STUDIA INFORMATICA 2012
Volume 33 Number 2A (105)

Damian ZAPART, Tomasz WALLER, Magdalena TKACZ
Instytut Informatyki, Uniwersytet Śląski

PORÓWNANIE WYDAJNOŚCI RELACYJNEJ I NIERELACYJNEJ BAZY DANYCH W KONTEKŚCIE PRZECHOWYWANIA DANYCH Z MIKROMACIERZY DNA

Streszczenie. W artykule porównana została wydajność dwóch wersji systemów zarządzania bazami danych pod kątem możliwości zastosowania ich w gromadzeniu danych z mikromacierzy DNA. W analizie porównawczej uwzględniono metody składowania danych: oparte na modelu nierelacyjnym (noSQL) oraz metodę gromadzenia danych mikromacierzowych zgodną z modelem relacyjnym Obiekt-Atrybut-Wartość. Do implementacji relacyjnej bazy danych wykorzystany został system Microsoft SQL Server 2012. Implementacja nierelacyjnej bazy danych wykonana została w programie Raven DB. Ocena efektywności każdej z metod gromadzenia danych oparta została na próbkowaniu czasu procesora (ang. *CPU sampling*) za pomocą oprogramowania Microsoft Visual Studio Profiler.

Słowa kluczowe: relacyjne bazy danych, nierelacyjne bazy danych, porównanie wydajności, model Obiekt-Atrybut-Wartość, Microsoft SQL Server, Raven DB, NoSQL, mikromacierze DNA

RELATIONAL AND NON-RELATIONAL DATABASE EFFICIENCY COMPARISION WITH DNA MICROARRAY DATA STORING

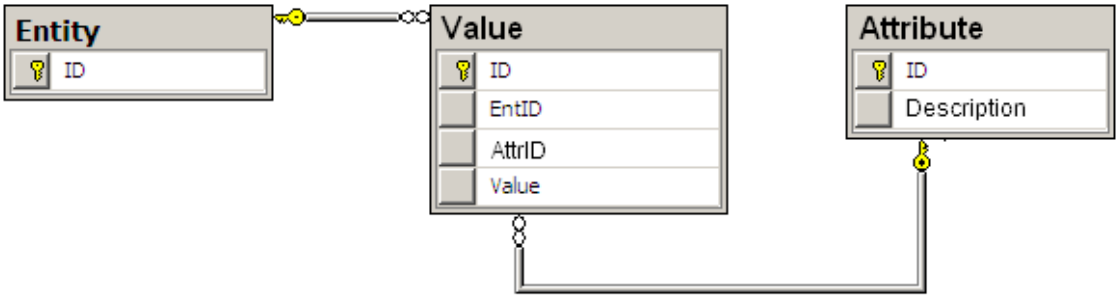
Summary. The paper compares the efficiency of two versions of database management systems from the possibility of using them in storing data from DNA microarray point of view. In comparative analysis 3 methods of storing data were taken into account: basing on non-relational model (noSQL) and the method of storing microarray data according to the relational model called entity-attribute-value. Microsoft SQL Server 2012 system was used to implementation of relational database. Implementation of non-relational database was performed in Raven DB. Evaluation of efficiency of each of the method of storing data was based on CPU sampling using Micrisoft Visual Studio Profiler.

Keywords: relational database, non-relational database, entity–attribute–value model, Microsoft SQL Server, Raven DB, NoSQL, DNA microarray

1. Wstęp

Mikromacierze DNA to w biologii molekularnej popularna technologia wykorzystywana do pomiarów aktywności transkrypcyjnej (ekspresji) dziesiątek tysięcy genów. Znajdują one szerokie zastosowanie w genotypowaniu, diagnostyce medycznej oraz farmacji [5].

System zarządzania bazą danych (ang. *database management system*, DBMS) może stanowić doskonałe narzędzie wspomagające gromadzenie i przetwarzanie danych z mikromacierzy DNA. W literaturowych bazach danych istnieje już kilka interesujących doniesień i opracowań naukowych poświęconych problematyce wykorzystania DBMS w składowaniu i analizie danych mikromacierzowych [1, 2, 3, 4]. Organizacja danych z wielu eksperymentów mikromacierzowych zgodnie z klasycznym modelem relacyjnym jest jednak co najmniej kłopotliwa do zrealizowania. Główne wyzwanie stanowi opracowanie uniwersalnej struktury bazy dla zróżnicowanej i trudnej do przewidzenia liczby atrybutów charakteryzujących poszczególne doświadczenia. Jednym ze sposobów rozwiązania tego problemu jest zastosowanie modelu danych Obiekt-Atrybut-Wartość (ang. *entity-attribute-value model*, EAV) [2, 4] (patrz rysunek 1).



Rys. 1. Projekt prostej bazy danych oparty na modelu Obiekt-Atrybut-Wartość
Fig. 1. The schema of simple database based on Entity-Attribute-Value model

Model EAV umożliwia gromadzenie danych o dynamicznie zmieniającej się strukturze. W reprezentacji tej wykorzystuje się trzy tabele (pierwsza tabela zawiera identyfikator obiektu, druga nazwę atrybutu, a trzecia wartość atrybutu). Możliwe jest także stworzenie jednej tabeli z trzema kolumnami (obiekt, atrybut, wartość). W przypadku danych mikromacierzowych tabele typu EAV są tworzone są przez: identyfikator próbki/mikromacierzy, identyfikator genu (ProbsetID) oraz wartość ekspresji genu. Zastosowanie modelu Obiekt-Atrybut-Wartość w projekcie bazy danych umożliwia przechowywanie danych z wielu eksperymentów mikromacierzowych zróżnicowanych zarówno pod względem typów mikromacierzy, jak i liczby próbek.

Strukturę danych typu EAV cechuje jednak olbrzymi przyrost liczby wierszy. W przypadku eksperymentu mikromacierzowego opartego na mikromacierzach Affymetrix typu HG-U133 liczba krotek w tabeli *Value* dla każdego doświadczenia równa będzie iloczynowi N*22283, gdzie N to liczba badanych próbek, a 22283 to liczba analizowanych genów. Ponieważ duża liczba danych gromadzonych w jednej tabeli jest charakterystyczna dla rozwią-

zań opartych na nierelacyjnych bazach danych, autorzy artykułu postanowili sprawdzić, czy zastosowanie bazy danych typu noSQL (Raven DB) może być bardziej wydajne niż implementacja w relacyjnym systemie zarządzania bazą danych (Microsoft SQL Server).

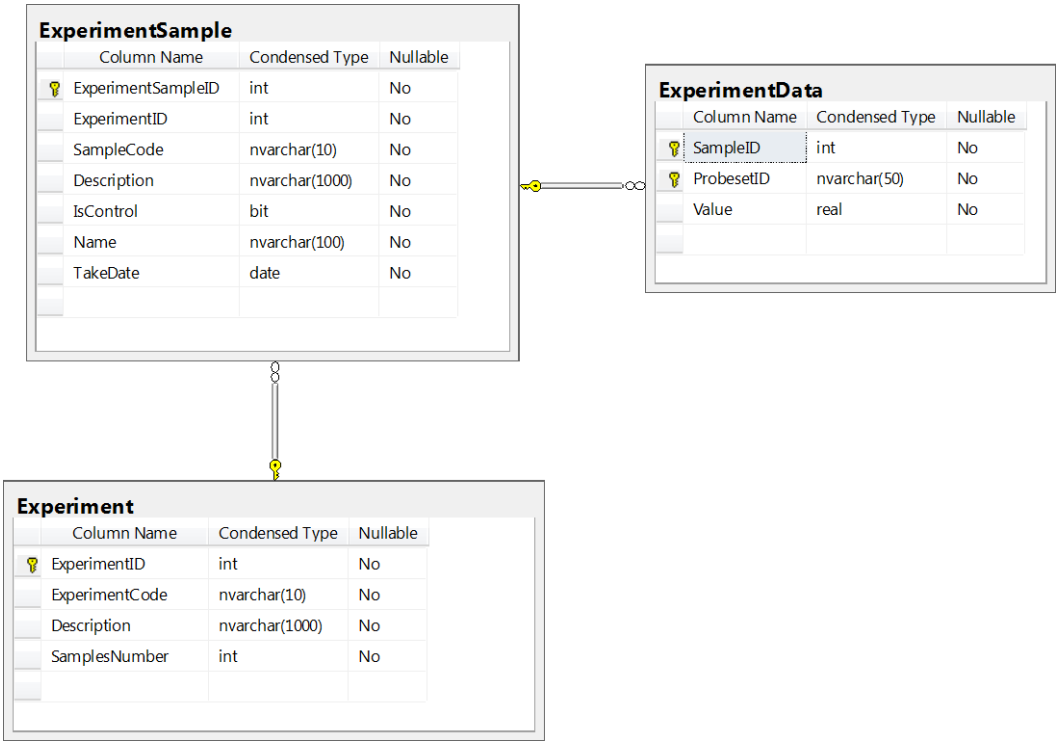
2. Przykładowe struktury danych stosowane w bazie danych mikromacierzowych

Na potrzeby wykonania analizy porównawczej uwzględniono cztery, opracowane przez autorów, sposoby gromadzenia danych mikromacierzowych w bazie danych. Sposób pierwszy to gromadzenie danych oparte na przystosowanej do danych mikromacierzowych strukturze typu EAV. Wykonana adaptacja polegała na wyeliminowaniu sztucznego klucza głównego na tabeli *ExperimentData* na rzecz pary kolumn *SampleID* oraz *ProbesetID* (patrz rysunek 2). Uzasadnieniem takiego podejścia był fakt, iż w obrębie pojedynczego eksperymentu dany *ProbesetID* może wystąpić tylko i wyłącznie raz na danej mikromacierzy. Unikalność *ProbesetID* zapewnia specyfika technologii firmy Affymetrix. Eliminacja sztucznego klucza głównego była również motywowana aspektami wydajnościowymi. Aby zapewnić wysoką skalowalność rozwiązania, należałoby zastosować typ danych bigint (8 bajtów), co w połączeniu z szybko przyrastającą liczbą rekordów w tabeli spowodowałoby zbędny rozrost zajmowanej przez tabelę przestrzeni dyskowej. Ponadto miałoby to negatywny wpływ na wydajność wyszukiwania. Dodatkowo, dzięki zastosowaniu naturalnego klucza głównego, dane z tabeli zostały fizycznie rozmieszczone na dysku twardym w optymalny sposób. W związku z tym nie będą występować „skoki" głowicy na różne sektory dysku, co również pozytywnie wpływa na wydajność wyszukiwania oraz wydajność odczytów. Implementacja opisanej struktury danych wykonana została w systemie zarządzania relacyjną bazą danych Microsoft SQL Server 2012.

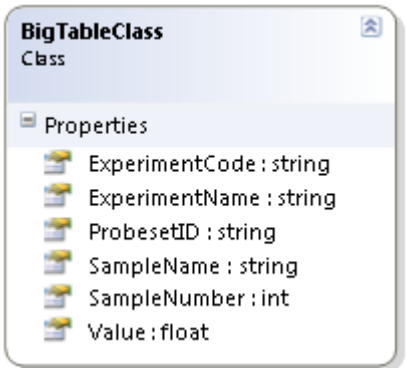
Kolejne metody gromadzenia danych mikromacierzowych opracowane zostały na darmowej, nierelacyjnej bazie danych (NoSQL [10]) o nazwie Raven DB. Rozwiązanie to, oparte na .NET Framework 4.0, umożliwia gromadzenie obiektów o dowolnej strukturze danych dzięki wykorzystaniu możliwości serializacji obiektów oraz, znanej z rozwiązania problemu asynchronicznej komunikacji klient-serwer, technologii JSON [11] (ang. *JavaScript Object Notation*).

Pierwszy opracowany sposób gromadzenia danych w bazie typu NoSQL zakłada wykorzystanie pojedynczej struktury danych, złożonej jedynie z typów prostych. Z uwagi na prostotę struktury danych podczas zapisu danych konieczne jest tworzenie osobnego rekordu danych dla każdego *ProbesetID* dla każdej mikromacierzy DNA. Podejście to jest więc zbliżone do modelu EAV w relacyjnych bazach danych pod kątem liczby rekordów. Z powodu

dużej ilości danych przechowywanych w jednej strukturze metoda ta nosi nazwę *BigTable* [6, 7, 8, 9] (patrz rysunek 3).



Rys. 2. Projekt relacyjnej bazy danych mikromacierzowych oparty na modelu Obiekt-Atrybut-Wartość
Fig. 2. The schema of microarray database based on Entity-Attribute-Value model

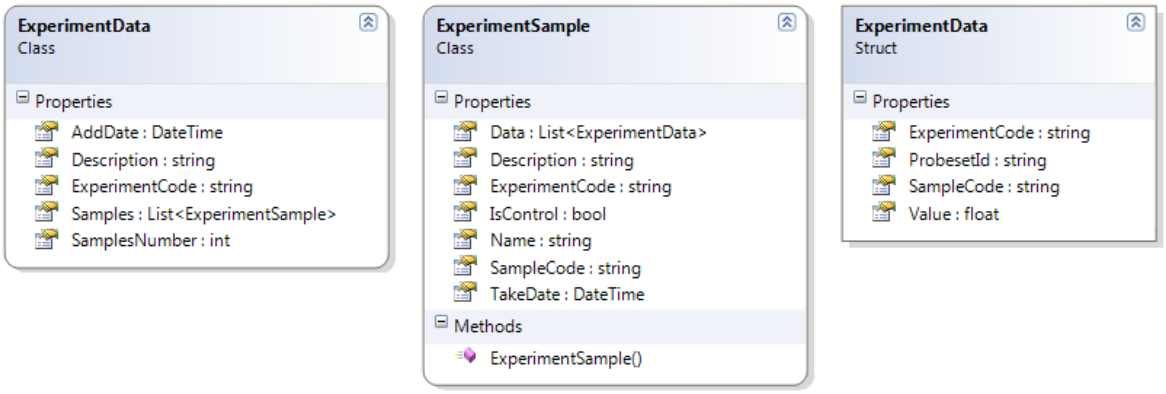


Rys. 3. Klasa pojedynczej tabeli danych w bazie NoSQL – *BigTable*
Fig. 3. The class of single table in NoSQL database – *BigTable*

Kolejna opracowana struktura danych w modelu nierelacyjnym wykorzystuje mechanizm serializacji grafu obiektów oraz ich gromadzenie w dynamicznej strukturze bazy danych. W dalszej części artykułu podejście to nosi nazwę *Object Graph*.

Klasa najwyższego poziomu *ExperimentData* reprezentuje pojedynczy eksperyment mikromacierzowy, w skład którego wchodzi generyczna lista obiektów reprezentujących kolejne mikromacierze w eksperymencie – *ExperientSample*. Każda z mikromacierzy ma z kolei generyczną listę obiektów reprezentujących dane dla poszczególnej próbki w eksperymencie (patrz rysunek 4). Podejście to zapisuje w bazie danych tylko jeden rekord danych dla każde-

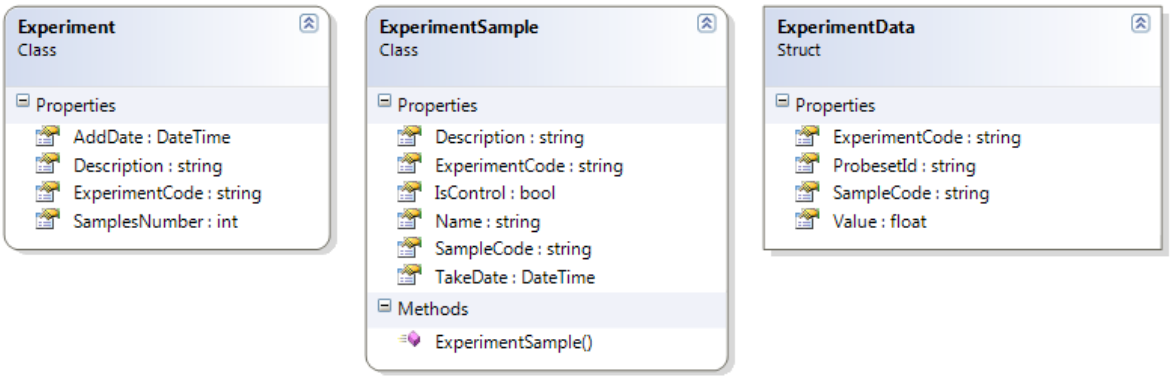
go eksperymentu, jednak w rekordzie tym przechowywana jest informacja o całym eksperymencie.



Rys. 4. Model grafu obiektów w bazie NoSQL – *Object Graph*
Fig. 4. The object graph model in NoSQL database – *Object Graph*

Opisane rozwiązanie z założenia przerzuca obciążenie z aplikacji klienckiej bezpośrednio na silnik bazodanowy – w szczególności przy wywołaniach asynchronicznych. Wpływa to pozytywnie na wydajność aplikacji, ale nie zmienia czasu trwania całej operacji na danych.

Trzecie opisane przez autorów podejście wykorzystane w nierelacyjnej bazie danych odzwierciedla model EAV (jego nazwa w eksperymencie to *NoSQL EAV*) w modelu relacyjnym. Na potrzeby tego modelu struktura danych z przystosowanego modelu EAV została tak zmodyfikowana, aby każdy z obiektów składowych był niezależny i jedynie referował do obiektu nadrzędnego za pomocą kodu. W modelu tym każdy z potomków w obiekcie podrzędnym przechowywał informację o kodzie obiektu nadrzędnego, jednak nie miał do niego jawnie przypisanej referencji (w modelu relacyjnym podejście takie nie jest możliwe) – w ten sposób autorzy artykułu starali się oddać idee relacji 1:N.



Rys. 5. Model EAV w modelu nierelacyjnej bazy danych – *NoSQL EAV*
Fig. 5. The EAV model in non-relational database model – *NoSQL EAV*

Z informatycznego punktu widzenia ważną informacją jest to, że domyślnie system Raven DB generuje unikatowy indeks każdego egzemplarza obiektów przechowywanego w bazie danych. Indeksy te, wywodzące się z adresów protokołu REST, mają postać *nazwa_obiektu/unikalne_id* i to właśnie przy ich wykorzystaniu fizycznie przechowywane są dane w bazie danych. Z uwagi na specyfikę wykorzystanych przez autorów artykułu danych

zostały stworzone dodatkowe indeksy (w wykorzystanej bazie danych jeden typ obiektu może mieć wyłącznie jeden zdefiniowany indeks), które zostały założone na kod eksperymentu w każdym z podanych modelów danych (patrz rysunek 6). Kolumna ta została wybrana z uwagi na fakt, że najczęściej stanowi ona cel klauzuli WHERE zapytań LINQ oraz cechuje się wysoką selektywnością i niską fragmentacją (kod eksperymentu był tworzony funkcją haszującą).

```
DocumentStore store = new DocumentStore()
              { Url = System.Configuration.ConfigurationManager.AppSettings["RavenDbUrl"] };
store.Initialize();
store.DefaultDatabase = "PublikacjaFinal";
store.DatabaseCommands.PutIndex("BigTableClassIndex",
               new IndexDefinitionBuilder<BigTableClass, BigTableClass>()
{
    Map = exs => from ex in exs
                 select new
            {
                ExperimentCode = ex.ExperimentCode
            },

    Stores = { { x => x.ExperimentCode, FieldStorage.Yes } }
});
```

Rys. 6. Przykładowa definicja indeksu w bazie danych Raven DB
Fig. 6. An example of index definition in Raven DB database

3. Eksperyment

Tabela 1
Charakterystyka eksperymentów biologicznych, z których pochodzą dane z mikromacierzy

Nazwa eksperymentu	Liczba mikromacierzy HG-U133A	Właściciel danych
Gen receptora estrogenowego w skoliozie idiopatycznej u chorych leczonych operacyjne metodą C-D	14	Katedra Biologii Molekularnej ŚUM
Rola podjednostek receptorowych IFN gamma oraz genów szlaku apoptotycznego, proliferacyjnego i zapalnego w polipach jelita grubego	6	Katedra Biologii Molekularnej ŚUM
Prognozowanie ryzyka progresji zmian śródbłonkowych i raka szyjki macicy na podstawie analizy profilu ekspresji genów angiogennych i limfoangiogennych z zastosowaniem techniki QRT-PCR i mikomacierzy oligonukleotydowych	6	Katedra Biologii Molekularnej ŚUM
Wykorzystanie małych interferujących RNA (shRNA) do hamowania ekspresji genów szlaku PI3K/AKT	11	Katedra Biologii Molekularnej ŚUM
Profilowanie transkrypcji pacjentów z chorobą Parkinsona	45	ArrayExpress. E-GEOD-6613

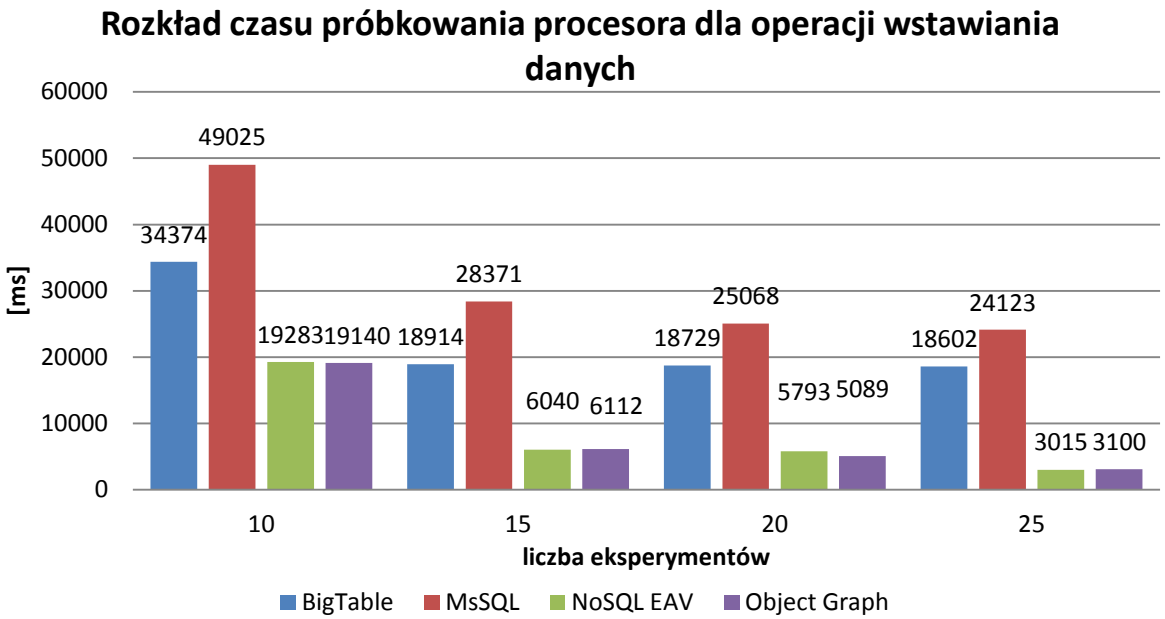
Celem eksperymentu było porównanie wydajności czterech, opisanych w rozdziale 2, metod gromadzenia danych. Na potrzeby eksperymentu wykorzystano dane pochodzące z pięciu eksperymentów mikromacierzowych (patrz tabela 1).

Dane z eksperymentów były importowane z plików do bazy danych – przy wykorzystaniu stworzonego na potrzeby publikacji autorskiego programu – początkowo w liczbie 10 eksperymentów, a następnie od 10 do 30 eksperymentów z krokiem 5 eksperymentów dla każdej z czterech metod przechowywania danych. Każdorazowo podczas importowania danych do baz (dane były wstawiane w sposób synchroniczny) mierzony był czas procesora (ang. *CPU Sampling*) przypadający na odpowiednią funkcję odpowiedzialną za wstawianie danych dla konkretnego modelu danych. Po wgraniu zestawu 10, 15, 20, 25 oraz 30 eksperymentów dla każdej z metod przechowania danych były wykonywane dwa testy efektywności dostępu do zgromadzonych danych. Pierwszy test polegał na pobraniu wszystkich danych z losowo wybranego eksperymentu, drugi natomiast polegał na pobraniu losowego ProbesetID dla uprzednio wylosowanego elementu (proces losowania nie był częścią funkcji podlegającej ocenie, więc nie wpłynął na końcowe wyniki). Zgromadzone wyniki zaprezentowano w kolejnym rozdziale.

4. Wyniki

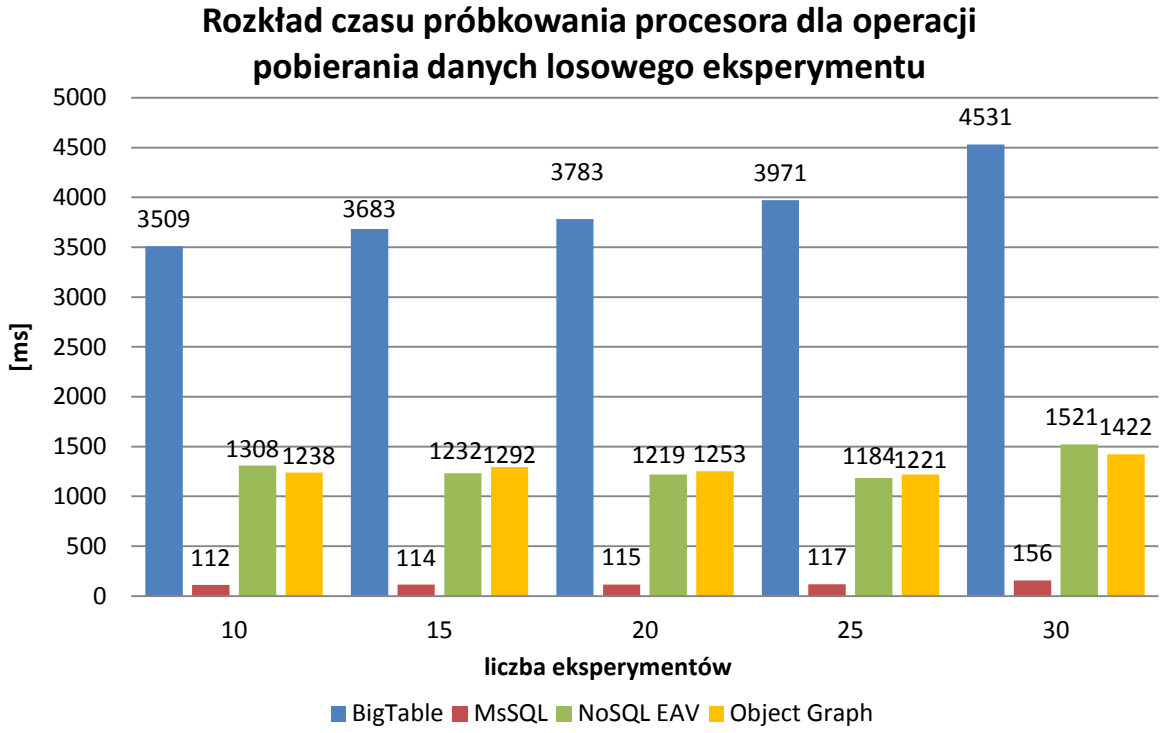
Po wykonaniu opisanego w rozdziale 3 eksperymentu dla zestawu 10, 15, 20, 25 oraz 30 doświadczeń mikromacierzowych dla każdej z czterech metod gromadzenia danych otrzymano wyniki, które zostały poddane wizualizacji i analizie.

Na rysunku 7 przedstawione zostały rozkłady czasów procesora dla każdej z funkcji realizującej funkcjonalność wstawiania do bazy danych dla każdej z czterech metod wykorzystanych w eksperymencie. Z uwagi na wykorzystanie dwóch różnych klas baz danych (baz relacyjnej oraz nierelacyjnej) czasy importu danych znacznie się różnią. Dodatkowo w obrębie bazy nierelacyjnej metoda *BigTable* znacznie różni się od pozostałych dwóch metod, co z pewnością związane jest z liczbą komend przesyłanych do serwera bazodanowego. Ponadto na podstawie zaprezentowanego wykresu można zauważyć, iż pozostałe dwie metody, *NoSQL EAV* oraz *Object Graph*, pod względem czasu importu zestawu danych pozostają na podobnym poziomie czasów procesora. Warto również zauważyć, że pomimo wzrostu liczby eksperymentów czas importu danych sukcesywnie spadał dla wszystkich czterech metod.



Rys. 7. Rozkład czasu procesora dla operacji wstawiania danych do baz danych przy różnej liczbie eksperymentów

Fig. 7. The CPU time distribution for an operation of inserting data into databases with various number of experiments



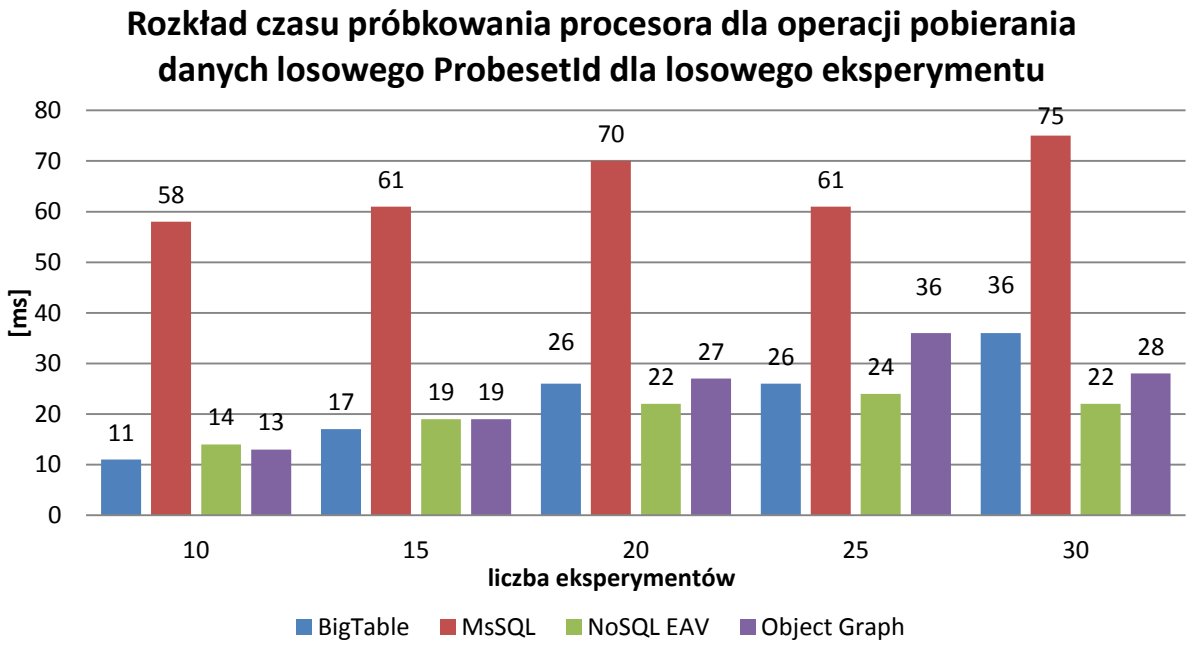
Rys. 8. Rozkład czasu procesora dla operacji pobierania danych losowego eksperymentu dla różnej liczby eksperymentów

Fig. 8. The CPU time distribution for data retrieval for the random experiment from various number of experiments

Wyniki kolejnego etapu eksperymentu, polegającego na pobraniu pełnego zestawu danych dla losowego eksperymentu, zostały zaprezentowane na rysunku 8. Z przedstawionego

na nim wykresu można wywnioskować, że w podejściu relacyjnym model EAV cechuje bardzo wysoka wydajność w porównaniu do modelu nierelacyjnego. Dodatkowo można zauważyć ciągły przyrost czasu pobierania pełnych danych eksperymentu dla modelu *BigTable*. Jednak warto zauważyć, że wydajność pozostałych dwóch modeli nierelacyjnych (*NoSQL EAV* oraz *Object Gaph*) nie zmienia się w sposób znaczący wraz z przyrostem liczby eksperymentów, ale nadal pozostaje o wiele gorsza w stosunku do modelu relacyjnego.

Ostatnim etapem przeprowadzonego eksperymentu badania wydajności było pobranie wartości pojedynczego *ProbesetId* dla wszystkich próbek w obrębie jednego, losowo wybranego eksperymentu. Zebrane wyniki zostały zaprezentowane na rysunku 9. Na jego podstawie można wywnioskować, że model wykorzystujący relacyjny silnik bazy danych jest znacznie mniej wydajny niż silnik NoSQL. Dodatkowo warto zauważyć, że pomimo przyrostu liczby eksperymentów czas wyszukania poszczególnego *ProbesetId* nie ulega zmianie w znaczący sposób.



Rys. 9. Rozkład czasu procesora dla operacji pobrania danych losowego ProbesetID dla losowego eksperymentu przy różnej liczbie eksperymentów

Fig. 9. The CPU time distribution for data retrieval operations for random ProbesetID for random experiment from various number of experiments

5. Wnioski

Po przeprowadzeniu eksperymentu będącego tematem niniejszego artykułu można stwierdzić, że wykorzystanie modelu nierelacyjnej bazy danych (NoSQL) w procesie gromadzenia danych mikromacierzowych jest jak najbardziej uzasadnione. Przewaga koncepcji bazy nierelacyjnej nad koncepcją relacyjną pojawia się już w momencie umieszczania da-

nych w bazie danych. Można to uzasadnić tym, że w przypadku silnika Microsoft SQL Server 2012 konieczne jest wstawianie dużej liczby rekordów do trzech tabel, podczas gdy w przypadku bazy RavenDB operacja wstawienia poszczególnego eksperymentu może być traktowana jako operacja atomowa, w podejściu *Object Graph* dodawany jest bowiem tak naprawdę tylko jeden nowy obiekt (jednak jest on dużego rozmiaru). Wprawdzie wykorzystanie serializacji jest operacją kosztowną, jednak, jak wykazano w eksperymencie (patrz rysunek 7), jest o wiele bardziej efektywne niż wstawianie wielu tysięcy rekordów. Dodatkowo w kwestii importu danych podejście *Object Graph* jest najbardziej wydajne spośród wszystkich trzech modeli nierelacyjnych – pozostałe dwa podejścia dla każdego eksperymentu wstawiają wiele osobnych obiektów danych. Niestety w przypadku pobierania uprzednio wstawionego eksperymentu mikromacierzowego model nierelacyjny wydaje się znacznie mniej efektywny niż klasyczne podejście relacyjne, wsparte wykorzystaniem wzorca EAV (patrz rysunek 8). Powodem tego stanu rzeczy jest odpowiednie, fizyczne rozłożenie danych na dysku serwera. Jako że dane zgromadzone są w uporządkowanej strukturze – zgodnie z kluczem podstawowym składającym się z *SampleID* oraz *ProbesetID* – ich odczyt jest zoptymalizowany pod kątem wydajności. Nie zachodzi również konieczność wykonywania operacji takich jak skanowanie tabeli czy też skanowanie indeksu, ponieważ ze względu na opisany klucz podstawowy możliwe jest wykonanie operacji przeszukania indeksu i tym samym pobranie całego zakresu danych bez konieczności wykonywania nadmiarowych przesunięć głowic dysku.

Z kolei model nierelacyjny w chwili obecnej jest bardziej przystosowany do pobierania pojedynczej wartości z całego zbioru zgromadzonych uprzednio obiektów, co też uwidoczniono na rysunku 9, zarazem będącym dowodem wyższości modelu nierelacyjnego nad klasycznym podejściem relacyjnym. O ile podczas importu oraz pobierania danych całego eksperymentu model *BigTable* nie cechował się wysoką wydajnością, o tyle w przypadku pobierania losowego *ProbesetID* okazał się podejściem najbardziej efektywnym, czego nie można powiedzieć o podejściu EAV w modelu relacyjnym.

W eksperymencie wykazano, że model *Object Graph* w przypadku nierelacyjnej bazy danych Raven DB wykazuje najlepszą wydajność średnią. Wprawdzie jego wykorzystanie podczas pobierania danych całego eksperymentu nie cechowało się najkrótszym czasem próbkowania procesora, jednak na tle wszystkich trzech przeprowadzonych w eksperymencie analiz wypada on najlepiej.

BIBLIOGRAFIA

1. Gramacki A., Gramacki J.: Modelowanie typu Entity-Attribute-Value w bazach danych. PAK, Vol. 53, nr 5/2007.

2. Hong-Hai D., Toralf K., Erhard R.: Comparative Evaluation of Microarray-based Gene Expression Databases. Conference paper. BTW 2003, Datenbanksysteme für Business, Technologie und Web, Tagungsband der 10. BTW-Konferenz, Leipzig 2003.
3. Tsoi L. C., Zheng W. J.: A Method of Microarray Data Storage Using Array Data Type. Comput. Biol. Chem., 2007.
4. Masys D.: Database designs for microarray data. Pharmacogenomics J., Vol. 1(4), 2001, s. 232÷233.
5. Jarząb B., Gubała E., Lange D.: Mikromacierze DNA i profil ekspresji genów raka brodawkowatego tarczycy. IV Konferencja Sekcji Endokrynologii Molekularnej PTE, Poznań 2004.
6. Chang F., Dean J., Ghemawat S., Hsieh W. C., Wallach D. A., Burrows M., Chandra T., Fikes A., Gruber R. E.: Bigtable: A Distributed Storage System for Structured Data. Google, Inc., 2006.
7. Lakshman A., Malik P.: Cassandra – A Decentralized Structured Storage System. Facebook 2009.
8. Peng D., Dabek F.: Large-scale Incremental Processing Using Distributed Transactions and Notifications. Google, Inc.
9. Scholz J.: Coping with Dynamic, Unstructured Data Sets – NoSQL a Buzzword or a Savior? TU Vienn.
10. Neubauer P.: Graph Databases, NoSQL and Neo4j. 2010.
11. Crockford D.: The application/json Media Type for JavaScript Object Notation (JSON). 2006.

Wpłynęło do Redakcji 15 stycznia 2012 r.

Abstract

The main goal of this article was to select the most efficient method of storing data from DNA microarray experiment in database system. We compared the performance of relational and non-relational database management systems from the possibility of using them in storing microarray data. In comparative analysis 3 methods of storing data were based on non-relational model (noSQL) and the 1 method of storing microarray data was according to the entity-attribute-value relational model. Implementation of relational database was performed in Microsoft SQL Server 2012 system. Raven DB system was used to implementation of non-relational database. Evaluation of efficiency of each of the method was based on CPU sampling using Microsoft Visual Studio Profiler.

We proved that storing the data from microarray experiment in non-relational is reasonable. Tests showed that the most efficient method of storing microarray data is the *Object Graph* based on non-relational Raven DB database system.

Adresy

Damian ZAPART: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 39, 41-200 Sosnowiec, Polska, damian.zapart@gmail.com.

Tomasz WALLER: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 39, 41-200 Sosnowiec, Polska, tomek@bioinformatyka.com.pl.

Magdalena TKACZ: Uniwersytet Śląski, Instytut Informatyki, ul. Będzińska 39, 41-200 Sosnowiec, Polska. magdalena.tkacz@us.edu.pl.