MySQL에서 기능 보완 기법의 설계 및 구현

신상철*, 이화민*, 윤태명**, 유헌창*, 정순영*
*고려대학교 컴퓨터교육과
**고려대학교 컴퓨터학과
E-mail:sangchul@comedu.korea.ac.kr

A Design and Implementation of Techniques for Supplying the Additional Functions in MySQL

Sang-Chul Shin*, Hwa-Min Lee*, Tae-Myoung Yoon**, Heon-Chang Yu*, Soon-Young Jung*
*Dept of Computer Science Education, Korea Univ.
**Dept of Computer Science & Engineering, Korea Univ.

요약

데이터베이스 관리 시스템은 데이터베이스를 효율적이며 편리하게 관리하기 위해서 사용되는 어플리케이션이다. 이를 위해서 데이터베이스 관리 시스템은 여러 가지 기능들을 사용자에게 제공한다. 이들 데이터베이스 관리 시스템 중에 하나인 MySQL은, 일반적인 데이터베이스 관리 시스템들이 지원해주는 몇 가지 기능들을 지원해주지 않아 사용자들은 불편함을 겪게 된다. 따라서 본 논문에서는 이들 기능들을 보완할 수 있는 기법을 제시하고, 제시된 기법에 따른 설계와 구현을 수행한다.

1. 서론

데이터베이스 관리시스템은 사용자와 데이터베이스와의 인터페이스로써 사용자의 데이터베이스로의 접근과 데이터 조작에 있어서 사용자에게 편리함을 제공해 준다. 이를 위해서 데이터베이스 관리시스템은 여러 가지 기능들, 즉 트랜잭션, 커밋과 롤백, 확장된 질의어, 트리거와 저장 프로시저, 뷰 등과 같은 기능들을 갖추게 된다[1]. 그러나 이러한 기능들은 사용자에게 편의를 제공해 주지만 데이터베이스 관리시스템의 수행속도를 낮추는 단점이 있다. 예를 들어 트랜잭션의 경우 사용자에게 있어서는 편리함을 제공하겠지만, 데이터베이스 관리시스템에 있어서는 데이터 회복과 병행 수행이라는 부수적인 작업을 처리해야 하기 때문에 트랜잭션을 사용하지 않는 질의보다 많은 시간을 필요로 한다. 그래서 몇몇 데이터베이스 관리시스템은 이런 기능들을 제거하기도 한다. 그리고 이런 제거는 데이터 조작이 간단하고, 속도가 중요시되는 상황에서 많이 이루어지고 있다.

그러나 때때로 복잡한 데이터 조작이 필요로 할 때가 있고, 이 때 사용자는 매우 불편함을 겪게 된다. 이런 상황에서 수행속도를 중요시하는 데이터베이스 관리시스템에게 유연성을 줄 필요가 있다. 즉 사용자에게 선택적으로 편리함을 주는 기능들을 제공하는 것이다.

따라서 본 논문에서는 대표적인 속도 중심의 데이터베이스 관리시스템인 MySQL의 기능을 보완함으로써 데이터베이스 관리시스템의 수행 속도와 사용자의 편리함에 대한 선택권을 사용자에게 줄 수 있는 시스템을 어떻게 구축할 수 있는지를 보이고자 한다.

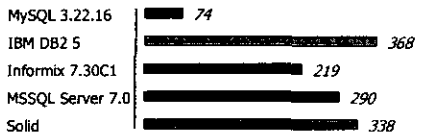
2. 관련 연구

우선 사용한 MySQL이 연구대상으로써 적합한지를 보이기 위해서 다른 데이터베이스 관리시스템과 비교해 보고자 한다. MySQL과 다른 데이터베이스 관리시스템과의 사양 비교는 <표 1>에서, 질의 처리 속도 비교는 <그림 1>에 제시되었다. 기능 비교는 Mark Kirkwood에 의해 이루어진 실험 결과이고[2], 속도 비교는 wisconsin 벤치마킹 프로그램에 의해 얻어진 결과이다[3].

<표 1> MySQL과 다른 데이터베이스 관리시스템과의 기능비교

Database	MySQL	Postgresql	Oracle
Transactions	N	Y	Y
Row Locking	N	Y	Y
Secure	Y	Y	Y
Fail Safe	N	Y	Y

<표 1>에서 나타나듯이 MySQL에는 많은 기능이 제공되지 않고 있다. 특히 MySQL에서 대표적으로 부재된 기능에는 트랜잭션, Sub-Select, 트리거와 저장 프로시저, 외래 키, 뷰 등이 있다[4].



<그림 1> wisconsin에 의한 benchmark

<그림 1>의 수치는 wisconsin 프로그램 자체의 시간 수치이다. 결과를 보면 MySQL이 다른 데이터베이스 관리 시스템보다 높은 수행 속도를 보여주고 있다.

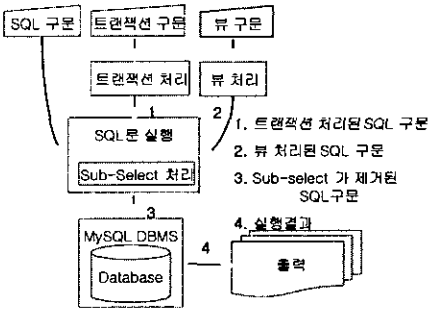
이들 결과들을 보면 MySQL이 속도 중심의 데이터베이스 관리 시스템으로써 많은 기능들을 배제시킨 것을 알 수 있다. 이에 따라 MySQL을 기능 보완할 데이터베이스 관리시스템으로 선정하여 기능 보완 기법을 설계하고 구현하였다.

3. 보완 기법의 설계 및 구현

데이터베이스 관리시스템의 수행 속도와 사용자의 편리함에 대한 선택권을 사용자에게 주기 위한 방법으로 먼저 생각해 볼 수 있는 것은 해당 데이터베이스 관리 시스템, 여기서는 MySQL의 소스 자체를 수정하여 MySQL 작동 시 옵션을 통해서 사용자에게 데이터베이스 관리 시스템의 속도와 기능들 중 하나를 선택하게 하는 것이다. 이것이 가능한 것은 MySQL의 경우 소스가 공개되어 있기 때문이다. 하지만 이 경우 개발에 너무 많은 시간이 소요될 뿐더러 그렇게 개발한다면 소스가 공개되어 있는 MySQL에 대해서만 적용되는 방법이기 때문에 더욱 일반적인 방법이 필요하다. 즉 미들웨어 형태로 데이터베이스 관리 시스템의 기능들을 구현하는 것이다. 이러한 미들웨어 형태를 포함하는 데이터베이스 클라이언트를 개발함으로써 과연 어떻게 미들웨어 형태로 이런 기능들을 보완할 수 있는지를 보이고자 한다. 여기서 보완할 기능들은 뷰, 트랜잭션, Sub-Select이다.

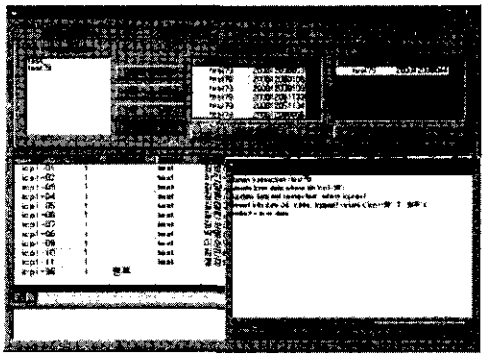
개발 환경으로, Pentium 셀러론 333Mhz, RAM 256MB, 프로그래밍 언어는 비주얼베이직 6.0이다. 사용된 MySQL의 버전은 3.23.27-beta이며, 사용된 ODBC 드라이버는 MySQL ODBC Driver 2.50.32.0 이다. 우선 전체적인 시스템 구성도는 <그림 2>와 같다.

시스템에서 입력으로는 Sub-Select가 포함된 SQL 구문, 트랜잭션 구문, 뷰 구문을 받는다. Sub-Select를 포함한 SQL 구문은 바로 Sub-Select 처리 모듈로 넘겨지고, 트랜잭션 구문과 뷰 구문은 트랜잭션 처리와 뷰 처리가 되어진 후 Sub-Select 처리 모듈로 넘겨진다. 그 다음 Sub-Select 처리가 된 SQL 구문을 MySQL DBMS에 넘겨주고 실행된 결과 값을 반환하여 보여 준다.



<그림 2> 시스템 구성도

<그림 3>은 시스템의 전체적인 인터페이스를 보여준다.



<그림 3> 인터페이스

3.1 Sub-Select 부재 보완

임시 저장 장소를 이용해서 Sub-Select 부재를 보완하도록 한다. Sub-Select의 결과는 일정한 형식을 가진 값들의 나열이거나, 단일 값의 형태로 상위 질의어에 반환되게 된다. 우선 Sub-Select의 결과를 배열에 저장한 다음 이 값들을 가지고 상의 질의어와 함께 다시 실행시키는 방법을 사용한다. 그리고 최종적으로 하나의 SQL 질의 문이 만들어질 때까지 이 과정을 반복하도록 한다. 스택을 자료구조로 사용한다. 그리고 Sub-Select는 "("와 ")" 사이에 있다는 점을 이용해서 Sub-Select를 실행시킨다. Sub-Select를 제거하는 알고리즘은 <표 2>와 같다.

3.2 뷰 부재 보완

먼저 간단한 뷰의 문법을 정의한다[5].

CREATE VIEW view_name AS (table_names WHERE condition) [fields]

여기에서 *view_name*은 생성할 뷰 이름을 뜻하며, *table_names*는 접근할 테이블들의 이름이며 *condition*은 조건절이다. 그리고 *fields*는 프로젝션될 필드들이다. 그리고 뷰 실행의 알고리즘은 <표 3>과 같다.

<표 2> Sub-Select 제거 알고리즘

①		Sub-Select가 존재하는 문자열을 입력받는다.
②		"(", ")" 의 위치를 찾아서 배열에 저장하고 위치순으로 정렬한다.
③		제일 처음의 "(" 문자 이전의 문자열들을 스택에 넣는다.
④	1	괄호와 괄호사이의 문자열과 문자열 앞의 괄호를 같이 읽는다.
	2	괄호가 "("이면 "("와 "(" 이후의 문자열은 스택에 저장시킨다.
	3	괄호가 ")"이면 ")" 이전의 문자열, 즉 스택의 Top에 위치한 문자열을 꺼내서 실행시킨다. 실행결과를 "(", ")"로 싸고 "(" 이전의 문자열과 ")" 이후의 문자열과 합쳐서 스택에 저장시킨다.
⑤		④의 과정을 TOP이 0이 될 때, 즉 하나의 문자열만이 스택에 저장될 때까지 실행시킨다.

<표 3> 뷰 실행 알고리즘

①	뷰 구문을 읽는다.
②	제일 처음 나타나는 "("와 제일 마지막에 나타나는 ")"사이의 구문을 읽고, 제일 처음 나타나는 "["와 제일 마지막에 나타나는 "]"사이의 구문을 읽은 다음, "select" 문자열에다 후자의 문자열을 덧붙인 다음, 여기에다 "from" 문자열을 덧붙이고, 여기에다 다시 전자의 문자열을 덧붙여서 SQL 쿼리를 만들어 낸다.
③	실행시킨다.

3.3 트랜잭션 부재 보완

트랜잭션의 경우 트랜잭션을 처리하는 부분뿐만 아니라 롤백을 위한 방법을 제시한다. 롤백을 위해서 로그 파일을 만드는데, 트랜잭션의 반대효과를 낼 수 있는 SQL 문들의 집합인 또 다른 트랜잭션을 포함한다. 그리고 롤백 시 이 로그 파일에 포함된 트랜잭션을 실행시킨다. 전체적인 알고리즘은 <표 4>와 같다.

<표 4> 트랜잭션 처리 알고리즘

①	트랜잭션 구문을 입력받고 SQL 문들을 뽑아낸다.
②	SQL문들이 접근하는 모든 테이블에 적절한 Lock을 걸어준다.
③	SQL문들을 실행하기 전 SQL문들을 가지고 Log를 생성해 준다.
④	SQL 문들을 실행한다.
⑤	모든 Lock을 풀어준다.

그리고 롤백을 위한 로그 파일 작성시 다음과 같은 규칙에 따라서 SQL 문들을 생성해 낸다.

INSERT 문일 경우 INSERT문에 나타나는 테이블과 FIELD들과 각 FIELD에 해당하는 VALUE 값을 읽어와서 이들로 DELETE 문을 만든다. 예를 들어서 INSERT INTO DATA(ID, ICPNO, ICPPOS) VALUES('ICP1-10', 1, '총무')일 경우 DELETE FROM DATA WHERE ID='ICP1-10' AND ICPNO=1 AND ICPPOS='총무' 문을 생성시킨다.

UPDATE 문일 경우 UPDATE에서 사용된 TABLE, SET 구문, WHERE절을 뽑아서 이들을 가지고 UPDATE 문에 의해 영향을 받은 테이블을 다시 원상복구시키는 UPDATE 문을 작성하도록 한다. 예를 들어서 UPDATE DATA SET ICPNO=3 WHERE ICPPOS='평회원' 일 경우 ICPPOS가 '평회원'인 모든 레코드를 검색해서 각 레코드에서 ICPNO와 Primary key 에 해당하는 값을 뽑은 후에 각 레코드에 대해서 UPDATE DATA SET ICPNO=<원래의 ICPNO 값> WHERE ICPPOS='평회원' AND <Praimary key 속성이름>=<Primary key 값>를 작성해 준다.

DELETE 문일 경우 DELETE에서 사용된 TABLE과 WHERE절을 뽑아서 이들을 가지고 DELETE 문에 의해 영향을 받은 테이블을 다시 원상복구시킬 수 있는 INSERT 문을 작성하도록 한다. 예를 들어서 DELETE FROM DATA WHERE ICPNO=1 문일 경우 먼저 ICPNO가 1인 모든 회원을 조회해서 이들에 대한 모든 정보를 가지고 INSERT 문을 작성한다. 즉 DELETE 문에 해당하는 모든 레코드에 대해서 INSERT INTO DATA (모든 필드들) VALUES(모든 값들) 문을 작성하여서 나중에 다시 정보를 INSERT 함으로써 DELETE에 의한 영향을 원상태로 돌릴 수 있게 한다.

4. 결론

이상과 같이, 속도 중심의 데이터베이스 관리시스템이 지원하지 않는 기능들을 어떻게 미들웨어 형태로 구현할 수 있는지를 본 논문에서 보였다. 이런 미들웨어의 사용을 통해서 사용자들은 속도 중심의 데이터베이스 관리시스템의 속도뿐만 아니라 지원되지 않는 기능들도 선택적으로 사용할 수 있게 될 것이다.

향후 연구 과제로는 트리거와 저장 프로시저, 외래키 등의 기능을 미들웨어 형태로 제공할 수 있는 기법들을 고안하고, 설계에 있어서 좀 더 효율적인 알고리즘을 개발하는 것이다

참고문헌

[1] 데이터베이스 시스템, 北川博之 저/ 이원규, 유헌창, 김현철 역, 홍릉과학출판사, 2000

[2] A comparison of 4 databases, Mark Kirkwood http://www.postgresql.org/mhonarc/pgsql-general/1999-11/msg00227.html, 2000

[3] MySQL Benchmarks http://www.mysql.com/information/benchmarks.html, 2000

[4] MySQL Reference Manual(3.23.27-beta) , TcX, 2000

[5] An introduction to Database Systems 6th edition , C. J. Date, Addition Weley, 1994