Politechnika Poznańska

Instytut Informatyki

Przedmiot**:**

Hurtownie danych i przetwarzanie analityczne

Prowadzący**:**

dr hab. inż. Robert Wrembel prof. nadzw

Temat projektu:

Benchmark TPC-DS

Autorzy:

Groszewski Damian nr indeksu : 138769

Grzebielucha Marcin nr indeksu : 138770

Tulej Wojciech nr indeksu : 138794

1. **Wstęp**

Pomiary wydajności systemów praktycznie nigdy nie są realizowane analitycznie. Zawsze pociągają za sobą konieczność przeprowadzenia eksperymentów. Natomiast eksperyment wymaga symulacji jak najbardziej realistycznego obciążenia wybranej konfiguracji systemu. Rozwiązaniem idealnym byłoby budowanie nowego środowiska eksperymentalnego na potrzeby każdego wdrożenia, ponieważ wdrożenia bywają do siebie podobne, dlatego w przemyśle informatycznym narodziła się dziedzina badania wydajności systemów przy użyciu standardowych benchmarków.

Benchmark jest aplikacją, która maksymalnie obciąża system informatyczny w celu pomiaru jego wydajności. Zwykle benchmarki dzieli się na uniwersalne i aplikacyjne. Benchmarki uniwersalne starają się jednakowo traktować wszystkie elementy systemu, natomiast benchmarki aplikacyjne specyfikują obciążenie systemu modelujące charakterystykę typowej aplikacji z wybranej dziedziny zastosowań, ponieważ wydajność systemu komputerowego może się istotnie różnić dla różnych typów aplikacji. W praktyce benchmarki aplikacyjne cieszą się większym zainteresowaniem, każdy benchmark aplikacyjny powinien spełniać podstawowe kryteria takie jak :

- adekwatność do dziedziny zastosowań,
- implementacja w różnych systemach i architekturach komputerowych,
- skalowalność,
- nieskomplikowany.

W praktyce benchmarki najczęściej służą porównywaniu rozwiązań według następujących kategorii:

- różne oprogramowanie i różny sprzęt dla tej samej klasy aplikacji,
- różne oprogramowanie na tym samym sprzęcie,
- różne komputery należące do tej samej rodziny,
- różne wydania oprogramowania na tym samym komputerze.

Benchmarki implementują najczęściej producenci sprzętu i oprogramowania, którzy chcą udokumentować jakość swoich produktów bądź swoją przewagę nad konkurencją. Czasami są to zespoły wdrażające system informatyczny, które mają możliwość testowania sprzętu wypożyczonego przez dostawcę. Kluczową korzyścią jaką dzięki standardowym benchmarkom uzyskuje konsument jest łatwość porównywania alternatywnych rozwiązań przy pomocy prostych współczynników typu cena/wydajność.

Najpopularniejsze benchmarki SPEC to:

- SPECapc - aplikacje z intensywnym przetwarzaniem grafiki,
- SPEChpc96 - aplikacje przemysłowe na stacjach końcowych,
- SPEComp2001 -aplikacje wykorzystujące OpenMP,
- SPECcpu2000 -intensywne wykorzystywanie procesora,
- SPECweb99 -wydajność serwerów WWW,
- SPECjbb2000 -Java po stronie serwera,
- SPECjvm98 -wydajność maszyny wirtualnej Java,
- SPECsfs97 -wydajność serwerów NFS,
- SPECmail2001 -wydajność serwera poczty elektronicznej,
- BAPCo- opracowuje standardy badania wydajności stacji klasy PC.

Najpopularniejsze benchmarki BAPCo to:

- WebMark - zastosowania internetowe,
- SysMark - aplikacje biurowe,
- MobileMark - notebooki - wydajność akumulatorów,
- SysMarkDB - serwery baz danych.

Z punktu widzenia zastosowań bazodanowych, największe znaczenie ma konsorcjum TPC (Transaction Processing Performance Council), którego celem jest definiowanie standardowych benchmarków dla systemów komputerowych służących do przechowywania danych i przetwarzania transakcji. Założone w 1988 roku TPC zrzesza takich producentów, jak: Acer, Bull, Compaq, Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Netscape, Oracle, SCO, Siemens, SGI, Sun, Sybase, Toshiba. Co kilka lat TPC publikuje nowe specyfikacje benchmarków dla różnych klas systemów baz danych.

2. Wykorzystane technologie

PostgreSQL

PostgreSQL to obiektowo-relacyjny system baz danych o otwartym źródle, wykorzystujący i rozszerzający język SQL w połączeniu z wieloma funkcjami, które bezpiecznie przechowują i skalują nawet najbardziej złożone dane. Zyskał dobrą reputację dzięki swojej sprawdzonej architekturze, niezawodności, integralności danych, rozszerzalności, zestawowi funkcji, a także dzięki zaangażowaniu społeczności open source w tworzenie oprogramowania.

PostgreSQL zawiera wiele funkcji pomagających deweloperom tworzyć aplikacje, administratorów wspiera w problemach ochrony integralności danych i budowaniu środowisk odpornych na awarie. Pomaga w zarządzaniu danymi bez

względu na to, jak duży i mały jest zestaw danych. Można definiować własne typy danych, budować niestandardowe funkcje oraz pisać kod z różnych języków programowania bez przebudowania bazy danych.

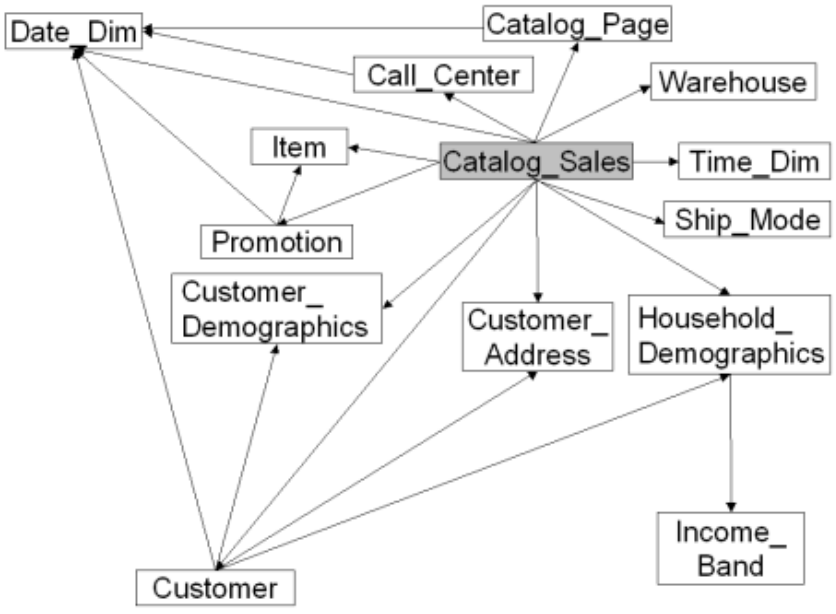
HBase

HBase to rodzaj bazy danych NoSQL. Technicznie rzecz biorąc, HBase jest bardziej składnicą danych niż bazą danych dlatego, że brakuje jej wielu funkcji, które można znaleźć w typowych bazach SQL, takich jak określone typy kolumn, indeksy pomocnicze, wyzwalacze i zaawansowane języki zapytań.
HBase oferuje jednak wiele innych cech, które sprawiają, że jest tak często wykorzystywany w zastosowaniach komercyjnych.
Najważniejsze z tych cech to:

- Skalowalność liniowa i modułowa,
- Ściśle spójny zapis i odczyt,
- Automatyczne przełączanie awaryjne między serwarami RegionSerwer,
- Blokowanie pamięci podręcznej dla zapytań w czasie rzeczywistym,
- Zapytania przekazywane za pomocą filtrów po stronie serwera,
- Obsługa eksportu danych za pomocą podsystemu metryk Hadoop,
- Interfejs JavaAPI.

3. Implementacja benchmarka TPC-DS

Spośród kilku dostarczanych przez firmę TPC schematów wybrany został jeden, który został przedstawiony poniżej.



Rysunek 1 Pierwotny schemat hurtowni danych.

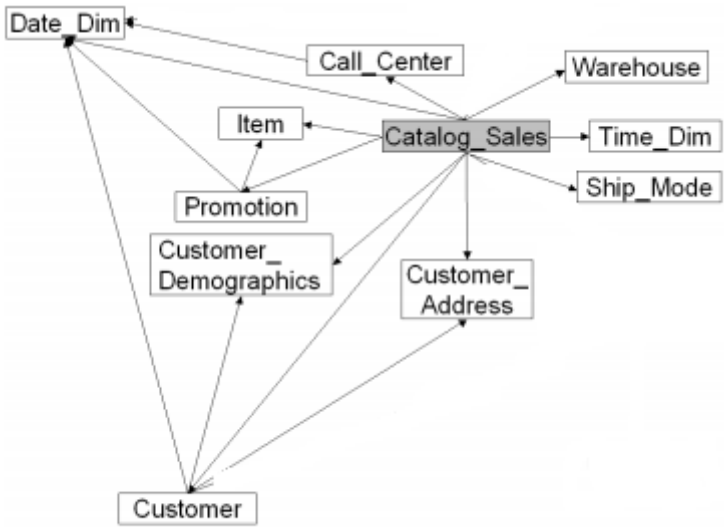
Firma TPC narzuca taki schemat i jest on zawarty w narzędziach przez nią dostarczanych. Znajduję się w pliku tpcds.sql. Fragment jego postaci został przedstawiony na zrzucie ekranu poniżej.

```
create table dbgen_version
(
    dv_version                varchar(16)                   ,
    dv_create_date            date                          ,
    dv_create_time            time                          ,
    dv_cmdline_args           varchar(200)
);

create table customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
    ca_location_type          char(20)                      ,
    primary key (ca_address_sk)
);

create table customer_demographics
(
    cd_demo_sk                integer               not null,
    cd_gender                 char(1)                       ,
    cd_marital_status         char(1)                       ,
    cd_education_status       char(20)                      ,
    cd_purchase_estimate      integer                       ,
    cd_credit_rating          char(10)                      ,
    cd_dep_count              integer                       ,
    cd_dep_employed_count     integer                       ,
    cd_dep_college_count      integer                       ,
    primary key (cd_demo_sk)
);
```

Rysunek 2 Fragment budowy pliku ze schematem benchmarka TPC-DS.

Pierwszym z naszych zadań było uproszczenie podstawowego schematu, tak aby można było je później zaimplementować w bazach danych Postgre i HBase. W celu uproszczenia schematu bazy danych należało połączyć niektóre tabele w jedną spójną całość. W wykorzystywanych narzędziach TPC-DS należało dostosować zawartość pliku tpcds.sql do naszych potrzeb (przenieść kolumny jednych tabel do innych).

Po uproszczeniu schemat hurtowni danych wygląda następująco. Uproszczenie polegało na wcieleniu tabeli Income_Band do tabeli Household_Demographics, a następnie obu już połączonych tabel do głównej tabeli hurtowni danych, czyli do Catalog_Sales. Takie połączenie sprawiło, że główna tabela ma teraz 88 atrybutów.



Rysunek 3 Schemat uproszczonej hurtowni danych.

Do wygenerowania danych testowych zostały użyte narzędzie z oficjalnej strony internetowej TPC.org. Narzędzia pozwalają wygenerować dane w skali od jednego gigabajta do nawet kilku terabajtów. Za pomocą narzędzi można również generować przykładowe rozbudowane zapytania.

Przygotowanie narzędzi do pracy:

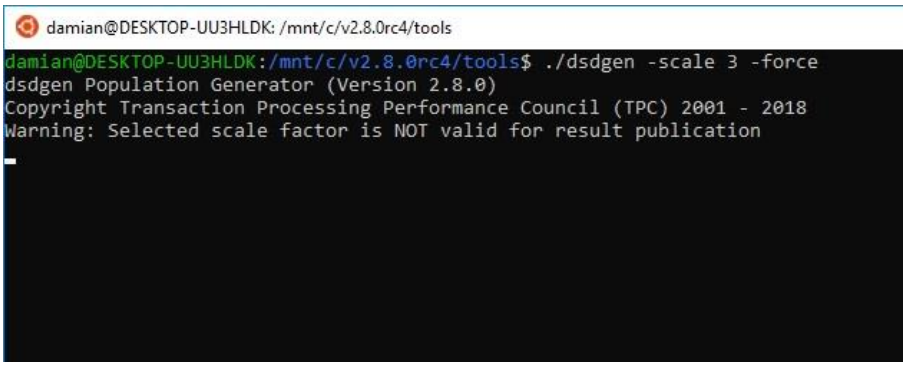
- Pobranie narzędzi z oficjalnej strony TPC.org,
- Rozpakowanie archiwum, które zawiera wszystkie potrzebne pliki,
- Przejście do folderu TPC-DS: cd TPC-DS,
- Przejście do folderu tools: cd tools,
- Najważniejsza część to kompilacja narzędzi za pomocą komendy make dostawca_systemu: make LINUX

Aby wygenerować przykładowe dane należy użyć narzędzia dsdgen. Będąc w folderze z narzędziami używamy komendy:

./dsdgen -scale 3 -force

Za pomocą skali określamy wielkość generowanych danych, w tym przypadku 3GB. Parametr force pozwala zaoszczędzić miejsce na dysku, ponieważ

podmienia wcześniejsze wersje generowanych danych, a jeśli generujemy dane po raz pierwszy to przydzielona zostaje odpowiednia ilość zasobów dysku.



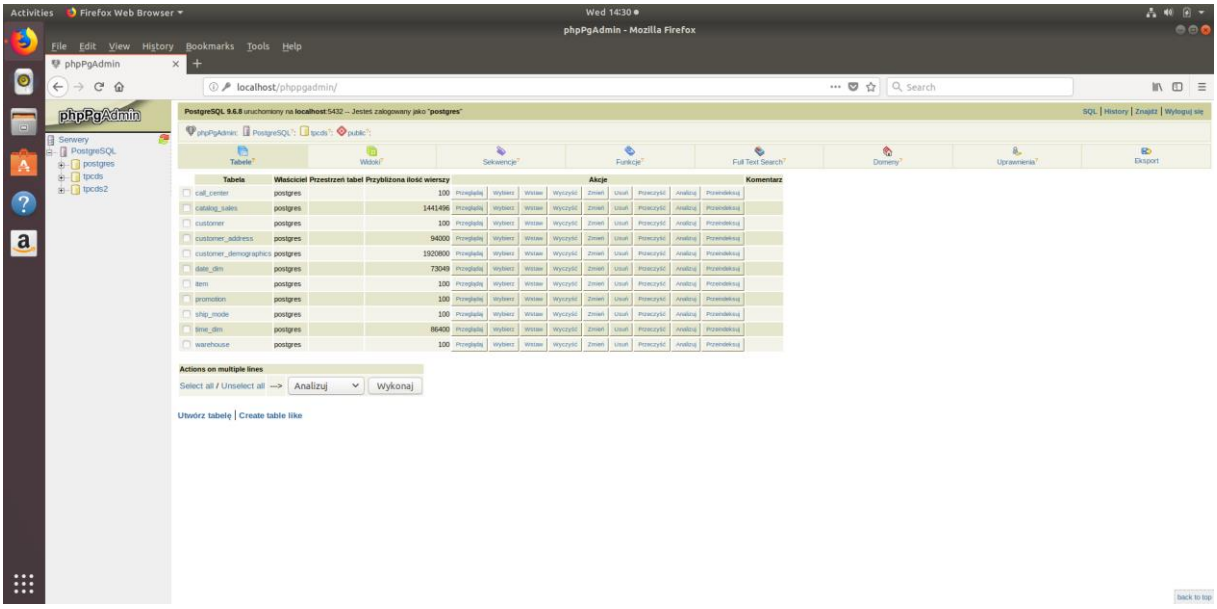
Rysunek 4 Generowanie przykładowych danych za pomocą narzędzi TPC-DS.

W celu ułatwienia oraz przyspieszenia pracy napisany został skrypt, który pozwala stworzyć bazę danych tpcds w systemie PostgreSQL. Jedyne czego wymaga od użytkownika to wprowadzenie niezbędnych parametrów takich jak ,na przykład rozmiar generowanego wolumenu danych.

```
echo "Ladowanie danych do tabel"

psql -c "COPY call_center.dat FROM 'call_center.csv' delimiter
'|' csv;"
psql -c "COPY customer FROM 'customer.csv' delimiter '|' csv;"

psql -c "COPY customer_address FROM 'customer_address.csv'
delimiter '|' csv;"

psql -c "COPY customer_demographics FROM
'customer_demographics.csv' delimiter '|' csv;"
```

Rysunek 5 Fragment skryptu.

Skrypt ten ułatwia tworzenie odpowiedniej bazy danych zgodnej ze schematem benchmarka TPC-DS, o którym wspominaliśmy wcześniej. Przedstawiony poniżej zrzut ekranu z narzędzia PhpPgadmin jest to już działająca baza danych zawierająca wszystkie założenia związane z pierwszym uproszczeniem schematu oryginalnego.



Rysunek 6 Widok tabel uproszczonej hurtowni danych w phpPgAdmin.

Generowanie testowych zapytań odbywa się za pomocą narzędzia dsqen. Używamy do tego komendy:

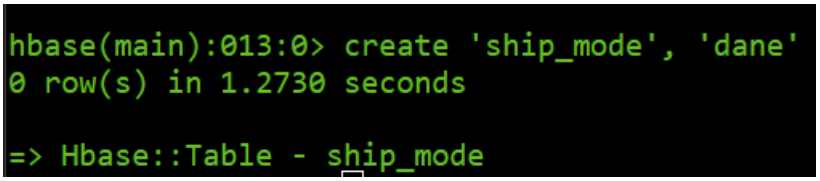
./dsqgen –template query1.tpl –directory query_templates

–dialect postgres –scale 1

Parametr template określa wzorzec jakiego używa, najczęściej ma on związek z wielkością generowanych danych. Kolejny parametr to directory, który określa gdzie zostaną zapisane wygenerowane zapytania. Ważnym parametrem jest także określenie dialektu, ponieważ w SQL mogą występować różnice zależnie od producenta bazy danych.

HBase

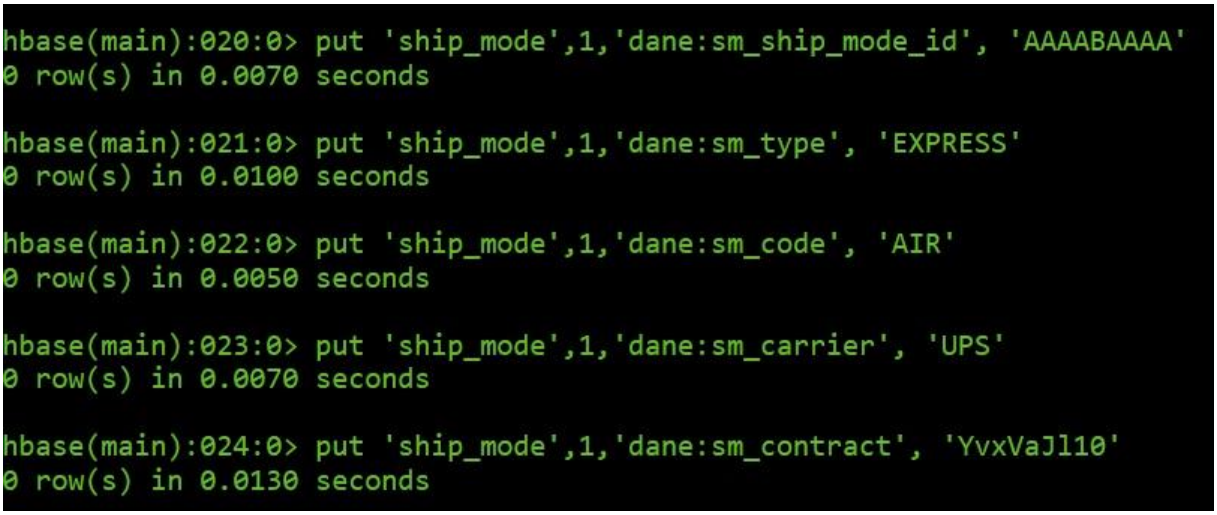
Tworzenie bazy danych HBase jest całkowicie odmienne, ponieważ są to rozwiązania typu NoSQL. Podstawową różnicą w porównaniu do standardu SQL jest zupełnie inna składnia. Przykładowo w NoSQL zamiast polecenia Insert stosujemy put. Kolejną kluczową różnicą jest to, że podczas tworzenia tabel definiujemy rodzinę tabeli, która może zostać wykorzystana do określenia nazw kolumn na wzór znany nam z SQL. Na poniższym zrzucie ekranu widać proces tworzenia jednej z tabel benchmarku TPC-DS - ship_mode, której rodzina nazwana została – dane.



Rysunek 7 Tworzenie tabeli ship_mode w HBase.

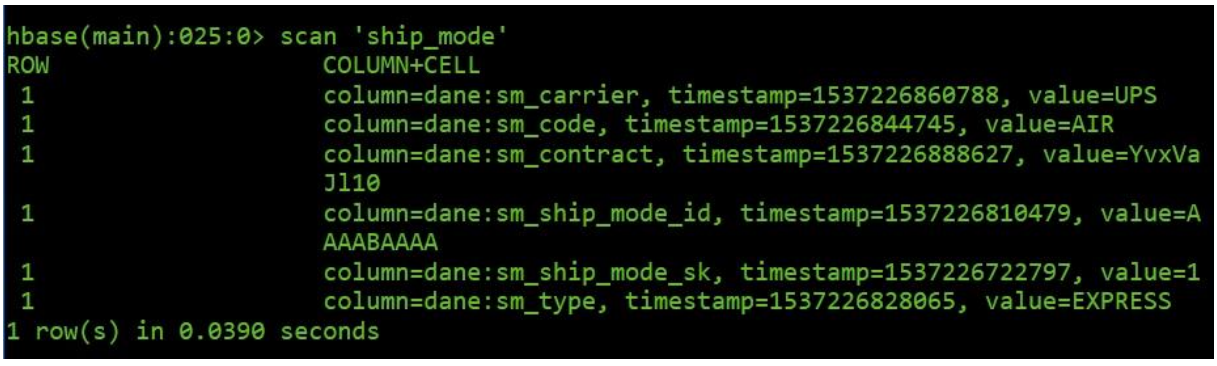
W HBase Shell jest konieczne ręczne tworzenie tabel, ponieważ nie ma opcji uruchamiania skryptów jak to było możliwe w przypadku bazy danych PostgreSQL. Każda tabela musi być tworzona osobno. Jednym z ułatwień jest tutaj właśnie użycie rodziny tabeli do, której później już przy samym wypełnaniu jej danymi możemy odnosić się wybranymi nazwami atrybutów.

Kolejny krok to wypełnienie tabeli danymi. Proces ten polega na przypisaniu nazw kolumn do wybranej rodziny tabeli.



Rysunek 8 Polecenie put do tabeli ship_mode w HBase.

Tyle kroków jest potrzebnych, aby wprowadzić jeden wiersz tabeli. W celu sprawdzenia danych w tabeli używamy polecenia scan ‘nazwa_tabeli’.



Rysunek 9 Wynik polecenia scan na tabeli ship_mode.

Narzędzia dostarczane przez firmę TPC pozwalają generować dane do wybranego przez siebie formatu pliku. Zalecane jest stosowanie plików z rozszerzeń .dat lub .csv. My wybraliśmy .csv, ponieważ zarówno PostgreSQL jak i HBase pozwalają na import danych z tego typu pliku.

Do implementacji bazy danych HBase użyliśmy wirtualnej maszyny HortonWorks, która pozwoliła przyspieszyć pracę, ponieważ ma już zainstalowane środowisko HBase Shell, a także przydatny do wykonania niezbędnych operacji na plikach system HDFS.

Import danych z pliku .csv

W celu załadowania danych z pliku .csv użyte zostało narzędzie implementowane domyślnie w maszynie HortonWorks, które współpracuję z Hbase Shell.

Pierwszym krokiem było przesłanie wygenerowanych przez narzędzia plików do systemu plików maszyny HortonWorks. Następnie zostały one załadowane do rozproszonego systemu plików HDFS, który jest elementem systemu Hadoop, najczęściej używanego w zastosowaniach Big Data.

Komenda pozwalająca przerzucić plik .csv do systemu HDFS.

```
scp nazwa_pliku.csv root@sandbox.hortonworks.com:/home/hbase
```

W tej chwili załadowany plik .csv jest widoczny lokalnie, aby był widoczny później w HBase Shell należy użyć jeszcze jednej komendy, która na czas obecnej sesji będzie przechowywać je w katalogu tymczasowym widocznym dla każdego użytkownika, także dla powłoki HBase Shell.

```
hadoop dfs -copyFromLocal nazwa_pliku.csv /tmp
```

W taki sposób pliki są gotowe do załadowania z poziomu HBase Shell. Pliki ładuję się osobno, każdy po kolei za pomocą bardziej skomplikowanej komendy.

```
hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=, -
Dimporttsv.columns="HBASE_ROW_KEY,lista_atrybutów po przecinku,np. x,y,z "
nazwa_tabeli hdfs://sandbox.hortonworks.com:/tmp/nazwa_pliku.csv
```

W przypadku tej komendy podajemy nazwę tabeli, następnie w ścieżce hdfs nazwę pliku .csv odpowiedniego dla danej tabeli, oraz co najważniejsze musimy wypisać wszystkie atrybuty tabeli do której ładujemy dane – wraz z ich rodziną tabel.

4. Wnioski

Implementacja benchmarku TPC-DS po uproszczeniu podstawowego schematu powiodła się dla baz danych PostgreSQL i HBase. Działania związane z importem danych z plików .csv były trudniejsze w przypadku bazy danych HBase ze względu jej specyficznego podejścia do tworzenia tabel, które jest naturalne dla standardu NoSQL. Drugie uproszczenie schematu nie powiodło się, ponieważ już dość duża główna tabela hurtowni danych po przyjęciu kolejnych atrybutów zaczęła generować błędy związane z niezgodnością kluczy podstawowych i obcych.