C-LAN 환경하에서의 완전한 SQL 기능을 제공하는 다수 사용자 관계 DBMS

조 성 배, 신 동 일, 최 윤 철
연세대학교 전산과학과

A multi-user relational DBMS with full set of SQL functionality in PC-LAN environment

Sung-Bae Cho, Dong-Il Sin, Yoon-Chul Choy
Department of Computer Science, Yonsei University

요 약

본 연구에서는 종래의 microcomputer용 관계 DBMS들에 비해 보다 완전한 SQL 들을 제공하는 Microcomputer용 relational DBMS를 설계, 구현하였다.
이 시스템은 IBM PC-XT들을 연결하는 Omni-net환경하에서 다수 사용자가 사용할 수 있도록 concurrency control을 simulate하는 방식을 이용하였다.

I. 서론

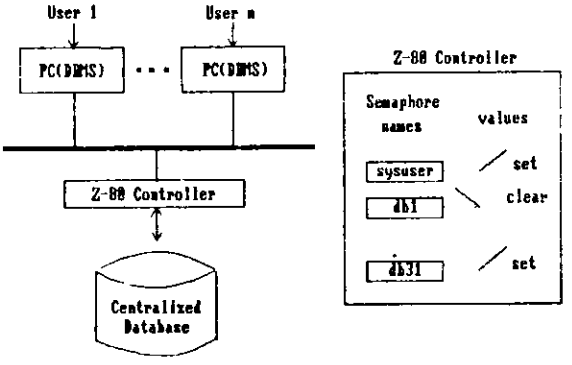
E.F. Codd에 의해 제안된 relational model[4]은 각종 데이타를 table로 표현함으로써 그 구조가 간단하고, 높은 data independence를 보장한다. 이러한 이유에서 IBM사의 system R[7], Berkeley대학의 INGRES[8] 등의 실험용 DBMS들이 mainframe에서 실현되었고, 1980년대에 들어와서 microcomputer의 성능이 향상되고, 가격이 저렴해짐에 따라 dBASE, Condor 3, Knowledge Man, Unify, Oracle, Rbase5000 등 수많은 microcomputer용 relational DBMS가 등장하게 되었다.

이러한 microcomputer 용 package 들이 relational model을 표방하는데 반하여, 실제로는 microcomputer의 여러가지 제약때문에 본래의 relational model 에서는 기능상 벗어남을 볼 수 있다. 특히, SQL을 Query로 하는 system의 경우에는 효율성 문제때문에, 대체로 그 기능이 매우 제한적으로 설계되었다.

본 논문에서는 common database 로 부터 다수 사용자의 다양한 요구를 SQL을 써서 보다 융통성있게 access할 수 있는 DBMS를 PC-LAN (Local Area Network) 환경하에서 구성하였다. 특히, concurrency control이 완전히 지원되지 않는 상황에서 다수 사용자 시스템 설계를 위하여 semaphore를 simulate하는 방식을 적용하였다. 또한 효율적인 screen manager를 이용함으로써 보다 편리한 user interface가 가능하게 되었다.

II. 시스템 조직의 개요

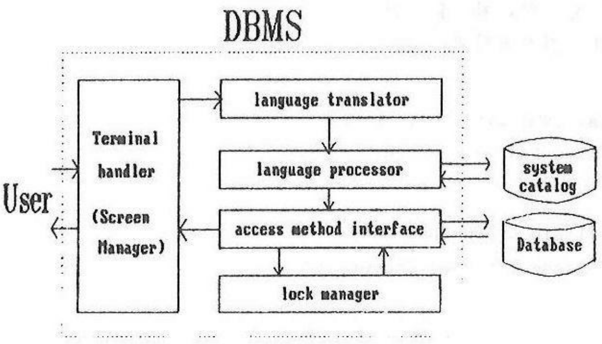
이 시스템은 <그림-1> 에 나타난 것과 같이 데이타 공유를 위하여 데이타를 중앙에서 집중 관리하되, 각 user 의 work station 이 DBMS copy를 각각 소유하는 process-per-user 구조 [6]를 채택하였다.



<그림-1>

각 DBMS의 내부조직은 <그림-2>에서와 같이 다섯가

지 기능으로 대별할 수 있으며, user name과 password로 log-in한 사용자의 SQL문을 처리하여 사용자에게 편리한 형태로 출력한다.



<그림-2>

전체 시스템을 관리하기 위하여 필요한 데이타를 보관하는 system catalog는 SQL문이 실행됨에 따라 dynamic 하게 갱신되어 현재 등록된 user, database, relation, column등의 정보를 유지하며, 아래의 4가지 table로 이루어져 있다.

① SYSUSER : Database를 access하기 위해 등록된 user name과 각종정보를 보관한다.

② SYSDB : Database에 대한 정보를 보관한다.

③ SYSREL : 각 Database내의 relation들에 대한 정보를 보관한다.

④ SYSCOL : 한 Relation내의 column에 대한 정보를 보관한다.

III. 시스템 설계및 구현

앞절에서 언급한 바와같이 이 시스템은 5개의 큰 module로 구성되어 있으며, 본 절에서는 각 module들의 기능및 구조를 설명한다.

1. language translator

SQL은 SELECT-FROM-WHERE를 기본구조로 갖는 non-procedural command로서 [5], 단순한 검색뿐만 아니라 데이타 베이스의 관리를 지원하는 language이며, translator 는 이와같은 SQL문을 scanner, parser, code생성기, semantic분석기[3] 를 통해서 처리하기 쉬운 내부 code로 변환하는 역활을 한다.

1) lexical분석기

크기가 255인 input buffer의 character stream을 token으로 나누는 것으로 19개의 state를 가는 finite table에 의해 identifier, string, number, relational operator, delimiter등을 인식할 수 있도록 했다. Identifier와 reserved word는 reserved word strategy [3] 를 사용하여, identifier로 인식된 것을 binary search로 table look up하여 reserved word와 구별한다. 이때, 보통의 compiler경우에는 symbol table을 만들지만, 여기서는 단순히 stack에 저장하여 code생성기에서 사용하도록 하였다.

2) syntax분석기

syntax분석에는 여러가지 방법이 있지만, 여기서는 LALR(1) parsing기법을 사용하는데, 이는 SLR과 Canonical LR의 중간형태로 parser로는 가장 적당하다고 생각되기 때문이다. 구현과정에서 PC용 YACC의 제한성때문에 YACC를 이용하여 parsing table만을 만들고, program 상에서 drive routine을 작성하여 shift와 reduce action을 수행하는 방식을 적용하였다. 본 시스템의 SQL syntax는 60개의 terminal, 45개의 nonterminal, 102개의 grammar rule, 187개의 state를 가진다.

3) code생성기와 semantic분석기

code생성방법은 syntax directed traslation기법을 써서 syntax분석과정에서 reduce가 일어났을때, 해당 rule이 어떤 operation을 필요로 할 경우 code를 생성한다. 이때 생성된 code는 51개의 operation code 중 한 opcode와 하나의 operand를 가지며, semantic 분석기에 의해서 from clause와 select clause에 명시된 column들의 base table이 올바른지, 또 where clause 중의 expression 들의 type이 일치하는지 검사한 후에 해당 processor로 넘긴다.

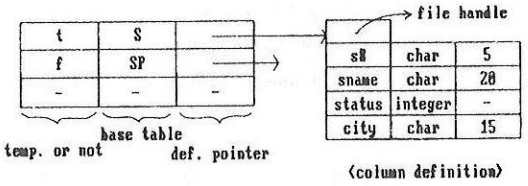
2. language processor

생성된 code로 부터 low-level의 file handler를 이용하여 실제적인 일을 수행하는 부분으로 SQL의 종류에 따라 네가지 processor 가 있는데, QLP를 제외한 나머지 processor는 단순히 acces method function 을 조합하여 처리한다.

1) Query Language Processor - SELECT

LR parsing의 속성으로 WHERE문중의 expression들은 postfix형태로 출력되므로, <그림-3> 과 같은 data structure 와 stack operation을 이용하여 설계하였다. 기본 algorithm은 주어진 predicate를 만족하는 중간 file을 만들어 나가는 것으로 다음과 같으며, 이때

nested query의 처리를 위해 recursive 방식을 이용하였다.



<그림-3>

```
Recursive-function QLP ();
   Push eof mark to P_stack ;
   make dynamic table <fig-3> with base tables of
        FROM-clause ;  /* At first, file handle
                          points original file */
   While  (codes are not exhausted)
      if  (operand)
          if  (nested query)
              recursively call QLP ();
          else
              Push code to P_stack ;
      else if  (boolean operator)
          Pop two predicates from P_stack ;
          if  (join predicate)
              Push predicate to J_stack ;
          else
              process the operation ;
             /*make temporary file from given fi-
              les by applying boolean operator,
              and replace the file handle of
              dynamic table with it */
      advance code pointer ;
   While  (J_stack is not empty)
      Pop predicates from J_stack and process JO-
      IN ;
   dispose all temporary files from dynamic table;
   return  (pointer of result);
```

2) Data Manipulation Language Processor
   - INSERT, DELETE, UPDATE

3) Data Definition Language Processor
   - ACCESS, CREATE, DROP, EXPAND

4) Data Control Language Processor
   -BEGIN TRANS, END TRANS, PASSWORD, QUIT

3. Access method interface [10]

Data independence를 높이기 위하여 상부 interface와 분리하여 data에 대한 low-level operation을 수행하는 부분으로 concurrency를 유지하기 위하여 lock manager의 통제하에 DB를 access한다.

1) Database단위로 조작하는 기능
   - Create_DB, Access_DB, Drop_DB

2) Relation단위로 조작하는 기능
   - Create_Rel, Drop_Rel, Expand_Rel

3) Tuple단위로 조작하는 기능
   - _INSERT, _DELETE, _UPDATE

   이때 disk I/O는 nonkey field에 대해서는 C language가 제공하는 기본 routine중 가장 융통성있는 read와 write를 사용하고, primary key의 index는 B-tree를 이용한다. 이때 B-tree의 최대 depth는 4로서, 저장 가능한 최대 index는 global variable을 setting하여 정해준다.

4) 보조 기능을 하는 routine들
   - User_check, Define_user, Cancel_user
   - DB_check, Rel_check, Column_check
   - Compute_ID, Delete_ID

4. lock manager

다수의 사용자가 데이타를 공유하려 할때 lost update등의 문제가 [2] 발생하므로, 공유 데이타에 대한 access 를 제어하는 기능이 필요하다. Concurrency control을 위해서는 DBMS가 수행되는 OS의 여러 기능[6]에 의해 process간의 communication을 해야하는데, microcomputer의 OS는 그와같은 기능을 지원하지 못하므로 Omninet의 Semaphore facility를 이용하도록 하였다. 이때 Omninet의 Semaphore는 단순히 32개의 hardware flag과 이 flag의 set, clear operation만을 제공하므로 아래와 같은 방법으로 해결하였다.

1) lock granularity는 file단위로 physical locking을 하지만 각 hardware flag에 대응하는 critical data는 data file이 아니라, wait queue를 포함한 각 DB의 상태를 나타내는 data structure로 한다. 이때 critical data를 shared image로 만들 수 없으므로 file로 만들어서 LOCK후에 local area에 copy하여 수정하고, UNLOCK전에 rewrite한다.

2) process의 block과 wake up을 구현하기 위해 status table을 file로 만들어서, block되어 있는 process는 file의 해당위치를 계속 check하면서 clear되기를 기다리다 다른 process가 이 부분을 clear하면 Wake up되고, 자신이 block상태에 들어가기 위해서는 해당위치를 set한다.

3) critical section에 해당되는 monitor는 procedure로 작성하여, relation을 access하려는 process가 수행과정에서 READ_BEGIN, READ_END, WRITE_BEGIN, WRITE_END, DELETE_END중의 한 entry를 호출하면, 그 relation이 속하는 DB의 상태에 대응하는flag을 LOCK한 후에 원하는 relation이 사용가능한지 검사하여 wait queue에 들어가서 기다리거나 해당 작업을 수행한다.

4) 병행수행도를 높이기 위하여 lock은 retrieve만을 위한 SLOCK과 update를 위한 XLOCK의 2가지를 사용하며, 각 process는 global과 local lock table을 가지고 있어서 현재 자신이 lock하고 있는 table들의 이름을 유지하는데, 이것은 단일 SQL문과 transaction문을 구분하기 위한 것이다.

5) Deadlock을 해결하기 위해서 모든 relation에 고유한 인식번호가 증가하는 순서대로 preclaim방법에 의해서 lock을 수행하도록 하고, 무한 연기 가능성을 배제하기 위해서는 wait queue를 만들어 FIFO 전략을 쓴다.

5. Screen Manager

사용자와 시스템사이의 interface로 다양한 window 처리기능을 이용하여 되도록 사용자가 편리하도록 한다. 주요 기능은 사용자로 부터 SQL문을 입력받고, 출력될 결과를 display하는 것으로, 이때 결과는 result window 내에 display되며, 사용자의 제어에 의해 window자체가 왼쪽, 오른쪽, 위, 아래로 움직일 수 있어서 어떤 크기의 결과 table도 자유롭게 볼 수 있다. 이 module은 IBM PC의 ROM BIOS call[9]과 memory mapped display 방식을 최대로 이용하여 만들어 졌으며, IBM PC의 화면 display를 다루는데 필요한 거의 모든 기능을 갖추고 있다.

IV. 결 론

의 사용자가 SQL을 사용하여 데이타베이스를 access할 수 있는 방법을 연구하고 prototype을 개발하였다. 여기서 개발된 DBMS는 기존의 마이크로 컴퓨터용 관계 DBMS에 비해 보다 완전한 SQL질의어를 제공하여 그 기능이 우수하다. 또한 매우 강력한 screen manager를 제공함으로써 편리한 user interface가 가능하게 되었다.

하지만, 실제이용하기 위해서는 화일 액세스 방식을 보다 효율적으로 개선해야 할 과제가 남아 있으며, programming language의 완전한 기능을 이용하기 위하여 host language내에서 embedded되어 사용될 수 있도록 해야 할 것이다.

참 고 문 헌

1. C.J. Date, An Introduction to Database Systems. Vol I, (4th ed.) Addison-Wesley, 1986
2. C.J. Date, An Introduction to Database Systems. Vol II, Addison-Wesley, 1983
3. A.V. Aho and J.D. Ullman, Principles of Compiler Design, Addison-Wesley, 1977
4. Codd, E.F, A relational model of data for large shared data banks. comm. ACM 13, 6,377-387
5. M.M. Astrahan et al., Implementation of a Structured English Query Language, Comm. ACM 18, 10, p.p. 580-588, Oct. 1975
6. M. Stonebraker, Operating System Support for Database management, comm. ACM, Vol 24, 7, p 412-p418
7. M.M. Astrahan et al., System R· Relational Approach to Database Management, ACM Trans. Database Syst. Vol. I, no. 2. pp 97-137
8. M. Stonebraker et al., The Design and Implementation of INGRES, ACM Trans. Database Syst., Vol 1, 3, pp189-222
9. P. Norton, Programmers Guide to the IBM PC, microsoft Press, 1985
10. 홍의경, 다수 사용자를 위한 릴레이셔날 DBMS의 하부 인터페이스의 설계및 구현, 석사논문, KAIST 1983