Warszawa, 19/6/2006

mgr inż. Tomasz Pieciukiewicz

Katedra Systemów Informacyjnych,

PJWSTK

Autoreferat planowanej rozprawy doktorskiej

Zapytania rekurencyjne w bazach danych

Wiele zadań istotnych z punktu widzenia praktycznych zastosowań informatyki – a w szczególności baz danych - to zadania rekurencyjne. Problemy takie jak Bill of Materials, wiele zadań związanych z przepływami pracy, transportem czy sieciami społecznymi najwygodniej formułować w postaci rekurencyjnej. Coraz więcej przetwarzanych danych zapisanych jest w formatach, dla których przetwarzanie rekurencyjne jest najbardziej naturalną techniką działania – dotyczy to np. danych zapisanych w formatach takich jak XML, RDF, czy też w obiektowych bazach danych.

Niestety, funkcjonalność dostarczana przez używane obecnie języki zapytań nie zaspokaja istniejących potrzeb w tym zakresie. Wsparcie dla przetwarzania rekurencyjnego w najpopularniejszym języku zapytań – SQL – w zasadzie nie istnieje. Standardy SQL-89 i SQL-92 nie zawierają jakichkolwiek konstrukcji umożliwiających formułowanie zapytań rekurencyjnych. Nowsze standardy SQL (SQL-3, SQL-2000, SQL-99) definiują co prawda odpowiednie konstrukcje, jednak mało prawdopodobne jest, aby pełna implementacja któregoś z tych eklektycznych, rozdmuchanych standardów kiedykolwiek ujrzała światło dzienne. Niektóre systemy zarządzania bazami danych, takie jak Oracle i DB2 zapewniają co prawda możliwość zadawanie tego typu zapytań, stosowane w nich rozwiązania są jednak bardzo ograniczone, w niektórych zaś wypadkach (DB2), zadawanie skomplikowanych zapytań okazuje się bardzo trudne dla typowego użytkownika.

Istnieją oczywiście języki zapytań zbudowane z myślą o dostarczeniu użytkownikom możliwości zadawania zapytań rekurencyjnych – język Datalog i jego liczne pochodne. Niestety języki te mają szereg wad, które zdyskwalifikowały je jako języki ogólnego zastosowania – między innymi trudności w zadawaniu zapytań z warunkami na wartości kolumn/atrybutów, koncentracja na opisywaniu zależności pomiędzy danymi zamiast na wyborze danych interesujących użytkownika, wreszcie kłopotliwy sposób formułowania bardziej złożonych zapytań – składających się tak naprawdę z serii niezależnych od siebie, osobno definiowanych stwierdzeń opisujących dane. Języki te okazały się zbyt skomplikowane dla użytkownika – zarówno ich składnia, jak i semantyka oraz sposób ich prezentacji przez twórców stały się przeszkodą nie do przebycia dla potencjalnych użytkowników spoza środowisk akademickich.

Inne języki zapytań – takie jak np. języki zapytań dla XML (Xqery i pochodne) problem zapytań rekursywnych albo ignorują albo spychają na dalszy plan, jako zbyt skomplikowany. Część z nich dostarcza odpowiednie rozwiązania, są to jednak rozwiązania kalekie, ograniczone pod względem funkcjonalnym i trudne w użyciu.

Najbardziej naturalne dla wielu programistów, wydawałoby się, podejście do problemów rekurencyjnych – użycie funkcji rekurencyjnych w odpowiednim języku programowania pozwalającym na łączenie się z wybraną bazą danych – również nie jest pozbawione wad.

Języki programowania zintegrowane z SZBD, takie jak Transact SQL czy PL/SQL, są poważnie ograniczone przez swoich twórców – zazwyczaj nie mogą wykorzystywać typów masowych wykorzystywanych przez ich SZBD, często głębokość rekurencji jest sztucznie ograniczana przez silnik SZBD. Wykorzystanie procedur zewnętrznych w stosunku do SZBD, łączących się z bazą przy pomocy jakiegoś API, obarczone jest zazwyczaj poważnym narzutem czasowym, jak również podobną niezgodnością impedancji i typów, jak w wypadku języków programowania zintegrowanych z wybranym SZBD. Zadawanie „zapytań" w ten sposób jest mało praktyczne – wymaga napisania dużej ilości kodu, którego jedynie niewielka część związana jest bezpośrednio z zadaniem.

Większość prac badawczych dotyczących zapytań rekurencyjnych, jakie prowadzone były dotąd w różnych ośrodkach, skupiała się na zagadnieniach takich jak różne algorytmy ewaluacji zapytań pozwalające na optymalizację czasu wykonania zapytania, zwiększenie siły ekspresji tego typu zapytań etc. Prace te jednak są jedynie próbami usprawnienia istniejących, dalekich od doskonałości rozwiązań. Znacznie bardziej obiecujące wydaje się odejście od istniejących rozwiązań na rzecz świeżego spojrzenia na problem. Dobrym polem dla tego typu eksperymentu może być język SBQL – język zapytań oparty na podstawach teoretycznych znanych z języków programowania ogólnego zastosowania, znacznie elastyczniejszy i prostszy do rozbudowy od klasycznych języków zapytań opartych na algebrze relacji lub logice.

W rozprawie zaproponowane zostaną konstrukcje dla języka SBQL, wspierające trzy podstawowe paradygmaty programowania rekurencyjnego:

- Operator tranzytywnego domknięcia – konstrukcja pozwalająca na wykonywanie obliczeń rekurencyjnych wewnątrz zapytań SBQL, realizująca równanie stałopunktowe $wynik = q_1 \cup wynik.\ q_2$, gdzie q_1 i q_2 to podzapytania otrzymane przez ten operator.
- Układy równań stałopunktowych – nowa konstrukcja w SBQL, pozwalająca definiować układy równań (każde równanie w formie przypisania wyniku działania zapytania mogącego korzystać ze wszystkich używanych w układzie równań zmiennych na zmienną), które będą ewaluowane do momentu osiągnięcia przez układ punktu stałego;
- Rekurencyjne procedury funkcyjne bezszwowo zintegrowane z językiem zapytań – korzystające z wspólnego z językiem zapytań środowiska i składu danych, mogące wykonywać dowolne zapytania i wykorzystywać ich wyniki, jak również możliwe do wywołania z zapytania;

W ramach pracy wszystkie trzy konstrukcje zostaną zaimplementowane dla prototypowej bazy danych jOdra. Przedstawione zostanie porównanie tych konstrukcji z podobną funkcjonalnością proponowaną w innych językach zapytań, na przykładach popularnych problemów rekurencyjnych takich jak np. Bill of Materials. Zaproponowane i zaimplementowane zostaną również metody optymalizacji dla operatora tranzytywnego domknięcia i układów równań stałopunktowych.