Politechnika Krakowska im. Tadeusza Kościuszki
Wydział Fizyki, Matematyki i Informatyki

Zaawansowane technologie baz danych

Projekt: *Porównanie systemu SQL Server z systemem CouchDB.*

Data oddania:
27-05-2018

Kierunek: *Informatyka Stosowana*
Rok akademicki: *2017/2018*

Wykonali:
Przemysław Kleszcz,
Krzysztof Tatar

Spis treści

1. Wprowadzenie

W zakresie projektu zaliczeniowego zrealizowano opracowanie porównawcze na temat relacyjnej bazy danych MSSQL Server oraz dokumentowej - CouchDB. MSSQL Server to system DBMS wspierany i rozpowszechniany przez Microsoft. Jest to główny produkt bazodanowy tej firmy, który charakteryzuje się tym, iż jako język zapytań używany jest przede wszystkim Transact-SQL, który stanowi rozwinięcie standardu ANSI/ISO. CouchDB to rozwijany przez fundację Apache wysoko wydajny, nierelacyjny silnik baz danych napisany w języku Erlang zorientowany na dokumenty. Dostęp do systemu realizuje się za pomocą API korzystającego z mechanizmu REST. Poniższe opracowanie ma za zadanie przybliżyć tematykę obu baz danych oraz ukazać różnicę pomiędzy relacyjnym a dokumentowym podejściem do składowania danych.

2. Cel projektu

Celem projektu jest analiza możliwości obu silników bazodanowych. W tym celu obie bazy zostaną zainstalowane oraz wypełnione odpowiednio dużą ilością danych umożliwiających przeprowadzenie testów. Nakreślony zostanie mechanizm migracji zbioru danych do wymienionych powyżej technologii. Opracowana zostanie aplikacja w języku C#, dzięki której możliwe będzie odbycie dokładnych pomiarów czasu dostępu do obu baz danych oraz prezentacja rezultatów na wykresie. Na koniec, na podstawie przeprowadzonych pomiarów zostaną przedstawione różnice w działaniu obu baz oraz wnioski z wykonanego projektu.

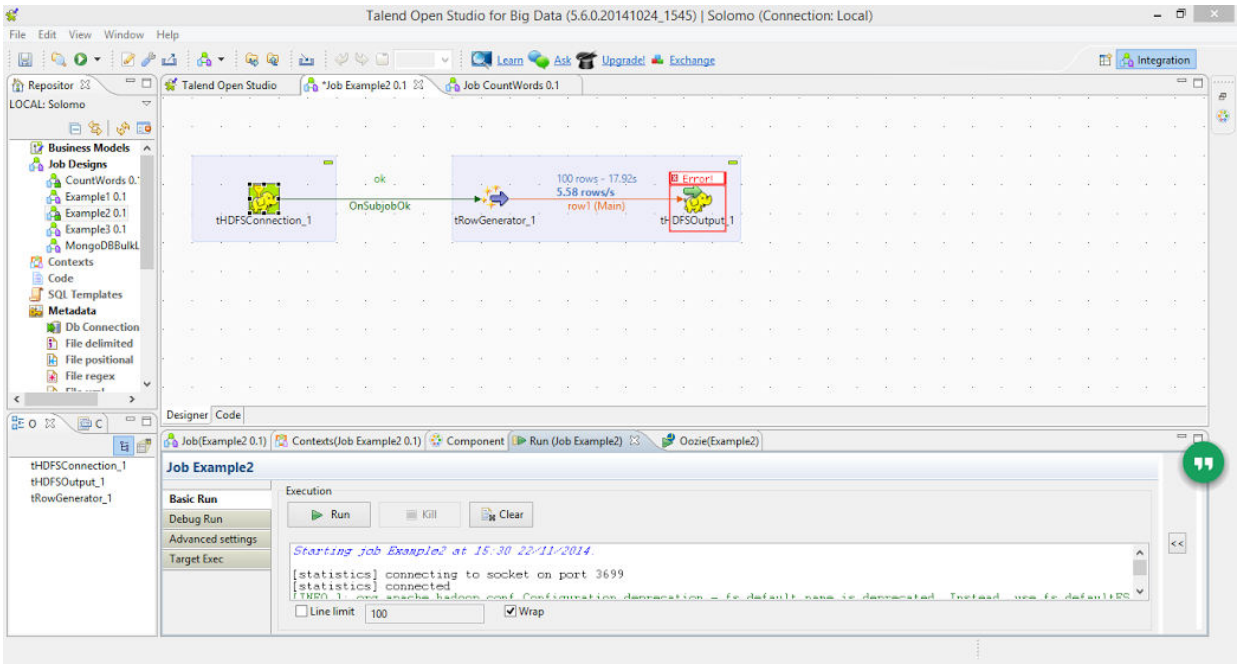
3. Środowisko

Bazy danych zostały zainstalowane na komputerze z czterordzeniowym procesorem Intel, pracującym pod kontrolą systemu operacyjnego Windows 10. Ten system operacyjny został wybrany z powodu ułatwionego procesu instalacji na tej platformie oraz uproszczonej procedurze zasymulowania testów. Warto wspomnieć, że oba systemy baz danych mogą pracować również pod kontrolą systemów Mac OS X oraz Linux.

W celu uzupełnienia baz danymi, wybrany został zestaw Microsoft Adventure Works. Jest to gotowy zbiór danych udostępniany przez firmę Microsoft prezentujący rozbudowany i spójny model bazodanowy, gotowy do bezpośredniego importu i wykorzystania. Wersja Adventure Works jest co roku udoskonalana, optymalizowana i dostrajana do najnowszej wersji systemu bazodanowego Microsoft SQL Server.

Zestaw Microsoft Adventure Works zapewnił dane dla bazy danych Microsoft SQL Server. Należało przygotować analogiczny zbiór danych po stronie CouchDB. W tym celu zamiast konstruować od podstaw proces migracji lub przesyłać odpowiednie dane ręcznie, wykorzystany został system Talend Open Studio for Big Data.

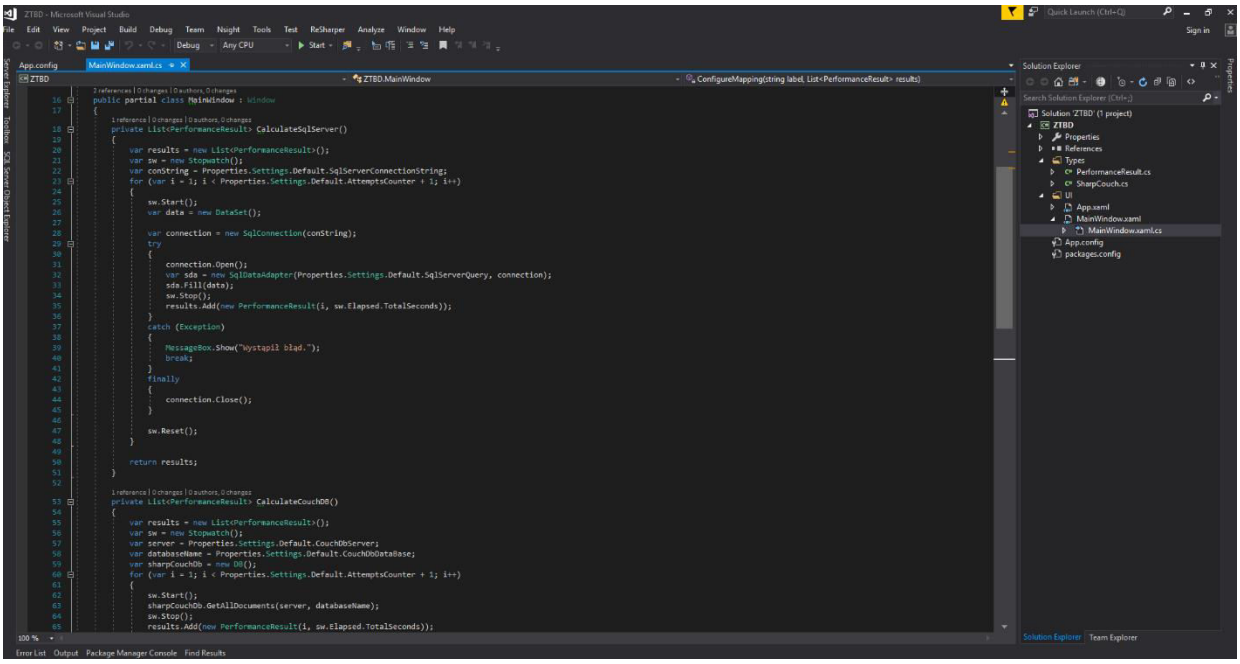
Jest to oprogramowanie wspomagające integrację zbiorów danych z różnych źródeł. Począwszy od plików csv, xml, zdalnych usług internetowych po bazy danych i zagregowane magazyny danych.



Oprogramowanie korzysta z odpowiednich mechanizmów zwanych konektorami do pobierania danych z źródła, odpowiedniego ich przekształcenia, prezentacji a następnie zapisu w lokalizacji docelowej. W celu ustalenia transmisji danych z bazy SQL Server do bazy CouchDB, w pierwszej kolejności należało przeprowadzić konfigurację konektora do systemu MSSQL. Wymagało to podania wszystkich parametrów połączeniowych oraz określenia w sposób jawny lokalizacji instalacji bazy SQL Server. Następnym krokiem była analogiczna konfiguracja konektora do bazy CouchDB. Na końcu należało stworzyć reguły mapowania danych pochodzących z bazy SQL Server do systemu CouchDB. Obie powyższe bazy interpretują dane w odmienny sposób, zatem niezbędne jest nakreślenie standardu odwzorowania pojedynczego rekordu na dokument.

Implementacja aplikacji dokonującej pomiaru wydajności silników bazdanowych została opracowana w języku C# w środowisku *Microsoft Visual Studio.* Jest to zintegrowane środowisko produkcyjne (IDE) zaprojektowane przez firmę *Microsoft.* Jest używane do produkcji programów komputerowych na platformę Windows. Za jego pomoc można produkować zarówno kod natywny jak i zarządzany przez platformę .NET. Oprócz pełnego wsparcia dla platformy .NET, Visual Studio oferuje wiele narzędzi i dodatków usprawniających proces implementacji dla wielu innych technologii – również bazodanowych. Edytor narzędzia jest w pełni zintegrowany z technologią – zapewnia odpowiednie podświetlenie i weryfikację składni. Wbudowane narzędzia w wygodny sposób wspomagają nawiązywanie połączenia z bazami danych oraz zarządzanie ich właściwościami wprost z edytora.

Zrzut ekranu z programu Microsoft Visual Studio:



4. Analiza

Po zainstalowaniu baz danych na testowym komputerze oraz zmigrowaniu danych pomiędzy systemami, kolejnym krokiem projektu była analiza poszczególnych baz. W pierwszej kolejności zbadana została poprawność poprzedniego kroku migracji. Należało skontrolować czy wszystkie pola zostały bezbłędnie zaimportowane. Zrewidowana została poprawność na poziomie merytorycznym (czy dane nie zostały przekłamane) oraz na poziomie typów danych (czy zastosowane zostały typy danych określone podczas kroku odwzorowania).

Poniższy zrzut ekranu ukazuje sposób reprezentacji tabeli osób w bazie MSSQL Server. Jest to standardowa baza relacyjna.

	BusinessEntityID	PersonType	NameStyle	Title	FirstName	MiddleName	LastName	Suffix	EmailPromotion	AdditionalContactInfo
1	1	EM	0	NULL	Ken	J	Sánchez	NULL	0	NULL
2	2	EM	0	NULL	Terri	Lee	Duffy	NULL	1	NULL
3	3	EM	0	NULL	Roberto	NULL	Tamburello	NULL	0	NULL
4	4	EM	0	NULL	Rob	NULL	Walters	NULL	0	NULL
5	5	EM	0	Ms.	Gail	A	Erickson	NULL	0	NULL
6	6	EM	0	Mr.	Jossef	H	Goldberg	NULL	0	NULL
7	7	EM	0	NULL	Dylan	A	Miller	NULL	2	NULL
8	8	EM	0	NULL	Diane	L	Margheim	NULL	0	NULL
9	9	EM	0	NULL	Gigi	N	Matthew	NULL	0	NULL
10	10	EM	0	NULL	Michael	NULL	Raheem	NULL	2	NULL
11	11	EM	0	NULL	Ovidiu	V	Cracium	NULL	0	NULL
12	12	EM	0	NULL	Thierry	B	D'Hers	NULL	2	NULL
13	13	EM	0	Ms.	Janice	M	Galvin	NULL	2	NULL
14	14	EM	0	NULL	Michael	I	Sullivan	NULL	2	NULL
15	15	EM	0	NULL	Sharon	B	Salavaria	NULL	2	NULL
16	16	EM	0	NULL	David	M	Bradley	NULL	1	NULL
17	17	EM	0	NULL	Kevin	F	Brown	NULL	2	NULL
18	18	EM	0	NULL	John	L	Wood	NULL	2	NULL

Poniższy zrzut ekranu ukazuje sposób reprezentacji analogicznych danych
Po stronie CouchDB w formacie JSON.

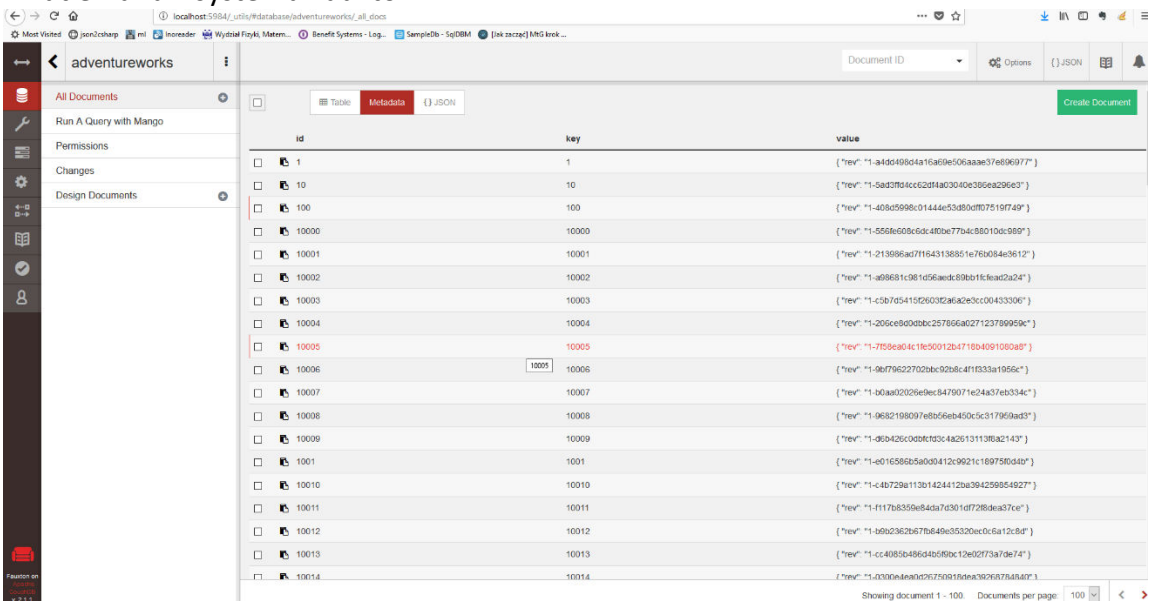

```
{
 "id": "1",
 "key": "1",
 "value": {
  "rev": "1-a4dd498d4a16a69e506aaae37e896977"
 },
 "doc": {
  "_id": "1",
  "_rev": "1-a4dd498d4a16a69e506aaae37e896977",
  "BusinessEntityID": "1",
  "PersonType": "EM",
  "NameStyle": "false",
  "Title": [],
  "FirstName": "Ken",
  "MiddleName": "J",
  "LastName": "Sánchez",
  "Suffix": [],
  "EmailPromotion": "0",
  "AdditionalContactInfo": [],
  "Demographics": "<IndividualSurvey xmlns=\"http://schemas.microsoft.com/
  "rowguid": "92C4279F-1207-48A3-8448-4636514EB7E2",
```

W celu późniejeszej implementacji programu testującego, należało najpierw wnikliwie przeanalizować charakterystykę obu silników. Ważnym punktem pracy było zbiorcze zestawienie najważniejszych cech systemów. Decyzje podjęte w trakcie analizy zestawienia miały mieć odzwierciedlenie w opracowanym oprogramowaniu testującym.

Nazwa	CouchDB	SQL Server
Model bazodanowy	Baza dokumentowa	Baza relacyjna
Język implementacji	Erlang	C++
Wykorzystanie schematu	Nie wykorzystuje	Wykorzystuje
Typowanie	Nie	Tak
Wykorzystanie SQL	Nie	Tak
Metody dostępu	REST	JDBC, ODBC, ADO…
Obsługa transakcji	Nie	Tak
Wspracie systemów	Windows, Linux,OS X, BSD…	Windows, Linux …

Obie bazy danych wyróżnia odmienny model przechowywania danych. Opierają się na różnych technologiach, działających w ramach różnych platform. Charakteryzują je odmienne metody dostępu do repozytorium. Zatem nasuwa się wniosek, że jedynym wyjściem by skutecznie przetestować powyższe systemy jest opracowanie aplikacji, która korzysta z oddzielnych metod dostępu do danych. Brak wspólnego dostępnego interfejsu uniemożliwia skorzystanie z tego samego medium. Dla MSSQL oczywistem rozwiązaniem jest metoda korzystająca z technologii ADO .NET – podstawową biblioteką Microsoftu do wykonywania zapytań. Polega ona na przekazywaniu do silnika bazodanowego zapytań SQL, przekazanych za pośrednictwem programu. Jedyną metodą dostępu do danych w systemie CouchDB jest REST. Zatem komunikacja z bazą jest oparta na protokole HTTP. System Fauxton, który stanowi integralną część systemu CouchDB – przetwarza żądania HTTP, przekształca je w wewnętrzne zapytania do bazy oraz zwraca odpowiedź do klienta interfejsu.

Zrzut ekranu z systemu Fauxton:



5. Implementacja klienta testującego

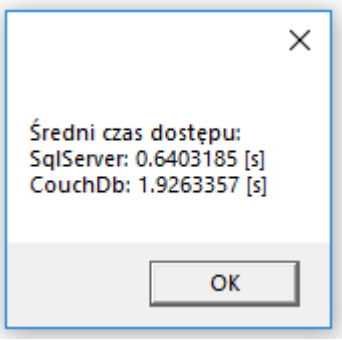
Aplikacja klienta testującego została wykonana w języku C# (Platforma .NET) w technologii WPF (Windows Presentation Foundation). Aplikacja została zaopatrzona w stosowny plik konfiguracyjny, który jest odpowiedzialny za przechowywanie odpowiednich parametrów połączeniowych do baz danych, zapytań które zostaną wykonane oraz parametru określającego ilość prób wykonania zapytań do baz.

Plik konfiguracyjny klienta:

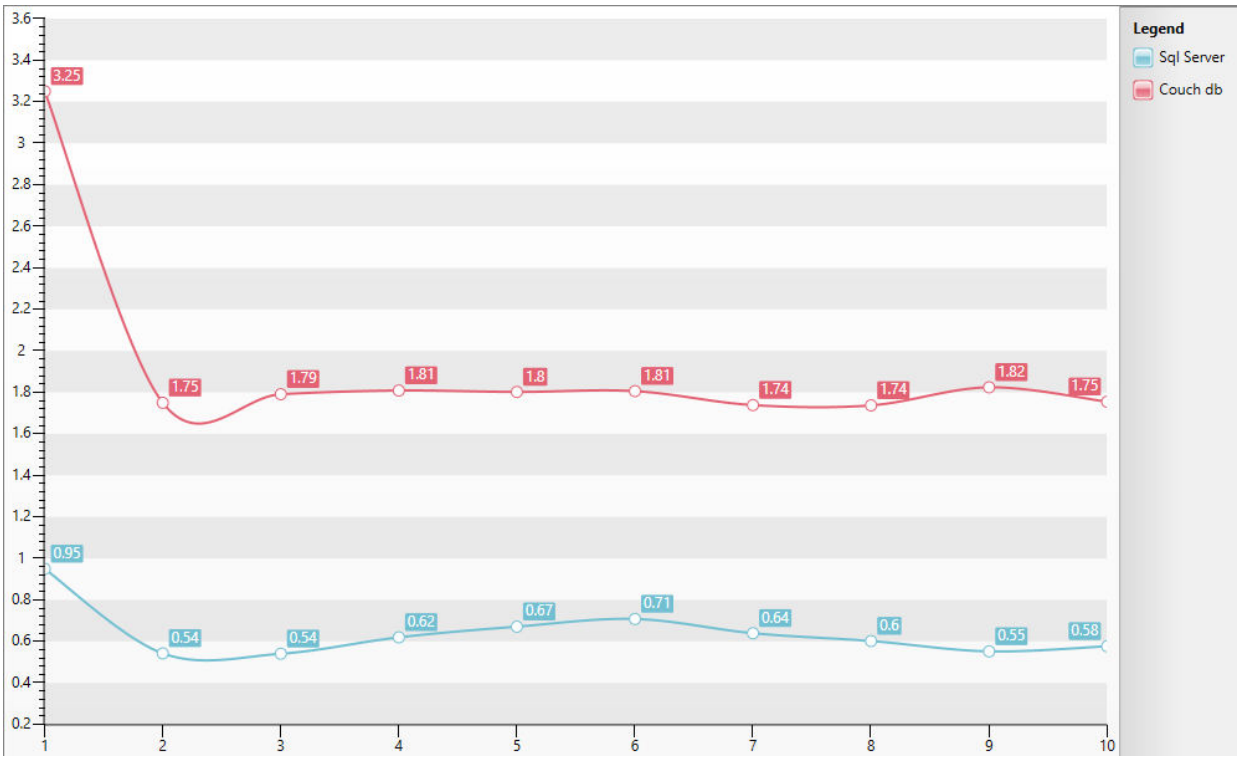
```
<ZTBD.Properties.Settings>
            <setting name="SqlServerConnectionString" serializeAs="String">
                <value>Initial Catalog=AdventureWorks2017;Data Source=.\SQLEXPRESS;Integrated
Security=SSPI;</value>
            </setting>
            <setting name="AttemptsCounter" serializeAs="String">
                <value>10</value>
            </setting>
            <setting name="SqlServerQuery" serializeAs="String">
                <value>select * from [AdventureWorks2017].[Person].[Person]</value>
            </setting>
            <setting name="CouchDbServer" serializeAs="String">
                <value>http://localhost:5984</value>
            </setting>
            <setting name="CouchDbDataBase" serializeAs="String">
                <value>adventureworks</value>
            </setting>
</ZTBD.Properties.Settings>
```

Zastosowanie pliku konfiguracyjnego eliminuje potrzebę ponownej kompilacji klienta w celu modyfikacji parametrów połączenia I testowania.

Główny kod programu jest odpowiedzialny za wywołanie dwóch metod – wykonanie testu dla bazy MSSQL oraz CouchDB. Po wykonaniu wszystkich testów w ilości opisanej parametrem *AttemptsCounter,* aplikacja wyświetla informację o średnim czasie dostępu z wszystkich prób.



Na koniec program sporządza wykres czasu dostępu do obu baz danych dla wszystkich zrealizowanych prób. Widoczne na wykresie opóźnienie dla pierwszej próby jest związane z dokonaniem pierwszego połączenia oraz inicjalizacją wszystkich wymaganych zasobów.



Kod odpowiedzialny za główną funkcjonalność aplikacji:

```
private void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
  var sqlServerResults = CalculateSqlServer();
  var couchDbResults = CalculateCouchDB();
  ConfigureMapping("Sql Server", sqlServerResults);
  ConfigureMapping("Couch db", couchDbResults);
  var sqlServerAvg = sqlServerResults.Select(x => x.Time).Average();
  var couchDbAvg = couchDbResults.Select(x => x.Time).Average();
  MessageBox.Show($"Średni czas dostępu:\nSqlServer: {sqlServerAvg}[s]\nCouchDb:
{couchDbAvg} [s]");
}
```

```
public MainWindow()
{
  InitializeComponent();
  Loaded += MainWindow_Loaded;
}
```

```
private void ConfigureMapping(string label, List<PerformanceResult> results)
{
    var mapping = new SeriesMapping
    {
          LegendLabel = label,
          SeriesDefinition = new SplineSeriesDefinition(),
          ItemsSource = results
     };

     mapping.ItemMappings.Add(new ItemMapping("Attempt", DataPointMember.XValue));
     mapping.ItemMappings.Add(new ItemMapping("Time", DataPointMember.YValue));
     radChart.SeriesMappings.Add(mapping);
}
```

```
private List<PerformanceResult> CalculateCouchDB()
{
    var results = new List<PerformanceResult>();
    var sw = new Stopwatch();
    var server = Properties.Settings.Default.CouchDbServer;
    var databaseName = Properties.Settings.Default.CouchDbDataBase;
    var sharpCouchDb = new DB();
    for (var i = 1; i < Properties.Settings.Default.AttemptsCounter + 1; i++)
    {
         sw.Start();
         sharpCouchDb.GetAllDocuments(server, databaseName);
         sw.Stop();
         results.Add(new PerformanceResult(i, sw.Elapsed.TotalSeconds));
         sw.Reset();
     }

    return results;
}
```

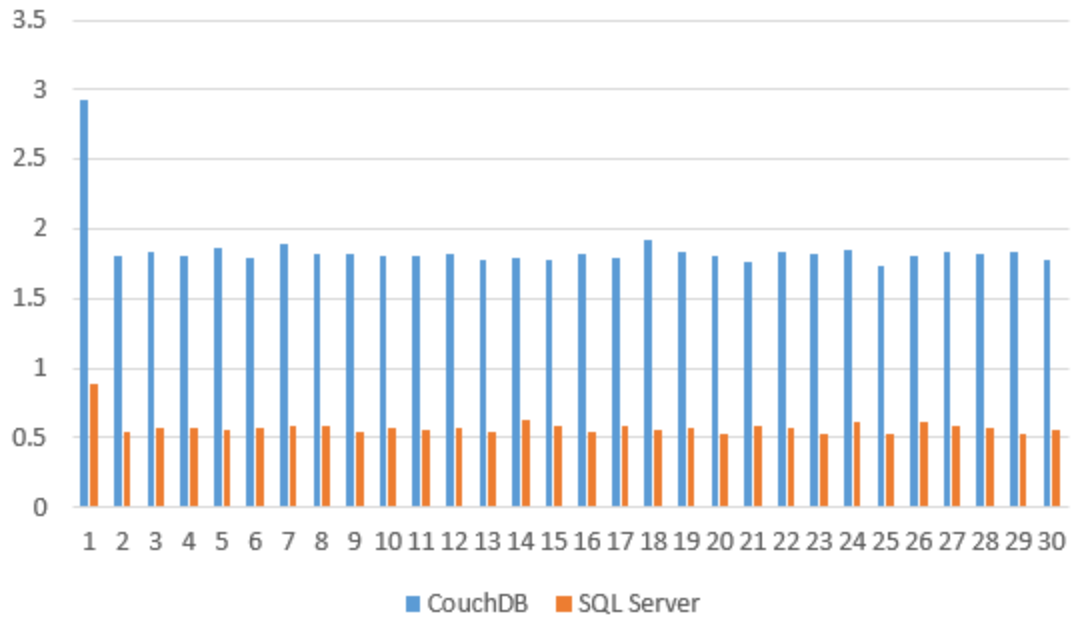
```
private List<PerformanceResult> CalculateSqlServer()
{
     var results = new List<PerformanceResult>();
     var sw = new Stopwatch();
     var conString = Properties.Settings.Default.SqlServerConnectionString;
     for (var i = 1; i < Properties.Settings.Default.AttemptsCounter + 1; i++)
     {
          sw.Start(); var data = new DataSet();
          var connection = new SqlConnection(conString);
          try
          {  connection.Open(); var sda = new
SqlDataAdapter(Properties.Settings.Default.SqlServerQuery, connection);
             sda.Fill(data); sw.Stop();
             results.Add(new PerformanceResult(i, sw.Elapsed.TotalSeconds));
           }
           catch (Exception)
           {
              MessageBox.Show("Wystąpił błąd."); break;
           } finally { connection.Close(); }
                sw.Reset();
       }
            return results;
}
```

6. Pomiary

W celu dokonania właściwych pomiarów czasu za pomocą klienta ustalono zbiór danych w wysokości **20 000** rekordów po stronie MSSQL Server oraz analogiczny magazyn danych po stronie CouchDB. Ilość prób wykonania zapytania odczytu ustalono na **30** powtórzeń. Zastosowano proste zapytanie SQL do bazy SQL Server oraz metodę odczytu analogicznych danych (o tych samych warunkach) dla CouchDB.

Średni czas odczytu
SQLServer: 0.57581929 [s]
CouchDB: 1.852948873 [s]

Zbiorcze porównanie prób dostępu:



Test wykazał, że w przypadku prostego zapytania odczytu, czas odpowiedzi CouchDB jest ponad trzykrotnie dłuższy od MSSQL Server. Dla większych porcji danych czas dostępu dodatkowo zwiększa się. W przypadku bardziej złożonych zapytań czas wydłuża się jeszcze bardziej. Wiąże się z to ograniczoną możliwością modelowania relacji za pośrednictwem bazy dokumentowej.

7. Wnioski

Sposób pozyskiwania i aktualizacji danych w bazie couchDB różni się znacząco od podejścia w standardowych bazach relacyjnych. Nie wykorzystujemy języka SQL. W zamian uzyskujemy dostęp do danych za pomocą języka JavasScript (za pośrednictwem Fauxton) lub żądań HTTP. Opracowana przez nas aplikacja obrazuje, że czas dostępu do systemu CouchDB jest znacznie większy w stosunku do SQL Server. Wiąże się to z wykorzystanymi mechanizmami wyższego poziomu – serializacją danych oraz sposóbem ich przechowywania i narzutem czasowym protokołu HTTP.

Powyższe rozważania pozwalają wysunąć wniosek, że bazy dokumentowe świetnie nadają się jako prosty magazyn nierelacyjnych danych. W przypadku gdy zachodzi konieczność zamodelowania skomplikowanej relacji lub potrzeba zapewnienia wysokiej wydajności i dostępności – warto sięgnąć po bazę relacyjną.

8. Bibliografia

1. Patrick LeBlanc, Microsoft SQL Server 2012 – Krok po kroku, APN Promise 2013

2. Itzik Ben-Gan, Microsoft SQL Server 2012 T-SQL Fundamentals, Microsoft Press 2012

3. Adam Jorgensen, Microsoft SQL Server 2012 Bible, John Willey & Sons 2012

4. William Durkin, SQL Server 2017 Developer's Guide, Packt 2018

5. J. Chris Anderson, CouchDB The Definitive Guide, O'Reilly 2010

6. M.C Brown, Getting Started with CouchDB, O'Reilly 2012