Porównanie algorytmów połączenia relacji NestedLoop oraz SortMergeJoin w zależności od selektywności oraz rozmiaru bufora.

Jędrzej Klorek, Jakub Malczewski

PUT Poznań, 25 lutego 2019

Wprowadzenie

Połączenie w relacyjnej bazie danych jest operacją łączącą krotki relacji. Jednym z najprostszych algorytmów (Hector Garcia-Molina 2011, str. 641), który może zostać wykorzystany do realizacji zadania jest złączenie zagnieżdżone, dale zwane NestedLoop. Jest to nieoptymalny algorytm w wielu sytuacjach połączeniowych choć wciąż przydatny. Klasyczny NestedLoop jest z zasady niezależny od rozmiaru bufora (niezbędna jest jego minimalna ilość) jednak nie jest to regułą pośród algorytmów. Intuicyjnie, wraz ze wzrostem poziomu normalizacji relacji bazy danych, w aplikacji użytkowej może zachodzić coraz silniejsza i liczniejsza potrzeba dokonywania połączeń, co za tym idzie, wydajność staje się kluczowa. Przykładem algorytmu, który często jest wydajniejszy oraz wykorzystuje bufor do przyspieszenia połączenia jest algorytm złączenia oparty na sortowaniu, zwany dalej SortMergeJoin. Optymalizacji wyboru algorytmu połączeniowego na podstawie zapytania dokonuje moduł optymalizatora zapytań. Operacja ta dokonywana jest na podstawie statystyk oraz założeń dokonywanych przez producenta bazy danych. W normalnej pracy moduł dla programisty jest transparentny tj. nie musi zdawać sobie sprawy z jego istnienia. W przypadku optymalizacji zapytań System Relacyjnej Bazy Danych (SRBD) udostępnia narzędzia do podglądu decyzji podjętych przez moduł. Aby móc zweryfikować poprawność decyzji należy porównać ze sobą dostępne algorytmy w zależności od wybranych parametrów, czym zajmiemy się w dalszej cześci artykułu.

Metody

W celu porównania pracy SortMergeJoin z NestedLoopJoin określono parametry opisujące warunki pracy algorytmów, oraz dokonano wyboru ich implementacji. Na tej podstawie powstał program zaimplementowany w języku Python dokonujący symulacji algorytmów (“DB_SortMergeJoin” 2019). Symulator posłużył do zebrania danych, reprezentujących wydajność algorytmu, które zostały opracowane w niniejszym artykule.

Wybranymi parametrami opisującymi warunki pracy algorytmu są rozmiar bloku, rozmiar relacji, selektywność zapytania oraz rozmiar dostępnego bufora. Przyjęty rozmiar bloku to 10 wierszy. Rozmiar relacji został określony na 100.

Przez selektywność zapytania rozumiemy współczynnik obliczony wg. wzoru:

$$Sel = \frac{< \text{ilość krotek wynikowych} >}{< \text{ilość krotek relacji } R > * < \text{ilość krotek relacji } S >}$$

gdzie relacje R i S są łączonymi zbiorami krotek.

Rozmiar bufora określony został w ilości bloków. Dodatkowo bufor zaimplementowany w programie symulujących oczekuje odgórnego podziału pamięci na rzecz relacji R oraz S. Algorytmy połączeniowe często wychodzą od podstawowego algorytmu, NestedLoop stąd obecne są w nim pętla zewnętrzna - iterująca się (zazwyczaj) po kroktach lewej relacji oraz wewnętrzna. W przypadku SortMergeJoin optymalizacja polega na uporządkowaniu danych co pozwala efektywniej sterować przebiegiem iteracji. Aby zbadać zachowanie obu pętli podział bufora jest niezbędny.

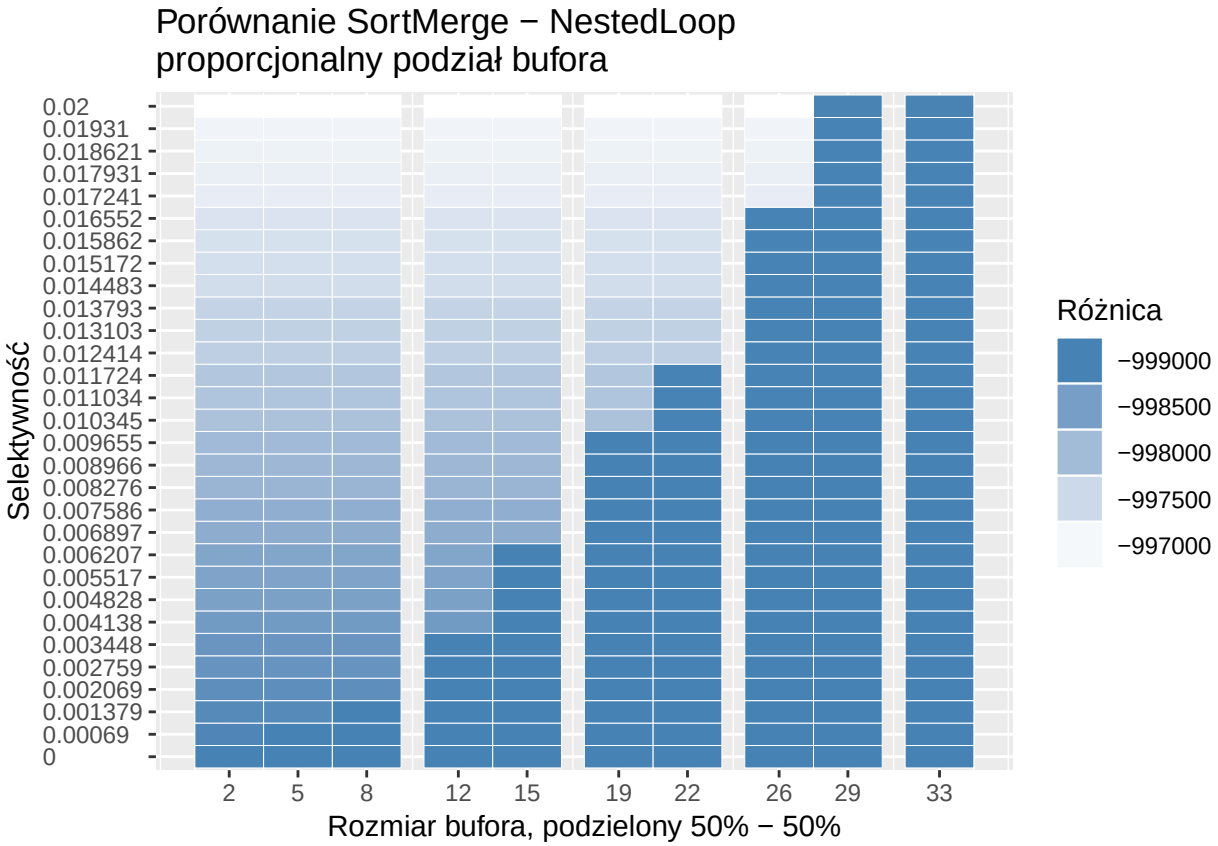
W przypadku SortMergeJoin spodziewanymi punktami wpływającymi na pracę algorytmu są miejsca gdy ilość krotek relacji podlegającymi połączoeniu jest większa niż dostępny bufor.

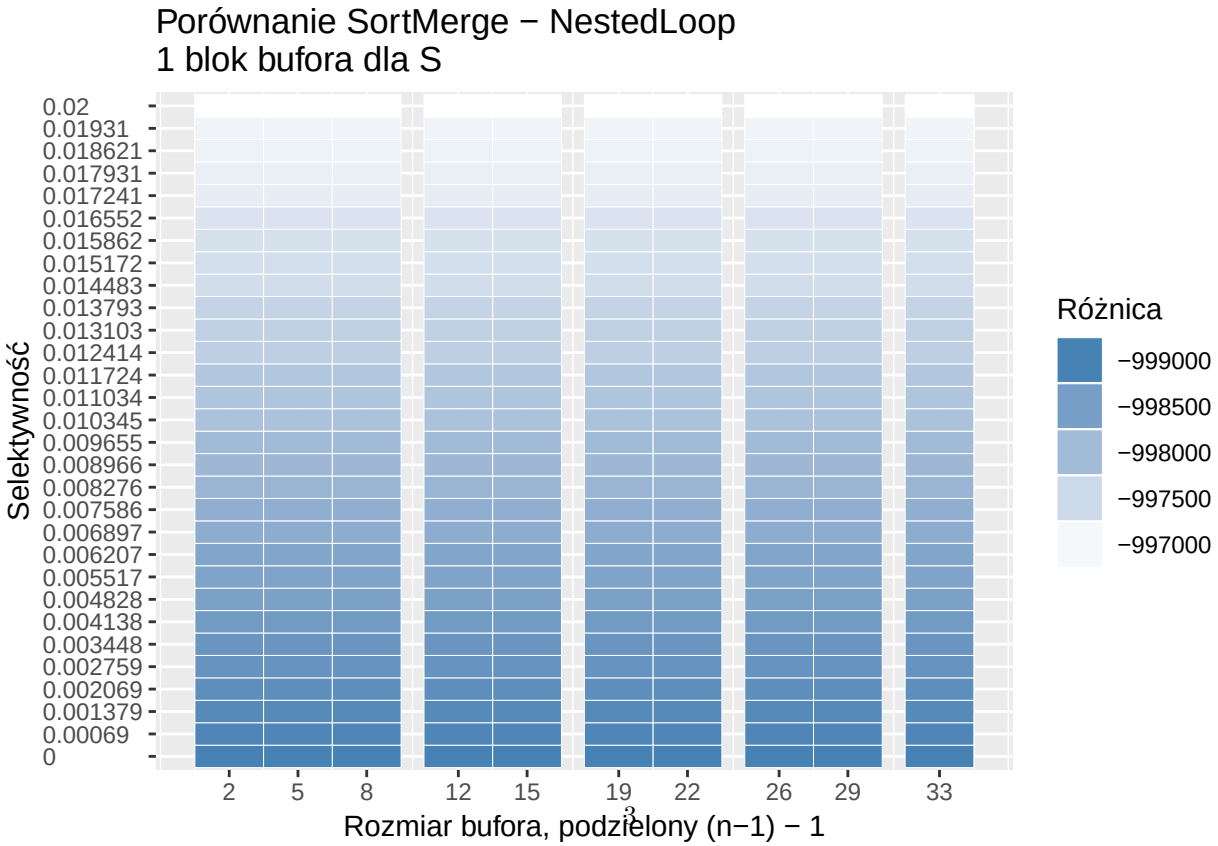
Badanym parametrem algorytmu jest ilość dostępów do dysku w celu odczytania bloku danych należących do przetwarzanej relacji. Całkowicie został pominięty aspekt zapisu wyniku ze względu na złożoność pamięciową zależną wyłącznie od rozmiaru wyniku oraz sposobu dostarczania go do aplikacji użytkowej.

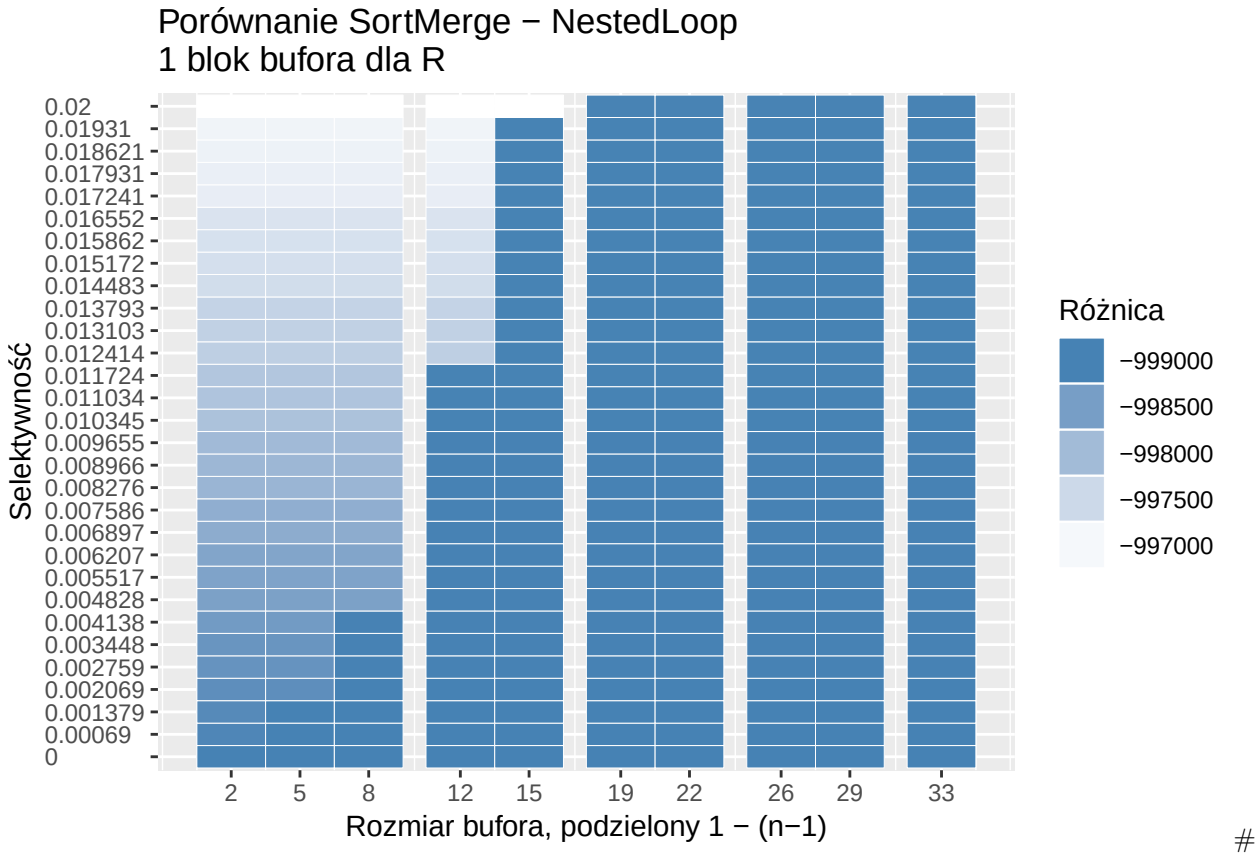
W symulatorze zostały zaimplementowane dwie wersje algorytmu NestedLoop. Pierwsza z nich jest klasyczna, oparta o dwie pętle, druga (BlockNestedLoop) wykorzystuje fakt odczytywania danych w blokach i wykorzystuje trzy pętle. W efekcie pętla przetwarzająca prawą relację nie odczytuje bloków co każdy wiersz lewej relacji a co każdy blok. Implementacja SortMergeJoin została oparta o pseudokod podany przez Héctor García-Molina (Hector Garcia-Molina 2011, str. 650), w tej implementacji zakładamy, że zbiory krotek są już posortowane a ilość dostępów wykorzystywana do sortowanie zostanie obliczona (Hector Garcia-Molina 2011, rozdz. 15.4.7). Dzięki temu uproszczeniu symulator nie musi przetrzymywać zbiorów danych w pamięci operacyjnej w zamian generując je w momencie gdy są potrzebne.

Wyniki

Porównanie NestedLoop do SortMergeJoin - stały przydział bufora







#

Dyskusja

Wnioski

References

"DB_SortMergeJoin." 2019. [online] URL: https://github.com/writ3it/DB_SortMergeJoin.

Hector Garcia-Molina, Jennifer Widom, Jeffrey D. Ullman. 2011. *Systemy Baz Danych: Kompletny Podręcznik.* Wydawnictwo Helion.