Optymalizacja zapytań a kompresja danych w bazie DB2 9

Agnieszka Węgrzyn, Jacek Tkacz, Adam Wojciechowski

Streszczenie: W artykule przedstawiono problem optymalizacji zapytań w języku SQL w bazie DB2 9. Baza danych DB2 9 ma możliwość kompresji danych, która polega na kompresowaniu wierszy tablic baz danych metodą wyszukiwania powtarzających się danych w wierszach i budowania słowników przypisujących im krótkie klucze numeryczne. W artykule zamieszczono wyniki z przeprowadzonych testów porównujących wyszukiwanie danych w przestrzeniach nieskompresowanych a skompresowanych.

Słowa kluczowe: bazy danych, kompresja danych, plan wykonania zapytania.

1. WPROWADZENIE

Na całym świecie można zaobserwować nieustający wzrost zapotrzebowania na łatwy i szybki dostęp do informacji. Liczba informacji ciągle wzrasta, zarówno tych ważnych jak i tych bezużytecznych. Do przechowywania, zarządzania i udostępniania danych poza sprzętem niezbędna jest technologia informatyczna. Często klasyczne bazy danych nie radzą sobie z bardzo dużą ilością danych oraz ciągle zwiększającym się zapotrzebowaniem na dostęp do nich. Zarówno producenci sprzętu jak i oprogramowania wychodzą naprzeciw wciąż rosnącemu zapotrzebowaniu na składowanie danych jak i dostęp do nich.

2. METODY KOMPRESJI DANYCH W BAZIE DB2 9.5

Ważnym aspektem systemów bazodanowych jest optymalizacja. Firma IBM w swojej bazie DB2 proponuje szereg udoskonaleń oszczędnościowych związanych z kompresją danych [1,2]. Jedną z najważniejszych własności funkcjonalnych systemów bazodanowych jest jednak jak najszybszy dostęp do informacji i wydawać by się mogło, że kompresja poza zmniejszeniem ilości przechowywanych danych wniesie znaczne spowolnienia. Zarówno czas na skompresowanie danych w momencie ich wprowadzania do bazy, jak i czas potrzebny na dekompresję w momencie uzyskania dostępu do nich powinny wykluczyć przydatność tego typu rozwiązań. Zaproponowane przez firmę IBM mechanizmy niewiele jednak mają wspólnego z kompresją najczęściej spotykaną w technologiach informatycznych, choć wynikiem działania wprowadzonych udoskonaleń jest znaczne zmniejszenie rozmiaru danych, a dodatkowo zmniejszenie kosztu dostępu do nich, a tym samym zwiększenie wydajności.

W rozdziale tym zostaną przedstawione i pokrótce opisane różne sposoby kompresji danych w bazie IBM DB2 w wersji 9.5. W celu przeprowadzenia eksperymentu wykorzystano kompresję wierszy.

2.1. KOMPRESJA WIERSZY

W wersji 9.5 systemu baz danych DB2 firmy IBM wprowadzono technologię venom kompresującą dane w tabeli [3,4,5]. Dla powtarzających się wzorców danych budowany jest słownik kompresji, który zawiera krótki klucz do reprezentowania wzorców. Wówczas w tabeli dane powtarzające się zastępowane są krótkim kluczem. Słownik kompresji jest tworzony dla całej tabeli. Wiersze dodawane po zbudowaniu słownika są także kompresowane. Tabele, które zbudowano jako nieskompresowane, można skompresować poprzez wydanie odpowiedniego polecenia SQL. Aktywację kompresji wierszy podczas tworzenia tabeli wykonuje się poleceniem:

```
CREATE TABLE <nazwa> compress yes;
```

Po wprowadzeniu danych należy użyć polecenia:

```
REORG TABLE <NAZWA TABELI>;
```

Tabela zostanie przeszukana pod kątem powtarzających się wpisów i wzorów.

Kiedy powtarzające się dane zostaną zebrane, słownik kompresji automatycznie utworzony podczas tworzenia tabeli, zostanie zaktualizowany. Powtarzające się wpisy zamienione zostają na krótkie numeryczne klucze. Słownik kompresji przechowuje oryginalne dane odpowiadające wcześniej przypisanym kluczom numerycznym. Rozmiary słownika wahają się od 50 kb do 150 kb. Jeżeli efekt kompresji niektórych danych jest równy, lub większy rozmiarowi danych oryginalnych, kompresja nie następuje.

Aktywacja kompresji wartości na istniejącej już tabeli odbywa się przy pomocy polecenia:

```
ALTER TABLE <NAZWA TABELI> COMPRESSION
YES;
```

Po wykonaniu powyższego polecenia zostanie utworzony słownik kompresji, a wszystkie już istniejące wiersze zostaną skompresowane. Po poleceniu ALTER TABLE nie jest wymagane użycie polecenia REORG. Należy mieć jednak na uwadze to, że w przypadku używania poleceń INSERT, IMPORT INSERT, LOAD

INSERT, REDISTRIBUTE, na tabelach posiadających włączoną kompresję oraz wygenerowany słownik kompresji, wprowadzane dane kompresowane są jako pojedyncze wiersze, nie biorąc pod uwagę całej tabeli. Może się, więc zdarzyć, że w tabeli będą występować wiersze skompresowane i nieskompresowane. Aby tego uniknąć, po użyciu wymienionych instrukcji, konieczne jest ponowne przebudowanie słownika kompresji przy użyciu polecenia REORG. Tym sposobem, w momencie tworzenia słownika kompresji, pod uwagę brana będzie cała tabela. Przebudowę słownika kompresji należy wykonać również w przypadku znaczącej modyfikacji danych przy użyciu instrukcji UPDATE.

Dezaktywacja kompresji wierszy wykonywana jest przy pomocy polecenia:

```
ALTER TABLE <NAZWA TABELI> COMPRESSION
NO;
```

Aby nastąpiła likwidacja słownika kompresji, również w tym przypadku należy użyć polecenia REORG.

Kompresja wierszy nie występuje w przypadku danych typu LONG, VARCHAR, LOB i XML (przechowywanego w postaci hierarchicznej).

2.2. KOMPRESJA WARTOŚCI NULL I WARTOŚCI DOMYŚLNYCH

Kolejnym sposobem kompresji danych w bazie DB2 jest kompresja wartości NULL i domyślnych. W tabelach przechowujących dużą liczbę wartości domyślnych i NULL należy zastosować klauzulę VALUE COMPRESSION, która spowoduje zastosowanie nowego formatu wiersza danych. Aktywacja kompresji wartości na istniejącej już tabeli odbywa się przy pomocy polecenia:

```
ALTER TABLE <NAZWA TABELI> ACTIVATE
VALUE COMPRESSION;
```

Kompresja tabeli nie działa wstecz, więc kompresja będzie aktywna tylko dla wpisów wykonanych po tym poleceniu. Jeżeli kompresja ma być aktywowana dla wcześniejszych wpisów należy użyć polecenia:

```
REORG TABLE <NAZWA TABELI>;
```

Dezaktywacja kompresji wykonywana jest przy pomocy polecenia:

```
ALTER TABLE <NAZWA TABELI> DEACTIVATE
VALUE COMPRESSION;
```

Dezaktywacja kompresji, analogicznie jak aktywacja, nie działa wstecz, więc w celu pełnego dezaktywowania mechanizmu należy również wykonać polecenie REORG.

2.3. KOMPRESJA FORMATU XML

W przypadku, gdy w dokumentach XML przechowywanych w bazie DB2 znajduje się wiele znaczników powtarzających się, zastępowane są one krótkimi znacznikami liczbowymi [6,7]. Ten sposób kompresji danych XML nie tylko oszczędza miejsce na dysku, ale również powoduje wyższą wydajność zapytań XML.

Kompresja danych typu XML możliwa jest poprzez zastosowanie opcji inline podczas tworzenia, lub modyfikacji istniejącej już tabeli. Standardowo dane typu XML przechowywane są w osobnym obszarze bazy danych nazwanym XDA. Po użyciu opcji inline dane XML, będą przechowywane razem z resztą danych na stronach tabeli. Do stworzenia nowej tabeli z opcją inline dla kolumny typu XML należy użyć instrukcji:

```
CREATE TABLE <NAZWA TABELI>(<NAZWA
KOLUMNY> XML INLINE LENGTH<INTEGER>);
```

Parametr LENGTH oznacza maksymalny rozmiar dokumentu XML, jaki może zostać przechowany. Maksymalny rozmiar danych XML, dodany do wielkości danych standardowych, przechowywanych w jednym wierszu, nie może przekroczyć rozmiaru strony tabeli. W przypadku, gdy dokument XML przekracza założony rozmiar LENGTH, lub dane przekraczają maksymalny rozmiar, to zostanie zapisany on w obszarze XDA.

Aby dodać kolumnę z opcją inline do istniejącej tabeli należy użyć instrukcji:

```
ALTER TABLE <NAZWA TABELI>ADD COLUMN
<NAZWA KOLUMNY> XML INLINE
LENGTH<INTEGER>;
```

Aby zastosować opcję inline na istniejącej już kolumnie należy użyć instrukcji:

```
ALTER TABLE <NAZWA TABELI>ALTER COLUMN
<NAZWA KOLUMNY XML> SET INLINE
LENGTH<INTEGER>;
```

Firma IBM zakłada, że w kolejnej wersji bazy kompresja danych typu XML nie będzie wymagała już użycia opcji inline, gdyż będzie bezpośrednio przeprowadzana w obszarze XDA. Podejście takie spowoduje, że operacja ta będzie dużo łatwiejsza do przeprowadzenia i może przynieść jeszcze lepsze rezultaty związane z oszczędnością miejsca jak i zwiększenia wydajności samych zapytań.

3. WYNIKI EKSPERYMENTÓW

W rozdziale tym zamieszczone zostaną wyniki z przeprowadzonych testów porównujących wyszukiwanie danych w przestrzeniach nieskompresowanych a skompresowanych.

Eksperyment przeprowadzono na serwerze SUN V890, na platformie Solaris 10 oraz serwerze baz danych IBM DB2 w wersji 9.5.

3.1. BADANIE KOMPRESJI WIERSZY

W celu przeprowadzenia eksperymentu zostały przygotowane dane w dwóch przestrzeniach tabel, w których umieszczono po jednej tabeli, z których każda przechowuje ponad milion rekordów (dokładnie tych samych rekordów). W celu sprawdzenia sposobu działania kompresji, część danych w tabelach powtarza się. Dla poszczególnych tabel i zapytań dla tych tabel przygotowano plan wykonania, na podstawie, którego oszacowano koszt dostępu do danych skompresowanych i nieskompresowanych.

Dane do celów eksperymentalnych zostały przygotowane w sposób losowy za pomocą poniższego polecenia SQL. Utworzono tabelę abonent w przestrzeni nieskompresowanej.

```
create table abonent (
```

```
ida bigint not null GENERATED ALWAYS AS
IDENTITY
(START WITH 1, INCREMENT BY 1,no cycle,
no cache),
data_wpisu date not null,
wplata bigint not null,
nazwisko char(100) not null,
primary key (ida))
in USER_NCOMPRESS index in
USER_NCOMPRESSI;
```

```
INSERT INTO abonent (wplata,
data_wpisu, nazwisko)
WITH abonent_(wplata) AS
(VALUES(2) UNION ALL
 SELECT wplata+1 FROM abonent_
WHERE wplata < 100001 )
 SELECT wplata,
  CURRENT DATE - ((18 * 365) +
RAND()*(47*365)) DAYS,
  TRANSLATE (CHAR(BIGINT(RAND()
* 10000000000)), 'abcdefgHij',
'1234567890')
FROM abonent_;
```

Polecenie insert wprowadza do tabeli abonent 100.000 losowo wygenerowanych rekordów. W celu przeprowadzenia testów ilość rekordów zwielokrotniono (do 1 100 000 rekordów). Zostały skopiowane te same dane, które zostały wcześniej wygenerowane. W tym celu utworzono tabelę tymczasową, z której kopiowano dane do tabeli abonent. Dzięki takiej operacji dane w tabeli abonent powtarzają się i mechanizm kompresji wierszy zadziała.

Utworzono tabelę abonentc, z włączoną kompresją wierszy (tabela jest identyczna jak tabela abonent), do której skopiowano dane z tabeli abonent.

Ponadto dla obu tabel utworzono indeksy, które zostały nałożone na kolumnę nazwisko oraz wplata. Indeksy dla tabeli abonentc zostały również skompresowane.

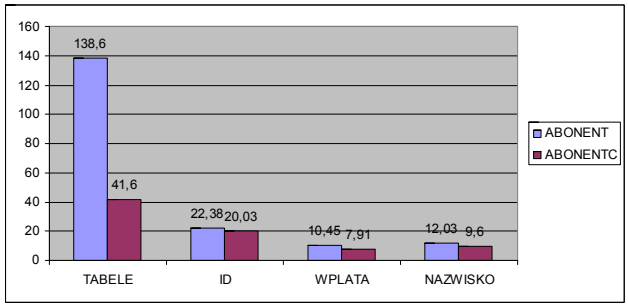
3.1.1. Oszacowanie wielkości danych

Obie tabele zostały umieszczone w osobnych przestrzeniach. W związku z tym możliwe było oszacowanie wielkości obu tabel. Pierwsza tabela, abonent zajmuje około 138,6 MB, natomiast skompresowana tabela abonentc zajmuje 41,6 MB, czyli o 97 MB mniej niż tabela abonent. W przypadku indeksów postąpiono podobnie, czyli indeksy umieszczono w osobnych przestrzeniach, i odpowiednio zajmują one, dla tabeli abonent (rys.1):

- indeks dla kolumny `ID` zajmuje 22,38 MB;
- indeks dla kolumny `nazwisko` zajmuje 12,03 MB;
- indeks dla kolumny `wplata` zajmuje 10,45 MB.

Natomiast dla tabeli `abonentc`:

- indeks dla kolumny `ID` zajmuje 20,03 MB;
- indeks dla kolumny `nazwisko` zajmuje 9,6 MB;
- indeks dla kolumny `wplata` zajmuje 7,91 MB.



Rys. 1 Wykres rozmiarów tabel testowych, przed i po kompresji

Po wprowadzeniu kompresji dla tabeli `abonentc` oszacowano również wielkość słownika kompresji. W tym celu wykonano zapytanie:

```
select tabname, dictionary_size from
SYSIBMADM.ADMINTABINFO where tabname
like 'ABONENT%';
```

dla którego otrzymano wyniki:

```
DICTIONARY_SIZE
ABONENT              0
ABONENTC             61824
```

Z czego wynika, że dla tabeli `abonent` nie został utworzony słownik kompresji, natomiast dla tabeli `abonentc`, słownik kompresji zawiera 61824 bajty. Dla tabel, które nie podlegają kompresji, słownik kompresji nie jest tworzony.

3.1.2. Oszacowanie kosztów wykonania zapytań

W celu obserwacji planu wykonania zapytania należy włączyć i uaktualniać statystyki.

```
RUNSTATS ON <NAZWA TABELI> AND
INDEXES ALL;
```

Jednostka TIMERON jest skumulowanym kosztem i obejmuje wszystkie czynności od początku przetwarzania zapytania.

Tabela wyników TIMERON dla zapytań wykonanych dla obu tabel została zamieszczona poniżej.

Tabela1.
Koszty wykonania zapytań

Lp.	Polecenie SELECT	Liczba pobranych wierszy	TIMERON	
			abonent	*abonentc*
1	`select * from abonent where wplata=6000`	11	68,982	68,982
2	`select * from abonent where wplata>6000`	1034011	18567,050	6310,187

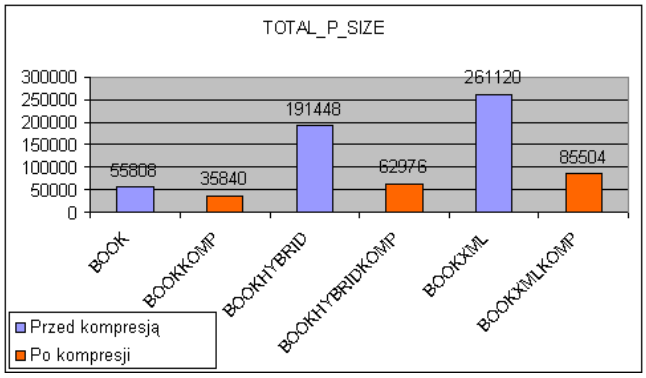
3	Select * from abonent where wplata<6000	65 978	17838,554	5532,752
4	Select * from abonent where wplata<5000 or wplata>6000	1088989	18860,152	6603,289
5	Select * from abonent where wplata>5000 and wplata<6000	10989	17591,054	5382,691
6	Select * from abonent where nazwisko like 'H%';	122738	17696,939	5414,479
7	Select * from abonent where nazwisko not like 'H%';	977262	18567,050	6310,187
8	Select * from abonent where nazwisko>'cią g znaków';	735548	18455,291	6113,963
9	Select * from abonent where nazwisko>'cią g znaków' and wplata<6000;	43637	15036,860	5550,838
10	Select * from abonent where data_wpisu>'1 944-06-11';	1066208	18567,050	6310,187
11	Select * from abonent where data_wpisu<'1 944-06-11';	33737	18569,992	6310,187
12	Select * from abonent where data_wpisu>'1 944-06-11' and wplata<8000;	85404	17945,089	5626,060
13	Select * from abonent where data_wpisu>'1 944-06-11' and wplata<8000 and nazwisko<'cią g znaków';	7062	2504,878	913,498

Jednostki TIMERON dla tabeli ABONENTC jest od 65% do 69% mniejszy od jednostek TIMERON dla tabeli abonent. Pomimo kompresji danych w tabeli ABONENTC, koszt dostępu do danych jest znacznie mniejszy od kosztu dostępu do danych w tabeli abonent. Na podstawie przeprowadzonego eksperymentu można wnioskować, że kompresja danych w bazie IBM DB2 w wersji 9.5 znacznie zmniejsza zajmowany obszar dysku oraz koszt dostępu do danych. Dodatkowe badania wykazały również, że włączenie obu rodzajów kompresji dla tej samej tabeli nie generuje lepszych efektów niż zastosowanie jednej wybranej metody.

3.2. Badanie kompresji danych XML

W rozdziale tym przedstawione zostaną możliwość kompresji tabel zawierających dane w klasycznej postaci relacyjnej, hybrydowej (relacyjne i XML) oraz zawierających tylko dane XML. Do badań wykorzystane zostały trzy pary tabel (BOOK, BOOKHYBRID, BOOKXML), każda para o identycznej strukturze z włączoną i wyłączoną kompresją wierszy [9]. Przyrostek KOMP dołączony do nazwy tabeli oznacza wariant skompresowany. Wszystkie tabele zawierają identyczne, wygenerowane automatycznie dane. W przypadku tabeli hybrydowej oraz XML, część danych lub wszystkie zostały przesunięte z obszaru relacyjnego do dokumentu XML. Pola XML zostały utworzone z opcja inline opisaną w rozdziale 2.

W wyniku przeprowadzonych badań otrzymano rezultaty przedstawione na rys.2.



Rys. 2. Wykres rozmiarów tabel testowych, przed i po kompresji

Porównując dane przedstawione na wykresie (rys. 2) widoczne jest, że w przypadku tabeli relacyjnej rozmiar fizyczny zmniejszył się prawie dwukrotnie, natomiast w przypadku tabeli zawierającej dane typu XML trzykrotnie.

Wydajność zapytań po kompresji danych wzrasta praktycznie dla każdego zapytania. Dla zapytań zawierających kolumny typu XML wydajność wzrasta w znacznym stopniu. Spadek wydajności zaobserwować można w tabelach hybrydowych, dla zapytań zawierających kolumny innego typu niż XML. Spowodowane jest to przechowywaniem wartości XML na stronach tabel, wraz z wcześniej znajdującymi się tam danymi. Skutkuje to znacznym zagęszczeniem danych na stronach tabeli, w porównaniu z tabelą bez kompresji. Jeżeli z tabeli hybrydowej częściej pobierane są dane typu XML, to wydajność zapytań rośnie. W przypadku, gdy częściej pobierane są dane innego typu należy rozważyć opłacalność kompresji dla tak zaprojektowanej tabeli hybrydowej. Zaletą będzie znaczące zmniejszenie rozmiaru tabeli, a wadą spadek wydajności zapytań.

4. Podsumowanie

Ważny nurt badań w zakresie baz danych koncentruje się wokół metod optymalizacji z jednej strony w celu zmniejszenia zajmowanie powierzchni dysku. Ma to szczególne znacznie przy coraz większych baza danych gdzie przetwarzane są miliony rekordów. W ramach tego

nurtu firma IBM zaproponowała w najnowszej wersji serwera baz danych DB2 9.5 metodę kompresji danych opartą na kompresji wierszy, która oprócz znaczącego zmniejszenia przestrzeni dyskowej jednocześnie skraca czas dostępu do skompresowanych informacji. W przeprowadzonych pracach stwierdzono to doświadczalnie na różnych strukturach danych i na różnych sposobach dostępu do tych danych.

LITERATURA

[1] 9 Starter Kit, SDJ Extra Nr 21, Software-Wydawnictwo Sp. z.o.o., 2007

[2] Dokumentacja IBM dla DB2
http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp
http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp
http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db29.doc.xml/db2z_nodetest.htm

[3] Rav Ahuja, Introducing DB2 9, Part 1: Data compression in DB2 9
http://www.ibm.com/developerworks/data/library/techarticle/dm-0605ahuja/index.html?S_TACT=105AGX11&S_CMP=ART#xml

[4] Bob Scheier. DB2 Deep Compression, Part 2
http://www.ibmdatabasemag.com/shared/printableArticle.jhtml?articleID=206800834

[5] Victor Chang, Yun Han Lee, Row compression in DB2 9
http://www.ibm.com/developerworks/data/library/long/dm-0610chang/index.html

[6] Vitor Rodrigues, Matthias Nicola, XMLTABLE by example, Part 1: Retrieving XML data in relational format
http://www.ibm.com/developerworks/data/library/techarticle/dm-0708nicola

[7] Vitor Rodrigues, Matthias Nicola, XMLTABLE by example, Part 2: Common scenarios for using XMLTABLE with DB2
http://www.ibm.com/developerworks/data/library/techarticle/dm-0709nicola

[8] Whei-Jen Chen, Art Sammartino, Db2 9 PureXML Guide, International Technical Support Organization 2007

[9] Radosław Żółtek, Struktury XML w bazie IBM DB2 v9.5, Praca magisterska napisana pod kierunkiem Prof. M.Adamskiego i dr. inż. J.Tkacza, Wydział Elektrotechniki, Informatyki i Telekomunikacji, Uniwersytet Zielonogórski, 2009

dr inż. Agnieszka Węgrzyn
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Podgórna 50
65-246 Zielona Góra
tel.: 68 328 2484
e-mail: A.Wegrzyn@iie.uz.zgora.pl

dr inż. Jacek Tkacz
Uniwersytet Zielonogórski
Wydział Elektrotechniki, Informatyki i Telekomunikacji
Instytut Informatyki i Elektroniki

ul. Podgórna 50
65-246 Zielona Góra
tel.: 68 328 2526
e-mail: J.Tkacz@iie.uz.zgora.pl

dr inż. Adam Wojciechowski
Politechnika Poznańska
Wydział Informatyki i Zarządzania
Instytut Informatyki

ul. Berdychowo
60-965 Poznań
tel.: 61 665 2983
e-mail:
Adam.Wojciechowski@cs.put.poznan.pl