Nowe metody przepisywania zapytań dla strukturalnych i semi-strukturalnych baz danych

Autoreferat pracy doktorskiej

Marta Jadwiga Burzańska

marzec 2011

Języki zapytań są podstawowym mechanizmem komunikacji aplikacji z bazą danych. Ich konstrukcja oparta jest o założenie, że programista zamiast mówić, w jaki sposób dane powinny być przeszukiwane, powinien skupić się na informacji, jakie dane chce wydobyć z bazy. Algorytmy wykonywania zapytań są ukryte przed programistą, co redukuje czas pisania zapytań oraz pozwala systemowi zarządzania bazą danych (SZBD) wybrać jak najlepszy algorytm zależnie od wielkości i rozkładu danych. Ciężar usprawniania dostępu do bazy spoczywa więc na module optymalizatora, który dobiera odpowiedni plan wykonania zapytania. Optymalizacja zapytań jest zatem bardzo ważnym zadaniem SZBD. Od lat była i nadal będzie przedmiotem badań wielu naukowców. Współcześnie istnieje wiele opracowanych metod optymalizacyjnych działających na różnych etapach optymalizacji: semantycznego przepisywania zapytań, reguł transformacji planów do optymalizacji kosztowej specjalnych podzbiorów planów wykonania zapytania. Ponieważ optymalizacja zapytań jest problemem NP-trudnym, jej praktyczna realizacja jest heurystyczna. Zawsze jest miejsce, aby opracować nowe metody optymalizacyjne.

Przedstawiona rozprawa prezentuje wyniki badań nad nowymi metodami optymalizacji przez przepisywanie dla języków zapytań. W wyniku tych badań opracowane zostały cztery algorytmy optymalizacyjne, których wspólną ideą jest ograniczenie zużycia zasobów systemu przy jednoczesnym zmniejszeniu czasu ewaluacji. Dwa spośród opracowanych technik koncentrują się wokół optymalizacji zapytań rekurencyjnych, które z samej swojej natury zużywają sporo zasobów systemowych w czasie ewaluacji. Główne wyniki badań nad tymi algorytmami zostały opublikowane w [Burz09, Burz10a]. Przedstawiona rozprawa spina te tematyki i prezentuje szerszą perspektywę przeprowadzonych badań.

Rozdział pierwszy rozprawy wprowadza w tematykę odpytywania danych obiektowo-relacyjnych oraz semi-strukturalnych. Omówiony w nim został między innymi obiektowo-relacyjny język SQL. Jest to najpopularniejszy obecnie język zapytań. Pomimo że od kilkudziesięciu lat jest obecny na rynku baz danych, jest stale rozwijany. W dzisiejszych czasach większość prac badawczych dotyczących optymalizacji języków zapytań opracowana jest w oparciu o ten język.

Jednocześnie prace te stanowią podstawy dla analogicznych metod optymalizacyjnych dla innych języków. W drugim rozdziale rozprawy zaprezentowana została koncepcja Podejścia Stosowego do konstrukcji języków (Stack Based Approach). Główną ideą tego podejścia jest połączenie koncepcji języka zapytań z wysokopoziomowym językiem programowania poprzez oparcie semantyki języka o przetwarzanie na stosie środowisk. Rozdział ten prezentuje również koncepcję języka SBQL (Stack Based Query Language) opracowanego na początku lat 90-tych przez K. Subietę [SBQL, Subi94]. Język ten stanowi realizację założeń SBA i dedykowany jest do odpytywania danych semi-strukturalnych. Istnieje wiele prac badawczych poświęconych optymalizacji i rozwojowi języków zapytań, które bazują na SBA bądź SBQLu. Jednakże jest to język stale rozwijany i z tego powodu jest ciągle miejsce na nowe koncepcje optymalizacyjne. Pierwsza wersja tego języka została zaimplementowana w systemie LOQIS [Subi90a, Subi90b]. Do chwili obecnej istnieje szereg projektów i systemów bazujących na SBQLu, w tym europejskie projekty eGov Bus i VIDE. Od roku 2007 SBQL jest rozważany przez grupę OMG jako standard budowy dla języków zapytań dla obiektowych baz danych [OMG07]. Jedną z implementacji języka SBQL jest opracowany przeze mnie język PySBQL łączący w sobie koncepcje SBA z konstrukcjami zaczerpniętymi z popularnego języka programowania jakim jest Python [Burz07, Burz09a]. Język PySBQL wraz z opracowanym dla niego składem danych stanowił platformę testową dla niektórych prezentowanych w rozprawie algorytmów optymalizacji. Opis tego języka wraz z opisem badań jemu poświęconych umieszczony został w rozdziale drugi rozprawy po opisie języka SBQL .

Główną tematyką rozprawy jest optymalizacja zapytań przez przepisywanie. Jest to bardzo szeroka tematyka pełna różnorodnych zastosowań i podejść do całej gamy języków zapytań. Próba opisu wszystkich istniejących algorytmów mogłaby zakończyć się utworzeniem wielotomowego dzieła. Jednakże badania naukowe nad jakąkolwiek tematyką powinny zostać poprzedzone gruntowną analizą istniejących rozwiązań. Rozdział trzeci rozprawy prezentuje kilka wybranych algorytmów optymalizacji poprzez przepisywanie. Są to algorytmy, których koncepcje miały istotny wpływ na opracowane techniki optymalizacyjne.

Zaprezentowane w rozprawie nowe algorytmy przepisywania zostały opracowane dla strukturalnych (relacyjnych i obiektowych) i półstrukturalnych języków zapytań. Dwa spośród nich zajmują się optymalizacją zapytań rekurencyjnych i zostały opisane w rozdziale piątym rozprawy. Pozostałe dwa algorytmy skupiają się na optymalizacji nie-rekurencyjnych zapytań języka SBQL. Oba z tych algorytmów koncentrują się na optymalizacji złączeń dwóch funkcji, bądź operatorów z semi-strukturalnych języków zapytań. Pierwsza z tych technik oparta jest o zamianę podstawowych planów zapytań dla operatorów na konstrukcję opartą o funkcję reduce. Idea takiej zamiany pochodzi z technik optymalizacyjnych znanych z funkcyjnych języków programowania. Główną inspiracją były algorytmy taniej deforestacji dla języka Haskell [Wadl90, Gill96, Grus98, Joha01,

Voig08], stąd też robocza nazwa dla prezentowanego algorytmu – prosta deforestacja zapytań SBQL. W czwartym rozdziale rozprawy prezentowane są między innymi główne założenia, jakie musi spełnić zapytanie, aby można je było zoptymalizować. W trakcie badań nad tym algorytmem odkryłam, że sposób rozstrzygania tej kwestii pokrywa się z metodą sprawdzania założeń dla istniejącego algorytmu wykrywania podzapytań niezależnych [Plod00, Subi04, Piec10]. Interesującą kwestią jest, że w sytuacjach, gdy nie można zastosować prostej deforestacji, zazwyczaj ma zastosowanie metoda wyłączania podzapytań. Rozprawa prezentuje również zbiór podstawowych planów zapytań jakie można zastosować do głównych operatorów SBQLa, opis algorytmu optymalizacji przykład zastosowania oraz wyniki testów wydajnościowych przeprowadzonych przy pomocy bazy danych odpytywanej językiem PySBQL. Wyniki pokazały, że zastosowanie tej metody optymalizacji poprawia w niewielkim stopniu czas ewaluacji, jednakże w znacznym stopniu zmniejsza ilość zasobów zużywanych przez system.

Badania nad tym algorytmem zaowocowały, niejako przy okazji, opracowaniem innego algorytmu optymalizującego złożenie operatorów i funkcji w SBQLu. Konkretnie złożenie operatora nawigacji (kropki) z funkcjami agregującymi. Celem tej techniki, podobnie jak w poprzednim algorytmie, była redukcja rozmiarów struktur pośrednich tworzonych w czasie ewaluacji zapytania. Prezentowane w rozprawie wyniki testów eksperymentalnych zostały zgromadzone przy użyciu tej samej konfiguracji sprzętowej i bazy danych co w przypadku algorytmu deforestacji. W tym przypadku algorytm uzyskał znaczący wynik zmniejszenia zużycia zasobów systemowych przy jednoczesnym przyśpieszeniu ewaluacji.

Piąty rozdział rozprawy poświęcony jest dwóm algorytmom optymalizującym zapytania rekurencyjne. Prace nad nimi są elementem ogólnych badań nad przetwarzaniem zapytań rekurencyjnych w bazach danych [Burz09, Przy10, Burz10, Burz10a]. Pierwszy z algorytmów opracowany został dla języka SQL. Oparty jest o dobrze znaną technikę „przesuwania predykatów” (predicate pushing), jednak zbadane zostało jej zastosowanie w nowym kontekście. Ponieważ podczas przetwarzania zapytań rekurencyjnych przechowywanie struktur pośrednich ma bardzo duże znaczenie, algorytm skupia się na redukcji tych struktur poprzez odpowiednie przepisanie zapytania. Bazuje na optymalizacji rekurencyjnych wyrażeń tablicowych, których konstrukcja umożliwia przesunięcie predykatów z zapytania zewnętrznego do zapytania inicjującego CTE. W rozprawie zaprezentowano koncepcję algorytmu, przykład użycia oraz wyniki testów eksperymentalnych. Testy przeprowadzone przy pomocy bazy danych IBM DB2 pokazały, że zastosowanie prezentowanego algorytmu, w zależności od ilości kroków rekurencyjnych i rozmiaru danych, pozwala na nawet 10-krotne przyśpieszenie ewaluacji

Druga metoda opracowana jest dla języka SBQL i optymalizuje złożenie funkcji przetwarzania danych z bazy z rekurencyjnym operatorem tranzytywnego domknięcia operatora nawigacji.

Konstrukcja tego algorytmu jest rozwinięciem idei deforestacji zapytań SBQL dla zapytań rekurencyjnych. Podobnie jak w przypadku deforestacji, ta metoda również bazuje na przepisywaniu zapytań wpierw do postaci planów zapytań, a następnie na przepisywaniu tych planów do zoptymalizowanej postaci. Pomimo że wyniki testów eksperymentalnych nie są tak imponujące jak w przypadku optymalizacji zapytań SQL, zastosowanie proponowanego algorytmu dla języka PySBQL zarówno przyśpieszyło ewaluację, ale co jest istotniejsze – miało znaczny wpływ na zmniejszenie konsumpcji zasobów systemowych.

Przygotowywana rozprawa miała na celu zaprezentowanie czterech algorytmów optymalizacyjnych nakierowanych na obniżenie zużycia zasobów systemowych podczas ewaluacji zapytań. Docelowo planuję zintegrować algorytmy opracowane dla języka SBQL na stałe z głównymi implementacjami tego języka oraz z platformą PySBQL. Algorytm opracowany dla języka SQL został włączony jako element optymalizacyjny do projektów badawczych nad bibliotekami odwzorowania obiektowo-relacyjnego (ORM). Wstępne wyniki tych badań, przygotowane dla rozszerzenia biblioteki ORM języka Python o zapytania rekurencyjne, zostały opublikowane w [Burz10a]. Trwają prace nad opracowaniem rozszerzeń rekurencyjnych dla pakietu Hibernate (biblioteka ORM dla języka Java). Innym nurtem badań, częściowo zasygnalizowanym w rozprawie, jest przeniesienie algorytmów optymalizacji języka SBQL na bazy z rodziny NoSQL. Obecnie prowadzę badania nad użyciem języka PySBQL jak wrappera na systemy obsługujące systemy map/reduce. Równoległym nurtem obecnie prowadzonym jest badanie nad przepisywaniem zapytań rekurencyjnych na zoptymalizowane zapytanie przetwarzane liniowo.

Bibliografia

[Burz07] M. Burzańska and P. Wiśniewski. PySBQL - Python-Like Query Language Constructed Using Stack Base Approach. Annales UMCS, Informatica, 2007, pages 143-151

[Burz09] M. Burzańska, K. Stencel and P. Wiśniewski. Pushing Predicates into Recursive SQL Common Table Expressions. In Proc. ADBIS 2009, LNCS 5739, Springer-Verlag, 2009, pages 194-205

[Burz09a] M. Burzańska and P. Wiśniewski. L-Value and R-Value Concept - Proposition to Solve Ref & Deref Chaos in SBQL Languages Family. Pol. J. Environ. Stud. Vol. 18 no. 3B, 2007, pages 143-151

[Burz10] M. Burzańska, K. Stencel and P. Wiśniewski. Intermediate Structure Reduction Algorithms for Stack Based Query Languages. In Proc. ASEA'10, CCIS 1(117), Springer-Verlag, 2010, pages 317-326

[Burz10a] M. Burzańska, K. Stencel, P. Suchomska, A. Szumowska, and P. Wiśniewski. Recursive Queries Using Object Relational Mapping. In Proc. FGIT'10, LNCS 6485, Springer-Verlag, 2010, pages 564-576

[Gill96] A. J. Gill. Cheap deforestation for non-strict functional languages. PhD thesis, The University of Glasgow (1996)

[Grus98] T. Grust, M. H. Scholl. Query deforestation. Technical report, Database Research Group, University of Konstanz, 1998

[Joha01] P. Johann. Short cut fusion: Proved and improved. In LNCS 2196, Springer-Verlag, 2001, pages 47–71

[OMG07] OMG Object Database Technology Working Group: Next-Generation Object Database Standardization, OMG White paper, 2007. Available at `http://www.omg.org/docs/mars/07-09-13.pdf`

[Piec10] T. Pieciukiewicz. Recursive Queries in Databases. PhD thesis, Polish-Japanese Institute of Information, Warsaw, 2010

[Plod00] J. Płodzień. Optimization Methods in Object Query Languages. PhD thesis, Institute of Computer Science, Polish Academy of Science, Warsaw, 2000,

[Przy10] P. Przymus, A. Boniewicz, M. Burzanska and K. Stencel. Recursive query facilities in relational databases: a survey. In proc. DTA/BSBT'10, CCIS 1(118), Springer-Verlag, 2010, pages 89-99

[Subi90a] K. Subieta. LOQIS: The Object-Oriented Database Programming System. In Proc. East/West Database Workshop 1990, Springer, Kiev, USSR, 1990, pages 403-421.

[Subi90b] K. Subieta, M. Missala, K. Anacki. The LOQIS System. Technical Report 695, Institute of Computer Science Polish Academy of Sciences, Warsaw, Poland, 1990.

[Subi94] K. Subieta, C. Beeri, F. Matthes and J. Schmidt. A Stack-Based Approach to Query Languages. In Proc. East/West Database Workshop, 1994, pages 159-180.

[Subi04] K. Subieta. Teoria i konstrukcja obiektowych języków zapytań. Editors of the PJWSTK, Warsaw, 2004

[SBQL] Stack Based Query Language: Recursive Operators. Available at `http://www.sbql.pl/Topics/SBQL%20Recursive.html`

[Voig08] J. Voigtländer. Semantics and Pragmatics of New Shortcut Fusion Rules. In Proc. FLOPS 2008, LNCS 4989, Springer-Verlag, 2008, pages 163-179,

[Wadl90] P. Wadler. Deforestation: Transforming programs to eliminate trees. Theor. Comput. Sci. 73(2), 1990, pages 231–248