Automatyczne strojenie fizycznej struktury bazy danych

Łukasz Twardokęs

Instytut Informatyki Wydziału Elektroniki i Technik Informacyjnych,
Politechnika Warszawska
L.Twardokes@stud.elka.pw.edu.pl.com

Streszczenie. Niniejsza praca stanowi wprowadzenie do tematyki automatycznego strojenia fizycznej struktury bazy danych. Celem jej jest ogólne zapoznanie odbiorców z problemem oraz narzędziami wykorzystywanymi przy automatycznym doborze indeksów, widoków zmaterializowanych oraz partycji. Opisowi architektury produktów wiodących producentów baz danych towarzyszy szczegółowe wyjaśnienie sposobu ich działania.

Słowa kluczowe: SQL, indeksy, widoki zmaterializowane, partycje, automatyczne strojenie

1 Wstęp

Język SQL jest podstawowym sposobem komunikacji z systemami baz danych. Łatwość konstruowania poleceń powodowana deklaratywnością jest niewątpliwie wielką jego zaletą. Aby pobrać dane wystarczy bowiem podać ich opis. Informacje o sposobie przechowywania danych oraz różnych metodach dostępu do nich nie są potrzebne, to bowiem system bazy danych na podstawie analizy zapytania wybierana najbardziej odpowiednią metodę dostępu do zmagazynowanych rekordów. Niestety prostota tworzenia zapytań okupiona jest złożonością procesu ich analizy oraz optymalizacji, którego wynik - plan wykonania zapytania - nie zawsze jest przez to optymalny. Problem efektywnego wybierania danych, potęguje dodatkowo fakt ciągłego przyrostu ilości magazynowanych informacji, z którymi współczesne systemy baz danych muszą sobie radzić.

Optymalizacja zapytań jest niewątpliwie najważniejszym procesem wpływającym na wydajność całego systemu bazy danych [1], ale kwestią nie do pominięcia jest jednak odpowiedni fizyczny projekt bazy danych. Właściwie zaprojektowane indeksy, partycje czy też perspektywy zmaterialiowane potrafią bowiem w sposób znaczący poprawić wydajność procesu pobierania danych. Niniejsze opracowanie nie wgłębia się w proces optymalizacji zapytań, odsyłając czytelnika do [2], szczególną za to uwagę poświęca właściwej strukturze bazy danych, a dokładniej narzędziom, które w sposób automatyczny dobierają wyżej wymienione obiekty do aktualnego obciążenia bazy danych.

Niniejsze opracowanie zorganizowane jest w następujący sposób. Cześć 1 stanowi wprowadzenie do omawianej tematyki. W części 2 przedstawiono motywacje, jaka sprawiła, że tego typu narzędzia powstały i są ciągle rozwijane. Z kolei część 3 skupia się na opisie różnych metod przeszukiwania przestrzeni rozwiązań wykorzystywanych w narzędziach optymalizacyjnych. Część 4 przedstawia ogólną architekturę istniejących na rynku rozwiązań, na szczególną uwagę zasługuje w niej jednak podrozdział 4.2 opisujący architekturę narzędzia COLT, które dzięki swej całkowitej automatyzacji (nie wymaga żadnej obsługi ze strony administratora/dewelopera) wyznacza kierunek rozwoju tego typu narzędzi. Na zakończenie, w części 5, podsumowano wcześniejsze rozważania.

2 Motywacja do badań i rozwoju

Ciągły rozwój biznesu oraz tendencja czy wręcz czasami konieczność wdrożenia rozwiązań informatycznych, opartych przeważnie o bazę danych sprawiają, iż ilość wymagających administracji baz rośnie w dużym tempie. Dodatkowo rozwój technologiczny, możliwości sprzętowe oraz konieczność redukcji kosztów sprawiają, iż powstające rozwiązania są dużymi i bardzo złożonymi systemami. Dla przykładu popularne aplikacje bazodanowe oparte o rozwiązania firmy SAP [3] operują przeciętnie na ponad 30000 obiektach bazodanowych, w skład których wchodzą głównie tabele i indeksy. Administracja takimi bazami danych to zatem nie lada wyzwanie oraz olbrzymi koszt. I tu ujawnia się pierwsza i jedna z bardziej istotnych motywacji, kierująca twórcami zautomatyzowanych narzędzi administratorskich - *przyczyny ekonomiczne*. Dobrze wykwalifikowany administrator baz danych, którego zadaniem jest utrzymanie tych złożonych systemów, to duży koszt dla każdego przedsiębiorstwa. Obserwowany ostatnio na rynku trend wzrostowy płac administratorów sprawia, iż całkowity koszt utrzymania systemów został zdominowany przez koszt kapitału ludzkiego, nie zaś sprzęt czy oprogramowania. W takich realiach celowe, gdyż przynoszące realną redukcję kosztów, wydaje się odciążenie administratorów baz danych z czynności żmudnych, czasochłonnych i rutynowych. Doskonałym obszarem do zastosowania automatyzacji jest właśnie strojenie fizycznej struktury bazy danych.

Kolejnym powodem wprowadzania automatyzacji, wspierającej administrację bazami danych jest ich rozmiar i złożoność. Jak już wspomniano wcześniej, rozwój biznesu - głównego odbiorcy rozwiązań bazodanowych - sprawia, iż tworzone systemy są coraz bardziej złożone. Bardzo często ich *złożoność* jest tak duża, że przekracza możliwości ludzkiego umysłu. W takim przypadku jedynym możliwym rozwiązaniem staje się zlecenie wykonania pracy odpowiedniemu systemowi komputerowemu, który właściwie zaprogramowany poradzi sobie z zadaniem.

Aby prawidłowo dostroić strukturę fizyczną bazy danych, konieczna jest znajomość typowego zestawu poleceń SQL, wykonywanego na tej bazie. Niestety, w niektórych przypadkach zestaw taki jest nieznany, gdyż polecenia są wykonywane w zależności od aktualnych potrzeb - polecenia ad hoc, lub też poleceń jest zbyt dużo i mogą się często zmieniać, jak to ma miejsce w bazach danych tworzo-

nych na serwerach hostingowych. W takich przypadkach, człowiek może okazać się bezradny lub niewydajny, konieczne są zatem narzędzia, które będą w sposób ciągły śledziły obciążenie bazy i na tej podstawie automatycznie dostrajały fizyczną strukturę bazy danych.

Wszystkie wymienione argumenty kierują uwagę badaczy i projektantów na problem automatyzacji części czynności administracyjnych, co zgodne jest z ogólnym trendem budowania systemów informatycznych - samo-zarządzalnych lub automatycznych [4] - i znajduje potwierdzenie w nowych aplikacjach, takich jak DB2 Design Advisor w bazie danych DB2, Database Tuning Advisor w Microsoft SQL Server 2005 oraz SQL Access Advisor w Oracle.

3 Obszar optymalizacji

Istnieje kilka struktur, które mogą poprawić działanie poleceń SQL, spośród nich w obszarze zainteresowań producentów narzędzi automatycznie strojących fizyczny projekt bazy danych, są: indeksy, perspektywy zmaterjalizowane, partycje oraz klastry. Należy przy tym zaznaczyć, że nie wszystkie narzędzia wspierają każdy z wymienionych typów obiektów. Obecnie pełne wsparcie, wśród wiodących producentów baz danych oferuje jedynie IBM w swoim produkcie DB2 Design Advisor. Pozostali dostawcy, tacy jak Mictosoft i Oracle ograniczają się do indeksów, perspektyw zmaterializowanych oraz partycji, w przypadku Oracle te ostatnie dostępne są dopiero od wersji 11g.

Biorąc pod uwagę wszystkie wymienione powyżej sruktury, przestrzeń przeszukiwań z jaką muszą zmierzyć się narzędzia jest bardzo duża. Oznaczając jako NI - ilość możliwych indeksów, jako NM - ilość widoków zmaterializowanych, jako NP ilość partycji oraz ewentualnie jako NC ilość możliwych do stworzenia tabel klastrowanych wynosi ona

$$2^{NI+NM+NP+NC}$$

a to głównie za sprawą tego, że różne struktury potencjalnie mogą ze sobą oddziaływać. Jest to olbrzymia ilość możliwości do analizy, a co za tym idzie olbrzymi nakład czasowy konieczny na zbadanie które z konfiguracji dają najlepsze efekty optymalizacyjne. Tworząc swoje narzędzie, badacze z IBM analizowali kilka możliwych rozwiązań [3], z których zostało wybrane jedno, najlepsze. Kolejne podrodziały prezentują pokrótce rozważane przez Nich podejścia.

3.1 Podejście iteracyjne

Podejście iteracyjne jest najprostszą metodą wyszukiwania odpowiednich struktur wspomagających dostęp do danych. Polega ono na ignorowaniu interakcji i zależności istniejących pomiędzy różnymi strukturami i analizie każdej z nich w oderwaniu od pozostałych, tak jak sama nazwa wskazuje - iteracyjnie - jeden po drugim. Niestety zmniejsza to znacząco możliwość znalezienia dobrego rozwiązania, gdyż indeksy i widoki zmaterializowane są dość mocno ze sobą powiązane. Jeżeli bowiem na oryginalnych danych zostanie stworzony widok i jakieś

zapytanie z niego będzie korzystać, eliminując tym samym użycie tabeli, indeks na pierwotnych danych jest zbędny, zaś na indeksowanym widoku wskazany. Taka sytuacja z pewnością nie zostanie wykryta w tym podejściu. Kolejnym przykładem mogą być partycje i indeksy. Partycjonowanie tabeli w większości przypadków wymaga również podziału założonych na tej tabeli indeksów (stworzenie oddzielnego indeksu dla każdej z partycji), co nie zostanie zauważone przy wykorzystaniu podejścia iteracyjnego.

3.2 Podejście zintegrowane

Mimo iż złożoność koncepcyjna wydaje się podobna do wcześniejszego rozwiązania podejście zintegrowane jest znacznie bardzie złożone obliczeniowo, a więc i czasochłonne. Koncepcja ta bazuje na przeszukiwaniu pełnej, opisanej we wstępie rozdziału, przestrzeni rozwiązań, korzystając jedynie z pewnych technik heurystycznych częściowo ograniczających ilość kandydatów do analizy. Niewątpliwą jednak zaleta podejścia zintegrowanego jest jego dokładność. Dzięki wszystkich struktur łącznie, 'chybienia' opisane przy okazji omawiania podejścia iteracyjnego nie będą mieć miejsca.

3.3 Podejście hybrydowe

Rozwiązaniem kompromisowym, wydaje się zastosowanie podejścia iteracyjnego w przypadkach, kiedy związku między wspomagającymi wydajność strukturami nie ma, lub jest niewielki oraz podejścia zintegrowanego w przypadku silnej zależności. Należy zatem wpierw rozważyć, które z struktur są od siebie silnie zależne, a których zależność jest słaba.

Zależności między strukturami Na podstawie doświadczeń praktycznych i badań autorzy [3] wyróżnili dwa rodzaje zależności występujących między strukturami wspierającymi dostęp do danych:

zależność silną , (A silnie zależy od B) która występuje gdy zmiana w wyborze struktury B często pociąga za sobą zmianę w wyborze struktury A.

zależność słabą , (A słabo zależy od B) gdy wybór innego zbioru struktur B rzadko pociąga za sobą zmianę w wyborze struktur A

Wszystkie analizowane struktury (indeksy, perspektywy zmaterializowane, partycje oraz klastry) zostały następnie zbadane pod względem istnienia tych zależność, zaś wyniki tej analizy przedstawione są w Tabeli 1.

Jak można zauważyć tabela nie jest symetryczna względem przekątnej, a zatem wprowadzona zależność nie jest symetryczna. Z tego właśnie względu opisany poniżej algorytm hybrydowy rozróżnia trzy przypadki.

W sytuacji, gdy pomiędzy strukturami występuje obustronnie silna zależność, ciężko jest rozdzielić analizę, gdyż łatwo można stracić dobre rozwiązania. Dlatego też w takim przypadku stosowane jest podejście zintegrowane, które co

A/B	Indeksy	Widoki zmaterializowane	Partycje	Klastry
Indeksy	–	silna	słaba	słaba
Widoki zmaterializowane	silna	–	słaba	silna
Partycje	słaba	silna	–	słaba
Klastry	słaba	silna	słaba	–

Tablica 1. Klasyfikacja zależności między strukturami A oraz B

prawda jest bardziej czasochłonne, jednak zapewnia znacznie większą dokładność. W pozostałych przypadkach, to znaczy gdy zależność między strukturami A i B jest słaba lub jednostronnie silna, stosowane jest podejście iteracyjne. Aby jednak uzyskać jak najlepsze efekty wybierana jest dokładnie kolejność iteracji oraz gdy to konieczne wyodrębniane są specjalne przypadki. Podsumowując zatem, gdy między strukturami A oraz B występuje

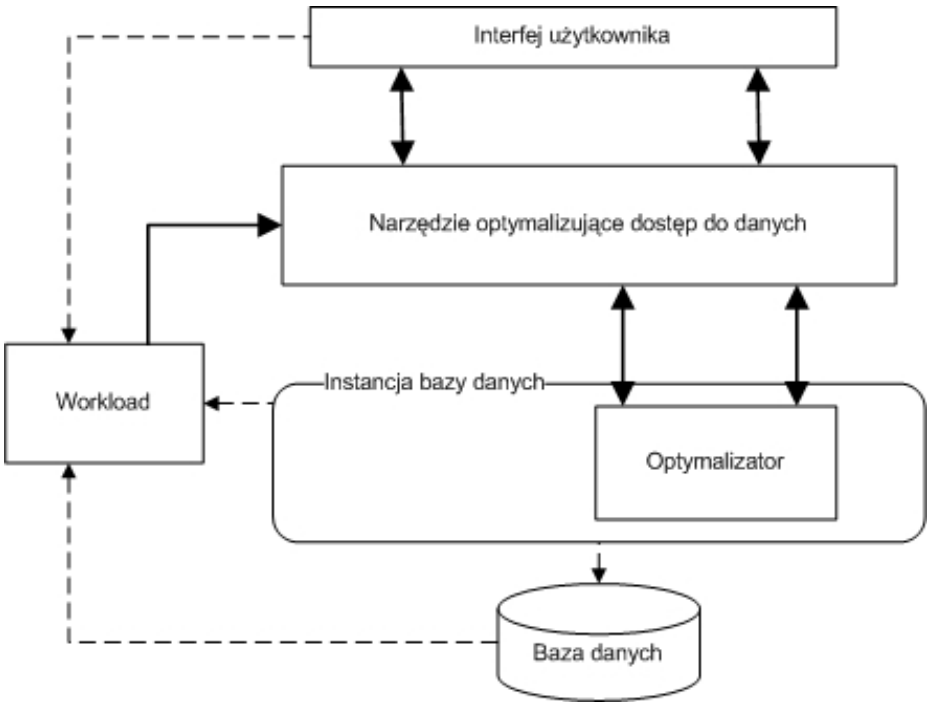
obustronnie silna zależność - stosowane jest podejście zintegrowane.
jednostronnie silna zależność (jedynie B zależy silnie od A) - stosowane jest podejście iteracyjne, jednak w taki sposób aby rozwiązanie dla B było wyszukiwane przed rozwiązania dla A
obustronnie słaba zależność - stosowane jest podejście iteracyjne, bez zwracania uwagi na

4 Narzędzia optymalizacyjne

Dostępne na rynku narzędzia optymalizujące fizyczną strukturę bazy danych, najczęściej zintegrowane z zestawem narzędzi administracyjnych i dostarczane wraz z komercyjnymi bazami danych budowane są w dość zbliżony sposób, ich ogólna architektura przedstawiona została w sekcji 4.1. Narzędzia te charakteryzują się tym, że rzeczywiście znacząco wspomagają administratora w strojeniu bazy danych, jednak wymagają jego interakcji, szczególnie w fazie interakcji. Pewnym kierunkiem rozwoju, który został zaprezentowany w sekcji 4.2 są narzędzia w pełni automatyczne, które całkowicie autonomicznie zajmują się strojeniem i implementacją odpowiednich struktur wspomagających dostęp do danych.

4.1 Ogólna architektura

Wiodący producenci baz danych wraz z swym produktem, dostarczają szereg narzędzi administracyjnych wspomagających administratorów w diagnostyce i utrzymaniu systemów. W skład tych narzędzi wchodzą również aplikacje wspomagające optymalizację dostępu do danych. Szczegóły dotyczące architektury oraz sposobu działania można znaleźć w [3], [5] oraz w [6]. Na podstawie przedstawionych w tych pracach informacji można zaprezentować ogólną architekturę tego typu narzędzi.



Rysunek 1. Ogólna architektura narzędzia doradczego

Elementem architektury, z którym bezpośrednio współpracuje administrator bazy dancyh jest interfejs użytkownika. Spełnia on dwie zasadnicze funkcje, pierwsza to dostarczenie danych wejściowych dla procesu optymalizacji, takich jak (opcjonalnie) obciążenie, czyli zestaw uruchamianych zapytań pod które jest optymalizowany dostęp do danych, oraz budżet przestrzeni dyskowej, czas uruchomienia optymalizacji, gdyż może ona być uruchomiona z opóźnieniem. Drugą funkcją interfejsu jest prezentacja wyników procesu optymalizacji, czyli zestawu sugestii, wskazówek, których implementacja poprawi wydajność zapytań.

Jak zostało wspomniane obciążenie bazy danych (Workload) może być pobierane za pomocą interfejsu użytkownika, czyli poprzez wpisanie zestawu poleceń SQL, nie jest to jednak jedyna droga pozyskania tych danych. Każdy z omawianych serwerów przechowuje w swojej pamięci operacyjnej zestaw ostatnio wykonywanych na bazie zapytań i on również może posłużyć jako wejściowe obciążenie. Dodatkowo, niektóre serwery umożliwiają zapisanie istniejącego w danym momencie obciążenia, a następnie wykorzystanie go jako jako dane wejściowe i to jest trzecia droga zdobycia najważniejszych danych do strojenia.

Po pozyskaniu wszystkich danych wejściowych następuje uruchomienie procesu strojenia. Dokładny przebieg tego procesu zależy oczywiście od dostawcy narzędzia, jednak istnieje jednak wspólna cecha tych narzędzi, a mianowicie wszystkie one korzystają z dostarczanego z bazą danych i wykorzystywanego do normalnego wykonywania zapytań optymalizatora. Uruchamiany zazwyczaj w specjalnym trybie, służy on przede wszystkim do oceny kosztu zaproponowanych przez narzędzie rozwiązań. Dzięki zastosowania specjalnego trybu jego uruchomienia możliwe jest stworzenie planu wykonania zapytania bez konieczności jego wykonywania, co jednak ważniejsze również bez konieczności tworzenia struktur przy których analiza przebiega. Dzięki takiemu rozwiązaniu zaoszczędzany jest czas jak i przestrzeń dyskowa.

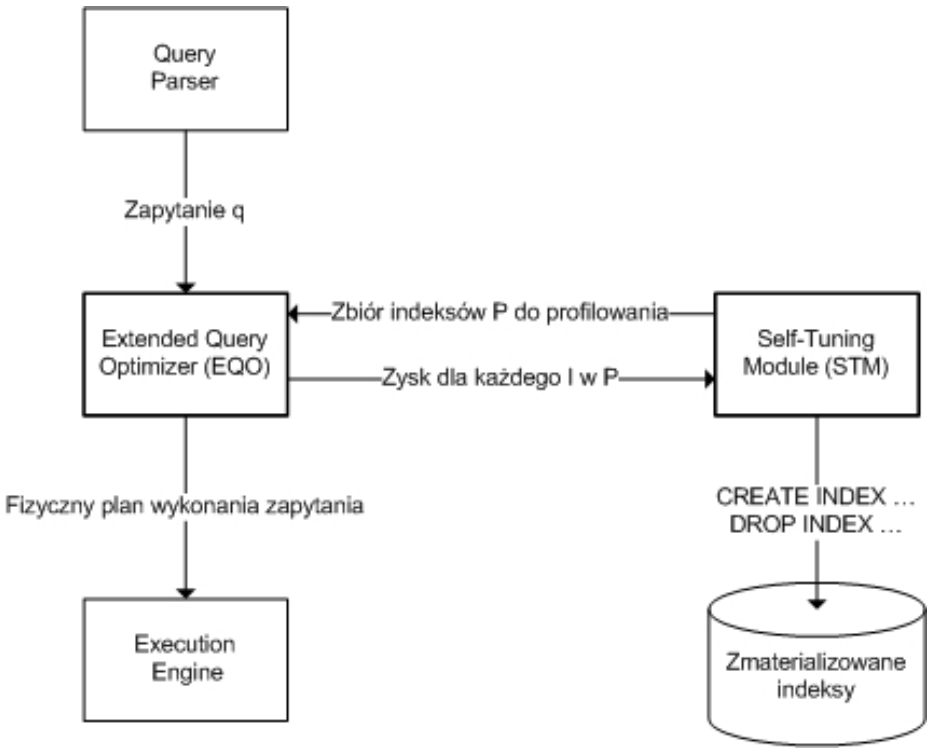
Wynikiem działania każdego z narzędzi jest zestaw rekomendacji, które prezentowane są użytkownikowi za pomocą interfejsu użytkownika. W tym momencie użytkownik decyduje o ewentualnej implementacji wskazówek.

4.2 Continous On-Line Database Tuning [7]

Opisane we wcześniejszym podrozdziale narzędzia mają za zadanie wsparcie administratora lub projektanta, który dzięki swojej wiedzy i doświadczeniu oraz pomocy narzędzi potrafi prawidłowo zaprojektować wymienione struktury. Rozwiązanie takie sprawdza się jednak jedynie wtedy, gdy obciążenie, czyli zestaw często zadawanych poleceń SQL, jest stabilny i mało zmienny, gdy jednak zapytania nie są znane, bądź zmieniają się dość często, tego typu podejście jest nieefektywne lub wręcz niemożliwe. Bazy danych do których zadawane są zapytania ad-hoc, lub serwery baz danych firm hostingowych są tego dobrym przykładem. Rozwiązaniem w tym przypadku wydaje się więc pełna automatyzacja procesu dopasowywania struktury fizycznej bazy danych do bieżącego jej obciążenia. Tego typu rozwiązaniem jest COLT (Continuous On-Line Database Tuning). Na podstawie zadawanych zapytań oraz dostępnej przestrzeni dysko-

wej, na bieżąco dopasowuje fizyczną strukturę bazy danych w celu uzyskania jak najlepszej wydajności zapytań.

Proces dynamicznego strojenia struktury fizycznej bazy jest oczywiście całkowicie niewidoczny dla użytkownika, a nawet administratora. Możliwe jest to dzięki zastosowanie specyficznej architektury (Rysunek 2). Moduł odpowiedzialny za strojenie stanowi bowiem rozszerzenie podstawowego mechanizmu optymalizacji zapytań. Podczas wykonywania zapytania, oprócz znalezienia optymalnego planu wykonania zapytań, opartego o istniejące aktualnie w bazie struktury danych, mechanizm ten wykonuje kilka dodatkowych czynności mających na celu sprawdzenie czy istniejąca konfiguracja indeksów i perspektyw zmaterializowanych jest optymalna lub czy można znaleźć lepszą.



Rysunek 2. Architekrura COLT, na podstawie [7]

Wykonywane przez COLT zadania rozdzielone są pomiędzy dwa jego zasadnicze moduły: Extended Query Optimizer (EQO) oraz Self-Tuning Module (STM). Pierwszy z nich odpowiedzialny jest przede wszystkim za proces optymalizacji zapytań, a więc wybór najlepszej metody dostępu do danych wyspecyfikowanych w zapytaniu, dodatkowo jednak wycenia zysk możliwy do osiągnięcia po implementacji wirtualnych struktur wspomagających dostęp do danych, zaproponowa-

nych przez STM. Zadaniem drugiego z komponentów jest wybór struktur, które będą poddane dalszej szczegółowej analizie przez EQO oraz dodawanie i usuwanie, czyli generalnie implementacja wybranych struktur, w ramach dostępnego budżetu przestrzeni dyskowej.

Proces optymalizacji pojedynczego zapytania przebiega następująco. Po przekazaniu zapytania do wykonania EQO wyznacza najlepszy, na podstawie istniejących struktur wspierających dostęp do danych, plan wykonania zapytania, drugi z modułów w tym czasie wyznacza wszystkie możliwe dla danego zapytania struktury, dzieląc je na podstawie przybliżonych (domniemanych) zysków na trzy podzbiory: Zaimplementowane (M), Gorące (H) oraz Zimne (C). Zależnie od przyjętej strategii część z tych elementów poddawana jest bardziej szczegółowej analizie podczas kolejnej fazy, tzw. analizy 'what-if'. Otrzymawszy zbiór struktur do profilowania - analizy, EQO sprawdza, czy wykorzystanie pewnej struktury może poprawić - zmniejszyć koszt wykonania zapytania, porównując go z kosztem wyliczonym dla planu opartego o istniejące indeksy i widoki zmaterializowane. Analiza taka przeprowadzana jest dla każdej z przekazanych struktur, zaś jej wynik jest przekazany do STM w postaci listy struktur, wraz z zyskiem jaki możne dzięki nim uzyskać. Na podstawie tak przygotowanych danych, biorąc dodatkowo pod uwagę ilość dostępnej przestrzeni dyskowej, STM decyduje, o implementacji lub usunięciu odpowiednich elementów. W celu poprawy efektywności i zmniejszenia nakładu potrzebnego na sam proces implementacji takich struktur nie odbywa się w momencie wykonania każdego zapytania, a co pewnie ustalony interwał zapytań (10).

5 Podsumowanie

Niewątpliwie narzędzie automatyzujące procesy administracji i obsługi różnych systemów, nie tylko baz danych, stanowią ważną dziedziną rozwoju współczesnej informatyki [4]. Wzrost skomplikowania i rozmiarów systemów sprawia, iż są one konieczne, gdyż człowiekowi ciężko jest ogarnąć i zanalizować z wystarczającą dokładnością dużą część z współcześnie tworzonych systemów. Problem strojenia fizycznej struktury bazy danych doskonale nadaje się do zastosowania automatyzacji, gdyż oprócz opisanego powyżej rozmiaru, stanowi również dobry przykład czynności czasochłonnych i żmudnych. Należy jednak pamięta, że istniejące na runku narzędzia wspomagają, lecz nie zastępują całkowicie człowieka. Dostarczają one bowiem zbiór podpowiedzi, a zatem znacząco zmniejszają rozmiar problemu, to jednak człowiek podejmuje ostateczną decyzję o kształcie fizycznej struktury bazy, implementując lub nie zaproponowane wskazówki. Co prawda przykład narzędzia COLT, przedstawionego w części 4.2 pokazuje, że podjęto już próby całkowitej automatyzacji omawianego procesu, aby jednak takie podejście było wydajne i dawało zbliżone rezultaty z automatyzacją wspieraną przez człowieka - eksperta, konieczne są dalsze badania.

Literatura

1. Garry, M., Corrigan, P.: Oracle Performance Tuning, O'Reilly, (1996).
2. Twardokęs, Ł.: Zapytania SQL: wydajność, metody optymalizacji, Instytut Informatyki, Wydział Elektroniki i Technik Informacyjnych.
3. Zilio, D.C., Rao, J., Lightstone, S., Lohman, G., Storm, A., Garcia-Arellano, C., Fadden, S.: DB2 Design Advisor: Integrated Automatic Physical Database Design, Materiały 30^{th} VLDB Conference, strony 1098-1109, Toronto, Canada 2004.
4. AUTONOMIC VISION and MANIFESTO:
`http://researchweb.watson.ibm.com/autonomic/manifesto/`,
Zasoby internetowe .
5. Agrawal, S., Chaudhuri, S., Kollar, L., Marathe, A., Narasayya, V., Syamala, M.: Database Tuning Advisor for Microsoft SQL Server 2005, Materiały 30^{th} VLDB Conference, strony 1098-1109, Toronto, Canada 2004.
6. Dageville, B., Das, D., Dias, K., Yagoube, K., Zait, M., Ziauddin,M.: Automatic SQL Tuning in Oracle 10g, (2004). Materiały 30^{th} VLDB Conference, strony 1098-1109, Toronto, Canada 2004.
7. Schnaiter, K., Abiteboul, S., Milo, T., Polyzotis, N.: COLT: Continuous On-Line Database Tuning. SIMOD 2006, strony 793-795, Chicago, USA 2006.