데이터웨어하우스에서 효율적인 분석데이터 추출시 PL/SQL을 이용한 질의 최적화

정승경*
*고려대학교 컴퓨터공학과
e-mail:jseungky*@korea.ac.kr

Query optimizing use PL/SQL for Efficient Extracting Analysis Data in data Warehouse

Seung-Kyung Jeong*
*Dept of Computer Engineering, Korea University

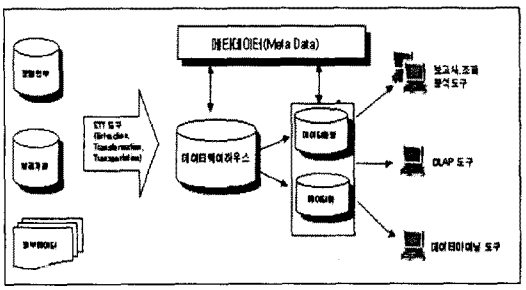
요 약

기존에 기업의 의사결정을 위해 사용되는 데이터 웨어하우스나 데이터 마트에서 성능향상을 위한 많은 연구들이 있었다. 하지만 복잡한 업무요건이 추가되고 기업의 요구가 다양해짐에 따라 RDBMS의 성능은 점점 낮아지고 이를 해결하기 위한 필요성이 요구되었다. 따라서 본 논문에서는 이를 해결하기 위해 버퍼기능을 하는 PL/SQL Package를 구현하여 효율적인 질의 최적화 방법을 제안하고자 한다. 그리고 본 논문에서 제안한 방법이 기존의 방법보다 성능이 좋다는 것을 실험을 통해 증명해 보였다.

1. 서론

오늘날과 같이 경쟁이 심화되는 환경에서 기업이 경쟁력을 유지하기 위한 그리고 기업 내의 의사결정을 지원하는 정보 기반 시스템은 반드시 필요한 수단이다. 데이터 웨어하우스(DW: Data Warehouse)와 데이터 마트(Data Mart)는 의사 결정자들이 정보를 근거로 판단하고, 제품과 서비스에서의 경쟁력 우위를 확보하도록 만들 수 있는 하나의 통합된 데이터 저장 공간이다[1].

(그림 1)에서 보듯이 데이터웨어하우스는 기업의 운영계 시스템에서 발생한 경영데이터 및 외부 데이터 등을 주제별로 통합하여 다양한 형태의 분석을 가능하게 하는 통합 시스템을 의미하며, 데이터마트는 특정 부서의 의사결정 프로세스 지원을 목적으로 하는 부서별 또는 부문별 데이터웨어하우스다. 현재 여러 금융기관에서는 신 BIS제도와 관련하여 고객 신용평가 분석을 위한 다양한 데이터마트 모델들이 연구되고 있다[2]. 하지만 이런 데이터마트 모델들은 대용량데이터베이스에서 세분화된 분석데이터를 요구한다. 또한 이런 분석데이터를 추출하기 위한 업무요건이 복잡할수록 RDBMS의 성능은 저하된다. 따라서 본 논문에서는 데이터웨어하우스에서 데이터마트 구축시 오라클 DBMS의 PL/SQL을 이용하여 효과적으로 분석데이터를 추출 할 수 있는 방법을 제시하고자 한다.

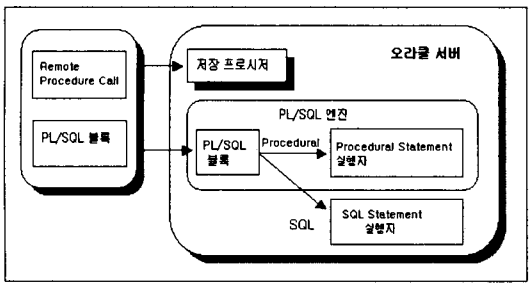


(그림 1) 데이터웨어하우스 아키텍처

본 논문의 구성은 다음과 같다. 2장에서는 기존에 연구되었던 데이터 웨어하우스의 성능향상 방법을 살펴보고, 3장에서는 본 논문에서 제안한 성능향상 방법과 결과를 살펴본다. 그리고 마지막으로 4장에서 결론을 맺는다.

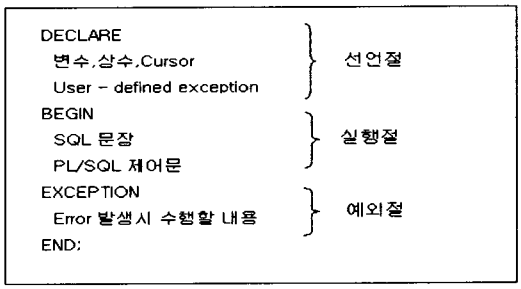
2. 관련연구

DW성능향상 기법으로는 실체화 뷰, 파티션 분할, 테이블 압축, 분석함수, 질의 최적화 등 많은 기술들이 소개되었다[3]. 이중 분석함수는 SQL을 확장한 프로그램 언어인 PL/SQL(Procedural Language extension to SQL)을 이용하여 개발자가 임의로 함수를 만드는 방법이다. PL/SQL은 SQL의 데이터조작과 질의문에 블록구조 및 절차적 단위로 된 코드를 포함 할 수 있으며 절차적 프로그래밍을 가능하게 한 강력한 트랜잭션 처리 언어이다[4]. (그림 2)는 오라클에서 PL/SQL이 물리적으로 어떻게 처리되어지는지를 나타낸다.



(그림 2) PL/SQL 처리 구조

PL/SQL은 (그림 3)과 같은 블록구조를 갖으며 SQL로는 얻을 수 없는 PL/SQL 만의 절차적 프로그래밍 기능을 갖고 있다. 또한 강력한 프로그램을 작성하기 위해 서브 블록들을 큰 블록에 포함시켜서 블록 내에서 논리적으로 관련된 문장들을 그룹화 시킬 수 있으며 복잡한 문제에 대한 프로그래밍시에 적절히 나누어진 모듈들의 집합으로 구성하여 모듈화 된 프로그램을 개발 할 수 있다. 그리고 성능향상을 위하여 PL/SQL은 여러 SQL문장을 단일 블록으로 묶고 한 번의 호출로 블록 전부를 서버로 보내기 때문에 네트워크 통신량을 줄일 수 있어서 응용프로그램의 성능을 향상시킬 수 있다[5]. 또한 PL/SQL은 저장프로시저 및 함수, 패키지, 데이터베이스 트리거와 같은 프로그램을 제작하는데 사용된다. 저장프로시저 및 함수는 복잡한 여러 가지 작업을 하나의 모듈로 만들 수 있고, PL/SQL 블록으로 매개변수를 받아서 반복해서 사용할 수 있다는 장점이 있다.



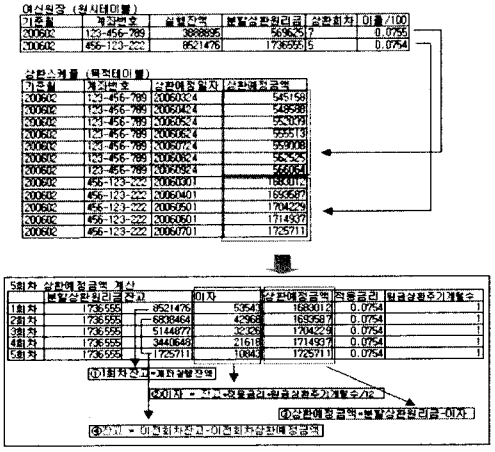
(그림 3) PL/SQL의 블록 구조

3. PL/SQL을 이용한 성능향상 기법

본 장에서는 DW의 성능향상을 위해 본 논문에서 구현한 PL/SQL을 이용한 성능향상 방법을 설명한다. 본 논문에서는 성능평가를 위해 금융기관의 여신업무 데이터마트를 예로 사용하였다.

3.1 버퍼기능을 하는 PL/SQL Package 구현

금융기관의 데이터마트 모델들은 수치와 관련된 복잡한 분석데이터를 요구하고 있어 응용프로그램들이 복잡해지고 이를 튜닝하기 위해 여러 가지 방법들이 이용되고 있다. (그림 4)는 원시테이블을 기준으로 데이터를 가공하여 목적테이블을 작성하는 예를 보여준다.



(그림 4) 원시테이블을 기준으로 목적테이블 생성

데이터 가공시 (그림 4)에서 상환예정금액이라는 필드의 값을 구하기 위해서는 잔고를 계산하여야 하는데 이는 이전 row의 잔고에 영향을 받으며 또한 이전 회차의 상환예정금액에 다시 영향을 받게 된다. 또한 여러 개의 row가 한 그룹으로 구성 되서 이전 row의 변경된 값을 다음 row의 input으로 사용한다. 하지만 기존 SQL로는 위와 같은 그룹데이터 처리가 어려웠다. 이를 위해 [6]에서는 한 그룹의 집단화 값을 구하면서 소속 튜플들의 정보도 같이 보여주는 것을 SQL로 쉽게 처리할 수 있는 방법을 제안하였다. 이 연구에서는 제안된 방법에 기반 해서 분석함수가 표준화된 방법으로 제안되었다[7]. 제안된 분석함수중 lag함수가 이전row에 대한 값을 참조하는 함수이나 이것은 참조값이 고정되어있는 것은 쉽게 사용할 수 있으나 위의 예시처럼 그룹별로 잔고가 가변적으로 변하면서 집단화 값을 계산할 때는 유용하지 않다. 이런 복잡한 요건은 드문 경우이기는 하지만 이럴 경우 SQL사용을 포기하고 (그림 5)처럼 프로그램 안에서 특정부분을 반복처리 하는 Loop문을 사용하여 이전 값을 임시변수에 저장한 후 다음 처리문에서 이전 값을 참조해야 하기 때문에 RDBMS의 성능을 저해하는 요인이 되었다. 따라서 본 논문에서는 버퍼기능을 하는 PL/SQL 패키지를 구현함으로써 RDBMS의 성능을 향상시키고자 한다.

```
Function_declare_cursor() : 여신원장(원시테이블)에서 작업대상 데이터 select
Function_fetch_table() : 작업대상 데이터 fetch
Function_rpay_sche_cal(): 상환예정금액 계산
Function_insert_table() : 상환스케쥴(목적테이블)에 가공된 데이터 insert

Function_declare_cursor();
while(end_data)
   Function_fetch_table();
   for i=1 to 상환회차
     Function_rpay_sche_cal();
     Function_insert_table();
  loop
loop
```

(그림 5) FETCH를 이용한 목적테이블 생성

(그림 6)은 PL/SQL을 이용하여 목적테이블을 생성하는 예를 보여준다. 아래의 방법을 사용하면 조건절에서 생성건수만큼 열을 복제한 후 SELECT절에서 PL/SQL을 이용하여 상환예정금액을 계산한 후 목적테이블을 생성한다. 이는 하나의 Query로 구성되어있기 때문에 FETCH를 이용한 것 보다 성능이 좋다.

```
LN_MAS_TB : 여신원장테이블(원시테이블)
PLAN_TB   : 상환스케쥴테이블(목적테이블)
COPY_TB   : NO 필드만 갖는 테이블로 열복제를 할때 사용한다.
            (여신원장의 상환회차만큼 복제하기 위해 사용)

INSERT INTO PLAN_TB
SELECT 기준월, 계좌번호, 상환예정일자, 상환예정금액계산
       (PL/SQL PACKAGE를 이용한 SQL LOGIC 구현)
FROM
     (SELECT T2.NO, T1.*
        FROM LN_MAS_TB T1, COPY_TB T2
       WHERE T2.NO <= 상환회차)
```

(그림 6) PL/SQL을 이용한 목적테이블 생성

(그림 7)은 본 논문에서 구현한 저장함수(SET_CHA)와 리턴함수(GET_CHA)를 가지는 PL/SQL Package 생성문이다. 즉 2개의 함수를 생성해서 하나의 Package 로 구성하였다.

```
CREATE OR REPLACE PACKAGE BUFFER_PKG
IS
   V_CHA    number ;
   FUNCTION GET_CHA(I_VARCHAR number) RETURN number ;
   FUNCTION SET_CHA(I_VARCHAR number) RETURN number ;
END;
/
CREATE OR REPLACE PACKAGE BODY BUFFER_PKG
IS
  FUNCTION SET_CHA(I_VARCHAR  number )
  RETURN number IS
  BEGIN
   V_CHA     := I_VARCHAR   ;
   RETURN     V_CHA        ;
  END;
  FUNCTION GET_CHA(I_VARCHAR  number  )
  RETURN number IS
  BEGIN
   RETURN     V_CHA        ;
  END;
END;
/
```

(그림 7) 버퍼기능을 하는 PL/SQL Package 생성

```
SELECT
      T3.NO,T3.ACCT_MGMT_NO,T3.EXEC_NO,T3.EXEC_BAL,T3.C_RPAY_AMT
    , CASE WHEN T3.NO = 1 THEN
             EXEC_BAL
           ELSE
             BUFFER_PKG.get_cha('') --저장된 값을 참조,
             -(T3.C_RPAY_AMT-( BUFFER_PKG.get_cha('')*IRT*PRIN_RPAY_CYC_MMS_CNT/1200 ))
      END "잔고"
    , BUFFER_PKG.set_cha --임시변수에 저장,
      (
        CASE WHEN T3.NO = 1 THEN
               EXEC_BAL
             ELSE
               BUFFER_PKG.get_cha('')
               -(T3.C_RPAY_AMT-( BUFFER_PKG.get_cha('')*IRT*PRIN_RPAY_CYC_MMS_CNT/1200 ))
        END
      )
FROM
   (
    ..(중략)
    ) T3
```

(그림 8) PL/SQL Package를 이용한 SQL문

(그림 8)은 생성된 PL/SQL Package를 이용하여 SET_CHA함수로 이전 row의 값을 저장하고 다음

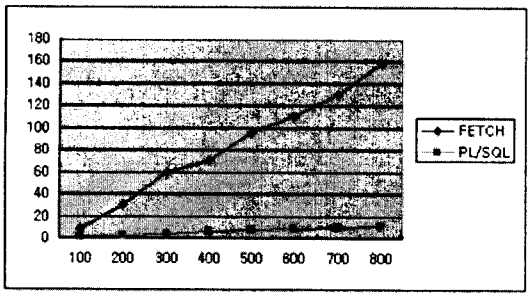
row에서 GET_CHA함수로 저장된 값을 다시 불러와서 임시기억장소인 버퍼와 같은 기능을 할 수 있도록 하였다.

3.2 성능평가

본 절에서는 제안한 모델을 기반으로 은행에서 필요자금을 대출한 고객의 매월 상환예정원금을 구해서 월별로 관리할 수 있는 테이블을 구축하는 알고리즘의 성능을 실험을 통해 평가하였다. 즉 (그림 5)와 (그림 6)의 알고리즘을 구현한 프로그램으로 성능평가 실험을 하였다. 원시테이블인 여신 원장건수는 1,087건이며 한건당 상환회차 만큼 복제가 되므로 목적테이블의 최종건수는 29,779건이다.

3.2.1 분포도에 따른 성능비교

아래 (그림 9)의 결과는 데이터 분포도에 따른 Elapsed Time을 비교한 것이다. 여기서 데이터 분포도란 목적테이블의 최종건수를 의미한다. 아래의 결과에서 살펴보면 FETCH는 한건의 여신원장에 대해서 상환회차 만큼 서브루틴을 반복 처리하므로 분포도가 높아질수록 성능은 크게 저하된다. 이에 반해 PL/SQL은 분포도가 올라가도 성능의 저하가 크지 않은 것을 볼 수 있다.



(그림 9) 분포도에 따른 성능비교 도표

(그림 9)에서 보듯이 본 실험에서 FETCH와 PL/SQL을 비교해 보았을 때 분포도가 올라갈수록 PL/SQL을 이용하는 것이 성능향상에 도움이 됨을 알 수 있다.

3.2.2 성능평가 방법에 따른 비교

아래의 <표 1>은 29,779건의 최종 목적테이블을 생성한 후 수행시간, 코딩의 양, 복잡도를 이용하여 성능을 비교한 것이다.

<표 1> 성능평가 비교

성능평가	수행시간	코딩의 양	복잡도
FETCH	5,880 sec	415 line	61,733
PL/SQL	420 sec	244 line	29,780

FETCH와 PL/SQL을 비교해본 결과 수행시간에서 가장 큰 차이가 발생하였고, 반복 처리하는 복잡한 서브로직이 많아질수록 FETCH를 사용한 프로그램이 PL/SQL을 사용한 프로그램보다 코딩의 양과 복잡도가 높아짐을 알 수 있다.

4. 결론

본 논문에서는 데이터 웨어하우스나 데이터 마트와 같은 의사결정을 위한 시스템의 성능을 향상시키기 위한 질의 최적화 방법을 제안하였다. 본 논문에서 제안한 방법은 가변적으로 값이 바뀌면서 집단화 값을 계산하는 것과 같이 복잡한 요건의 쿼리를 실행시키기 위해 버퍼기능을 하는 PL/SQL Package를 구현하였으며 기존의 FETCH 보다 시간이나 복잡도면에서 좋다는 것을 실험을 통해 알 수 있었다.

참고문헌

[1] 박건, "데이터 웨어하우스와 데이터 마트구축 사례연구", 숭실대학교 석사논문, 2004.

[2] 서종민, "금융기관의 고객신용위험 관리를 위한 데이터마트 모델에 관한 연구", 숭실대학교 석사논문, 2003.

[3] 이상원, "데이터웨어하우스를 위한 데이터베이스 기술 소개", 정보과학회지 제21권 제10호, 2003.

[4] Scott Urman, "ORACLE PL/SQL Programming", 1997.

[5] 김 병 훈 , "PL/SQL을 이용한 효율적인 성적조회 및 졸업관리 시스템 설계 및 구현", 동신대학교 석사논문, 2001.

[6] D.Chatziantoniou, K.Ross, "Querying Multiple Features in Relational Databases," Proceedings of VLDB 1996.

[7] Oracle Corp., "Analytic SQL Features in Oracle 9i," Oracle Technical White Paper 2001.