|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Politechnika Świętokrzyska w Kielcach Wydział Elektrotechniki, Automatyki i Informatyki** | | | | |
| **Projekt Bazy Danych** | | | | |
|  | **HURTOWNIA DANYCH – SIEĆ SKLEPÓW PRASA, KSIĄŻKA, MUZYKA (EMPIK)** | | **Dominik Szczerek**  **Konrad Moćko**  **GRUPA:2ID14A**  **Kamil Plewa**  **GRUPA:2ID14B** | |
|  | | Data oddania  sprawozdania :  13.06.2016 | | Ocena : |
|  | | | | |

**Wprowadzenie -**

* Projekt 'Sieci sklepów prasa/muzyka/ksiazka - EMPIK' realizowany jest przez hurtownie danych w oparciu o technologię Oracle. Hurtownia danych to zródło danych dla analityków, którzy na podstawie wyników zapytań skierowanych do hurtowni mogą w końcowym rozrachunku, zwiększyć zyski firmy. Oczywiście hurtownia wykonana w ramach projektu jest uproszczona i nie oddaje według mnie realiów prawdziwej hurtowni danych, jednakże nie moje rozważania na ten temat są istotą tego projektu. Hurtownia dotyczy sieci sklepow prasa/ksiazka/muzyka – (Empik). Za tabele faktow zostala wybrana tabela sprzedaz . Zawiera ona odniesienie do wymiaru czasu,miejsca i lokalizacji (tabel wymiarow), a także atrybuty ilosc, wartosc a także typ platnosci (gotowka, karta, bon).Cały schemat bazy danych hurtowni został zaprezentowany ponizej, w punkcie 3. sprawozdania
* 3(jeżeli byłyby problemy z widocznoscia, schemat znajduje się również, w osobnym pliku PLIKI\DANE\er\_model v2.png)

(4.Pliki sql zostały zamieszczone w załączniku)

5.

Na początku wygenerowaliśmy przykładowe dane do tabel stworzonych na potrzeby hurtownii

Po wygenerowaniu wszystkich danych do naszej tabeli przeszliśmy do następnego kroku który polegał na napisaniu plików sterujących **ctl** do każdych z osobna tabel.

Poniżej jest pokazana budowa z dowolnie wybranej tabeli:

Zawartość przykladowego pliku sterującego: dzien***.ctl***

load data

infile 'C:\Users\Szczominik\Desktop\nasze\_2\dzien.csv'

append into table dzien

fields terminated by ';'

(id\_dnia,miesiac,dzien)

Gdy zostały napisane 11 plików sterujących dla każdej tabeli, mogliśmy przejść do załadowania

danych do naszej hurtowni danych.

Żeby można było załadować nasz ***pliki csv*** musieliśmy wywołać narzędzie **sqlldr**. Dokładniej przedstawiam poniżej skrypt który ładuje mi wszystkie ***pliki csv:***

sqlldr hr@localhost/hr control=wojewodztwo.ctl

sqlldr hr@localhost/hr control=miasto.ctl

sqlldr hr@localhost/hr control=sklep.ctl

sqlldr hr@localhost/hr control=rok.ctl

sqlldr hr@localhost/hr control=miesiac.ctl

sqlldr hr@localhost/hr control=dzien.ctl

sqlldr hr@localhost/hr control=pracownik.ctl

sqlldr hr@localhost/hr control=kategoria.ctl

sqlldr hr@localhost/hr control=produkt.ctl

sqlldr hr@localhost/hr control=typ\_platnosci.ctl

sqlldr hr@localhost/hr control=sprzedaz.ctl

Gdzie (***hr***) oznacza – ***nazwa użytkownika***, a (***hr***)oznacza – ***hasło***

Cały ten skrypt jest zapisany w pliku **skrypt.bat** który można później uruchomić w celu załadowania wszystkich plików ***csv*** naraz.

**Poniżej przedstawiam cześć obrazka pokazująca uruchomienie tego skryptu**



Po zakończonym ładowaniu wszystkich plików w katalogu zostały nam utworzone do każdej z tabel osobno pliki z rozszerzeniem ***\*.log.*** Z nich możemy wyczytać np. czy wszystkie rekordy zostały wypełnione i czy nie nastąpiły jakieś bledy.

Poniżej przedstawiam przykładowy plik: ***(dzien.log)***

SQL\*Loader: Release 11.2.0.2.0 - Production on Pt Cze 10 19:54:01 2016

Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.

Control File: dzien.ctl

Data File: C:\Users\Szczominik\Desktop\nasze\_2\dzien.csv

Bad File: dzien.bad

Discard File: none specified

(Allow all discards)

Number to load: ALL

Number to skip: 0

Errors allowed: 50

Bind array: 64 rows, maximum of 256000 bytes

Continuation: none specified

Path used: Conventional

Table DZIEN, loaded from every logical record.

Insert option in effect for this table: APPEND

Column Name Position Len Term Encl Datatype

------------------------------ ---------- ----- ---- ---- ---------------------

ID\_DNIA FIRST \* ; CHARACTER

MIESIAC NEXT \* ; CHARACTER

DZIEN NEXT \* ; CHARACTER

Table DZIEN:

366 Rows successfully loaded.

0 Rows not loaded due to data errors.

0 Rows not loaded because all WHEN clauses were failed.

0 Rows not loaded because all fields were null.

Space allocated for bind array: 49536 bytes(64 rows)

Read buffer bytes: 1048576

Total logical records skipped: 0

Total logical records read: 366

Total logical records rejected: 0

Total logical records discarded: 0

Run began on Pt Cze 10 19:54:01 2016

Run ended on Pt Cze 10 19:54:01 2016

Elapsed time was: 00:00:00.15

CPU time was: 00:00:00.05

(wszystkie pliki csv,ctl i log zostały umieszczone w osobnym folderze)

6.Zapytania do bazy

**ROLL UP**

**--------------JAKIE PRODUKTY KUPIONO W JAKIM MIESCIE--------------------------**

SELECT m.nazwa\_miasta,p.nazwa\_produktu FROM sprzedaz s,sklep w,produkt p,miasto m

where s.sklep=w.id\_sklepu and s.produkt=p.id\_produktu and m.id\_miasta=w.miasto

GROUP BY ROLLUP(m.nazwa\_miasta,p.nazwa\_produktu)

**-------ILOSC PRODUKTOW SPRZEDANYCH W DANYM SKLEPIE-------------------**

SELECT sklep.miasto,produkt.nazwa\_produktu, Sum(ilosc) FROM sprzedaz, produkt,sklep,miasto

WHERE sprzedaz.produkt=produkt.id\_produktu AND sklep.miasto=sprzedaz.sklep

GROUP BY rollup(sklep.miasto, sprzedaz.produkt, nazwa\_produktu) ORDER BY sklep.miasto

**operator ROLLUP** który stanowi rozszerzenie **klauzuli GROUP BY** zapytania SELECT, pozwalające na wyznaczaniu wartości funkcji grupowych na rosnących poziomach agregacji. Dzięki zastosowaniu tego mechanizmu bez konieczności stosowania dodatkowych zapytań możemy w jednym zapytaniu uzyskać podsumowanie . Mechanizm ten zwiększa wydajność aplikacji.

**CUBE**

**-----------------------PRODUKTY PO CENIE MALEJĄCO--------------------------------**

SELECT produkt.nazwa\_produktu, cena, ilosc, (produkt.cena\*sprzedaz.ilosc) AS wpływy FROM sprzedaz, produkt WHERE

sprzedaz.produkt=produkt.id\_produktu GROUP BY cube(sprzedaz.produkt, produkt.nazwa\_produktu, cena, ilosc) ORDER BY sprzedaz.produkt, produkt.nazwa\_produktu;

operator **CUBE**, który stanowi **rozszerzenie klauzuli GROUP BY zapytania SELECT**. Zapytanie wykorzystujące operator **CUBE** jest semantycznie równoważne złożeniu wielu zapytań, stosujących grupowanie według wszystkich kombinacji podanych kolumn.

**GROUPING SETS**

**---------ile wynosila ogolna sprzedaż w danym miesiącu w danym miescie------------------**

SELECT miesiac.id\_miesiaca, rok.rok,miesiac.miesiac, miasto.nazwa\_miasta AS nazwa\_miasta, Sum(sprzedaz.ilosc) FROM miesiac,sklep, miasto, sprzedaz,rok

WHERE miesiac.rok = rok.id\_roku AND miasto.id\_miasta=sklep.miasto AND sklep.id\_sklepu=sprzedaz.sklep

GROUP BY rok.rok, Grouping sets ((miesiac.miesiac, miasto.nazwa\_miasta, miesiac.id\_miesiaca), (miasto.nazwa\_miasta))

ORDER BY rok.rok, miesiac.id\_miesiaca ;

**operator GROUPING SETS**, który umożliwia realizację wielu zadanych schematów grupowania w jednym przebiegu zapytania **SELECT**. Jego argumentem jest lista wielu zestawów kolumn grupujących

**PARTITION BY**

**---------------------------------------------sprzedaz najdroszych towarow-----------------------------------**

SELECT sprzedaz.sklep, miasto.nazwa\_miasta AS "miasto", produkt.nazwa\_produktu AS "nazwa produktu", ilosc, cena, Rank() OVER (PARTITION BY sprzedaz.sklep ORDER BY cena desc) AS ranking

FROM sprzedaz, miasto,sklep, produkt

WHERE miasto.id\_miasta=sklep.miasto AND sklep.id\_sklepu=sprzedaz.sklep AND produkt.id\_produktu=sprzedaz.produkt;

------------------------------**-PO TYPIE PLATNOSCI--**----------------------------------------------------------

SELECT sprzedaz.sklep, typ\_platnosci.nazwa\_platnosci AS "platnosc", produkt.nazwa\_produktu AS "nazwa produktu", ilosc, cena, Rank() OVER (PARTITION BY platnosc ORDER BY cena asc) AS ranking

FROM sprzedaz, miasto,sklep, produkt,typ\_platnosci

WHERE miasto.id\_miasta=sklep.miasto AND sklep.id\_sklepu=sprzedaz.sklep AND produkt.id\_produktu=sprzedaz.produkt AND sprzedaz.platnosc=typ\_platnosci.id\_platnosci;

**FUNKCJE STATYSTYCZNE**

**----statystyczna procentowy udzial poszczegolnych artykułów w ogólnej sprzedazy -------------**

SELECT produkt.nazwa\_produktu AS "Produkt", miasto.nazwa\_miasta AS "Miasto",rok.rok,produkt.cena ,ilosc, Sum(produkt.cena) over (PARTITION BY produkt.nazwa\_produktu) as "suma dochodu",

Round(100\*produkt.cena/(Sum(produkt.cena) over (PARTITION BY produkt.nazwa\_produktu)),5) "UDZIAL w %"

FROM sprzedaz, produkt, miasto, rok,sklep

WHERE produkt.id\_produktu=sprzedaz.produkt AND miasto.id\_miasta=sklep.miasto AND sklep.id\_sklepu=sprzedaz.sklep

ORDER BY produkt.nazwa\_produktu, rok.rok;

7.Brak zrobionego interfejsu

8.**WNIOSKI /PODSUMOWANIE**

Hurtownie danych, takie choćby jak i i nasza to świetny sposob na podsumowanie zdobytej wiedzy z baz danych a także jej zwiększenie. Mamy nadzieje, ze przez ten projekt udalo nam zrealizowac/ zaimplementowac hurtownie dancyh na temat sieci sklepow Empik, której choc fragment można byloby wykorzystac w realnej hurtowni, nie tylko takiej na potrzeby uczelni.