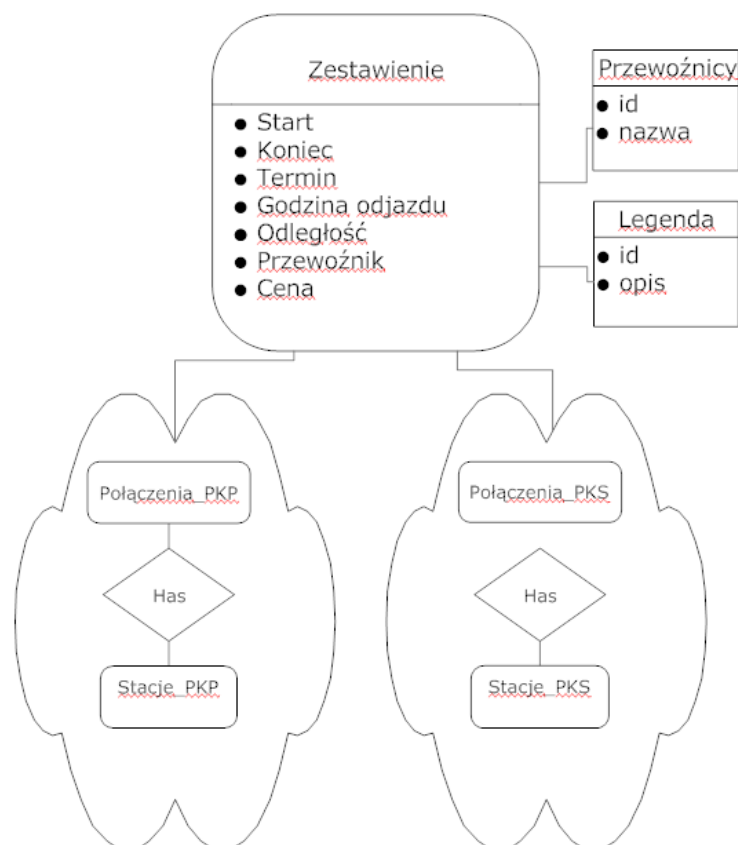
5.3.1. Model konceptualny – diagram związków encji

System Internetowego Rozkładu Jazdy zakłada następującą strukturę baz danych:

- Aplikacje klienckie (przewoźników) posiadają jednakowe struktury, składające się z tabeli z nazwami stacji oraz tabeli z definicją połączenia
- Identyfikatory poszczególnych miast muszą być jednakowe
- Baza zbiorcza posiada tabele z identyfikatorami poszczególnych przewoźników
- Każdy przewoźnik musi identyfikować się z góry ustalonym identyfikatorem



Rysunek 14 ERD baz przewoźników



Rysunek 15 ERD bazy zbiorczej

5.3.2. Model fizyczny bazy danych w środowisku MS SQL

Model fizyczny bazy danych w środowisku MS SQL:

Bazy klienckie:

Tabela 1. Model tabeli z nazwami stacji

Stacje		
Nazwa pola	Typ	Zezwól puste
Id	Int	nie
Nazwa	char(50)	nie

Tabela2. Model tabeli z połączeniami

Połączenia		
Nazwa pola	Typ	Zezwól puste
Id	Int	nie
Id_stacji_start	Int	nie
Id_stacji_koniec	Int	nie
Id_data_odjazdu	Int	nie
Godzina_odjazdu	Time(7)	nie
Odległość	Int	Tak
Id_przewoźnik	Int	nie
Cena	Money	tak

Baza zbiorcza:

Tabela 3. Model tabeli z nazwami przewoźników

Przewoźnicy		
Nazwa pola	Typ	Zezwól puste
Id	Int	nie
Nazwa	char(50)	nie

Tabela 4. Model tabeli z legendą

Data_odjazdu		
Nazwa pola	Typ	Zezwól puste
Id	Int	nie
Opis	Char(100)	nie

Tabele z danymi od różnych przewoźników są scalane w postaci unii:

Tabela 5. Model unii z zestawieniem połączeń

Zestawienie		
Nazwa pola	Typ	Zezwól puste
Start	Char(50)	nie
Koniec	Char(50)	nie
Termin	Char(50)	nie
Godzina odjazdu	Time(7)	nie
Odległość	Int	nie
Przewoźnik	Char(50)	nie
Cena	money	nie

5.4. Projekt aplikacji klienckich

W kolejnych podpunktach opisany został projekt aplikacji klienckich, operatorów przewoźników oraz webowej aplikacji użytkownika.

5.4.1. Interfejs operatora

Cały system Internetowego Rozkładu Jazdy oprócz głównej aplikacji klienckiej, składa również się z innych serwisów, które dostarczane są operatorom komunikacyjnym, takim jak PKP czy przewoźnikom autobusowym PKS. Dzięki nim poszczególni operatorzy mają możliwość dowolnej edycji następujących danych:

- połączenia między poszczególnymi stacjami
- stacje
- daty odjazdu (legenda)
- nazwy przewoźników

Istotą działania systemu jest tutaj mechanizm replikacji – po każdej edycji przez operatora swojej bazy danych, baza ta jest aktualizowana w bazie zbiorczej, która z kolei służy dalej użytkownikowi aplikacji klienckiej. W ten sposób za pomocą jednej zmiany, nowe informacje są aktualizowane w wielu miejscach jednocześnie

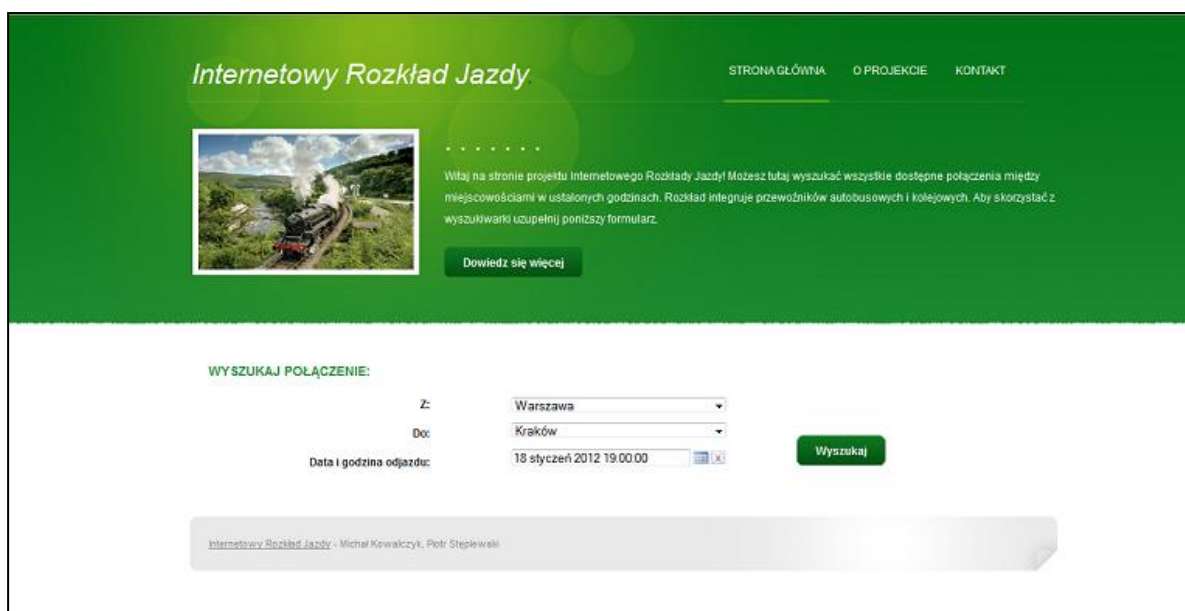
Administrator komunikuje się z aplikacją operatorską za pomocą interfejsu webowego. Do dyspozycji ma tabele zawierające autentyczne dane, które w wygodny i czytelny sposób mogą dowolnie aktualizowane, po czym zmiany są natychmiast zapisywane w bazie danych.

5.4.2. Interfejs aplikacji klienckiej

Aplikacja kliencka jest bazą systemu Internetowego Rozkładu Jazdy. Jest to aplikacja webowa, której główną funkcjonalnością jest udostępnianie użytkownikom informacji o aktualnych rozkładach jazdy wielu operatorów. Odbywa się to za pomocą wyszukiwarki połączeń. Źródłem danych dla aplikacji klienckiej jest zbiorcza baza danych, która powstaje w wyniku replikacji oryginalnych baz przewoźników.

Oprócz wyszukiwarki połączeń komunikacyjnych, opisanej w dalszej części dokumentacji, aplikacja oferuje również stronę z informacjami o całym systemie oraz formularz umożliwiający bezpośredni kontakt klienta z administratorem systemu.

Jeden z widoków aplikacji klienckiej wygląda następująco:



Rysunek 16 Główne okno aplikacji klienckiej

6. Implementacja elementów systemu

W kolejnych podrozdziałach opisane zostały sposoby implementacji konkretnych rozwiązań. Dotyczy to zarówno mechanizmu replikacji baz danych, jak i aplikacji webowych.

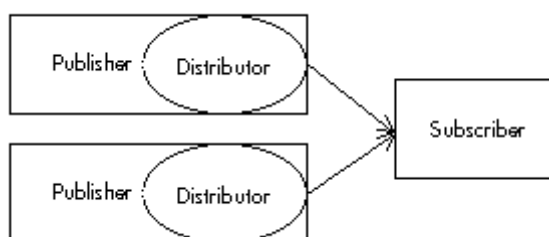
6.1. Realizacja konfiguracji bazy danych

System Internetowego Rozkładu jazdy został zrealizowany fizycznie na jednej instancji serwera MS SQL 2008. W skład rozproszonego systemu wchodzi następujące bazy danych:

- Baza_PKP – zawiera informacje dotyczące połączeń kolejowych
- Baza_PKS – zawiera informacje dotyczące połączeń autobusowych
- Baza_zbiorcza – zawiera informacje zebrane z powyższych baz przewoźników różnego typu

6.2. Realizacja mechanizmu replikacji

System działa w oparciu o mechanizm **replikacji transakcyjnej**. Struktura systemu pod względem mechanizmów replikacji przedstawia się następująco:

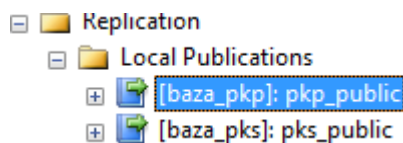


Rysunek 17 Topologia replikacji

Jest to więc system z centralnym subskrybentem, który nie może aktualizować danych publikatorów.

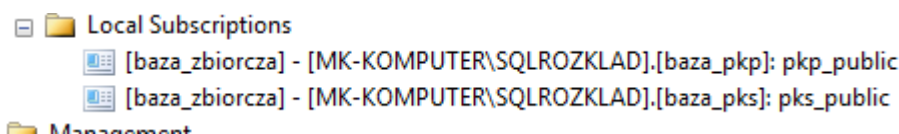
W aplikacji MS SQL 2008 serwer dystrybutora fizycznie jest uruchomiony na tej samej instancji, na której pracują publikatory.

Publikatory skonfigurowane są do udostępniania dwóch tabel: „stacje” oraz „połączenia”. Po skonfigurowaniu publikatorów, pojawiają się one na liście publikacji serwera MS SQL 2008:



Rysunek 18 Publikatory

Do obydwu publikacji przypisany jest jeden subskrybent:



Rysunek 19 Subskrybent

Subskrybent wraz z pierwszą migawką tworzy automatycznie wszystkie wymagane tabele. Następnie za pomocą uruchomionych w tle programów pośredniczących (opisanych w pkt 3.2) realizowane są wszystkie funkcje związane z replikacją transakcyjną (opisaną w pkt 3.3).

6.3. Realizacja wybranych funkcjonalności

Poniżej znajduje się opis implementacji wybranych funkcjonalności aplikacji klienckich, wraz z fragmentami kodu źródłowego.

6.3.1. Wyszukiwarka połączeń

Zaimplementowana w aplikacji klienckiej wyszukiwarka połączeń jest jedną z jej podstawowych funkcjonalności, która umożliwia użytkownikowi faktyczne odnalezienie interesującego połączenia komunikacyjnego na podstawie podanych informacji:

- stacja początkowa
- stacja docelowa
- data odjazdu
- godzina odjazdu

Dane te podaje użytkownik przy pomocy graficznego interfejsu – w kontrolkach *combobox* dostępne są nazwy miast ze stacjami początkowymi i końcowymi (pobierane z baz danych operatorów PKS i PKP), data i godzina odjazdu dla danego połączenia wybierana jest natomiast za pomocą specjalnego panelu, który przypomina tradycyjny kalendarz. Tym samym użytkownik może wybrać wyszukać interesujące go połączenie w prosty i czytelny sposób, pokazany na rysunku poniżej:

WYSZUKAJ POŁĄCZENIE:

Z:

Do:

Data i godzina odjazdu:

[Internetowy Rozkład Jazdy - Michał Kowalczyk, Piotr Stęplewski](#)

styczeń 2012

Pn	Wt	Śr	Cz	Pt	So	N
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

5 30 pm

9 styczeń 2012 17:30:00

OK Cancel

Rysunek 20 Interfejs wyszukiwarki połączeń komunikacyjnych

W dalszej kolejności wybrane informacje są przesyłane jako parametry do strony z wynikami wyszukiwania. Dodatkowo wybrana data oraz godzina zapisywane są w globalnej zmiennej, aby później móc korzystać z ich różnych postaci i form zapisu. Przesyłane parametry w praktyce zapamiętywane są postaci ciągu widocznego na pasku adresu (*QueryString*). Przykładowe zapytanie może zatem mieć ostateczną postać:

```
http://localhost:1127/ShowResults.aspx?Start=Warszawa&Stop=Kraków&Date=2012-01-09&DateNatural=9-20stycznia-2012
```

Po dobraniu parametrów wyszukiwania, aplikacja przekierowuje przeglądarkę do strony z wynikami zapytania, gdzie następuje filtrowanie rekordów bazy danych i ich ostateczne wyświetlenie. Wyszukiwarka została dodatkowo wyposażona w zabezpieczenia przed niepoprawnie wprowadzonymi danymi, m.in. przed brakiem wyboru daty lub godziny odjazdu. Fragment kodu obsługi wyszukiwarki:

```
if (DropDownListStart.SelectedValue == DropDownListStop.SelectedValue)
{
    ShowPopUpMsg("Stacja początkowa nie może być taka sama jak stacja końcowa.");
}
else if (DateTimePicker.Text == "")
{
    ShowPopUpMsg("Nie wybrano daty lub godziny odjazdu.");
}
else
{
    Global.Data = DateTimePicker.DateTime;
    Response.Redirect("ShowResults.aspx?Start=" + DropDownListStart.SelectedValue +
        "&Stop=" + DropDownListStop.SelectedValue +
        "&Date=" + DateTimePicker.DateTime.ToShortDateString() +
        "&DateNatural=" + DateTimePicker.DateTime.ToString());
}
```

6.3.2. Aplikacja operatora kolejowego

Podstawową funkcjonalnością aplikacji operatora kolejowego jest wyświetlanie oraz edycja zawartości bazy danych tego przewoźnika. Sama aplikacja komunikuje się z operatorem za pomocą interfejsu webowego, dostępnego z poziomu przeglądarki internetowej.

Program operatorski połączony jest bezpośrednio z bazą danych PKP i może wykonywać na niej różnego rodzaju operacje edycji informacji. Składa się on z następujących zakładek:

- połączenia
- stacje
- data odjazdu
- przewoźnicy

W każdej zakładce wyświetlona jest tabela, w której pokazane są aktualne rekordy (komponent *GridView*). Przykładowa strona z połączeniami operatora PKP wygląda następująco:

APLIKACJA OPERATORA PKP								
Strona główna Połączenia Stacje Data odjazdu Przewoźnicy								
POŁĄCZENIA								
	id	id_stacja_start	id_stacja_koniec	id_data_odjazdu	godzina_odjazdu	id_przewoźnik	id_odleglosc	cena
Edit Delete	1	1	2	1	13:00:00	2	120	21,0000
Edit Delete	2	2	3	2	09:30:00	1	80	17,0000
Edit Delete	3	2	4	1	17:17:00	1	220	30,0000
Edit Delete	4	3	5	3	13:30:00	2	13	8,0000
Edit Delete	5	1	2	8	15:30:00	1	120	25,0000
								<input type="button" value="Dodaj rekord"/>

Rysunek 21 Aplikacja operatora PKP - widok połączeń

Połączenie z bazą danych następuje za pomocą polecenia *ConnectionString*, w której określone są niezbędne parametry. Po lewej stronie tabeli, przy każdym rekordzie dostępne są polecenia edycji lub usunięcia całego wpisu. Za to działanie odpowiadają komendy, które zdefiniowane są następująco:

```

<asp:SqlDataSource ID="SqlDataSource5" runat="server"
    ConnectionString="<%"$ ConnectionStrings:baza_pkpConnectionString %>"
    DeleteCommand="DELETE FROM [T_Pkp_Polaczenie] WHERE [id] = @id"
    InsertCommand="INSERT INTO [T_Pkp_Polaczenie] ([id], [id_stacja_start],
[id_stacja_koniec], [id_data_odjazdu], [godzina_odjazdu], [id_przewoznik],
[id_odleglosc], [cena]) VALUES (@id, @id_stacja_start, @id_stacja_koniec,
@id_data_odjazdu, @godzina_odjazdu, @id_przewoznik, @id_odleglosc, @cena)"
    SelectCommand="SELECT [id], [id_stacja_start], [id_stacja_koniec],
[id_data_odjazdu], [godzina_odjazdu], [id_przewoznik], [id_odleglosc], [cena] FROM
[T_Pkp_Polaczenie]"
    UpdateCommand="UPDATE [T_Pkp_Polaczenie] SET [id_stacja_start] =
@id_stacja_start, [id_stacja_koniec] = @id_stacja_koniec, [id_data_odjazdu] =
@id_data_odjazdu, [godzina_odjazdu] = @godzina_odjazdu, [id_przewoznik] =
@id_przewoznik, [id_odleglosc] = @id_odleglosc, [cena] = @cena WHERE [id] = @id">
. . .
</asp:SqlDataSource>

```

Pod tabelą znajduje się formularz, który daje możliwość po zatwierdzeniu przyciskiem szybko dodać nowy rekord do bazy danych i automatycznie wyświetlić go na stronie. Implementacja zapisu nowych wartości do bazy danych podana jest poniżej:

```

SqlDataSource5.InsertParameters["id"].DefaultValue = txtPolId.Text;
SqlDataSource5.InsertParameters["id_stacja_start"].DefaultValue=txtPolStart.Text;
SqlDataSource5.InsertParameters["id_stacja_koniec"].DefaultValue=txtPolKoniec.Text;
SqlDataSource5.InsertParameters["id_data_odjazdu"].DefaultValue=txtPolData.Text;
SqlDataSource5.InsertParameters["godzina_odjazdu"].DefaultValue=txtPolGodzina.Text;
SqlDataSource5.InsertParameters["id_przewoznik"].DefaultValue=txtPolPrzewoznik.Text;
SqlDataSource5.InsertParameters["id_odleglosc"].DefaultValue=txtPolOdleglosc.Text;
SqlDataSource5.InsertParameters["cena"].DefaultValue=txtPolCena.Text;
SqlDataSource5.Insert();

```

6.3.3. Aplikacja operatora autobusowego

Na potrzeby projektu, aplikacja operatora PKS jest w swoim wyglądzie analogiczna do aplikacji operatora PKP. Podobną ma również funkcję – edycję istniejących i zapis nowych rekordów do bazy danych przewoźnika. Przykładowo, dodanie nowego przystanku do listy następuję według kodu:

```

<asp:SqlDataSource ID="SqlDataSourceStacje" runat="server"
    ConnectionString="<%"$ ConnectionStrings:baza_pksConnectionString %>"
    DeleteCommand="DELETE FROM [T_Pks_Stacja] WHERE [id] = @id"
    InsertCommand="INSERT INTO [T_Pks_Stacja] ([id], [nazwa]) VALUES (@id, @nazwa)"
    SelectCommand="SELECT [id], [nazwa] FROM [T_Pks_Stacja]"
    UpdateCommand="UPDATE [T_Pks_Stacja] SET [nazwa] = @nazwa WHERE [id] = @id">
. . .

SqlDataSourceStacje.InsertParameters["id"].DefaultValue = txtStacjeId.Text;
SqlDataSourceStacje.InsertParameters["nazwa"].DefaultValue = txtStacjeNazwa.Text;
SqlDataSourceStacje.Insert();

```

6.4. Testy funkcjonalne

- **Test 1.** *Nawigacja na stronach aplikacji klienckiej, kompatybilność z różnymi przeglądarkami*

Test ma na celu sprawdzenie czy wszystkie strony aplikacji klienckiej są odpowiednio linkowane. Sprawdzone zostały strona z informacjami o projekcie oraz strona z formularzem do administratora. Oprócz tego aplikacja została wypróbowana w następujących przeglądarkach: Firefox 9.01, Opera 11.52, IE 9.0.4.

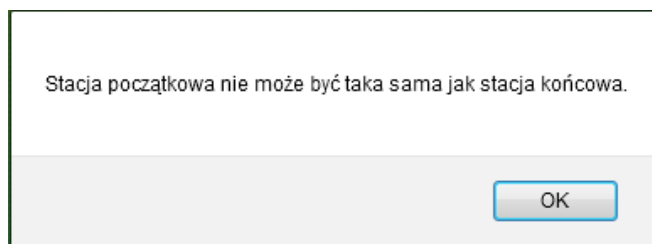


Rysunek 22 Test nawigacji po aplikacji klienckiej. Strona z informacjami o projekcie

Wynik testu: zaliczony

- **Test 2.** *Próba wyszukania połączenia z błędnie wprowadzonymi stacjami*

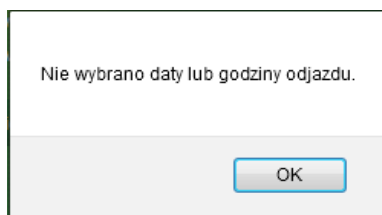
Użytkownik podaje w formularzu wyszukiwania tę samą stację początkową i końcową. Aplikacja nie powinna dopuścić do wyszukania takiego połączenia.



Rysunek 23 Test wyszukania połączenia z błędnie wprowadzonymi stacjami

Wynik testu: zaliczony

- **Test 3.** *Próba wyszukania połączenia z błędnie wprowadzoną datą i czasem odjazdu*
Użytkownik nie podaje daty ani czasu odjazdu dla szukanego połączenia. Aplikacja nie powinna dopuścić do wyszukania takiego połączenia.



Rysunek 24 Test wyszukania połączenia bez podania daty lub godziny odjazdu

Wynik testu: zaliczony

- **Test 4.** *Test pozytywny wyszukania połączenia przy prawidłowym zapytaniu*
Użytkownik podaje poprawne dane wejściowe dla wyszukiwarki. Aplikacja powinna przekierować użytkownika na stronę z wynikami wyszukiwania i wyświetlić pasujące rekordy.



Rysunek 25 Test pozytywny wyszukiwania połączeń

Wynik testu: zaliczony

6.5. Testy systemu replikacji

System replikacji baz danych testowany był z poziomu aplikacji operatorów, czyli tak jak powinien być wykorzystywany w praktyce. Sprawdzenie, czy replikacja rzeczywiście się wykonała, wykonywane było z poziomu programu MS SQL Management Studio, który oferuje bezpośredni dostęp do fizycznej bazy danych.

- **Test 1. Test replikacji przy dodaniu nowego połączenia operatora PKP**

Operator przewoźnika PKP w swojej aplikacji dodaje nowe połączenie. Powinno ono być natychmiast dodane do bazy danych, wyświetlone w tabeli aplikacji oraz powinna zostać wywołana replikacja bazy danych, aby w zbiorczej bazie pojawił się nowy rekord

APLIKACJA OPERATORA PKP

Strona główna

Połączenia

Stacje

Data odjazdu

Przewoźnicy

Połączenia

	id	id_stacja_start	id_stacja_koniec	id_data_odjazdu	godzina_odjazdu	id_przewoźnik	id_odleglosc	cena
Edit Delete	1	1	2	1	13:00:00	2	120	21,0000
Edit Delete	2	2	3	2	09:30:00	1	80	17,0000
Edit Delete	3	2	4	1	17:17:00	1	220	30,0000
Edit Delete	4	3	5	3	13:30:00	2	13	8,0000
Edit Delete	5	1	2	8	15:30:00	1	120	25,0000
	6	2	4	8	16:20:00	2	200	28

Dodaj rekord

Rysunek 26 Test replikacji przy dodaniu nowego połączenia

Po dodaniu nowego wpisu w aplikacji operatorskiej, nowe dane zostały zapisane w bazie danych PKP (sprawdzone w MS SQL Management Studio) i pokazane w zaktualizowanej tabeli w programie. Dodatkowo po sprawdzeniu zawartości zbiorczej bazy danych okazało się, że replikacja również zakończyła się sukcesem, co pokazane jest poniżej:

	id	id_stacja_start	id_stacja_koniec	id_data_odjazdu	godzina_odjazdu	id_odleglosc	id_przewoźnik	cena	id_legenda
1	1	1	2	1	13:00:00.0000000	120	2	21,00	1
2	2	2	3	2	09:30:00.0000000	80	1	17,00	2
3	3	2	4	1	17:17:00.0000000	220	1	30,00	1
4	4	3	5	3	13:30:00.0000000	13	2	8,00	2
5	5	1	2	8	15:30:00.0000000	120	1	25,00	1
6	6	2	4	8	16:20:00.0000000	200	2	28,00	NULL

Rysunek 27 Widok zbiorczej bazy danych po zakończonej replikacji

Wynik testu: zaliczony

- **Test 2. Test replikacji przy usunięciu istniejącego rekordu**

Operator danego przewoźnika usuwa jeden ze swoich rekordów połączenia za pomocą aplikacji operatorskiej. Rekord powinien zostać usunięty w bazie danych operatora oraz bazie zbiorczej, w tabeli aplikacji wpis powinien być również usunięty.

Wynik testu: zaliczony

- **Test 2. Test replikacji przy edycji istniejącego rekordu**

Operator danego przewoźnika edytuje jeden z swoich rekordów połączenia za pomocą aplikacji operatorskiej. Rekord powinien zostać zmieniony w bazie danych operatora oraz bazie zbiorczej, tabeli aplikacji wpis powinien być również zmieniony.

Wynik testu: zaliczony

6.6. Wnioski z testów

W wykonanych testach sprawdzane były najważniejsze funkcjonalności, które są kluczowe dla działania całego systemu, w szczególności systemu replikacji baz danych. Pominięte zostały dlatego np. testy formularza kontaktowego z administratorem, który jest usługą dodatkową.

Aplikacja kliencka posiada interfejs webowy, zatem jej kompatybilność z różnymi przeglądarkami oraz stabilność i bezpieczeństwo w trakcie używania są priorytetem. Oprócz tego, ważne było przetestowanie wyszukiwarki połączeń, która jest głównym łącznikiem aplikacji z rozproszonymi bazami danych. Sprawdzono jej zabezpieczenia pod względem wprowadzenia błędnych danych wejściowych oraz trafność wyników wyszukiwania, porównując je z fizyczną zawartością bazy danych.

Drugi etap testów dotyczył działania systemu replikacji baz danych. Kluczowe było w tym miejscu sprawdzenie poprawności zapisu danych do baz operatorskich przy wykorzystaniu aplikacji poszczególnych operatorów oraz sprawdzenie samego wywoływania procesu replikacji baz danych. Testy wykonane za pomocą aplikacji operatorskich weryfikowane były na poziomie fizycznej bazy danych.

Wszystkie z przeprowadzonych testów zakończyły się sukcesem. Cały system nie posiada jeszcze rozbudowanej funkcjonalności oprócz wyszukiwarki, dlatego testy nie były bardzo skomplikowane, można przyjąć jednak, że mogłyby funkcjonować w realnych warunkach.

7. Podsumowanie

W ramach niniejszego projektu wykonany został kompletny system informatyczny, który do swej funkcjonalności wykorzystuje obsługę rozproszonych baz danych. W ramach tego systemu powstały trzy niezależne aplikacje webowe – kliencka oraz dwie dostosowane dla administratorów poszczególnych przewoźników. Cały mechanizm replikacji rozproszonych baz danych zadziałał zgodnie z założeniami, co pokazały wykonane testy.

Projekt Internetowego Rozkładu Jazdy dał możliwość pracy w innym środowisku bazodanowym, niż ma to miejsce w wypadku większości aplikacji internetowych, które korzystają z lokalnej bazy danych. Zastosowanie mechanizmu replikacji transakcyjnej pokazuje, że w dość prosty sposób można zbudować system informatyczny, który do swej pracy wykorzystuje rozproszone, niejednorodne bazy danych. Nie dotyczy to jedynie występowania wielu baz w jednej aplikacji, ale również zakłada ich ulokowanie na różnych serwerach, w różnych częściach świata. Możliwe jest w ten sposób również połączenie ze sobą baz różnych producentów, np. MS SQL i Oracle, które często posiadają różne formaty zapisu danych.

Szczególnie wyraźnie widać zalety budowania tego rodzaju systemów w przypadku aplikacji o interfejsie webowym, do którego dostęp mamy za pomocą zwykłej przeglądarki internetowej. Tym sposobem cały system jest ogólnodostępny, a dane z których korzysta mogą pochodzić z wielu odległych od siebie źródeł. Może mieć to choćby ogromne zastosowanie w przypadku hurtowni danych i systemów wspomagających procesy decyzyjne.

Literatura

- [1] Coulouris G., Systemy rozproszone. Podstawy i projektowanie, WNT, Warszawa, 1998.
- [2] Payne C., ASP.NET dla każdego, Helion, Gliwice 2002.
- [3] Connolly R., ASP.NET 2.0. Projektowanie aplikacji internetowych, Helion, Gliwice, 2008.
- [4] Barney L., McLaughlin M., Oracle Database. Tworzenie aplikacji internetowych w AJAX i PHP, Helion, Gliwice, 2010.
- [5] Schafer S., HTML, XHTML i CSS : biblia, Helion, Gliwice, 2011.