Załadowanie danych do bazy POSTGRESQL

Pliki w formacie tekstowym wczytywane są po ich otrzymaniu z serwera wewnętrznego firmy. W tym momencie wykonywana jest konwersja kodowania na stronę kodową iso88592 (obowiązujący standard kodowania polskich znaków na stronach www tzw. iso-latin2). Proces ten wywoływany jest również jako cykliczne zadanie w crontab'ie następującym wpisem w plik konfiguracyjny:

```
# zasilenie dla emarket codziennie o 20:30

30 20 * * 1,2,3,4,5,6,7 /home/bazy/emarket/zaladunek 2>&1 | /var/qmai1/bin/qmai1-inject
```

Skrypt „załadunek" zbudowany jest podobnie jak skrypt konwersji danych na maszynie wewnętrznej i wykonywane działania wyświetla na standardowe wyjście (stdout) co jest poprzez sposób wywołania w corntab wysyłane jako raport do operatora. Jedyną różnicą w wywołaniu skryptu jest to, iż działa on na tej samej maszynie na której zainstalowany został system pocztowy Qmail i dlatego można było wykorzystać jeden z pocztowych programów usługowych do dostarczenia poczty bezpośrednio do adresata. Działania takie maja na celu informowanie obsługi lub administratorów o wszelkich występujących problemach a także o powodzeniu każdego z etapów działania sklepu internetowego. Poniższy skrypt prezentuje działania wykonywane w czasie importowania danych do bazy PostgreSQL:

Skrypt służy do załadowania danych do bazy danych eMarket. Jest to przykład skryptu automatyzującego proces wczytywania i konwersji danych do odpowiednich formatów przed załadowaniem ich do systemu. Poniżej przedstawiam omówienie jego przydatności oraz poszczególnych kroków:

1. Wysyłanie powiadomienia e-mail

Skrypt rozpoczyna się od wysłania e-maila, który informuje o rozpoczęciu procesu załadowania danych do bazy eMarket. Dzięki temu administrator lub odpowiednia osoba odpowiedzialna za system może monitorować stan procesu.

2. Definicja katalogów roboczych

Zdefiniowanie katalogów roboczych i ustawienie zmiennych systemowych, takich jak KAT_TMP i BINARIA, pozwala na łatwe zarządzanie lokalizacjami plików oraz narzędziami używanymi w procesie.

3. Sprawdzanie obecności wymaganych plików

Skrypt sprawdza, czy w katalogu roboczym znajdują się pliki wejściowe (z rozszerzeniem .IN). Jeśli pliki nie zostaną znalezione, skrypt przerywa działanie i wysyła powiadomienie o błędzie. To zapewnia, że proces nie rozpocznie się, jeśli brakuje niezbędnych danych.

4. Konwersja kodowania znaków

Skrypt dokonuje konwersji znaków w pliku CENNIK.IN z jednego formatu (LAT) na inny (ISO). Jest to ważne, gdyż różne systemy mogą używać różnych standardów kodowania, a konwersja zapewnia spójność danych.

5. Zablokowanie dostępu do bazy

Skrypt blokuje dostęp do bazy danych na czas wczytywania danych. Tworzenie pliku tymczasowego (emarket.tmp) zapewnia, że nikt nie będzie mógł modyfikować bazy danych podczas załadowania nowych danych, co zapobiega możliwym błędom lub kolizjom.

6. Wykonywanie skryptów załadunkowych

Po zablokowaniu dostępu do bazy, skrypt uruchamia serię skryptów, które odpowiadają za:

- **Usuwanie starych tabel** (90-usuntabełe),
- **Tworzenie nowych tabel** (20-załoztabełe),
- **Załadowanie danych** (30-załadujtabełe),
- **Tworzenie indeksów** (40-załozindeksy),
- **Ładowanie katalogu** (40-załadujkatałog),
- **Optymalizację** (45-optymałizuj),
- **Generowanie statystyk** (50-statystyka).

Te kroki są kluczowe dla prawidłowego załadowania danych i ich późniejszego przetwarzania w systemie.

7. Zakończenie procesu

Na końcu skryptu zostaje usunięty plik tymczasowy, który blokował dostęp do bazy, a także użytkownicy ponownie uzyskują dostęp do systemu. Skrypt informuje o zakończeniu procesu załadunku danych.

Przydatność skryptu:

Skrypt ten jest przydatny w systemach, które wymagają regularnego i zautomatyzowanego załadowania danych do bazy. Automatyzacja tych procesów zmniejsza ryzyko błędów ludzkich i zapewnia płynność operacji. Skrypt wspiera także konwersję danych, co jest istotne, gdy systemy używają różnych kodowań znaków. Dodatkowo, blokowanie dostępu do bazy podczas załadowania danych zapewnia integralność danych i chroni przed niepożądanymi zmianami w trakcie procesu.

Etapy załadunku danych do bazy zostały podzielone na osobne funkcjonalne części, co ułatwia ewentualne diagnozowanie nieprawidłowości oraz dalszy rozwój tego oprogramowania. Każdy ze skryptów wykonujących działania na bazie PostgreSQL otrzymał nazwę zbudowaną z początkowych dwóch cyfr

oznaczających jego logiczną kolejność w wykonaniu, analogicznie jak to ma miejsce w przypadku skryptów startowych w systemach UNIX typu SysV 4.2 gdzie takie uporządkowanie występuje oraz w systemach BSD w skryptach administracyjnych (daily, weekly, monthly itp.). Omówienie poszczególnych skryptów SQL zostało umieszczone w kolejnych rozdziałach traktujących o samej strukturze bazy danych. Wspomnieć należy, iż aby wykorzystywać bazę danych PostgreSQL, administrator bazy założył odpowiednich użytkowników bazy z hasłami dostępu i uprawnieniami pozwalającymi na czynności administracyjne. Głównym plikiem konfiguracyjnym PostgreSQL regulującym dostęp określonych użytkowników z określonych hostów jest plik konfiguracyjny „pg_hba.conf\ Dokładnie omawia to dokumentacja PostgreSQL. Sposób autoryzacji (czy system wymaga hasła czy też nie) jest definiowany w tym pliku. Na jego końcu przy domyślnej konfiguracji znajdują się wpisy: local all trust

host all 127.0.0.1 255.255.255.255 trust_____

Oznaczają one, że dla połączeń lokalnych (local) – czyli takich, gdzie łączymy się nie korzystając z socket'ów tcpip i dla dowolnej bazy (all), system ma przyjąć regułę nie pytania o hasło (trust) natomiast dla połączeń zdalnych, ale tylko z maszyny o adresie 127.0.0.1 (czyli z tego samego hosta, ale przez sockety tcpip), będzie obowiązywać ta sama reguła. Aby to zmienić należy zastąpić ostatnie słowo (trust) na „password" lub „crypt" (różnią się one metodą przesyłania hasła). Przykładowo zapis:

host all 192.168.1.10 255.255.255.0 password
~

Oznacza, że osoby łączące się z serwera o adresie 192.168.1.10 oraz z całej klasy C (czyli

w rzeczywistości adres ip może być typu 192.168.1.*) muszą podać hasło aby dostać się do dowolnej bazy. Chcąc wymusić aby

wzorcowy szablon bazy PostgreSQL był także chroniony można dokonać zapisu: host template1 192.168.1.10 255.255.255.0 password

co oznacza, że te same osoby będą mogły teraz dostać się tylko do bazy template1 i będą musiały podać hasło. Metod autoryzacji użytkowników jest wiele, między innymi także na podstawie protokołu ident. Przy pisaniu reguł dostępu należy pamiętać o kolejności. Zawsze użyta zostanie ta reguła która jest pierwsza w pliku i pasuje do sytuacji (regułki przeszukiwane są w kolejności od początku do końca pliku aż do znalezienia pierwszej

pasującej i na tym się kończy przeszukiwanie). Poniższy zapis byłby błędny, pozbawiony sensu poprzez swoją błędną hierarchię:

local all trust

host all 127.0.0.1 255.255.255.255 trust local templatel password

host templatel l27.0.0.1 255.255.255.255 password

Poprawnej konfigurację tej access listy przedstawia przykładowy zapis: local templatel password

host templatel l27.0.0.1 255.255.255.255 password local all trust

host all l27.0.0.l 255.255.255.255 trust_____

W przypadku konfiguracji rzeczywistej sklepu wpisy dopuszczające ustalają dostęp do bazy sklepu nazwanej jako „emarket” tylko i wyłącznie z serwera na którym działa sklep oraz dla określonego użytkownika z hasłem. Zapewnia to już zwiększony poziom bezpieczeństwa poprzez odrzucenie przez silnik bazy danych połączeń pochodzących zarówno od użytkowników lokalnych serwera jak i zdalne próby nawiązania

połączeń (choć to także zabezpiecza zastosowany firewall). PostgreSQL zapewnia ponadto zabezpieczenia dostępu do tabel zgodnie ze standardem SQL92 (www.postaresal .org/idocs/index.php?sq1 .html). Na zasadzie przywilejów na konkretne działania i operacje na krotkach, tabelach, indeksach, procedurach czy funkcjach. Można tu bardzo precyzyjnie określać co, kto jak i kiedy może uczynić ze zgromadzonymi danymi, czy może tylko czytać nasze dane czy też usuwać, poprawiać, dodawać nowe itp.