Dydaktyka Informatyki 12(2017)
ISSN 2083-3156
DOI: 10.15584/di.2017.12.24
http://www.di.univ.rzeszow.pl

Wydział
Matematyczno-Przyrodniczy UR
Laboratorium Zagadnień
Społeczeństwa Informacyjnego

Jacek WOŁOSZYN

Dr inż., Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 29, 26-600 Radom; jacek@delta.pl

KONCEPCJA MODELU STRUMIENIOWEGO POBIERANIA DANYCH W CZASIE RZECZYWISTYM NOTOWAŃ GIEŁDOWYCH

THE CONCEPT MODEL OF DOWNLOADING DATA STRIMING IN REAL TIME STOCK QUOTES

Słowa kluczowe: baza danych, giełda, pobieranie danych, model.
Keywords: databases, stock exchange, downloading, model.

Streszczenie

W poniższym artykule przedstawiono model koncepcyjny pobierania danych transakcji giełdowych w czasie rzeczywistym.

Summary

The following article presents a conceptual model of data collection exchange transactions in real time.

Wstęp

Opisane zagadnienie jest wstępem do budowy dużego złożonego projektu realizującego proces kupna-sprzedaży akcji na giełdzie papierów wartościowych w czasie rzeczywistym. W artykule tym pominięte zostaną wszelkie zagadnienia związane z budową całej infrastruktury, modelem bazy, użytkownikami, autoryzacją użytkownika, zarządzaniem portfelami itd. Opisany zostanie proces pobierania danych i gromadzenie ich w bazie danych na serwerze lokalnym.

Dane źródłowe

Aktualne dane giełdowe dotyczące transakcji są udostępniane na zasadzie połączenia z bazą danych na zdalnym serwerze, gdzie są gromadzone na bieżąco w sposób przyrostowy. Każda transakcja ma swój unikalny numer id, co narzuca od razu sposób postępowania przy pobieraniu danych.

Przykład:

SELECT * FROM tabela WHERE id > [ostatnio_pobrany_id][1]

Polecenie pozwala na odczyt z bazy. Powtarzanie go cyklicznie umożliwia pobieranie zbioru rekordów bez ich powtarzania się, a tym samym bez przypadkowego uruchomienia powtórnie transakcji. Jedynym warunkiem powinno być przechowywanie w zmiennej numeru id ostatniego odczytanego rekordu.

Tabela zawiera także informacje dotyczące nazwy instrumentu, daty, czasu, numeru transakcji, kursu, wolumenu, wartości itd.

Przykładowa odpowiedź na zapytanie SQL wygląda następująco:

90169528	1482509099		PLMSTSD00019		POLIMEXMS		2016-12-23		
17:04:59	570	3.67	125	458.75	03	A	XWAR		
90169527	1482509099		PLMSTSD00019		POLIMEXMS		2016-12-23		
17:04:59	569	3.67	350	1284.5	03	A	XWAR		
90169526	1482509099		PLMSTSD00019		POLIMEXMS		2016-12-23		
17:04:59	568	3.67	41	150.47	03	A	XWAR		
90169525	1482509099		PLMSTSD00019		POLIMEXMS		2016-12-23		
17:04:59	567	3.67	350	1284.5	03	A	XWAR		
90169524	1482509099		PLMSTSD00019		POLIMEXMS		2016-12-23		
17:04:59	566	3.67	134	491.78	03	A	XWAR		
90169523	1482509097		PLBSK0000017		INGBSK		2016-12-23		
17:04:57	182	157	150	23550	02	A	XWAR		
90169522	1482509096		PLCMPLD00016		SYGNITY		2016-12-23		
17:04:56	41	4.87	200	974	10	A	XWAR		
90169521	1482509094		PLPTRLI00018		PETROLINV		2016-12-23		
17:04:54	630	9.92	137	1359.04	10	A	XWAR		
90169520	1482509094		PLPTRLI00018		PETROLINV		2016-12-23		
17:04:54	629	9.92	1702	16883.84		10	A	XWAR	
90169519	1482509092		PLZATRM00012		GRUPAAZOTY		2016-12-23		
17:04:52	446	62	305	18910	02	A	XWAR		

Rys. 1. Przykład strumienia danych źródłowych

[1] DuBois Paul MySQL Cookbook, O'Reilly 2014.

gdzie kolejne pola oznaczają numer id, znacznik czasu, kod ISIN, nazwę instrumentu, datę transakcji, czas transakcji, numer transakcji, kurs, wolumen, wartość, grupę, kategorię, kod rynku.

Opis problemu

Ogólna koncepcja polega na utworzeniu bazy danych w lokalnym systemie i jej uaktualnianiu wraz z nadchodzącymi transakcjami z giełdy. Takie rozwiązanie zapewni lepszą wydajność systemu, pełen nadzór nad stanem systemu, pełne zarządzanie infrastrukturą i nie będzie obciążać źródłowego systemu dostarczania danych.

Zakładamy, że użytkownicy będą mieli dostęp do lokalnego systemu w ramach utworzonego dla nich profilowanego konta, a tym samym dostęp do aktualnych notowań oraz możliwości zawierania transakcji kupna-sprzedaży.

Oznacza to, że konieczne jest utworzenie w lokalnej bazie danych struktury tabeli, która będzie pobierała dane otwarcia z każdego dnia, a kolejne nadchodzące transakcje będą aktualizowały kursy akcji wraz z kolejnymi transakcjami realizowanymi na giełdzie.

Rozwiązanie problemu

Do utworzenia lokalnego systemu konieczne jest utworzenie tabeli[2] w lokalnej bazie danych, która będzie przechowywała dane wartości instrumentów giełdowych:

Przykład struktury tabeli:

```
nazwa = models.CharField(max_length=50, verbose_name="Nazwa instrumentu")
data = models.DateField(verbose_name="Data notowania")
otw = models.FloatField(verbose_name="Otwarcie")
maxim = models.FloatField(verbose_name="Maximum")
minim = models.FloatField(verbose_name="Minimum")
kurs = models.FloatField(verbose_name="Kurs")
wolum = models.FloatField(verbose_name="Wolumen obrotu")
itype = models.CharField(max_length=30, verbose_name="Typ Instrumentu")
iexch = models.CharField( max_length=30, verbose_name="Giełda")
ifeer = models.FloatField(verbose_name="Stawka opłaty")
```

Rys. 2. Struktura tabeli przechowująca dane o instrumentach w bazie lokalnej

[2] Tamże.

Wynikiem zapytania

```
SELECT * FROM wallstreet_notowaniaakt LIMIT 10
```

do tak utworzonej tabeli jest:

1	01CYBATON	2016-10-25	0.08	0.08	0.07	0.08	
	1363577Akcje	GPW	0.0019				
2	06MAGNA	2016-12-22	0	0	0	0.83	0
	Akcje	GPW	0.0019				
3	08OCTAVA	2016-10-25	1.07	1.07	1.07	1.07	476
	Akcje	GPW	0.0019				
4	11BIT	2016-12-22	126.4	126.4	119.25	145.55	6282
	Akcje	GPW	0.0019				
5	2CPARTNER	2016-10-25	0.53	0.53	0.53	0.5	242
	Akcje	GPW	0.0019				
6	2INTELLECT	2016-10-25	0	0	0	0.24	0
	Akcje	GPW	0.0019				
7	4FUNMEDIA	2016-10-26	5.66	5.66	5.46	5.65	72
	Akcje	GPW	0.0019				
8	5THAVENUE	2016-10-25	1.4	1.42	1.4	1.44	200
	Akcje	GPW	0.0019				
9	8FORMULA	2016-07-04	0	0	0	0.46	0
	Akcje	GPW	0.0019				
10	AATHOLD	2016-10-26	20.65	20.65	19.73	20.65	65
	Akcje	GPW	0.0019				

Rys. 3. Listing odczytu rekordów z bazy danych

gdzie poszczególne pola są zgodne ze strukturą przedstawioną w opisie jak na rys. 2.

Procedura zapisu poszczególnych rekordów do bazy danych jest mało skomplikowana i polega na utworzeniu poszczególnych rekordów z wykorzystaniem pętli[3], której działanie kończy się wraz z pobraniem ostatniego rekordu.

Należy oczywiście pamiętać o utworzeniu połączenia z bazą danych

```
dbrdf = MySQLdb.connect(host = 'IP/or domain',
                        port = 33**,
                        user = 'user',
                        passwd = '********',
                        db = 'db1')
```

oraz zapytaniu do bazy, które w tym przypadku wygląda następująco:

[3] A. Downey, *Python for Software Design*, Cambridge University Press 2009.

```
SELECT
        n.id, n.nazwa,  n.data, n.open,  n.max, n.min, o.kurs, n.wolumen,
            n.kateg
FROM
    notowania as n INNER JOIN kurs_odn
            as o on n.isin=o.isin
WHERE
        o.`data`= CURDATE() and n.`data`= CURDATE() and n.kateg="A"
```

Rys. 4. Zapytanie SQL odczytujące wybrane dane z bazy zdalnej

Otrzymane w ten sposób dane są zapisywane do lokalnej tabeli w kolejnych rekordach. Istotne jest, aby zapamiętać po skończonym zapisie id ostatniej transakcji, gdyż w ten sposób pobieranie danych zacznie się od nowych rekordów.

```
Notowania.objects.filter(nazwa=row[1]).update(nazwa=row[1], data=row[2],
otw=row[3], maxim=row[4], minim=row[5], kurs=row[6], wolum=row[7], ity-
pe='Akcje',iexch='GPW', ifeer=0.0019)
```

Rys. 5. Modyfikacja wartości notowań otwarcia dnia

Mając tak przygotowaną podstawową tabelę z notowaniami, uaktualnianą każdego dnia, należy zadbać o update jej poszczególnych rekordów wraz z nadchodzącymi informacjami o transakcjach[4] – rys. 5.

Realizacja tego zadania jest bardzo prosta. Należy każdy rekord otrzymany z zapytania wykorzystać do wykonania zapisu aktualnych danych:

```
Notowania.objects.filter(nazwa=z.nazwa).update(kurs=z.kurs, data=z.data)
```

Rys. 6. Modyfikacja tabeli notowania dla bieżących transakcji

W ten sposób użytkownik wyświetlający tabelę notowań w celu dokonania transakcji zawsze widzi jej aktualną wartość.

Notowania

Notowania

KOD	Nazwa	Data	Kurs otwarcia	Kurs maksymalny	Kurs minimalny	Kurs aktualny	Wolumen		
1	01CYBATON	25-10-2016	0,08	0,08	0,07	0,08	1363577,00	0,00%	KUP
2	06MAGNA	22-12-2016	0,0	0,0	0,0	0,83	0,00	0,00%	KUP
3	08OCTAVA	25-10-2016	1,07	1,07	1,07	1,07	476,00	0,00%	KUP
4	11BIT	22-12-2016	126,4	126,4	119,25	145,55	6282,00	15,15%	KUP

Rys. 7. Przykład prezentacji danych z tabeli

[4] M. Goodrich, R. Tamassia, M. Goldwasser, *Data Structures and Algorithms in Python*, Wiley 2013.

Wnioski

Przy pobieraniu strumienia danych ze zdalnego źródła zdecydowano się na utworzenie lokalnej bazy z tabelą, która będzie przechowywać gromadzone dane. Takie rozwiązanie ma swoje zalety i wady. Do wad możemy zaliczyć konieczność tworzenia bazy i tym samym obciążania serwera dodatkowymi zasobami co wraz ze wzrostem pojemności bazy, czyli zapisem coraz większej ilości danych będzie znacznym obciążeniem dla serwera. Jednak z drugiej strony stanowi to pewnego rodzaju uniezależnienie od warunków zewnętrznych, związanych z transmisją danych. W przypadku zakłóceń transmisji, a takie zapewne w czasie życia projektu pojawią się dane tracone byłyby bezpowrotnie, co w przypadku złożenia zlecenia kupna/sprzedaży przez użytkownika systemu uniemożliwiłoby transakcję. Zastosowanie natomiast przyjętego rozwiązania uniemożliwia pomięcie jakiejkolwiek transakcji, co najwyżej nastąpi ono opóźnione w czasie. Ponadto gromadzenia danych pozwala na odtworzenie stanu bazy w przypadku awarii lub wystąpienia błędu w systemie.

Literatura

Downey A., *Python for Software Design*, Cambridge University Press 2009.
Downey A., *Think Python*, O'Reilly 2012.
Goodrich M., Tamassia R., Goldwasser M., *Data Structures and Algorithms in Python*, Wiley 2013.
DuBois P., *MySQL Cookbook*, O'Reilly 2014.
Hellman D., *The Python Standard Library by Example*, Addison-Wesley 2011.
Hilpisch Y., *Derivatives Analytics with Python*, Wiley 2015.
Payne J., *Beginning Python*, Wrox 2010.