STUDIA INFORMATICA 2010
Volume 31 Number 2A (89)

Marcin GORAWSKI, Rafał WARDAS
Politechnika Śląska, Instytut Informatyki

RÓWNOWAŻĄCY OBCIĄŻENIA SYSTEM ETL, BAZUJĄCY NA MASZYNIE UCZĄCEJ

Streszczenie. Użytkownicy hurtowni danych wymagają zazwyczaj zarówno krótkiego czasu odpowiedzi na zapytania, jak i wysokiego poziomu świeżości pobieranych danych. Przedstawiony system LEMAT zarządzający procesem ekstrakcji danych ETL opiera się na koncepcji adaptacyjnego równoważenia obciążenia operacji zapytań i aktualizacji zgodnie ze zmieniającymi się potrzebami użytkownika. System LEMAT używa autorskiego algorytmu równoważenia obciążenia z użyciem maszyny uczącej z zaawansowanym klasyfikatorem zapytań LMWB. Zaprezentowana została również metoda adaptacji systemu LEMAT na podstawie zbieranych statystyk o zmieniających się warunkach pracy oraz jego reakcja na przeciążenia.

Słowa kluczowe: ETL, SVM, klasyfikacja, adaptacja, LMWB

THE WORKLOAD BALANCING ETL SYSTEM BASING ON A LEARNING MACHINE

Abstract. Data warehouses users usually expects both: short response time and high level of data "freshness". The LEMAT presented as the ETL process manager bases on a concept of a adaptive load balancing of queries and actualizations according to user changing needs. The LEMAT system uses new workload balancing algorithm that uses LMWB (*Learning Machine-based Workload Balancing*) with the advanced query classifier SVM (*Support Vector Machine*). Moreover the method of a LEMAT system adaptation is presented. This method bases on collection of changing work conditions characteristics and reactions to congestions.

Keywords: ETL, SVM, classification, adaptation, LMWB

1. Wprowadzenie

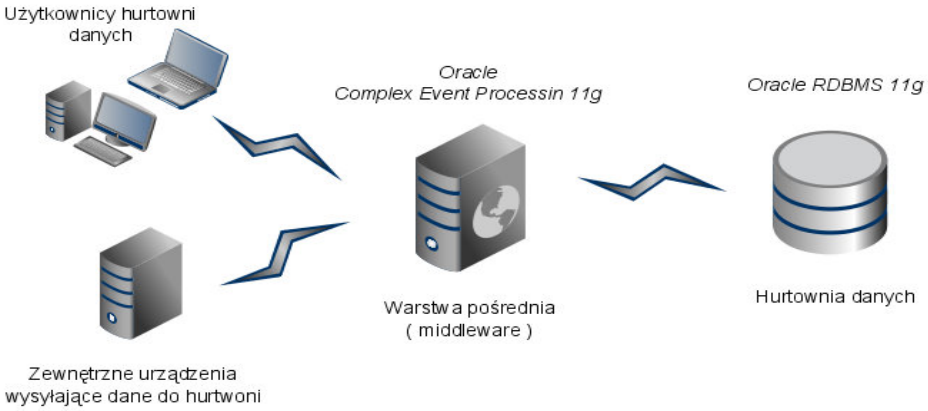
Jednym z istotniejszych problemów systemów hurtowni danych jest wymóg częstego aktualizowania danych w procesie ekstrakcji danych ETL (ang. *Extraction, Transformation, Load*). Proces ETL obejmuje ekstrakcję, transformację oraz ładowanie danych do hurtowni danych DW (ang. *Data Warehouse*). Proces ETL możne być budowany na podstawie uniwersalnych systemów ETL lub, co jest częstszą praktyką specjalizowanych aplikacji [1, 2, 4, 17, 18, 21, 23, 25, 26, 27], [5 – 14]^{w}. Taki klasyczny model procesu ekstrakcji danych jest autonomiczny i niepowiązany z procesem generowania zapytań przez użytkowników DW.

Rozbudowa procesu ETL tak, aby każde zapytanie użytkownika DW było powiązane z danymi oczekującymi na załadowanie, znacznie poszerza możliwości efektywnego wykorzystania zasobów takiego systemu. Ponadto, jeżeli założymy, że wymiar czasu oraz poziom jego agregacji ma kluczowe znaczenie przy zwracanym wyniku, to możemy określić, czy zapytanie wymaga bardzo aktualnych (świeżych) danych. W tym celu zaproponowane zostanie rozwiązanie, które poddaje analizie wszystkie zapytania oraz aktualizacje. Możliwe jest bardzo precyzyjne określenie przedziału czasu, w którym każde zapytanie zostanie wykonane w zależności od m.in. rozkładu danych, czy też różnych warunków filtracji. Proponowane rozwiązanie umożliwia zmniejszenie kosztów budowy i eksploatacji systemu DW odpornego na krótkotrwałe obciążenia jego zasobów. Posiadając wiedzę zawartą w zapytaniach możliwe jest wymuszanie procesu aktualizowania danych tylko dla zapytań szczegółowych. Dodatkowo, posiadając informację o tym, w jakim przedziale czasu wykona się zapytanie, można znacznie zwiększyć efektywność systemu DW (wykonujące się szybciej zapytania, krócej oczekują w kolejce) i jego odporność na krótkotrwałe obciążenia zasobów.

2. Schemat platformy LEMAT

Prezentowany autorski **równoważący obciążenia system ETL, bazujący na maszynie uczącej** (w skrócie: system LEMAT), zarządza w systemie DW procesem ETL za pomocą maszyny uczącej. W systemie LEMAT, z jednej strony użytkownicy DW z dużą intensywnością odpytują jej zawartość, a z drugiej do systemu napływają nieprzerwanie nowe dane, które mają się jak najszybciej znaleźć w hurtowni, a przy tym nie powodować jej przeciążenia. Wszystkim zarządza aplikacja zdarzeniowa w warstwie pośredniej, która nadzoruje obydwa procesy. W systemie LEMAT umieszczone są 2 grupy źródeł generujących dane, tj. użytkownicy DW oraz zewnętrzne urządzenia, które z innych systemów przesyłają nowe dane.

Schemat fizycznej platformy systemu LEMAT przedstawia rysunek 1.



Rys. 1. Schemat fizycznej platformy systemu LEMAT
Fig. 1. Physical schema of LEMAT system platform

Każda z tych grup jest reprezentowana przez specjalnie rozbudowaną kolejkę, która znajduje się we wnętrzu serwera aplikacji warstwy pośredniej (ang. *Middleware*) [19]. W warstwie pośredniej zawarta jest pełna wiedza o schemacie hurtowni danych typu ROLAP (ang. *Relational On-line Analytical Processing*) (rys. 1). W tym typie architektury bazę danych DW tworzy relacyjna baza danych o schemacie zorientowanym. Jeżeli podczas przetwarzania zapytań odebranych w serwerze aplikacji dokonuje się ich analizy pod kątem odwołań do poszczególnych fragmentów schematu danych DW, to obydwie kolejki przekształcają się w grupę kolejek o rozmiarze określonym przez liczbę tabel w docelowym schemacie DW. Elementy w takiej strukturze dla każdej z kolejek mogą być ze sobą powiązane dla zapytań odpytujących kilka tabel jednocześnie. Proponowana architektura systemu LEMAT w dużym stopniu opiera się idei dwupoziomowego algorytmu planowania WINE (ang. *Workload Balancing by Slection*) [24]. Algorytm WINE równoważy i nadaje priorytet zapytaniom oraz aktualizacjom, pozwalając maksymalnie zaspokoić oczekiwania użytkownika. Poddaje on również analizie zapytania, przyjmując nazewnictwo „partycji" dla każdej z tabel ze schematu DW w architekturze ROLAP. Koncepcja ta została w dużym stopniu zachowana. Metoda ustalania priorytetu i analizy zapytań jest o wiele bardziej skomplikowana i zdecydowanie różni się od algorytmu WINE. Wyeliminowana została konieczność definiowania oczekiwanego poziomu jakości danych przez użytkownika. Ponadto system LEMAT z autorskim algorytmem równoważenia obciążenia z użyciem maszyny uczącej z zaawansowanym klasyfikatorem zapytań (ang. *Learning Machine-based Workload Balancing,* LMWB). Klasyfikator opiera swoje działanie na bibliotece *wlsvm* (z pakietu oprogramowania do eksploracji danych o nazwie WEKA [22]) implementującej klasyfikator SVM (ang. *Support Vector Machine*). Powodem, dla którego został wybrany ten typ klasyfikatora, jest jego funkcjonalność. Potrafi on przetwarzać typy wyliczeniowe oraz liczbowe. Spośród kilkudziesięciu klasyfikatorów dostępnych w pakiecie oprogramowania do eksploracji danych WEKA tylko część spełniała

kryteria przetwarzania tak rozbudowanych danych wejściowych. W fazie testów okazało się, że SVM wygenerował model z największym współczynnikiem poprawnie sklasyfikowanych elementów. Testy systemu LEMAT zostały przeprowadzone w zbliżony sposób jak dla algorytmu WINE [24].

3. Analiza zapytań

Zarówno w przypadku aktualizacji danych, jak i zapytań użytkowników zapytania są zapisane w postaci języka DML (ang. *Data Manipulation Language*). W przypadku danych generowanych przez użytkownika mamy do czynienia jedynie z projekcją danych zaczynających się od SELECT, natomiast dla aktualizacji danych to operacje INSERT. Wykorzystując wzorce wyrażeń regularnych można uzyskać informacje o tym, jakich fragmentów schematu DW te zapytania dotyczą. Na tym etapie analiza aktualizacji dobiega końca, jednak w przypadku zapytań użytkownika potrzebna o wiele więcej informacji. Proces analizy zapytania użytkownika rozpoczyna się od wyliczenia jego funkcji skrótu, a następnie określeniu zdefiniowanego na potrzeby systemu parametru TGL (ang. *Time Granulation Level*), oznaczającego poziom granulacji czasu. W praktyce oznacza to, że występują dwa typy wyników szczegółowe i zagregowane. Szczegółowe wyniki, jeżeli korzystamy z dokładności na poziomie rekordu w tabeli ROLAP, a zagregowane wyniki, jeżeli używamy funkcji agregującej. Na potrzeby przykładu przyjmijmy założenia, że najmniejszą jednostką w tabeli wymiaru czasu jest pole typu DATE, czyli określające dzień. Łącząc w zapytaniu SQL powiązane tabele nie uzyskamy bardziej szczegółowego raportu niż dzienny, więc to jest wynik, który nie wymaga agregacji i jest na najwyższym poziomie szczegółowości. W testowanym systemie LEMAT najniższą jednostką wymiaru czasu jest dzień, tak więc posortowany według wagi poziom granulacji czasu wraz z wagą każdego z poziomów prezentuje się następująco:

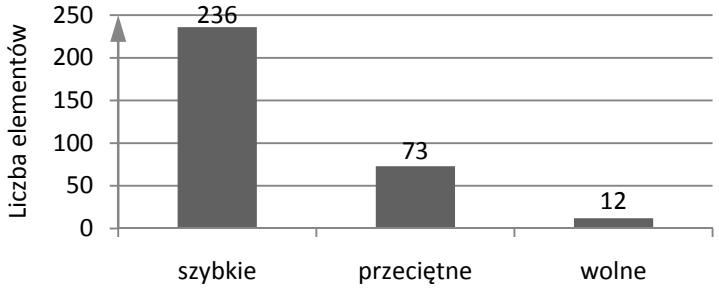
Tabela 1

Tabela wartości TGL

Jednostka czasu	**TGL**
Dzień	0
Tydzień	1
Miesiąc	2
Kwartał	3
Rok	4

Następnym etapem analizy jest określenie słów kluczowych języka SQL, które zostały użyte w zapytaniu. Ze względu na to, że duża część zbioru testowego używa dedykowanych konstrukcji, wyszukiwane są następujące słowa kluczowe: GROUP, RANK, ROLLUP, CUBE, GROUPING, PARTITION, ORDER.

Kolejnym etapem analizy jest szacowanie przedziału, w którym wykona się zapytanie. Zgodnie ze specyfikacją WEKA wymagane jest przekształcenie tych informacji o zapytaniu do formatu *.arff*, przeznaczonego dla klasyfikatora SVM (ang. *Suppor Vector Machine*). W tym momencie pojawiają się dwie istotne kwestie dotyczące funkcjonowania systemu LEMAT oraz zasady działania klasyfikatora. Po pierwsze klasyfikator musi najpierw zostać wytrenowany, by mógł pełnić swoją funkcję, stąd bezpośrednio po starcie systemu tworzony jest nowy model klasyfikacji na podstawie załączonego zbioru danych wzbogaconego o czas wykonania zapytań. Ponadto, zgodnie z założeniami, nie szacuje się czasu wykonania lecz przedział czasu, w jakim wykona się zapytanie. Zatem chcemy uzyskać informację, czy zapytanie wykona się (zrealizowane zostanie) wolno, z umiarkowaną prędkością czy też szybko. Jest to o tyle trudne, że posiadamy jedynie czas wykonania dla określonej grupy zapytań i nie jesteśmy w stanie określić, co to znaczy szybko czy też wolno. Żeby przekształcić taki zbiór na pozwalający klasyfikować czas wykonania do określonych klas należy założyć na kolumnę tabeli filtr dyskretyzacji, który tworzy histogram generujący trzy przedziały wartości uwzględniające rozkład wartości.



Rys. 2. Histogram czasu wykonania zapytań w systemie LEMAT
Fig. 2. Query execution time histogram from the training set in LEMAT system

Nietrudno zauważyć, że w zbiorze uczącym przeważająca część zapytań to zapytania szybkie. Wykorzystanie filtru dyskretyzacji dla trzech przedziałów powoduje utworzenie dwóch punktów, oddzielających przedziały czasowe. Pierwszy punkt wyznacza granicę na 1151 ms, drugi na 2302, co oznacza, że zapytania szybkie wykonują się mniej niż 1151 ms, zapytania przeciętne od 1151 do 2301, a zapytania wolne zaczynają się od 2302 ms. Posługując się przykładem, poniżej zaprezentowany został fragment zbioru uczącego we wcześniej wspomnianym formacie *.arff* :

```
01 @relation TrainingSet
01
02 @attribute TABLE_CHANNELS {0,1}
03 @attribute TABLE_COSTS {0,1}
04 @attribute TABLE_COUNTRIES {0,1}
05 @attribute TABLE_CUSTOMERS {0,1}
06 @attribute TABLE_PRODUCTS {0,1}
07 @attribute TABLE_PROMOTIONS {0,1}
08 @attribute TABLE_SALES {0,1}
```

```
@attribute TABLE_SALTES_TRANSACTIONS_EXT {0,1}
@attribute TABLE_SUPPLEMENTARY_DEMO {0,1}
@attribute TABLE_TIMES {0,1}
@attribute KEYWORD_GROUP {0,1}
@attribute KEYWORD_RANK {0,1}
@attribute KEYWORD_ROLLUP {0,1}
@attribute KEYWORD_CUBE {0,1}
@attribute KEYWORD_GROUPING {0,1}
@attribute KEYWORD_PARTITION {0,1}
@attribute KEYWORD_ORDER {0,1}
@attribute TIME_GRANULATION_LEVEL numeric
@attribute TABLES_COUNTER numeric
@attribute QUERY_LENGTH numeric
@attribute EXECUTION_TIME numeric

@data
0,0,1,1,0,0,1,0,0,0,1,1,0,0,0,0,1,4,2,277,1625
0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,1,3,4,518,2562
0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,4,1,137,128
```

W zaprezentowanym nagłówku bardzo wyraźnie występują dwie grupy atrybutów, jedna z prefiksem TABLE, druga z KEYWORD. Określają one, czy dane słowo kluczowe bądź tabela występuje w zapytaniu SQL. Nie bez znaczenia jest typ tych atrybutów, który nazywany jest typem nominalnym (odpowiada typowi wyliczeniowemu). Pozostałe kolumny opisują parametr poziomu granulacji czasu (TGL), którego wartość ma zakres od 0 do 4 i odpowiada jednostkom czasu od dnia do roku uporządkowanym rosnąco. Następne parametry są odpowiedzialne kolejno za zliczanie liczby tabel, długość zapytania SQL oraz czas wykonania.

Po założeniu filtru dyskretyzacji zachodzą pewne zmiany w zbiorze uczącym dotyczące czasu wykonania:

```
...
@attribute QUERY_LENGTH numeric
@attribute EXECUTION_TIME {'\'(-inf-1151]\'','\'(1151-2302]\'','\'(2302-
inf)\''}

@data
0,0,1,1,0,0,1,0,0,0,1,1,0,0,0,0,1,4,2,277,'\'(1151-2302]\''
0,0,1,1,1,0,1,0,0,1,1,0,0,0,1,0,1,3,4,518,'\'(2302-inf)\''
0,0,0,0,1,0,1,0,0,0,1,0,0,0,0,0,1,4,1,137,'\'(-inf-1151]\''
```

Zgodnie z opisanym wcześniej mechanizmem każdy z rekordów został przyporządkowany do jednej z klas szybkości wykonywania się zapytania (typ kolumny uległ zmianie). Każde z nowo pojawiających się zapytań użytkownika w systemie LEMAT przechodzi taki proces i jest transformowane do dwóch obiektów typu *Instance* (będącego odpowiednikiem powyższych przykładów, różniących się czasem wykonania). Instancja z czasem wykonania w postaci liczby rzeczywistej zostaje skopiowana do bufora cyklicznego i oczekuje na wykonanie, jest identyfikowana na podstawie wcześniej wyliczonej funkcji skrótu z zapytania SQL. Warto zapamiętać, że przechowywane instancje w buforze zostaną później użyte do ponownego trenowania klasyfikatora, jeżeli jego skuteczność będzie zbyt niska. Druga in-

stancja z czasem wykonania zdefiniowanym pod postacią przedziałów odpytuje klasyfikator i zwraca klasę szybkości w postaci liczby od 0 do 2 (dla ustalonych wcześniej trzech przedziałów). Tak otrzymana wartość, łącznie z zapytaniem, informacjami o użytych tabelach oraz parametrem TGL z pominięciem słów kluczowych jest przesyłana do kolejki. Jeżeli takie zapytanie już istnieje w kolejce, to jest ono wykrywane na podstawie funkcji skrótu i nowe zapytanie jest pomijane. W zamian za to wzorzec ma zwiększany parametr informujący o tym zdarzeniu, a stempel czasowy zapytania przyjmuje wartość usuwanego duplikatu. Analogiczna kolejka zostaje utworzona dla aktualizacji, z tym że tutaj wysyłane są jedynie grupy zapytań SQL z informacją, które tabele podlegają aktualizacji (bez wykrywania duplikatów).

4. Algorytm LMWB

Nasz algorytm równoważenia obciążenia z użyciem maszyny uczącej z zaawansowanym klasyfikatorem zapytań LMWB w systemie LEMAT kolejkuje zapytania wg priorytetu. Algorytm LMWB wylicza priorytety dla każdego zapytania w kolejce oraz ustanawia dla nich warunkową i selektywną aktualizację. Priorytet określa wzór:

$$priority = \frac{svmQoS + TGL + 1}{duplicatesCtr},$$

gdzie: *svmQoS* – wynik działania klasyfikatora może posiadać wartości od 0 do 2, *TGL* – poziom granulacji czasu, jego przedział wartości to od 0 do 4, *duplicatesCtr* – liczba duplikatów zapytania w kolejce.

Następnie wybierane jest najstarsze zapytanie użytkownika o najmniejszej wartości priorytetu. Jeżeli posiada ono poziom granulacji równy 0 lub 1, czyli odpowiada raportowi dziennemu bądź tygodniowemu, to automatycznie przed jego obsłużeniem z kolejki aktualizacji pobierane są wszystkie zapytania, które mają choć jedną tabelę wspólną z przetwarzanym zapytaniem. W systemie LEMAT mechanizm ten zaimplementowano za pomocą maski bitowej o rozmiarze określonym przez liczbę elementów w schemacie DW (co znacznie ułatwia wyszukiwanie). Tak wybrane aktualizacje są przesyłane w paczkach o rozmiarze nie przekraczającym 30 krotek do DW. Żeby ładowanie danych przebiegało w ten sposób, wymagane jest wyłączenie mechanizmu sprawdzania integralności bazy DW. Natomiast przy większej liczbie schematów danych (kostek) i bardziej zróżnicowanej strukturze danych istnieje możliwość przebudowy systemu LEMAT bez potrzeby wyłączania tego mechanizmu. Po zakończeniu procesu ładowania danych następuje wykonanie zapytania użytkownika, a w kolejnym etapie zapamiętanie czasu jego wykonania, który wraz z funkcją skrótu jest przesyłany do statystyk działania klasyfikatora SVM (ang. *Support Vector Machine*). Statyst-

ki oceniają skuteczność działania klasyfikatora i na ich podstawie podejmowana jest decyzja o ponownym uczeniu klasyfikatora. Czasy te (wyrażone w milisekundach) są porównywane z przedziałami czasu ustalonymi po wygenerowaniu modelu i na tej podstawie określana jest rzeczywista klasa szybkości wykonania zapytania. Jeżeli pokrywają się z tymi z bufora cyklicznego, wtedy zwiększane są dwa liczniki odpowiedzialne za zliczanie wszystkich poprawnie sklasyfikowanych elementów. W przeciwnym przypadku, jeżeli przedziały się nie zgadzają, wtedy zwiększany jest jedynie licznik wszystkich klasyfikacji. W kolejnym kroku sprawdzana jest liczba wszystkich klasyfikacji i skuteczność - jeżeli okaże się, że zostało wykonane co najmniej 300 prób, a skuteczność klasyfikatora spadła poniżej 80%, wtedy następuje ponowne wytrenowanie klasyfikatora na podstawie ostatnich kilkuset zapytań. Na tym etapie algorytm LMWB kończy cykl i przystępuje do wyboru kolejnego zapytania użytkownika. Jeżeli jego TGL jest większe niż 1, to raport prezentuje dane z podsumowań miesięcznych, kwartalnych bądź rocznych. Wtedy uruchamiany jest generator losowy, który podejmuje decyzję, czy ładować porcję najstarszych danych z kolejki, nie zważając na maskę bitową. Prawdopodobieństwo załadowania danych wynosi odpowiednio:

Tabela 2

Tabela podziału czasu dla generatora losowego

Jednostka czasu	TGL	Podział roku	Zakres zmiennej losowej
Miesiąc	2	365/12	30
Kwartał	3	365/4	91
Rok	4	365/1	365

Wyeliminowano przypadki, kiedy aktualizacje dla rzadko odpytywanych tabel, które mają z reguły mały rozmiar, będą nienaturalnie długo oczekiwały na obsłużenie, jednocześnie obniżając szybkość wyszukiwania danych w kolejce.

5. Testy systemu

Testy zostały przeprowadzone na maszynie z dwurdzeniowym procesorem Intel Pentium T2300, taktowanej zegarem 1.66 GHz, 3GB pamięci RAM oraz dyskiem 5400 RPM na systemie operacyjnym Windows XP SP2. Powodem wykorzystania tylko jednej maszyny jest fakt, że najwolniejszym komponentem systemu jest DW, która nawet gdyby pracowała na bardzo wysokiej klasy sprzęcie, to nie byłaby w stanie dorównać czasowi przetwarzania zapytań w warstwie pośredniej. Docelowa DW bazuje na przykładowym schemacie danych SH (ang. *Sales History*) dostarczanym wraz z bazą danych Oracle 11g Enterprise Edition. Zarówno aplikacja zdarzeniowa, jak i obsługa procesu ETL zostały zaimplementowane w serwerze aplikacji Oracle WebLogic Event Server w wersji 11gR1 [3, 15, 19, 20]. Zachowano

powtarzalność pomiarów, a wyniki pomiarów są przechowywane w niezależnej bazie danych mySQL 5.1.

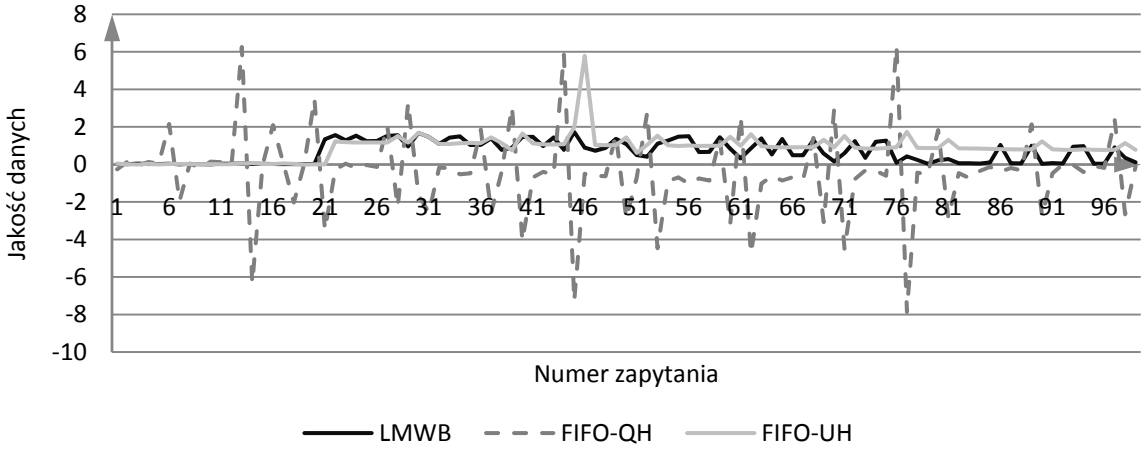
Celem pomiarów było porównanie trzech algorytmów, z których każdy w inny sposób zarządzał procesem ETL. System LEMAT opierający swoje działanie na algorytmie LMWB został porównany z systemami opierającymi się na dwóch typach algorytmów kolejek FIFO. Podobne testy zostały przeprowadzony dla algorytmu WINE. Pierwszym porównywanym algorytmem był FIFO-QH, który wykonuje wszystkie zapytania użytkowników przed aktualizacjami procesu ETL (nastawiony na obsługę użytkownika) i wyznacza idealny przypadek jakości usługi QoS (ang. *Quality of Service*). Drugim algorytmem był FIFO-UH, faworyzujący aktualizacje wyznaczone o współczynnik załadowania danych (ograniczony opóźnieniami wynikającymi z ich przetworzenia). Przeprowadzone pomiary mają na celu określenie jakości danych QoD (ang. *Quality of Data*).

$$QoD = \frac{\sum_{i=0}^{n} updatesCtr(i)}{\sum_{i=0}^{n} totalCtr(i)},$$

gdzie: *QoD* – jakość danych, *updateCtr* – liczba załadowanych aktualizacji od początku testu dla określonej tablicy, *totalCtr* – liczba wierszy w tabeli plus liczba wierszy załadowana od początku testu.

W przypadku pomiaru jakości usługi (QoS) mierzony jest czas oczekiwania zapytań w kolejce. Przeprowadzono szereg testów mających na celu sprawdzenie, jak zachowuje się system LEMAT dla zmiennych warunków obciążenia i niedoboru zasobów.

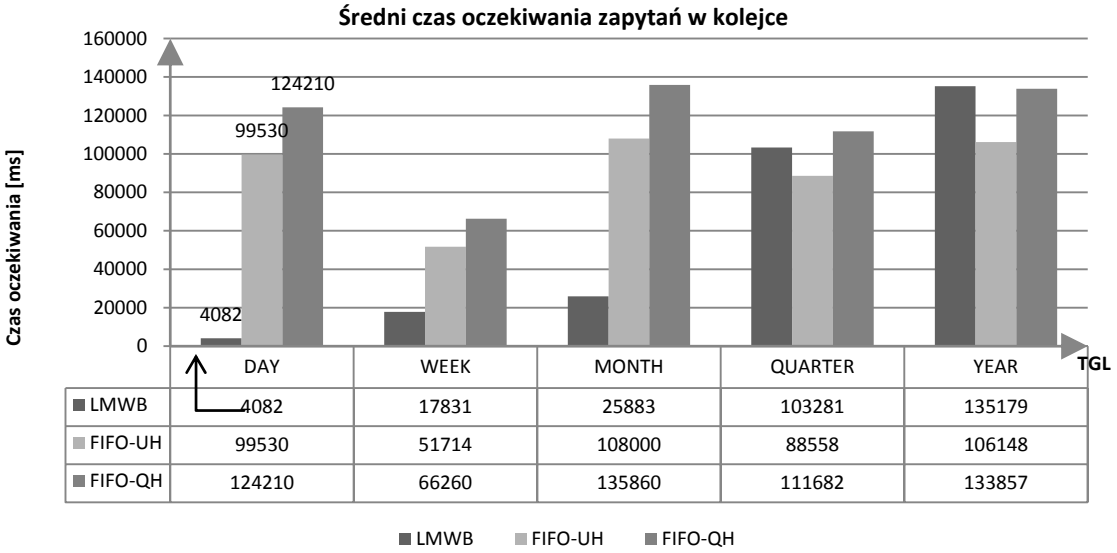
Wybrany fragment wyników z grupy testów prezentuje pomiar jakości danych (QoD) dla każdego z algorytmów. Test symulował przeciążenie DW (złożone zapytanie, które nie posiada indeksu i jest przetwarzane z użyciem powolnego dysku twardego) (rys. 3).



Rys. 3. Pomiar QoD w systemie LEMAT – stan przeciążenia zasobów DW
Fig. 3. Measurement QoD in LEMAT system – the state of filling of queues

Przeciążenie można zaobserwować w okolicy 20. zapytania, powoduje ono przepełnienie obydwu kolejek. Bezpośrednio po przetworzeniu zapytania wg algorytmu LMWB następuje ładowanie danych dla zapytania szczegółowego o niskim parametrze TGL. Okazuje się, że dzięki bardziej efektywnemu wykorzystaniu kolejki algorytm LMWB szybko uzyskuje podobny poziom jakości danych jak FIFO-UH, który faworyzuje aktualizacje. W dalszym etapie można zaobserwować jak w granicy 80. zapytania dla LMWB obniża się poziom jakości danych. Jest to zachowanie jak najbardziej prawidłowe i pokazuje, że przy uzyskaniu stabilności algorytm LMWB ładuje porcję selektywnie wybranych danych jedynie przy wystąpieniu zapytania o niskim TGL (by zapytania szczegółowe osiągały możliwie najwyższą jakość danych). Algorytm FIFO-QH przez cały czas faworyzuje zapytania i dopiero w okolicy 75. zapytania zapewnia ładowanie duzej porcji danych zaktualizowanych.

LMWB ma faworyzować zapytania szczegółowe, by zminimalizować liczbę danych, które trzeba załadować przed ich obsłużeniem. Dodatkowo faworyzowanie zapytań zduplikowanych bądź tych, których czas wykonania jest krótszy, wyraźnie przekłada się na uzyskanie lepszej wydajności systemu LEMAT (rys. 4).



Rys. 4. Pomiar QoS w systemie LEMAT
Fig. 4. Measurement QoS in LEMAT system

Zapytania z najbardziej szczegółowych raportów, mające niską wartość parametr TGL, były obsługiwane niemalże natychmiastowo, bez względu nad długość kolejki. Wraz ze wzrostem TGL czas oczekiwania zapytania w kolejce zaczyna rosnąć (QoS). Skuteczność klasyfikatora zapytań SVM podczas testów w przedziale pierwszych 150 zapytań wynosiła nie mniej niż 92%. Po przekroczeniu tej granicy i wygenerowaniu nowego modelu klasyfikacji, symulującego dostosowanie się klasyfikatora do warunków zmiennego obciążenia, sku-

teczność zmalała do 72% (duży wpływ mechanizmu *cache* RDBMS Oracle11g oraz zbyt mały zbiór zapytań).

6. Podsumowanie

Przedstawione testy wykazały, że DW z wbudowanym systemem LEMAT, tj. równoważącym obciążenia systemem ETL, bazujący na maszynie uczącej (realizującym algorytm LMWB), potrafi efektywnie równoważyć obciążenie. Uzyskuje również przewagę w porównaniu z klasycznymi algorytmami reprezentującymi dwa skrajne sposoby kolejkowania i ustalania priorytetu. Dzięki selektywnemu doborowi aktualizacji danych w systemie LEMAT system DW uzyskuje bardzo wysoką świeżość danych dla zapytań krytycznych. Ponadto, zautomatyzowanie procesu wyboru, czy dla danego zapytania użytkownika mają zostać załadowane aktualizacje, eliminuje przypadek, w którym użytkownik DW wymusi niepotrzebnie zbędną danych. Obiecujące okazuje się użycie klasyfikatora SVM do szacowania przedziału czasu wykonywania zapytania. Algorytm LMWB był testowany na skończonym i niezbyt dużym zbiorze danych, testy na danych rzeczywistych powinny pozytywnie wpłynąć na stabilność pracy klasyfikatora. W przypadku spadku wydajności wynikającej z funkcjonowania mechanizmu *cache* po kolejnym cyklu uczenia klasyfikator SVM powinien wygenerować właściwie funkcjonujący model. Żeby klasyfikator działał poprawnie, wymagane jest dodanie funkcjonalności umożliwiającej korektę oraz filtrowanie statystyk. W tak poprawionych warunkach prognozujemy, że skuteczność klasyfikatora ustabilizuje się na poziomie około 85%, co jest dobrym rezultatem biorąc pod uwagę fakt, że nawet błędna klasyfikacja nie oznacza błędnego sposobu kolejkowania zapytania.

BIBLIOGRAFIA

1. Bruckner R., List B., Schiefer J.: Striving towards Near Real-Time Data Integration for Data Warehouses. Data Warehousing and Knowledge Discovery, 4th International Conference, DaWaK'02, France, LNCS, Vol. 2454, 2002, s. 317÷326.
2. Bruckner R., Tjoa A.M.: Capturing Delays and Valid Times in Data Warehouses Towards Timely Consistent Analyses. Journal of Intelligent Information Systems, Vol. 19, No. 2, 2002, s. 169÷190.
3. Campos M., Milenova B.: Creation and Deployment of Data Mining-Based Intrusion Detection Systems in Oracle Database 10g. Oracle Data Mining Technologies 2005.

4. Galhardas H., Florescu D., Shasha D., Simon E.: Ajax. An Extensible Data CleaningTool. ACM SIGMOD, May 16-18, 2000, Dallas, Texas. ACM Press 2000, s. 590.
5. Gorawski M., Ciepluch M.: Przyrostowa ekstrakcja danych ETL(δ). Studia Informatica Vol. 27, No. 1, Wyd. Politechnika Śląska, Gliwice 2006, s. 27÷40.
6. Gorawski M., Jabłoński P.: Uniwersalne środowisko graficzne do modelowania procesów ekstrakcji i odtwarzania. Studia Informatica, Vol. 26, No. 3, Wyd. Politechniki Śląskiej, Gliwice 2005, s. 7÷28.
7. Gorawski M., Marks P.: Data Loading Based on UB-Tree Index Implemented in Design-Resume /JavaBeans Environment. Studia Informatica, Vol. 25, No. 1, 2004, s. 141÷153.
8. Gorawski M., Marks P.: Grouping and Joining Transformations in Data Extraction Process. AI Informatica, Annales Univ.Marii Curie-Skłodowska, Vol. 4. 2006, s. 135÷147.
9. Gorawski M., Piekarek M.: Rozproszony proces ekstrakcji danych z protokołem SimpleRMI. Red. S. Kozielski i in. Tom 2. Bazy danych. Modele, technologie, narzędzia Analiza danych i wybrane zastosowania. Wyd. Komunikacji i Łączności, 2005, s. 43÷50.
10. Gorawski M., Siódemak P.: Graficzne projektowanie aplikacji ETL. Studia Informatica, Vol. 24, No. 4(56), Wyd. Politechniki Śląskiej, Gliwice 2003, s. 345÷367.
11. Gorawski M.: Zaawansowane hurtownie danych. Gliwice: Wydaw. Politechniki Śląskiej, (Rozprawa habilitacyjna) 2009, s. 387.
12. Gorawski M.: 3 perspektywy procesu ekstrakcji danych. Red. J. S. Nowak, J. K. Grabara, Z. Szyjewski. Strategie informatyzacji i zarządzanie wiedzą. WNT, 2004, s. 295÷341.
13. Gorawski, M.: Charakterystyka procesu ekstrakcji danych. Studia Informatica, Vol. 24, No. 4(56), Wyd. Politechniki Śląskiej, Gliwice 2003, s. 211÷232.
14. Gorawski M.: Ekstrakcja i integracja danych w czasie rzeczywistym. Red. A. Kwiecień, P. Gaj. Współczesne problemy systemów czasu rzeczywistego. Wyd. Naukowo-Techniczne, Warszawa 2004, s. 435÷445.
15. Huiming Qu., Labrinidis A.: Preference-Aware Query and Update Scheduling in Web-database. Data Engineering, ICDE 2007.
16. Microsoft, Services Managing Data with Data Transformation. http://www.microsoft.com/technet/community/events/sql2000/tnt1-78.mspx.
17. Oracle Data Warehousing, OLAP Option to Oracle Database 11*g*, http://www.oracle.com/solutions/business_intelligence/dw_ home.html. 2009.
18. Oracle®, Complex Event Processing in the Real World September 2007
19. Oracle®CEP , IDE Dev. Guide for Eclipse Release 11gR1 (11.1.1), E14301-01, 2009.
20. Rahm E., Hai Do H.: Data Cleaning: Problems and Current approches. Bulletin of the Technical Committee on Data Engineering, Vol. 23. 2000.

21. Remco R. Bouckaert, Eibe Frank, Mark Hall - WEKA Manual for Version 3-6-1, University of Waikato – 2009.
22. Schrefl M., Thalhammer T.: On Making Data Warehouses Active. 2nd International Conference Data Warehousing and Knowledge Discovery, DaWaK'00, September 4-6, 2000, London. Lecture Notes in Computer Science, Vol. 1874, Springer 2000, s. 34÷46.
23. Thiele M., Fischer U., Lehner W.: Partition-based Workload Scheduling in Living Data Warehouse Environments, DOLAP'07, Portugal, ACM 2007.
24. Vassiliadis P., Simitsis A., Georgantas P., Terrovitis M.: A Framework for the Design of ETL Scenarios. Advanced Information Systems Engineering, CAiSE'03, Lecture Notes in Computer Science, Vol. 2681, Springer 2003, s. 520÷535.
25. Vassiliadis P., Simitsis A., Skiadopoulos S.: Modeling ETL Activities asGraphs. Design and Management of Data Warehouses, DMDW'02, May 27, 2002, Toronto. CEUR-WS.org 2002, s. 52÷61.
26. Vassiliadis P., Simitsis A., Skiadopoulos S.: Conceptual Modeling for ETL Processes. DOLAP'02, November 8, 2002, McLean, VA. ACM 2002, s. 14÷21.

Recenzent: Prof. dr hab. inż. Tadeusz Morzy

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

The article presents LEMAT system which supervises ETL process, which role is based on an estimating periods of the execution time of SQL queries with SVM classifier and an analyze of time dimension usage. Additionally, this paper presents a solution which allows system to accommodate to variable environment workload conditions by gathering statistics and retraining classifier. This complete solution LEMAT for middleware layer of complex computer system, which might reduce cost of hardware spent on database server. Use SVM classifier to analyze SQL queries allows middleware application to maximize effective usage of hardware, increase number of executed queries per second and quality of data while system overload.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, M.Gorawski@polsl.pl .

Rafał WARDAS: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, Rafal.Wardas@polsl.pl .