실시간 주기억장치 데이타베이스 시스템을 위한 질의 처리기의 설계 및 구현

(Design and Implementation of a Query Processor for Real-Time Main Memory Database Systems)

김 경 배 † 배 해 영 ††
(Gyoung-Bae Kim) (Hae-Young Bae)

요 약 본 논문에서는 주기억장치 데이타베이스의 특성을 반영하여 시간제약조건을 처리할 수 있는 실시간 주기억장치 데이타베이스시스템을 위한 질의 처리기를 설계하고 구현한다. 제안된 질의 처리기는 메타 데이타베이스를 이용하여 시간제약을 갖는 실시간 데이타를 유지 관리한다. 응용 프로그램의 작성을 위해서 CLI를 지원하고 있으며, 이를 확장한 확장 CLI와 저장 CLI를 지원하여 확장 CLI를 이용하여 실시간 트랜잭션의 정보를 CLI를 사용으로 표현할 수 있도록 하였고, 빈번하게 수행되는 트랜잭션을 지원하기 위해 저장 CLI를 지원한다. 제안된 질의 처리기는 주기억장치 실시간 데이타베이스 관리시스템의 질의 처리기로 구현하였으며, 성능평가를 통해서 시스템의 질의처리 능력과 실시간 데이타의 효율적인 관리를 통해서 종료시한을 만족하는 실시간 트랜잭션의 비율이 증가됨을 보였다.

Abstract In this paper, we design and implement a query processor of real-time main memory database systems, which reflect the characteristics of main memory database systems and satisfy timing constraints. The proposed query processor manages real-time data that has timing constraint by exploiting meta database. It supports CLI in order to make application programs. It also supports extended CLI and stored CLI. The former can be expressed the information on real-time transaction. The latter is designed to support frequently processed transaction. The proposed query processor is implemented as query processor of real-time database management systems. We present performance evaluation results that illustrate ratio of transaction, which satisfy deadline are increased by the query processing ability of system and the efficient management of real-time data.

1. 서 론

실시간 데이타베이스 시스템(real-time database systems)[1,2]은 트랜잭션이 종료시한(deadline)과 같은 시간적인 제약조건을 가지는 데이타베이스 시스템이다. 따라서 이러한 실시간 데이타베이스 시스템의 정확성은 논리적인 연산의 결과의 정확성뿐만 아니라 결과가 도출되는 시간에 의존한다. 예를 들어 이동 통신, 정보통신, 원자력 발전소 제어, 항공시스템 제어 등과 같은 실시간 시스템을 위한 실시간 데이타베이스 시스템에서 발생되는 트랜잭션은 논리적 정확성뿐만 아니라 부여된 제한 시간 내에 완료되는 적시성을 가져야만 한다[3]. 실시간 트랜잭션은 트랜잭션이 완료되어야 하는 시간을 나타내는 종료시한을 가지며, 이는 동시 발생하는 실시간 트랜잭션들의 처리를 위해 트랜잭션들 간에 우선 순위의 고려가 필요하다[4]. 실시간 트랜잭션의 우선 순위는 하드(hard), 소프트(soft), 펌(firm)으로 구별되는 트랜잭션의 타입과 트랜잭션 타입을 세분화하는 트랜잭션의 중요도, 그리고 트랜잭션의 종료시한에 의해 결정된다. 실시간 데이타[5]는 유효한 시간 내에 갱신이 일어나지 않으면 그 정보의 유효성을 상실하는 데이타를 말한다. 이러한 실시간 데이타는 정보 갱신 시간에 따라

· 본 연구는 일부 1999년도 한국전자통신연구원의 위탁과제로 진행되었음
† 학생회원 : 인하대학교 전자계산공학과
gbkim@kgi.co.kr
†† 종신회원 : 인하대학교 전자계산공학과 교수
hybac@dragon.inha.ac.kr
논문접수 : 1999년 1월 4일
심사완료 : 1999년 12월 7일

데이타의 일관성 여부가 결정되며, 이를 관리하기 위해 데이타가 실시간 데이타임을 정의하고, 시간 제약 조건을 기술할 수 있으며, 시스템 내에서 실시간 데이타의 갱신 시간 정보를 관리할 수 있어야 한다.

본 논문에서는 디스크로의 입출력 제거로 인해 트랜잭션의 예측성을 지원하는 주기억장치 데이타베이스 시스템을 이용하여 시간제약조건을 처리할 수 있는 실시간 주기억장치 데이타베이스 시스템[6,7,8]을 위한 질의 처리기를 설계하고 구현한다. 구현된 실시간 데이타베이스 시스템은 사용자의 응용 프로그램 작성을 위한 방법으로 CLI(Call Level Interface)[9,10,11]를 지원한다. 실시간 트랜잭션을 지원하기 위해 트랜잭션에 실시간 특성을 부여하고, 주기억장치 상주의 특성과 실시간 데이타를 정의할 수 있도록 CLI를 확장하였다. 확장된 CLI는 실시간 트랜잭션의 종류, 각 트랜잭션의 중요도, 그리고 종료시한 등을 설정할 수 있으며, 설정된 정보는 질의 처리 시스템의 스케줄링 시에 반영하여 처리한다. 또한, 빈번하게 발생하는 질의문을 위한 저장 CLI(Stored CLI)를 지원한다. 사용자는 자주 사용되는 질의를 저장 CLI로 정의하면, 정의된 저장 CLI는 재 수행 시 컴파일 시간을 줄이기 위해 실행플랜(execution plan)상태로 저장되고, 질의 처리기는 저장된 실행 플랜을 이용하여 처리함으로써 빠른 질의 처리를 가능하게 하여 실시간 트랜잭션이 종료시한을 초과하는 가능성을 최소화하였다. 제안된 질의 처리기는 주기억장치 실시간 데이타베이스 관리시스템의 질의 처리기로 구현하였으며, 성능평가를 통해서 시스템의 질의처리 능력과 실시간 데이타의 효율적인 관리를 통해서 종료시한을 만족하는 실시간 트랜잭션의 비율이 증가됨을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 질의 처리 시스템의 구조와 질의처리 과정에 대하여 설명하고, 3장에서는 데이타가 실시간 특성을 갖는 실시간 데이타를 지원하기 위한 기법을 설명하고, 4장에서는 CLI기법의 확장을 통해 실시간 트랜잭션의 특성을 정의하는 방법을 제안한다. 5장에서는 빈번하게 사용되는 질의 구문을 저장하고 사용하기 위한 저장 CLI와 장점을 설명하며, 6장에서는 본 논문에서 제안된 시스템의 성능을 평가하고, 7장에서 결론을 맺는다.

2. 시스템 구조

본 논문에서 제안하는 주기억장치 실시간 데이타베이스 시스템의 전체적인 구조는 그림 1과 같다.

제안하는 시스템은 사용자의 SQL[12]질의를 처리하기 위한 질의 처리기, 주기억장치 저장관리자, 메타 데이터베이스, 저장 CLI 데이타베이스, 주기억장치 상주 데이타베이스, 백업 데이타베이스, 그리고 안전로그로 구성된다. 사용자는 CLI를 이용하여 각종 응용 프로그램을 작성할 수 있으며, 작성된 응용 프로그램의 사용자 질의는 입력 큐를 통하여 서버에 전달된다. 다중 쓰레드로 구성된 서버의 질의 처리기는 사용자의 질의를 파싱하여 질의 실행 관리자를 통하여 데이타베이스에 대한 연산을 수행한다. 결과 값이 삽입, 갱신과 같이 단일 값을 갖는 경우는 출력 큐를 통해 전달되며, 셀렉트 연산과 같이 여러 개의 결과를 갖는 경우에는 출력 큐에는 해당 연산의 성공여부와 연산 결과의 튜플 개수 등의 정보만을 전달하고, 실제 결과는 공유메모리를 이용하여 한번에 전달한다. 즉, 큐를 이용하여 여러 개의 결과 값을 전달하게 되면 결과를 전송 받는 데 많은 시간이 소요되어 응용 프로그램의 실행시간이 지연되어 시간적인 제약조건을 위반하게 된다. 따라서 본 논문에서 제안하는 시스템에서는 빠른 결과 전달을 위하여 공유메모리에 연산의 결과를 저장하고, 응용 프로그램에서 저장된 결과를 자신의 작업 메모리로 한번에 복사하여 결과 값을 사용할 수 있게 하였다.

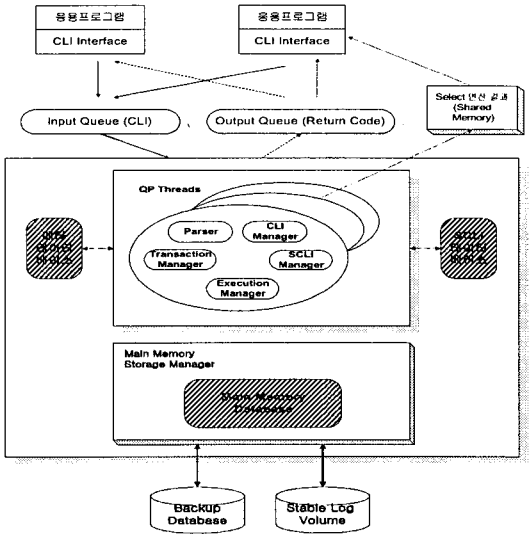

그림 1 주기억장치 상주 실시간 데이타베이스 시스템의 구조

3. 실시간 데이타의 지원

실시간 데이타는 데이타 자체가 시간적인 제약 조건을 지니는 것으로 실시간 데이타를 지원하기 위한 질의어는 데이타가 실시간 데이타임을 정의할 수 있어야 하

며, 실시간 데이타베이스 시스템에서 실시간 데이타는 실제 상태와 비교하여 그 정보의 가치를 유지하기 위하여 시간적 일관성을 유지할 수 있어야 하며, 시간적 일관성의 유지를 위해 시기 적절한 데이타의 검사와 처리를 지원해야 한다. 본 장에서는 실시간 데이타의 특징을 살펴보고, 이를 지원하기 위한 질의어의 확장과 실시간 데이타의 유지관리에 대하여 설명한다.

3.1 실시간 데이타의 특징

실시간 데이타를 지원하기 위해서는 다음과 같은 기능을 지원해야 한다.

① 실시간 데이타의 선언 기능 : 테이블의 생성 시에 생성된 테이블이 실시간 데이타를 포함하고 있음을 표현할 수 있어야 한다.

② 실시간 데이타의 제약조건 기술 : 선언된 실시간 데이타가 지니는 시간적인 제약조건을 설정할 수 있어야 한다. 예를 들면, 원자력 발전소의 상태를 제어하는 시스템에서 원자로의 온도를 기록하는 데이타는 계속적으로 원자로 시스템의 상태를 반영하기 위해 센서로부터 값을 입력받아 갱신할 수 있어야 하며, 제약조건인 갱신 주기를 기술하기 위한 방법이 필요하다.

③ 실시간 데이타의 제약조건 위반 검사 : 실시간 데이타에 대한 연산을 수행하는 경우 연산에 사용하는 실시간 데이타가 시간적인 제약조건을 위반하였는가의 여부를 검사하는 기능이 지원되어야 한다.

④ 실시간 데이타의 회복 기능 : 실시간 데이타가 시간적인 제약조건을 위반한 경우 이 데이타는 더 이상 유효하지 않다. 따라서 유효성을 상실한 실시간 데이타는 그 값을 회복하기 위한 작업이 필요하다.

위의 특성을 지원하기 위한 질의어는 시간정보를 유지하기 위해 데이타가 실시간 데이타임을 정의할 수 있어야 하며, 실시간 데이타베이스 시스템에서 실시간 데이타는 실제 상태와 비교하여 그 정보의 가치를 유지하기 위하여 시간적 일관성을 유지할 수 있어야 하며, 시간적 일관성의 유지를 위해 시기 적절한 데이타의 검사와 처리를 지원해야 한다.

3.2 실시간 데이타를 위한 SQL 데이타 정의어 확장

실시간 데이타를 위해서는 실시간 데이타임을 선언할 수 있어야 하고, 실시간 데이타가 갱신된 시간을 유지할 수 있어야 한다. 따라서 본 논문에서는 기존에 데이타베이스 시스템을 위한 표준 질의어인 SQL의 데이타 정의어에 실시간 데이타의 선언 기능을 확장한 ERTSQL (Extended Real-Time SQL)[13]을 사용하였다.

SQL DDL에서 테이블의 생성 구문에서 실시간 데이타 인 경우에는 **"WITH REALTIME DATA"** 구문을 추가한다. 테이블의 어트리뷰트 중에서 실시간 데이타인 경우에는 데이타의 형을 선언한 구문에 WITH REALTIME DATA를 추가하여 실시간 어트리뷰트임을 선언하고, 실시간 어트리뷰트의 갱신 연산 시에는 저장 관리자에서 변경된 데이타의 값뿐만 아니라 갱신된 시간을 저장한다.

```
base_table_element :
  column_def
  |table_constraint_def;
column_def :
  column_name data_type opt_timestamp;
opt_timestamp :
  |AS REALTIME DATA;
table_constraint_def :
  |AS REALTIME DATA check_constraint_def;
check_constraint_def :
  |CHECK CONSTRAINT time_expression;
```

그림 2 EBNF를 이용한 실시간 데이타의 정의

실시간 제약조건의 기술을 위해서는 실시간 어트리뷰트가 지니는 시간제약조건을 **"CHECK CONSTRAINT"** 구문을 두어 기술할 수 있도록 하였다. 실시간 데이타가 지니는 제약조건을 CHECK CONSTRAINT 다음에 설정할 수 있으며, 설정된 제약조건은 메타데이타베이스를 확장하여 저장한다. 실시간 데이타를 사용하는 트랜잭션 연산에서 메타 데이타베이스에 저장된 제약조건 정보를 이용하여 실시간 데이타가 변경된 시간이 제약조건을 초과하였는 가를 검사한다.

실시간 데이타를 위한 EBNF와 실시간 데이타 선언 예는 각각 그림2, 그림3과 같다.

```
CREATE TABLE PlantLine (
  id          CHAR(10) PRIMARY KEY,
  Name        CHAR(10),
  Operator    CHAR(20),
  Pressure    NUMERIC(5) AS REALTIME DATA
     CHECK CONSTRAINT INTERVAL(10),
  Temperature NUMERIC(5) AS REALTIME DATA
     CHECK CONSTRAINT INTERVAL(5));
```

그림 3 확장된 DDL을 이용한 실시간 데이타의 선언 예

그림 3의 예는, 원자력 발전소에서 각 라인에 대한 압력과 온도를 관리하는 테이블을 정의하는 구문으로, 압력과 온도가 실시간 데이타임을 나타낸다. 시스템은 압

력과 온도 데이타가 갱신될 때마다 그 갱신된 시간을 관리하게 된다. 또한, 시간적 일관성을 유지하기 위해 압력은 매 10초마다, 그리고 온도는 5초마다 데이타 값이 갱신되어야 함을 나타낸다.

3.3 실시간 데이타의 제약조건 및 일관성 유지

시스템에서는 실시간 데이타를 관리하기 위하여 기존 메타 데이타베이스를 확장하여 데이타가 저장되는 시간을 별도로 저장한다. 저장된 시간 정보는 데이타가 갱신될 때마다, 그 데이타가 갱신된 현재 시간으로 바뀌어 저장된다.

실시간 데이타의 타당성을 위한 시간적 일관성을 관리하기 위한 두 가지 방법이 있다. 첫째로, 시스템이 주어진 실시간 데이타의 시간제약조건마다 그 데이타의 시간적 일관성을 검사하고 갱신하는 방법이고, 둘째로 실시간 데이타에 대한 트랜잭션의 접근이 이루어질 때, 그 시간적 일관성을 검사하고, 시간제약조건을 위반했을 경우의 처리를 추가하는 방법이다. 첫 번째 방법은 데이타베이스 내에 존재하는 모든 실시간 데이타에 대해 유효간격 시간마다 일관성 위반 여부를 검사해야 하므로 시스템에 과부하가 따른다.

따라서, 본 연구에서는 사용하는 기법은 실시간 데이타에 접근하는 트랜잭션이 있을 때, 그 데이타에 대한 절대적 일관성 위반 여부를 검사하고, 시간 제약조건을

```
Class RtMetaDB {
Public :  // member function
RtMetaDB();
~RtMetaDB();
void  InitDB(  char *DBName );
void  CreateTbltbl(  chr *TblName);
void  DropTbltbl(char *pTableName );
void  CreateTblInfo( char *TblInfoName );
void  AddTblInfo(  chr *cpTableName, int iRecSize, int
        iFieldCount);
void  DropTblInfo(char *pTablename);
void  CreateTblIdx( char* TblNme);
int    AddTblIdx(char* TblName, char* Idxname, CSqlNameList*
         spFieldname, int Fieldcnt, int IdxNo, char* IdxType);
int    AddTblIdx(char* TblName, char* IdxName, char** spFieldname,
         int FieldCnt, int IdxNo, char IdxType);
int    DropTblIdx(  char *Tblname, char *Idxname );
                    .
                    .
                    .
                    .
};
```

그림 4 메타 데이타베이스 관리자 클래스의 구조

위반한 경우에 위반된 실시간 데이타를 처리하는 기법을 사용한다. 즉, 트랜잭션이 발생하면 메타 데이타베이스를 이용하여 실시간 데이타를 포함하는 연산인가를 검사하고, 실시간 데이타를 포함하는 경우는 실시간 데이타의 제약조건과 데이타의 갱신 시점을 비교하여 위반 여부를 검사한다. 실시간 제약조건을 위반하지 않은 트랜잭션은 정상적으로 수행하지만, 위반된 경우에는 데이타의 정확성을 보장하지 못하므로 실시간 데이타를 갱신하기 위한 연산을 수행하고, 해당 트랜잭션을 재 수행한다. 메타 데이타베이스 클래스의 구조는 그림4와 같다.

4. 실시간 트랜잭션을 위한 CLI의 확장

4.1 실시간 트랜잭션의 지원

본 연구에서 개발된 시스템에서 사용자에게 지원하는 데이타베이스의 응용프로그램 작성 방법으로 CLI를 사용한다. 데이타베이스 사용자는 확장된 SQL을 이용하여 질의를 작성하고, 작성된 질의는 CLI를 이용하여 데이타베이스 시스템에 요청된다.

본 연구에서는 실시간 특성을 갖는 실시간 트랜잭션을 지원하기 위한 방법으로는 기존의 CLI를 확장하였다. 기존의 CLI는 일반적인 데이타베이스 시스템을 지원하기 위한 구조를 지니고 있어 트랜잭션의 종류, 중요도, 시간적인 제약조건 등을 지닌 실시간 환경에 적용할 수 없다. 따라서 본 연구에서는 응용 프로그램을 위한 프로그래밍 인터페이스인 CLI를 실시간 트랜잭션을 위해 확장한 확장된 CLI를 제공한다. 확장된 CLI에서는 실시간 데이타베이스 시스템의 실시간 트랜잭션을 위하여 트랜잭션의 종료시한, 타입, 중요도 등을 설정할 수 있는 기능을 제공한다. 설정된 실시간 트랜잭션의 특성은 질의를 수행하는 실행 관리자에서 아래와 같이 트랜잭션의 우선순위를 반영하여 트랜잭션을 수행한다. 하드나 펌 트랜잭션은 소프트 트랜잭션 보다 우선적으로 처리된다. 또한, 트랜잭션의 유형을 기준으로 실행 중에 발생하는 종료시한의 초과를 처리한다. 예를 들면, 종료시한 내로 처리가 가능한 하드 트랜잭션의 경우는 가장 높은 우선순위를 부과하여 시간제약조건을 만족할 수 있는 하드 트랜잭션의 수를 최대화하였고, 펌 실시간 트랜잭션의 경우에 트랜잭션이 종료시한을 초과한 경우에는 트랜잭션을 취소시키는 정책을 사용한다. 부과된 우선순위는 실행 관리자에서 트랜잭션을 실행하기 위한 쓰레드의 우선순위로 할당되어 처리된다.

4.2 CLI의 확장

확장된 CLI에서 질의문 실행을 위한 수행 단계는 그림 5와 같다.

```
단계1:연결설정
단계2:질의결과 영역 설정
단계3:트랜잭션의 타입 설정
단계4:트랜잭션의 중요도 설정
단계5:트랜잭션의 데드라인 설정
단계6:실시간 데이타 위반시의 정책
단계7:질의문 실행
단계8:연결해제
```

그림 5 확장된 CLI의 수행 단계

확장된 CLI에서는 사용자가 트랜잭션의 특성을 고려하여 데이타베이스를 위한 응용프로그램을 작성할 때, 실시간 트랜잭션의 특성을 부여할 수 있으며, 실시간 데이타의 시간제약조건 위반시의 정책을 결정할 수 있다. 이를 위한 확장된 CLI의 함수들은 표 1과 같다.

확장된 CLI를 이용한 응용프로그램 예는 그림 6과 같다. 이 응용프로그램의 목적은 온도가 1000도가 넘은 발전소 라인을 찾는 트랜잭션을 수행하는 것이다. 이 트랜잭션은 하드 타입, 중(Middle)의 중요도, 0.53초의 종료시한을 갖는 트랜잭션이고, 실시간 데이타인 온도가 시간적 일관성을 위반하였을 경우에도 트랜잭션을 수행한다.

표 1 확장된 CLI의 함수

함 수	사 용 인 자	설 명
CLI_Connection	DB 이름	데이타베이스와 연결
CLI_SetTrType	SOFT_TR, FIRM_TR, HARD_TR	트랜잭션의 타입 설정
CLI_SetTrWeight	LOW_WEIGHT, MIDDLE_WEIGHT, HIGH_WEIGHT	트랜잭션의 중요도 설정
CLI_SetTrDeadline	시간	트랜잭션의 시간제약조건 설정
CLI_Set4RTDataFailure	CONTINUE, ABORT	실시간 데이타의 시간제약조건 위반시 실행여부 결정
CLI_ExecSQL	SQL 질의문과 데이타영역	질의문을 실행하고 데이타 영역에 질의 결과를 받음.
CLI_Disconnect	데이타베이스 이름	데이타베이스와 연결을 해제

5. 저장 CLI(Stored CLI)

5.1 저장 CLI의 역할

실시간 데이타베이스 시스템에서 발생하는 트랜잭션

```
CLI cli;
cli.CLI_Connection("PlantDB");
COM_AREA * pDataArea
        = new COM_AREA;
memset(pDataArea,0x00,sizeof(COM_AREA));
cli.CLI_SetTrType(FIRM_TR);
cli.CLI_SetTrLevel(MIDDLE_WEIGHT);
cli.CLI_SetTrDeadLine( "0.53");
cli.CLI_Set4RTDataFailure(CONTINUE);
cli.CLI_ExecSQL("SELECT id FROM PlantLine
    WHERE Temperature>1000;", pDataArea);
cli.CLI_Disconnect("PlantDB");
```

그림 6 확장된 CLI를 이용한 응용프로그램의 작성 예

의 특징 중 하나는 발생되는 트랜잭션의 종류가 단순하다는 것이다. 즉, 일반 데이타베이스 시스템에서는 다양한 종류의 트랜잭션이 발생하지만 실시간 데이타베이스 시스템에서는 시스템의 특성에 따라 기본적인 유형의 트랜잭션들이 자주 발생한다. 이렇게 자주 사용되는 트랜잭션의 질의어들을 매번 컴파일하여 수행하는 것은 실시간 트랜잭션이 종료시한을 위반할 수 있는 가능성을 크게 하는 것이다. 따라서, 확장된 CLI에서는 이러한 반복적으로 사용되는 질의문 들을 관리하고 사용할 수 있는 방법으로 저장 CLI를 제공한다.

5.2 저장 CLI의 지원

데이타베이스와 접속한 후 사용자는 저장 CLI를 선언할 수 있다. 사용자는 저장 CLI를 위한 함수를 사용하여 여러 번 수행하고자 하는 질의를 저장하고, 이를 삭제, 실행할 수 있다.

저장 CLI를 위한 함수를 사용한 예는 그림 7과 같다.

```
cli.CLI_DefineSQL("GetDangerousLine",
        "SELECT id FROM PlantLine
        WHERE Temperature > 1000;");
cli.CLI_ExecSQL("GetDangerousLine",pDataArea,1);
cli.CLI_Delete("GetDangerousLine");
```

그림 7 저장 CLI 함수를 사용한 예

5.3 저장 CLI의 처리 및 관리

질의문이 실행되는 과정을 살펴보면 다음과 같다. 질의문을 파싱하여 파스트리를 생성하고, 이 파스트리를 가지고 질의어 처리기는 질의를 실행한다. 저장 CLI는 자주 실행되는 질의문의 파스트리를 저장하고, 수행이 필요할 때 사용자의 질의를 컴파일하지 않고 저장된 파

스트리를 이용하여 질의를 수행한다. 이는 컴파일 수행 시 발생하는 파싱하는 시간을 단축시켜 트랜잭션의 전체 수행 속도를 향상시켜 실시간 트랜잭션이 종료시한 준수할 가능성을 높여 준다.

데이타베이스 관리 시스템에서는 저장 관리자를 두어 저장 CLI를 관리하며, 질의문을 저장할 때 질의문을 파싱하여 사용자가 정의한 이름으로 파스트리를 저장한다. 파스트리의 저장을 위해서 저장 CLI를 위한 데이타베이스를 생성하고, 생성된 저장 CLI 데이타베이스에서 유지 관리한다. 저장 CLI의 실행 시, 사용자가 정의한 이름을 사용하여 파스트리를 검색하고, 파스트리를 얻어 수행한다.

6. 성능 평가

본 논문에서 제안된 시스템은 Mr.RT 2.5[14]상에 구현되었으며, 구현된 시스템에 대한 성능을 트랜잭션의 수행시간에 대한 평가와 실시간 데이타의 관리 기능에 대한 성능 평가를 수행하였다.

성능평가에 사용된 데이타 테이블은 30개의 필드를 가지며, 튜플의 크기는 500 바이트이고, 테이블에 속한 전체 튜플의 개수는 500개로 하였다. 표 2는 트랜잭션의 수행 시간을 DBMS 서버와 클라이언트의 응용 프로그램 각각에서 트랜잭션의 평균 수행 시간을 구한 것으로 삽입, 셀렉트, 갱신 연산에 대한 평균 연산 수행 시간과 초당 수행되는 트랜잭션의 속도를 구한 것이다. 삽입과 갱신은 1개의 튜플에 대하여 수행되고, 셀렉트 연산은 전체 테이블을 검색하여 10%인 약 50개의 튜플이 검색되는 경우를 성능평가 데이타로 사용하였다.

표 2 트랜잭션 종류에 따른 연산 처리

항 목		INSERT	SELECT	UPDATE
평균수행 시간	DBMS 서버	54.5 ms	33.8 ms	37.0 ms
	응용프로그램	60.7 ms	36.4 ms	39.6 ms
초당수행 횟수	DBMS 서버	18.3 회	29.5 회	27.0 회
	응용프로그램	16.5 회	27.4 회	25.3 회

그림 8은 실시간 데이타가 제약조건을 위반하는 경우 발생하는 연산의 실패율을 검사한 것이다. 실시간 데이타에 대한 제약조건으로는 트랜잭션의 평균 실행 시간의 10배로 설정하여 수행하였다. 실시간 데이타가 정해진 제약조건 내에 갱신연산이 발생하지 않으면 실시간 데이타를 제약조건을 위반하게 된다. 본 논문에서 제안한 기법은 실시간 데이타의 제약조건 위반이 발생하면 대기중인 연산을 철회하고 실시간 데이타를 갱신하기 위한 회복 연산을 수행한다. 따라서 실시간 데이타에 대한 갱신연산이 실패율의 증가에 따라 회복 연산을 수행하는 제안 기법과 미 수행하는 기존 기법을 서로 비교하였다.

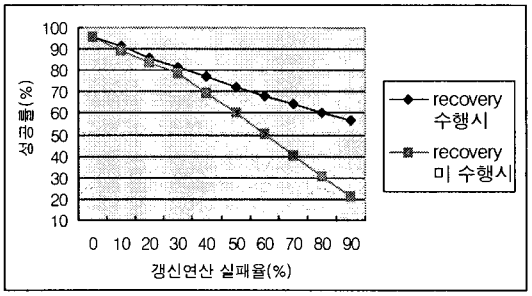

그림 8 실시간 데이타의 제약조건 발생 빈도에 따른 성공률

그림 9는 기존의 CLI 기법과 본 논문에서 제안하는 저장 CLI를 사용한 경우 파스트리를 생성에 소요되는 시간을 비교한 것이다. 질의를 직접 파싱하여 파스트리를 얻는 시간이 0.1msec 단위인데 반해, 저장 CLI를 이용해 파스트리를 얻는 시간은 0.01 msec단위로 직접 파싱하는 것보다 약 10배의 빠른 생성 속도를 보이고 있다. 결국, 빈번하게 발생되는 질의에 대해서 SCLI로 저장함으로서, 수행을 할 때 파스트리 생성속도의 감소로 트랜잭션이 종료시한을 만족할 확률을 더욱 높일 수 있게 된다.

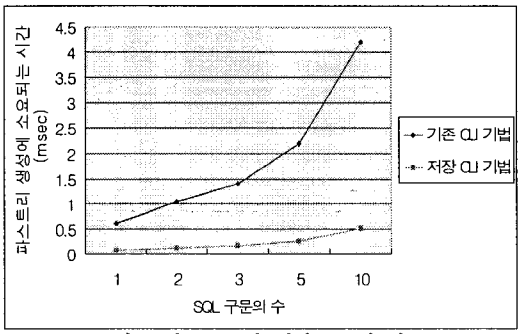

그림 9 기존 CLI와 저장 CLI의 비교

7. 결 론

본 논문에서는 주기억장치 데이타베이스의 특성을 반영하여 시간제약조건을 처리할 수 있는 실시간 주기억

장치 데이타베이스시스템을 위한 질의 처리기를 설계하고 구현한다. 제안된 질의 처리기는 주기억장치 데이타베이스를 기반으로 하며, 시간제약을 갖는 실시간 트랜잭션뿐만 아니라 메타데이타 베이스를 확장하여 시간적인 유효시간을 갖는 실시간 데이타를 지원한다.

응용 프로그램의 작성을 위해서 CLI를 지원하고 있으며, 이를 확장하여 실시간 트랜잭션의 정보를 CLI를 사용으로 표현할 수 있도록 하였다. 또한, 반복되어 자주 사용되는 질의 수행을 지원하기 위해 저장 CLI를 지원한다. 저장 CLI로 선언된 질의 구문은 저장 CLI관리자에 의해 데이타베이스 관리시스템에 실행 트리 형대로 저장되며, 한번 선언된 저장 CLI는 반복적인 수행 시에 질의 구문을 컴파일하지 않고 실행 트리를 바로 수행하여 해당 저장 CLI를 사용하는 트랜잭션의 실행 시간을 단축시킨다.

제안된 질의 처리기는 주기억장치 실시간 데이타베이스 관리시스템의 질의 처리기로 약 8만 라인의 C++코드로 구현하였으며, 성능평가를 통해서 시스템의 질의처리 능력과 실시간 데이타의 효율적인 관리를 통해서 종료시한을 만족하는 실시간 트랜잭션의 비율이 증가됨을 보였다.

참 고 문 헌

[1] B. Kao and H. Garcia-Molina, "An Overview of Real-Time Database Systems," *Real-Time Computing, Springer-Verag*, Vol.127, pp.261-282, 1994.

[2] K. Ramamritham, "Real-Time Databases," *International Journal of Distributed and parallel Databases*, pp.199-226, 1993.

[3] B. Purimetla, R. M. Sivasankaran, K. Ramamritham, and J. A. Stankovic, "Real-Time Databases: Issues and Applications," *Advances in Real-Time Systems*, Prentice Hall, pp.487-507, 1995.

[4] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions," *SIGMOD RECORD, ACM*, Vol.17, No.1, pp.71-81, 1988.

[5] S. J. Lee and H. Y. Bae, "Data Compression Management Mechanism for Real-Time Main Memory Database Systems," *DASFAA*, pp.230-237, 1995.

[6] D. Dewitt, R. Katz, F. Olken, L. Shapiro, M. Stonebraker, and D. Wood. "Implementation Technique for Main Memory Database Systems," *Proc. ACM SIGMOC Conf.*, Vol.14, No.2, pp.1-8, 1984.

[7] L. Gruenwald and S. Liu, "A Performance Study of Concurrency Control in a Real-Time Main Memory Database System," *SIGMOD RECORD, ACM*, Vol.22, No.4, pp.38-44, 1993.

[8] 차상균, 박장호, 박병대, 이성직, "M²RTSS: 주메모리 실시간 저장 시스템", 정보과학회지, 14권, 제2호, pp. 14-23, 1996.

[9] Don Chamberlin, *Using the new DB2*, Morgan Kaufmann, pp. 393-465, 1995.

[10] *Informix-CLI*, Informix Software Inc., pp.1-12, 1997.

[11] ISO/IEC 9075-3:1995, *International Standard for Database Language SQL - Part 3: Call Level Interface*, 1995.

[12] C. J. Date and H. Darwen, *A Guide to the SQL Standard*, 3rd Edition, Addison-Wesley, 1993.

[13] G. B. Kim, S. K. Cho, H. Y. Bae, K. C. Jung, and S. J. Lee, "Extended Call Level Interface(CLI) for Real-Time Main Memory Database Systems," *Proc. of Computer Science Information Technologies'99*, Russia, Vol.1, pp.36-42, 1999.

[14] 임정옥, 김경배, 조숙경, 이순조, 허대영, 배해영, "Mr.RT3.0 : 고성능 실시간 트랜잭션 처리를 위한 주기억장치 상주형 실시간 데이타베이스 시스템," 한국정보과학회 학술 발표 논문집, 25권 2호, pp.208~210, 1998.

김 경 배

1992년 인하대학교 전자계산공학과 학사
1994년 인하대학교 전자계산공학과 석사
1998년 인하대학교 전자계산공학과(박사과정 수료) 관심분야는 이동컴퓨팅, 실시간 시스템, 주기억장치 데이타베이스시스템, 웹GIS

배 해 영

1974년 인하대학교 응용물리학과 공학사. 1978년 연세대학교 전자계산공학과 공학석사. 1989년 숭실대학교 전자계산학과 공학박사. 1985년 Univ. of Houston 객원교수. 1992년 ~ 1994년 인하대학교 전자계산소장. 1982년 ~ 현재 인하대학교 전자계산공학과 교수. 관심분야는 데이타베이스, 멀티미디어시스템, 지리정보시스템, 실시간 시스템