STUDIA INFORMATICA 2013
Volume 34 Number 2B (112)

Wanda GRYGLEWICZ-KACERKA
Państwowa Wyższa Szkoła Zawodowa we Włocławku
Jarosław KACERKA
Politechnika Łódzka, Instytut Automatyki

MODEL HURTOWNI DANYCH ZASILANEJ W CZASIE RZECZYWISTYM, NA PODSTAWIE TECHNOLOGII ORACLE

Streszczenie. Artykuł obejmuje analizę metod ekstrakcji danych oraz wybór najszybszej metody odczytu zmienionych danych, a także ich przeniesienia do hurtowni danych celem przeprowadzenia analiz biznesowych. Przeniesienie oznacza zasilanie hurtowni danymi okresowo (cyklicznie) ładowanymi oraz danymi bieżącymi. W pracy zaprezentowano własną koncepcję rozwiązania zagadnienia, potwierdzającą, że przyjęty prototyp rozwiązania zagadnienia spełnia stawiane przed nim wymagania.

Słowa kluczowe: hurtownia danych, wspomaganie decyzji biznesowych

ORACLE BASED MODEL OF A DATA WAREHAUSE FED IN REAL-TIME

Summary. The paper presents an analysis of the methods of data extraction and the possibility of moving the extracted data into a data warehouse in order to perform business analysis in real time. This means periodic and up-to-date supply of the data warehouse. Presented is a concept of solution of the problem confirming that the proposed prototype solution meets the requirements.

Keywords: data warehouse, support of business decisions

1. Wstęp

Celem biznesowym jest aktualizacja danych w bazie docelowej za pomocą danych zmienionych w bazie źródłowej, przy przyjęciu założenia, że priorytetem jest uzyskanie możliwie najszybciej aktualnego stanu danych w bazie docelowej. W opisanym powyżej modelu istnie-

ją dwa główne zadania. Po pierwsze, identyfikacja zmian w bazie źródłowej istotnych dla bazy hurtowni, a po drugie, udostępnienie wyekstrahowanych danych w formie umożliwiającej efektywne wczytanie ich do bazy hurtowni.

Praca ma na celu przeprowadzenie analizy możliwości zastosowania najbardziej przydatnej metody ekstrakcji danych do hurtowni w możliwie najszybszym czasie.

Jedynym sposobem, gwarantującym aktualność danych w bazach hurtowni, jest wykonywanie z maksymalną częstotliwością procesu eksportu, ograniczonego tylko do zmienionych danych. Problem wychwytywania na bieżąco zmian w bazie okazał się na tyle istotny, że Oracle wbudował w bazę danych dedykowany do tego celu mechanizm *Change Data Capture* CDC.

Praca zawiera projekt implementacji rozwiązania problemu z wykorzystaniem mechanizmu CDC. Badania doświadczalne wykonane zostały dla danych z bazy danych B2. Artykuł zawiera algorytm rozwiązania zagadnienia.

2. Hurtownia danych czasu rzeczywistego

Hurtownie danych są bazami danych, integrującymi dane z różnych systemów baz danych działających w przedsiębiorstwie. Aktualne potrzeby biznesowe wymagają przetwarzania dużego wolumenu danych. Ich analiza jest zwykle podstawą do podejmowania decyzji biznesowych.

Zasilanie hurtowni danych odbywa się zwykle okresowo, np. raz na dobę. Aktualne potrzeby biznesowe wymagają częstej aktualizacji (bardzo dużej ilości) danych, co wyklucza zasilanie hurtowni w dużych interwałach czasowych. Odczytywanie prawie aktualnych danych z hurtowni wymaga wykonywania eksportu danych z maksymalną częstotliwością procesu eksportu i ograniczenia się tylko do zmienionych danych.

Można przyjąć, że aktualne potrzeby biznesowe wymagają przetwarzania danych w czasie rzeczywistym lub prawie rzeczywistym.

Wymaga to stosowania odpowiednich metod i narzędzi, pozwalających na pobieranie i analizę danych w krótkim okresie czasu (prawie w czasie rzeczywistym). Oznacza to również konieczność optymalizacji wszystkich procesów ETL (*Extraction Transformation and Loading*), a w szczególności wymusza na procesie ekstrakcji konieczność pobierania danych przyrostowo [3], [4], [6].

Hurtownia danych czasu rzeczywistego powinna być ładowana w taki sam sposób, jak każda inna klasyczna hurtownia danych, lecz z uwzględnieniem częstotliwości ładowania, której przebieg powinien zostać zoptymalizowany.

3. Metody ekstrakcji danych

3.1. Śledzenie znacznika postępu

Metoda polega na określeniu dla każdej tabeli źródłowej metody wyznaczenia następnika klucza głównego lub klucza unikalnego dla ostatnio przeniesionego wiersza. O ile nie są stosowane naturalne wartości atrybutów, zwykle stosuje się klucze sztuczne, w postaci ciągu liczb kolejnych, pobieranych z sekwencji. Zastosowane jako klucz główny lub jego element mogą one stanowić znacznik postępu. W takim przypadku porcję danych do przeniesienia należałoby wyznaczyć w następujący sposób:

```
Select max(PK) into PKc from TABLE;
Select * from TABLE where PK between PKl and PKc;
PKl:=PKc;
Gdzie:
PK – kolumna klucza głównego tabeli
PKl – max wartość ostatniego klucza pobranego w poprzedniej ekstrakcji
PKc – max wartość ostatniego klucza pobranego w bieżącej ekstrakcji
```

Często stosuje się również w tabelach dodawanie kolumn technicznych, zapisujących dokładną wartość czasu (timestamp) dodania wiersza do tabeli. Wartość tę można wykorzystać w sposób analogiczny do pokazanego powyżej numeru sekwencyjnego. Dodatkową korzyścią w tym przypadku jest możliwość określenia ważności danych w bazie hurtowni – opóźnienie w stosunku do bazy rzeczywistej.

Śledzenie znacznika postępu ma następujące zalety: całkowity brak ingerencji w bazę źródłową, możliwość powtórzenia procesu ekstrakcji od dowolnego punktu (ważne np. w przypadku uszkodzenia bazy hurtowni i próbie odzyskania jej do pewnego punktu w czasie), łatwość implementacji.

Metoda śledzenia znacznika postępu ma następujące wady: zastosowanie wyłącznie do tabel mających klucz główny (lub unikalny), śledzenie znacznika postępu na podstawie klucza głównego pozwala zauważyć wstawianie nowego wiersz, a nie wiersza aktualizowanego.

Uwaga: wykorzystanie mechanizmu timestamp w metodzie śledzenia znacznika postępu pozwala zauważyć wstawianie nowego wiersza (Oracle 10, SQLServer).

3.2. Dublowanie zapisów z użyciem wyzwalaczy

Metoda dublowania zapisów z użyciem wyzwalaczy polega na utworzeniu w bazie danych tabel pomocniczych, przechowujących dane będące ekstraktem z tabel źródłowych. Wartości są wpisywane do nich przy każdym dopisaniu wiersza do każdej tabeli źródłowej, poprzez wyzwalacze bazodanowe typu *after insert*. W efekcie, w bazie na bieżąco tworzone

są zbiory danych do zasilenia hurtowni danych, natomiast osobny proces odpowiada za cykliczne przenoszenie tych danych do hurtowni oraz usuwanie przeniesionych wartości.

Metoda ma następujące zalety: tworzenie na bieżąco ekstraktu danych, niski koszt pozyskania danych do ekstrakcji.

Metoda ma następujące wady: konieczność generacji wyzwalacza dla każdej tabeli źródłowej, konieczność dodatkowej alokacji miejsca na zdublowane wartości danych, dość złożony mechanizm zarządzania tabelami pomocniczymi ekstrakcji, wysoki poziom obciążenia I/O dla procesu rejestracji zmian.

3.3. Rejestracja zmian z użyciem wyzwalaczy

Metoda jest modyfikacją sposobu opisanego w punkcie „Dublowanie zapisów z użyciem wyzwalaczy". Różnica w działaniu obu metod polega na tym, że w tabelach pomocniczych ekstrakcji nie zapisuje się wszystkich wartości danych ekstrahowanych, a jedynie wartości klucza głównego wiersza tabeli źródłowej. Dzięki temu ilość miejsca potrzebnego na przechowanie informacji o danych źródłowych jest wielokrotnie mniejsza.

Metoda ma następujące zalety: niski koszt pozyskania danych do ekstrakcji, niewielka ilość miejsca wymagana do zapisania informacji o danych źródłowych, niski poziom obciążenia I/O dla procesu rejestracji zmian.

Metoda ma następujące wady: konieczność generacji wyzwalacza dla każdej tabeli źródłowej, konieczność oprogramowania w celu pobrania danych źródłowych na podstawie zarejestrowanych wartości kluczy głównych, dość złożony mechanizm zarządzania tabelami pomocniczymi ekstrakcji, wolniejszy mechanizm wyszukiwania elementu w zbiorze (ze względu na konieczność połączenia tabeli zmian z tabelą źródłową).

Uwaga: rejestracja zmian z użyciem wyzwalaczy wymaga, aby dla odczytania nowych lub zmienionych wierszy danych zastosować powiązania obu tabel (tabel zmian i tabeli źródłowej). Ten mechanizm staje się wolny z powodu koniecznego powiązania.

3.4. Użycie mechanizmu logminera

Wszystkie zmiany w bazie danych są rejestrowane w logach i są dostępne poprzez mechanizm logminera. Użycie tego mechanizmu polega na przetwarzaniu kolejnych logów archiwalnych osobnym procesem podłączonym bezpośrednio do bazy hurtowni danych.

Zapis do plików dziennika powtórzeń odbywa się w sposób cykliczny, tzn. zapełnienie jednej grupy powoduje przełączenie się na drugą i kontynuowanie zapisu. Po zapełnieniu ostatniej grupy następuje przełączenie się na pierwszą i nadpisanie informacji w niej zawartych.

Przy założeniu, że plik logu byłby przetwarzany zaraz bezpośrednio po przełączeniu na inną grupę plików dziennika powtórzeń w bazie źródłowej, stopień opóźnienia byłby zależny od wielkości logów i aktywności bazy. W przypadku gdyby aktywność bazy była niska, a stopień aktualności danych w hurtowni miałaby wysoki priorytet, należałoby zastosować wymuszanie przełączania logów. Mechanizm przełączania logów spowodowałby utworzenie nowego logu i uruchomienie mechanizmu logminera (bez czekania aż automatycznie zapełni się bieżący plik logu). Podłączenie logów i ich przetwarzanie odbywa się wyłącznie w bazie hurtowni, o ile ingerencja w bazę źródłową jest wykluczona.

Metoda ma następujące zalety: całkowity brak ingerencji w bazę źródłową oraz całkowity brak obciążenia bazy źródłowej procesem ekstrakcji.

Metoda ma następujące wady: bardzo skomplikowane oprogramowanie do przetwarzania logów, konieczność zgodności niektórych parametrów bazy hurtowni z bazą źródłową (w celu umożliwienia przetwarzania obcych logów) [4].

4. Wybór metody ekstrakcji danych

Na szczególną uwagę zasługują dwa mechanizmy:

- rejestracja zmian z użyciem wyzwalaczy,
- CDC (*Change Data Capture*) [15].

Metoda rejestracji zmian z użyciem wyzwalaczy ma zalety, bo jej działanie nie powoduje nadmiernego wzrostu obciążenia I/O. Natomiast odczyt nowych lub zmienionych wierszy danych wymaga powiązania tabel zmian i tabeli źródłowej, co oznacza spowolnienie odczytu i wyszukiwania danych.

Problem odczytu na bieżąco zmian w bazie jest w tym przypadku istotny [15]. Mechanizm CDC opisany w Oracle oferuje możliwość odczytania danych w bazie źródłowej i przekazanie ich do miejsca docelowej bazy danych. To rozwiązanie udostępnia dwa tryby pracy: synchroniczny i asynchroniczny.

W trybie synchronicznym zmienione dane są odczytane natychmiast po ich pojawieniu się w bazie źródłowej, dzięki zastosowaniu wyzwalaczy. W tym przypadku natychmiast po zatwierdzeniu transakcji w tabeli źródłowej, zmienione dane mogą być przeniesione prawie bez zwłoki do hurtowni. Mechanizm synchroniczny jest najszybszą i najprostszą metodą uzyskania zmienionych danych w bazie źródłowej, ponieważ wykorzystując wyzwalacze, reaguje natychmiast na wszelkie zmiany w bazie. Wadą tego mechanizmu jest zwiększenie obciążenia bazy danych, gdyż działanie wyzwalaczy nie pozostaje bez wpływu na zużycie mocy serwera bazy (szczególnie, gdy liczba tabel biorących udział w procesie ekstrakcji jest duża).

Spowolnienie procesów przetwarzania danych na tych tabelach mogą wprowadzić uruchamiane wyzwalacze, używane przez proces CDC.

W trybie asynchronicznym pracy mechanizmu CDC wykorzystuje pliki dziennika powtórzeń, przetwarzając zapisane w nich informacje o zmianach dla zatwierdzonych transakcji. Ze względu na konieczności dostępu do bieżących plików dziennika powtórzeń, tabele zmian muszą znajdować się w bazie źródłowej. Informacje o zmianach w tabelach źródłowych, wykorzystując proces LGWR, zapisywane są do bieżącego pliku dziennika powtórzeń (razem z innymi danymi, pochodzącymi z prowadzonych w bazie transakcji). Dane z tabel obsługiwanych przez mechanizm CDC, zapisywane są w tabelach zmian, skąd mogą być przeniesione do miejsca docelowego. Zaletą asynchronicznego trybu pracy jest fakt, że nie jest wymagane tworzenie dodatkowych tabel w źródłowej bazie danych.

Liczba możliwych trybów pracy mechanizmu CDC jest duża, co niewątpliwie decyduje o możliwości szerokich zastosowań. Można przyjąć, że implementacja mechanizmu CDC we wszystkich przypadkach trybów pracy będzie podobna.

4.1. Porównanie wybranych metod ekstrakcji danych

Porównanie obu metod pokazuje, że realizują one zbliżoną funkcjonalność. Implementując mechanizm CDC w systemie, należy pamiętać, że zmiany są zapisywane fizycznie w tabeli zmian, której rozmiar rośnie proporcjonalnie do aktywności systemu. W niniejszej pracy przyjęto założenie, że priorytetem użytkownika jest aktualność danych w hurtowni, natomiast wpływ procesu ekstrakcji danych na obciążenie bazy źródłowej nie ma znaczenia. Do tego celu najbardziej odpowiedni będzie tryb synchroniczny pracy mechanizmu CDC.

5. Charakterystyka metody wykorzystującej technologię CDC

Metoda wykorzystująca mechanizm CDC, pozwala zoptymalizować sam proces ekstrakcji danych. Wykorzystanie mechanizmu CDC do rejestrowania zmienionych danych w tabelach oraz możliwości ich udostępniania na podstawie wybranych innych obiektów bazy Oracle, pozwala na praktycznie natychmiastową rejestrację zmian zatwierdzonych transakcji. Wykorzystanie odpowiednich widoków i zapytań SQL pozwala na rejestrację zmienionych danych w tabelach, w obrębie bazy źródłowej hurtowni, w wyniku działania ostatniej transformacji wymuszonej przez cykliczny proces ETL.

Metoda ma następujące zalety: w bazie źródłowej przechowywane będą tylko nowe dane oraz kod składowany, obsługujący proces ekstrakcji, częstotliwość pobierania danych przez proces ETL wynika tylko z założeń funkcjonalnych hurtowni i uzależniona jest od wydajno-

ści środowiska. Zaletą tej metody jest również to, że jest dobrze udokumentowana w specyfikacji technicznej Oracle ETL [7].

5.1. Użycie mechanizmu CDC

Oracle dostarcza mechanizm rejestrowania zmienionych danych w tabelach oraz udostępnia te informacje poprzez zapytania do utworzonych w tym celu widoków.

Dla celu ekstrakcji danych z hurtowni, najbardziej odpowiednią metodą jest metoda z natychmiastową rejestracją zmian zatwierdzonych transakcji. Zmienione dane są rejestrowane w obrębie bazy źródłowej [13].

Użycie mechanizmu CDC wymaga: zidentyfikowania tabel źródłowych, z których dane mają być przenoszone do bazy hurtowni, określenia kolumn w tabelach źródłowych, z których dane mają być przenoszone do bazy hurtowni, utworzenia tabel rejestracji zmian, zdefiniowania widoków, poprzez które będą widoczne zmienione dane, aktywacji mechanizmu subskrypcji zmian. Ponieważ zarejestrowane dane są zapisywane wraz z wartością SCN (*System Change Number*), można użyć tej wartości jako wskaźnika czasowego zmian w bazie źródłowej oraz jako parametru dla procesu cyklicznego pobierania danych do hurtowni [15].

6. Prototyp rozwiązania zagadnienia zasilania hurtowni w czasie rzeczywistym

6.1. Założenia

Przyjmuje się następujące założenia dla bazy źródłowej:

- bazę stanowi relacyjna baza danych Oracle;
- dane źródłowe dla hurtowni zapisane są w tabelach zawierających zawsze klucz główny;
- znana jest struktura bazy, brak możliwości zmian struktur danych i algorytmów przetwarzania, nie jest dopuszczalna modyfikacja danych źródłowych;
- możliwość budowy własnych struktur danych w bazie źródłowej, w wydzielonym schemacie, oraz tworzenia wyzwalaczy na strukturach bazy źródłowej.

Dodatkowo przyjęto założenia uwzględniające spójność i kompletność danych:

- dane do ekstrakcji pobierane są z zatwierdzonych transakcji;
- pojedyncza transakcja bazodanowa zawiera dane biznesowe, wystarczająco spójne dla procesu ekstrakcji.

Znając logikę działania aplikacji bazy źródłowej oraz przy założeniu, że możliwa jest identyfikacja referencji pomiędzy tabelami, można założyć, że jest wykonalna identyfikacja

całej spójnej porcji danych, w przeciwnym przypadku należy przyjąć jedną z dwóch możliwości:

- jest akceptowalny stan chwilowej niekompletności danych w hurtowni, gdyż produkowane z niej dane zagregowane mają dopuszczalny stopień dokładności, np. 0,1%;
- dane są kompletowane w procesie przygotowania do transformacji, poprzez wybór z dostępnego zbioru przyrostu danych, podzbioru danych kompletnych.

6.2. Algorytm procesu ekstrakcji danych źródłowych

Ekstrakcja danych odbywa się z poziomu procesu sterującego przenoszeniem danych z bazy źródłowej do bazy hurtowni. Proces pobiera dane cyklicznie z częstotliwością wynikającą z założeń funkcjonalnych hurtowni lub z maksymalnie możliwą częstotliwością z punktu widzenia wydajności środowiska. Pobieranie danych jest poprzedzone wywołaniem funkcji w bazie źródłowej, dającej w wyniku odpowiedź o istnieniu lub braku nowych danych od czasu poprzedniej ekstrakcji.

Cykl ekstrakcji danych jest następujący:

- pytanie o nowe dane w bazie źródłowej,
- wywołanie funkcji NEW_EXTRACT składowanej w bazie,
- utworzenie widoku zmodyfikowanych danych dla okna czasowego okresu od poprzedniej ekstrakcji do chwili bieżącej,
- zapamiętanie maksymalnej wartości SCN bieżącego okna ekstrakcji dla każdej tabeli podlegającej ekstrakcji,
- wypełnienie tabeli meta danych wartościami parametrów bieżącej ekstrakcji,
- zwrot wartości identyfikatora bieżącej ekstrakcji lub wartości NULL do procesu sterującego ekstrakcją,
- pobranie przez proces ekstrakcji danych z bieżącego okna czasowego ekstrakcji, kolejno ze wszystkich widoków, na podstawie identyfikatora bieżącej ekstrakcji.

Zakończenie cyklu i rozpoczęcie nowego cyklu po ewentualnym czasie zwłoki lub bezzwłocznie, po przetworzeniu ekstraktu.

6.3. Implementacja mechanizmu CDC (dla przykładowych danych)

W bazie źródłowej zostanie utworzony:

- schemat EXT, przechowujący nowe dane,
- kod składowany, obsługujący proces ekstrakcji.

Dla każdej tabeli źródłowej zostanie utworzona, zgodnie z wymaganiami mechanizmu CDC:

- tabela zmian (Change Table), wg schematu nazewnictwa NAZWATABELI_CT.

Ta tabela będzie zawierała tylko te kolumny z tabeli źródłowej, które są istotne z punktu widzenia hurtowni. W przypadku zmiany spowodowanej instrukcją UPDATE, do tabeli będą zapisywane tylko wiersze o wartościach zatwierdzonych w transakcji. Dostęp do nowych danych będzie odbywał się poprzez perspektywy o nazwach utworzonych wg schematu NAZWATABELI_V.

Każda perspektywa będzie zawierała zestaw kolumn potrzebnych do zasilenia hurtowni oraz kolumny techniczne, z których istotne znaczenia mają:

OPERATION$ CHAR(2) – identyfikuje metodę zmiany wiersza (przyjmuje wartości: I – nowy wiersz, wprowadzony instrukcją INSERT, UN – wiersz zmodyfikowany po wykonaniu instrukcji UPDATE, D – wiersz usunięty instrukcją DELETE) oraz CSCN$ NUMBER – wartość SCN dla transakcji zatwierdzającej zmianę wiersza.

Do sterowania procesem cyklicznego pobierania nowych wartości zostanie utworzona tabela metadanych o nazwie E_CONTROL, o następującej strukturze: EXTRACT_NR NUMBER – identyfikator ekstrakcji, TIME_STAMP DATE – dokładny czas ekstrakcji, VIEW_NAME VARCHAR2(30) – nazwa perspektywy z danymi ekstrakcji, SCN_MIN NUMBER – największa wartość SCN dla poprzedniej ekstrakcji, SCN_MAX NUMBER – największa wartość SCN dla bieżącej ekstrakcji.

Proces ekstrakcji będzie sterowany zewnętrznie, poprzez funkcję: NEW_EXTRACT RETURN NUMBER.

Funkcja zwraca identyfikator ekstrakcji EXTRACT_NR NUMBER, lub wartość NULL, jeżeli w bazie nie ma nowych danych w żadnej tabeli źródłowej od czasu poprzedniej ekstrakcji. Jeżeli w bazie pojawiły się nowe dane, to można je pobrać do bazy hurtowni zapytaniem:

```
select <lista kolumn> from NAZWATABELI_V
   where CSCN$ > SCN_MIN; gdzie SCN_MIN jest wartością uzyskaną z zapytania:
select SCN_MIN from E_CONTROL
  where EXTRACT_NR = <wynik funkcji NEW_EXTRACT> and VIEW_NAME = NAZWATABELI_V;
```

Każde wywołanie funkcji NEW_EXTRACT tworzy kolejny zbiór nowych wierszy, a powtórne odczytanie danych może być wykonane na podstawie zapisanych w tabeli E_CONTROL danych.

6.4. Generacja kodu składowanego

Poniżej opisano sposób wygenerowania kodu składowanego oraz przykład ilustrujący użycie metody. Poniżej podane zostały wymagane skrypty:

Skrypt init.txt

```
compatible = 10.1.0
java_pool_size = 50000000;
job_queue_processes = 2
parallel_max_servers = <current value> + 5
processes = <current value> + 7
sessions = <current value> + 2
streams_pool_size = <current value> + 21 MB
undo_retention = 3600
```

Skrypt user.sql

```
compatible = 10.1.0
java_pool_size = 50000000;
job_queue_processes = 2
parallel_max_servers = <current value> + 5
processes = <current value> + 7
sessions = <current value> + 2
streams_pool_size = <current value> + 21 MB
undo_retention = 3600
```

Skrypt Table.sql

```
create table SOURCE_TABLES
(
  OWNER      VARCHAR2(30) not null,
  TABLE_NAME VARCHAR2(30) not null,
  VIEW_NAME  VARCHAR2(30) not null,
  COL_LIST   VARCHAR2(2000) not null
)
tablespace USERS
  pctfree 10
  initrans 1
maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );

-- Create/Recreate primary, unique and foreign key constraints

alter table SOURCE_TABLES
  add constraint SOURCE_TABLES_PK primary key (OWNER, TABLE_NAME)
  using index
  tablespace USERS
  pctfree 10
  initrans 2
  maxtrans 255
  storage
  (
    initial 64K
    minextents 1
    maxextents unlimited
  );
--------------------------------
-- Create table
create table E_CONTROL
(
  EXTRACT_NR number not null,
  TIME_STAMP date not null,
  VIEW_NAME  varchar2(30) not null,
  SCN_MIN    number not null,
  SCN_MAX    number not null
);
```

```
-- Create/Recreate primary, unique and foreign key constraints
alter table E_CONTROL
  add constraint E_CONTROL_PK primary key (EXTRACT_NR, VIEW_NAME);
```

Skrypt extraction.pck (ten skrypt obejmuje kilka stron tekstu, został umieszczony w bardzo skróconej postaci)

```
create or replace package extraction is
  -- Author  : WKACERKA
  -- Created : 2010-09-11 19:33:55
  -- Purpose :

  -- Public type declarations
  --type <TypeName> is <Datatype>;

  -- Public constant declarations
  --<ConstantName> constant <Datatype> := <Value>;
  -- Public variable declarations
  --<VariableName> <Datatype>;

-- Public function and procedure declarations
  function Prepare return number;
  procedure Erase;
  function New_Extract return number;
  procedure Clear_View;

end extraction;
/
create or replace package body extraction is
  -- Private type declarations
  -- type <TypeName> is <Datatype>;

  -- Private constant declarations
  --<ConstantName> constant <Datatype> := <Value>;
  -- Private variable declarations
  --<VariableName> <Datatype>;

-- Function and procedure implementations

function Prepare return number is
    ct_name  varchar2(30);
    s        varchar2(100);
  begin
……
……
……
```

Należy ustawić parametry bazy zgodnie z wartościami podanymi w init.txt, używając SQL*Plus, oraz wykonać poniższe czynności:

- podłączyć się do bazy jako użytkownik sys as sysdba,
- wykonać skrypt user.sql,
- podłączyć się do bazy jako ext/ext,
- wykonać skrypt tabele.sql,
- wykonać skrypt extraction.pck.

W wyniku tych operacji w schemacie ext zostaną utworzone tabele sterujące procesem oraz pakiet zawierający następujące procedury i funkcje:

- funkcja PREPARE – wykonuje wszystkie operacje wymagane do uruchomienia śledzenia zmian na tabelach, których opis został wcześniej wpisany do tabeli SOURCE_TABLES,

- procedura ERASE – usuwa skutki działania funkcji PREPARE, usuwa obiekty i kasuje subskrypcje oraz publikacje zmian,
- funkcja NEW_EXTRACT, procedura CLEAR_VIEW – czyści tabele zmian, jeżeli dane nie są już potrzebne do zasilania hurtowni.

Na potrzeby zobrazowania działania metody na dowolnych tabelach źródłowych, użyto, jako źródło metadadanych dla funkcji PREPARE, tabeli SOURCE_TABLES o strukturze:

```
SQL> desc SOURCE_TABLES
Nazwa                                     Wartość NULL? Typ
----------------------------------------- -------- ----------------
OWNER                                     NOT NULL VARCHAR2(30)
TABLE_NAME                                NOT NULL VARCHAR2(30)
VIEW_NAME                                 NOT NULL VARCHAR2(30)
COL_LIST                                  NOT NULL VARCHAR2(2000)
```

Do tabeli należy wpisać wiersze opisujące kolejno tabele źródłowe. Kolumna COL_LIST powinna zawierać listę kolumn w tabeli źródłowej, w postaci opisu: NAZWA_KOLUMNY_1 TYP_1,…., NAZWA_KOLUMNY_N TYP_N.

6.5. Przykład ilustrujący użycie metody CDC

Zakładamy schemat user1, który będzie zawierał tabele źródłowe:

```
create user user1 identified by user1
  default tablespace USERS
  QUOTA UNLIMITED ON SYSTEM
  QUOTA UNLIMITED ON SYSAUX;;
GRANT CREATE SESSION TO ext;
GRANT UNLIMITED TABLESPACE TO ext;
GRANT RESOURCE TO ext;
```

Następne operacje wykonujemy, będąc podłączonym do bazy jako:

- użytkownik ext – do wykonywania operacji związanych z ekstrakcją,
- użytkownik user1 – do wprowadzania zmian w tabelach źródłowych TAB1 i TAB2.

Jako użytkownik user1 zakładamy dwie tabele źródłowe TAB1 i TAB2:

```
create table TAB1
(COL1 number,
 COL2 varchar2(25));
alter table TAB1
  add constraint TAB1_PK primary key (COL1);
create table TAB2
( COL1 number,
  COL2 varchar2(25));
alter table TAB2
  add constraint TAB2_PK primary key (COL1);
```

Do tabeli SOURCE_TABLES wpisujemy opisy tabel źródłowych:

```
insert into SOURCE_TABLES
  ( owner, table_name, view_name, col_list )
 values
  ( 'USER1','TAB1','TAB1_V','COL1 NUMBER,COL2 VARCHAR2(25)');
insert into SOURCE_TABLES
```

```
  ( owner, table_name, view_name, col_list )
 values
  ( 'USER1','TAB2','TAB2_V','COL1 NUMBER,COL2 VARCHAR2(25)');
Commit;
```

Uruchamiamy mechanizm rejestracji zmian:

```
var n number
execute :n:=extraction.prepare;
```

Wykonujemy zmiany na tabelach źródłowych:

```
insert into user1.TAB1 values (1,'jeden');
insert into user1.TAB1 values (2,'dwa');
insert into user1.TAB2 values (1,'jeden');
commit;
```

Wykonujemy migawkę zmian:

```
exec DBMS_CDC_SUBSCRIBE.EXTEND_WINDOW(subscription_name => 'EXTRACTION_SUB');
```

Odczytujemy nowo zmodyfikowane dane z perspektyw TAB1_V i TAB2_V:

```
var n number
execute :n:=extraction.new_extract;
print n
--- jeżeli pojawiły się nowe zmiany wartość n jest nie jest null
select OPERATION$,COL1, COL2 from TAB1_v where CSCN$>
 ( select SCN_MIN from E_CONTROL
     where EXTRACT_NR = :n and
        VIEW_NAME = 'TAB1_V');
select OPERATION$,COL1, COL2 from TAB2_v where CSCN$>
 ( select SCN_MIN from E_CONTROL
     where EXTRACT_NR = :n and
        VIEW_NAME = 'TAB2_V');
```

Jeszcze jedna zmiana ilustrująca pobranie kolejnej porcji danych:

```
insert into user1.TAB1 values (3,'trzy');
update user1.TAB1 set col2='zmiana' where col1=2;
insert into user1.TAB2 values (3,'trzy');
commit;
-------------
exec DBMS_CDC_SUBSCRIBE.EXTEND_WINDOW(subscription_name => 'EXTRACTION_SUB');
var n number
execute :n:=extraction.new_extract;
print n
--- jeżeli pojawiły się nowe zmiany wartość n nie jest null
select OPERATION$,COL1, COL2 from TAB1_v where CSCN$>
 ( select SCN_MIN from E_CONTROL
     where EXTRACT_NR = :n and
        VIEW_NAME = 'TAB1_V');
select OPERATION$,COL1, COL2 from TAB2_v where CSCN$>
 ( select SCN_MIN from E_CONTROL
     where EXTRACT_NR = :n and
        VIEW_NAME = 'TAB2_V');
```

Na koniec można usunąć dane z tabel zmian oraz wyłączyć cały mechanizm:

```
execute extraction.clear_view; execute extraction.erase;
```

6.6. Implementacja mechanizmu CDC (dla Bazy B2)

Na podstawie danych z przykładowej bazy B2, zostały wykonane testy pokazujące przykłady wykorzystania mechanizmu CDC dla ekstrakcji danych z hurtowni.

Testy obejmują dwa widoki – widok tabeli faktów Fct.vSales i widok tabeli wymiaru Dim.vAccounts. Wynikiem zmian na tabelach źródłowych jest rezultat zapytania do utworzonych perspektyw vSales_ext i vAccounts_ext, które mają dokładnie taką strukturę, jak odpowiadające im widoki vSales i vAccounts (zawierają tylko dane pochodzące od wierszy zmodyfikowanych lub nowych).

Plik zawierający przykład implementacji CDC dla perspektywy vAccounts to test_vAccounts.sql. Plik zawierający przykład implementacji CDC dla perspektywy vSales to test_vSales. Pliki przykładów użycia mechanizmu CDC opisują w kolejnych krokach działanie CDC, dla ułatwienia są tam zawarte polecenia „conn" by wskazać na koncie jakiego użytkownika należy wykonać polecenie. Test został przeprowadzony za pomocą jednocześnie uruchomionych dwóch instancjach SQL*Plus, jednej z połączeniem jako SRC (schemat zawierający tabele przykładowej bazy) i drugiej z połączeniem jako EXT (właściciel pakietu EXTRACTION). Wyniki testów umieszczone są w pliku wyniki_test_vSales_vAccounts.txt. Przykładowy test mechanizmu CDC dla modelowej perspektywy vSales:

```
--1. Utworzenie schematu dla tabel modelowych
conn sys as sysdba
create user src identified by src
default tablespace users
temporary tablespace temp;
grant dba to src;
grant select any table to src;
--2. Utworzenie modelowych tabel -------
conn src/src
@vSales.sql
@inserts.sql-vSales.sql
--3. Przygotowanie mechanizmu CDC poprzez pakiet EXTRACTION
conn ext/ext
execute extraction.erase;
delete from SOURCE_TABLES;
commit;
insert into SOURCE_TABLES
values ('SRC','SUBITEMS','SUBITEMS_V','ID NUMBER(10),NUMBER_A
NUMBER(10),QUANTITY NUMBER(18),PRICERATE NUMBER(18),'|| 'SUBITEMVALUE
NUMBER(18),TYPE NUMBER(3),RELATEDSUBITEMID NUMBER(10),ARTICLEID
NUMBER(10),WAREHOUSEID NUMBER(10),'||'STOCKDATE TIMESTAMP(6),
STOCKDOCUMENTNUMBER VARCHAR2(100)' );
insert into SOURCE_TABLES values ('SRC','ITEMS','ITEMS_V','ID NUMBER(10),
HEADERID NUMBER(10), ARTICLEID NUMBER(10), ARTICLENAME
VARCHAR2(200),'||'ARTICLEDESCRIPTION VARCHAR2(1000), NUMBER_A NUMBER(10),
QUANTITY NUMBER(18), BASENETPRICE NUMBER(18),'||' BASEGROSSPRICE NUMBER(18),
DISCOUNT NUMBER(18),VATRATE NUMBER(18), FINALNETPRICE NUMBER(18),FINALGROSSPRICE
NUMBER(18),'||'VALUEDIRECTIONTYPE NUMBER(3),NETVALUE NUMBER(18), VATVALUE
NUMBER(18),GROSSVALUE NUMBER(18), PURCHASEVALUE NUMBER(18),'||'UNIT
VARCHAR2(20),UNITID NUMBER(10), QUANTITYINBASICUNIT NUMBER(18),PRECISION
NUMBER(3), NUMERATOR NUMBER(10),'||'DENOMINATOR NUMBER(10),
DECIMALREPRESENTATION NUMBER(1),VATRATEID NUMBER(10), CURRENCYID
NUMBER(10),'||'PRICETYPEID NUMBER(10), USERDISCOUNT NUMBER(18),
NETDOCUMENTCURRENCYVALUE NUMBER(18),'||'VATDOCUMENTCURRENCYVALUE UMBER(18),
GROSSDOCUMENTCURRENCYVALUE NUMBER(18), BASEDOCUMENTCURRENCYNETPRICE NUMBER(18),
```

```
'||'BASEDOCUMENTCURRENCYGROSSPRICE NUMBER(18), FINALDOCUMENTCURRENCYNETPRICE
NUMBER(18), '||'FINALDOCUMENTCURRENCYGROSSPRIC NUMBER(18),
FINALSYSTEMCURRENCYNETPRICE NUMBER(18), FINALSYSTEMCURRENCYGROSSPRICE
NUMBER(18)' );
commit;
var n number
execute :n:=extraction.prepare;
--4. Utworzenie perpektywy zmienionych danych vSales_EXT - analogu vSales -
conn src/src
@vsales_ext.sql
--5. Wykonanie pierwszej modyfikacji danych źródłowych
update items set ID=40001 where ID=40001;
update subitems set ID=40001 where ID=40001;
commit;
--6. Stworzenie pierwszej migawki zmian
conn ext/ext
execute :n:=extraction.new_extract;
print n
--7. Odczyt zmienionych danych
conn src/src
select * from vSales_ext; -- wynikiem jest jeden wiersz perspektywy, pochodzący
od zmodyfikowanych danych
--8. Wykonanie kolejnych modyfikacji danych źródłowych
update items set ID=40002 where ID=40002;
update subitems set ID=40002 where ID=40002;
update items set ID=40004 where ID=40004;
update subitems set ID=40004 where ID=40004;
commit;
--9. Stworzenie kolejnej migawki zmian
conn ext/ext
execute extraction.clear_view;
execute :n:=extraction.new_extract;
print n
--10. Odczyt zmienionych danych
conn src/src
select * from vSales_ext; --
KONIEC TESTU
```

Przykładowy test mechanizmu CDC dla modelowej perspektywy vAccounts:

```
--1. Utworzenie schematu dla tabel modelowych
conn sys as sysdba
create user src identified by src
  default tablespace users
  temporary tablespace temp;
grant dba to src;
grant select any table to src;
--2. Utworzenie modelowych tabel
conn src/src
@vAccounts.sql
@inserts.sql-vAccounts.sql
--3. Przygotowanie mechanizmu CDC poprzez pakiet EXTRACTION
conn ext/ext
execute extraction.erase;
delete from SOURCE_TABLES;
commit;
insert into SOURCE_TABLES values
 ( 'SRC','ACCOUNTS','ACCOUNTS_V','ACCOUNTID NUMBER(10),NAME
NVARCHAR2(150),NUMBER_A NVARCHAR2(100),'||
   'NEXTNUMBER NVARCHAR2(100), CREATEDATE DATE,ACCOUNTTYPEID NUMBER(10),ISACTIVE
NUMBER(1),'||
   'ISOPENINGBALLANCETOMOVE NUMBER(1),BALANCECONTROL NUMBER(5),DESCRIPTION
NVARCHAR2(400),'||
   'ISGROUPDIVIDED NUMBER(1),CURRENCYID NUMBER(10),PARENTACCOUNTID
NUMBER(10),TIMESTAMP TIMESTAMP(6),'||
   'ISHAVINGCHILDRENS NUMBER(1),DICTIONARYACCOUNTGROUPNAMEID
NUMBER(10),DICTIONARYACCOUNTSUBGROUPNAMEI NUMBER(10),'||
   'ACCOUNTINGPERIODID NUMBER(10),ISCLEARING NUMBER(1),PROTECTEDOBJECTID
NUMBER(10)' );
```

```
commit;
var n number
execute :n:=extraction.prepare;
--4. Utworzenie perpektywy zmienionych danych vAccounts_EXT - analogu vAccount -
--
conn src/src
CREATE OR REPLACE VIEW vAccounts_EXT
AS
   SELECT
     CASE
      WHEN ( acc.BalanceControl = '0' ) THEN 'Bez kontroli'
      WHEN ( acc.BalanceControl = '1' ) THEN 'Kontrola Wn'
      WHEN ( acc.BalanceControl = '2' ) THEN 'Kontrola Ma'
     END
      ACC_BalanceControl,
     cur.NAME ACC_Currency,
          CAST(acc.DESCRIPTION AS VARCHAR(255)) ACC_Description,
          CASE
                WHEN ( ( acc.DictionaryAccountGroupNameID IS NOT NULL )
                  AND ( acc.DictionaryAccountSubGroupNameI IS NULL ) ) THEN
cast(gr.NAME as NVARCHAR2(100))
                WHEN ( ( acc.DictionaryAccountGroupNameID IS NOT NULL )
                  AND ( acc.DictionaryAccountSubGroupNameI IS NOT NULL ) ) THEN
CASE
WHEN artgr.Id IS NOT NULL THEN artgr.Code
WHEN art.Id IS NOT NULL THEN art.Code
WHEN cstgr.Id IS NOT NULL THEN cstgr.Code
WHEN cst.Id IS NOT NULL THEN cst.Code   END
          ELSE NULL
             END ACC_DictionaryAccount,
          CAST(acc.NAME AS VARCHAR(255)) ACC_Name,
          CAST(acc.NextNumber AS VARCHAR(255)) ACC_NextAccountingPeriodNumber,
          CAST(acc.Number_A AS VARCHAR(255)) ACC_Number,
          acc.Number_A || '|' || period.Symbol ACC_OrgID,
          CAST(acc.IsClearing AS NUMBER) ACC_SettlementAccount,
          act.TYPE ACC_Type,
          paracc.Number_A || '|' || period.Symbol ACC_ParOrgID
     FROM EXT.Accounts_V acc
            LEFT JOIN AccountTypes act
             ON acc.AccountTypeID = act.ID
            LEFT JOIN DictionaryAccountGroupNames gr
             ON acc.DictionaryAccountGroupNameID =
gr.DictionaryAccountGroupNameID
            LEFT JOIN DictionaryAccountSubGroupNames sub
             ON acc.DictionaryAccountSubGroupNameI =
sub.DictionaryAccountSubGroupNameI
            LEFT JOIN Currencies cur
             ON acc.CurrencyID = cur.Id
            LEFT JOIN Accounts paracc
             ON acc.ParentAccountID = paracc.AccountID
            LEFT JOIN AccountingPeriods period
             ON acc.AccountingPeriodID = period.AccountingPeriodID
            LEFT JOIN ArticleGroupsBindings artgrbind
             ON artgrbind.AccountID = acc.AccountID
            LEFT JOIN Dic_ArticleGroups artgr
             ON artgr.Id = artgrbind.ArticleGroupId
            LEFT JOIN ArticlesBindings artbind
             ON artbind.AccountID = acc.AccountID
            LEFT JOIN Dic_Articles art
             ON art.Id = artbind.ArticleID
            LEFT JOIN CustomersGroupBindings cstgrbind
             ON cstgrbind.AccountID = acc.AccountID
            LEFT JOIN Dic_CustomerGroups cstgr
             ON cstgr.Id = cstgrbind.CustomersGroupID
            LEFT JOIN CustomersDictionaryBindings cstbind
             ON cstbind.AccountID = acc.AccountID
            LEFT JOIN Dic_Customers cst
             ON cst.Id = cstbind.CustomerID;
--5. Wykonanie pierwszej modyfikacji danych źródłowych
update accounts set timestamp=CURRENT_TIMESTAMP where accountid=32;
```

```
commit;
--6. Stworzenie pierwszej migawki zmian --
conn ext/ext
execute :n:=extraction.new_extract;
print n
--7. Odczyt zmienionych danych -----
conn src/src
select * from vaccounts_ext; -- wynikiem jest jeden wiersz perspektywy, pocho-
dzący od zmodyfikowanych danych
--8. Wykonanie kolejnych modyfikacji danych źródłowych ---
update accounts set timestamp=CURRENT_TIMESTAMP where accountid=33;
update accounts set timestamp=CURRENT_TIMESTAMP where accountid=34;
commit;
--9. Stworzenie kolejnej migawki zmian --
conn ext/ext
execute extraction.clear_view;
execute :n:=extraction.new_extract;
print n
--10. Odczyt zmienionych danych -----
conn src/src
select * from vaccounts_ext; -- wynikiem są dwa zmodyfikowane wiersze perspekty-
wy,pochodzące od zmodyfikowanych danych
------ KONIEC TESTU
```

7. Podsumowanie i wnioski

W opracowaniu przyjęto model architektury, w którym wyróżnione są dwa miejsca: baza źródłowa – system transakcyjny, baza docelowa, będąca hurtownią danych.

Głównym celem biznesowym jest aktualizacja danych w bazie docelowej, za pomocą danych zmienionych w bazie źródłowej. W pracy przyjęto założenie, że najważniejszym zadaniem jest uzyskanie możliwie szybko aktualnego stanu danych w bazie docelowej, co ma niewątpliwy wpływ na analizę procesów biznesowych.

Oracle oferuje mechanizm CDC, pozwalający na odczytanie zmienionych danych w bazie źródłowej i przekazanie ich do miejsca docelowej bazy danych. Możliwe są dwa tryby pracy mechanizmu CDC: synchroniczny i asynchroniczny.

W pracy zastosowano tryb synchroniczny, który używając wyzwalaczy, reaguje natychmiast na wszelkie zmiany w bazie. Oznacza to, że natychmiast po zatwierdzeniu transakcji na tabeli źródłowej, zmiana może być przetransportowana bez zwłoki do miejsca docelowego. Wadą tego mechanizmu jest zwiększenie obciążenia bazy oraz spowolnienie procesów przetwarzania danych (poprzez uruchamiane wyzwalacze używane przez proces CDC). Dodatkowo implementacja mechanizmu CDC w systemie wymaga zapisu zmienionych danych w tabeli zmian, której rozmiar rośnie proporcjonalnie do aktywności systemu.

W niniejszej pracy przyjęto założenie, że priorytetem użytkownika jest aktualność danych w hurtowni ,natomiast wpływ procesu ekstrakcji danych na obciążenie bazy źródłowej nie ma znaczenia. Z tego właśnie powodu uznano, że najbardziej odpowiedni będzie tryb synchroniczny pracy mechanizmu CDC.

Metoda wykorzystująca mechanizm CDC pozwala zoptymalizować sam proces ekstrakcji danych oraz wyznaczyć zmienione wiersze danych powstałych w wyniku działania ostatniej transformacji wymuszonej przez cykliczny proces ETL. Zmienione dane są rejestrowane w obrębie bazy źródłowej [13].

Metoda ma następujące zalety: w bazie źródłowej przechowywane są tylko nowe dane oraz kod składowany, obsługujący proces ekstrakcji, częstotliwość pobierania danych przez proces ETL wynika tylko z założeń funkcjonalnych hurtowni i uzależniona jest od wydajności środowiska. Zaletą jest również to, że jest dobrze udokumentowana w specyfikacji technicznej Oracle ETL [7].

W podanym prototypie rozwiązania zagadnienia zasilania hurtowni przyjmuje się, że mechanizm CDC to mechanizm rejestrowania zmienionych danych w tabelach i udostępniania tych informacji poprzez zapytania do utworzonych w tym celu widoków vAccounts oraz vSalles. Wyniki zmian są zapisywane odpowiednio w vAccounts_EXT oraz vSalles_EXT.

Zaprezentowany przykład trybu synchronicznego wprowadza najmniejsze opóźnienie z dostępnych trybów pracy. Opóźnienie to ma wymiar proporcjonalny do czasu potrzebnego na otwarcie okna zmian oraz odczytu i przetworzenia danych zapisanych w tabeli zmian. Alternatywnym rozwiązaniem jest definicja własnych wyzwalaczy bazodanowych, pracujących na tabelach źródłowych.

BIBLIOGRAFIA

1. Kimball R., Reeves L., Ross M.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses, Warren Thornthwaite. John Wiley & Sons, 1998.
2. Kimball R., Ross M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. John Wiley & Sons, 2002.
3. Poe V., Klauer P., Brobst S.: Tworzenie hurtowni danych, wspomaganie podejmowania decyzji. WNT, Warszawa 2000.
4. Hand D., Mannila H., Smyth P.: Eksploracja danych. WNT, Warszawa 2005.
5. Larose D. T.: Odkrywanie wiedzy z danych. PWN, Warszawa 2006.
6. Gorawski M.: Praktyczne aspekty projektowania hurtowni danych. Studia Informatica, Vol. 24, No. 4 (56), 2003.
7. Gorawski M.: Charakterystyka procesu ekstrakcji danych. Studia Informatica, Vol. 24, No. 4 (56), 2003.

8. Gorawski M.: Modelowanie procesu ekstrakcji danych. IV Krajowa Konferencja Metody i systemy komputerowe w badaniach naukowych i projektowaniu inżynierskim, Kraków 2003.
9. Gorawski M.: Architektura przemysłowych hurtowni danych. X Konferencja Systemy Czasu Rzeczywistego, Ustroń 2003.
10. Kimball R.: Data Warehouse Toolkit – Lifecycle Toolkit. John Wiley & Sons, 1998.
11. Han J., Kamber M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.
12. Chen Z.: Computational Intelligence for Decision Support. CRC Press, 2000.
13. Hand D., Mannila H.: Eksploracja danych. WNT, Warszawa 2005.
14. Bębel B., Wrembel R., Zadrożna A.: Implementowanie hurtowni danych – zagadnienia technologiczne. Materiały konferencyjne Hurtownie Danych i Business Intelligence, Warszawa 2004.
15. Oracle® Database Data Warehousing Guide 10g Release 1 (10.1) Part No. B10736-01 http://www.zsi.pwr.wroc.pl/zsi/missi2002/pdf/s204.pdf.

Wpłynęło do Redakcji 16 stycznia 2013 r.

Abstract

The study contains information about research on the B2 database. The aim of the research is to obtain consistent data in order to support effective business decisions.

A data warehouse is a type of database optimized for efficient processing of data for the purpose of data analysis. Traditionally data warehouse data is periodically supplied with data according to a specific cycle (usually daily or longer period).

The distinguishing feature of the problem examined is a requirement that available for business analysis is not only the data from the warehouse (supplied periodically) but also up from date information entered in transactional systems. Such requirement means that the data warehouse operating in real-time has to be optimized so that the delay between transaction completion in source database and transformation and storing in data warehouse is as short as possible. This means that all ETL processes need to be optimized and the data has to be retrieved incrementally. The size of retrieved data determines the delay of warehouse data actualization as well as the accuracy of the implementation of the real-time warehouse model.

For proper operation of the source system data completeness and consistency in the data warehouse is required. One of two possibilities has to be adopted:

- a state of temporary data incompleteness is acceptable because the aggregated data has an acceptable degree of accuracy, e.g. 0.1%;
- in the process of preparation for the transition data is completed by selecting a subset of complete data from the available set of incremental data.

The conclusion is that for a data warehouse operating in real-time, data extraction has to be performed with an acceptably small delay, such that there is no need to change data in the data warehouse.

In this study a custom method that using the CDC technology has been implemented, allowing to optimize the process of extracting data from a real-time data warehouse.

Adresy

Wanda GRYGLEWICZ-KACERKA: Państwowa Wyższa Szkoła Zawodowa we Włocławku, Instytut Nauk Społecznych i Informatyki, ul. 3 maja 17, 87-800 Włocławek, Polska, wanda.gryglewicz-kacerka@p.lodz.pl.
Jarosław KACERKA: Politechnika Łódzka, Instytut Automatyki, ul. Stefanowskiego 18/22, 90-924 Łódź, Polska, jarosław.kacerka@p.lodz.pl.