문서관리시스템을 위한 질의처리기 설계 및 구현

우 종 원† · 윤 승 현† · 유 재 수††

요 약

문서관리시스템은 문헌 정보들에 대한 정보 검색 및 관리를 효율적으로 지원하기 위한 시스템이다. 이러한 문서관리시스템은 하나의 테이블만으로 정보들을 관리하기 때문에 기존 데이타베이스관리시스템에서 사용되는 조인 연산과 뷰 연산 등 많은 비용이 소요되는 연산을 제공할 필요가 없다. 더불어 문서관리시스템은 기존 데이타베이스관리시스템에서는 제공되지 않았던 새로운 연산이 요구된다. 본 논문은 문서관리시스템에서 자료의 구조 정의, 처리 등을 표현할 수 있는 효과적인 데이타 언어를 정의한다. 특히 문서 검색에 필요한 Ranking 연산, Proximity 연산을 제공하도록 정의하고, 정의된 데이타 언어로 작성된 질의를 효율적으로 처리하기 위한 질의처리기를 설계하고 구현한다. 구현된 문서관리시스템을 위한 질의처리기는 기존 관계형 데이타베이스 관리 시스템의 질의처리기를 문서관리시스템의 질의처리기로 사용할 때 나타나는 성능저하 문제점을 해결할 뿐만 아니라 문서관리시스템에서 필요로 하는 새로운 연산을 제공한다.

Design and Implementation of a Query Processor for Document Management Systems

Jong-won Woo† · Seung-hyeun Yoon† · Jae-Soo Yoo††

ABSTRACT

The Document Management System(DMS) is a system which retrieves and manages library information efficiently. Since DMS manages the information using only one table, it does not need to provide join and view operations that spend high cost in traditional DBMS. In addition, DMS requires new operations because of their property. The operation has not been supported in existing DBMSs. In this paper we define a data language which represents the structure definition and process of data on the DMS. Especially we define Ranking and Proximity operation which is needed in Document Retrieval. We also design and implement a query processor to process the query constructed with the data language. When the existing query processors of relational DBMS are used as a query processor of DMS, they degrade the whole system performance. The proposed query processor not only overcomes such a problem but also supports new operation which is needed in DMS.

1. 서 론

현대 사회는 정보교류가 정치, 경제, 사회, 문화기술 등 다방면에서 급속히 이루어지고 있다. 이러한 상황에서 특히 문헌 정보를 보다 효율적이고 다양하게 검색하고 관리할 수 있는 문서관리시스템의 개발이 요구된다. 문헌 정보를 관리하는 문서관리시스템은 기존의 데이타베이스 관리 시스템과는 달리 변경이 거의 발생하지 않고 검색이 주가 되는 연산 특성을 갖으며, 하나의 테이블에서 문헌 정보를 관리하기 때문에 데이타

※ 본 논문은 정보통신부의 정보통신 우수시범학교 지원사업에 의하여 수행된 것입니다.
† 비 회 원 : 충북대학교 정보통신공학과
†† 정 회 원 : 충북대학교 전기전자공학부 교수
논문접수 : 1998년 5월 2일, 심사완료 : 1999년 4월 8일

베이스 관리 시스템에서 자주 발생하면서 가장 비싼 연산중의 하나인 조인 연산이 일어나지 않는 특성을 갖고 있다. 문서관리시스템은 자료의 물리적인 저장을 담당하는 하부 저장 시스템과 데이타의 구조 정의, 저장된 자료에 대한 처리를 담당하는 상부 데이타 언어 시스템으로 분류할 수 있다. 특히 상부 시스템은 특정한 데이타 언어의 실현화(embodiment)라고도 볼 수 있기 때문에, 자료의 구조 정의, 처리 등을 표현할 수 있는 효과적인 데이타 언어의 정의가 필요하다[10].

현재 정보를 효율적으로 관리하고 정보화 추세에 적극적으로 대처하기 위하여 많은 데이타베이스 관리 시스템이 여러 연구소나 기업에서 이미 개발하여 활용중이거나 활발히 개발 추진중이다. 이러한 데이타베이스 관리 시스템을 위한 여러 가지 이론적인 모델중에서 현재는 관계형 데이타 모델이 가장 대표적인 모델로 인정되고 있다. 가장 널리 사용되고 있는 관계형 데이타베이스 시스템의 데이타 언어는 국제 표준으로 채택된 SQL(Structured Query Language)이다[5,9,10]. 그러나, SQL 질의 처리기는 위에서 언급한 바와 같이 문서관리시스템의 질의 처리기로 사용하기에는 그 기능이 너무 다양하고 복잡한 문제점을 갖고 있다. 또한, 문서관리시스템의 질의 처리기가 제공하여야 할 기능들이 구체적으로 제시되지 못한 문제점을 안고 있다. 이러한 질의 처리기의 기능들이 정의되면 현 기술로 그 기능들을 제공하는 질의 처리기를 개발할 수 있다.

본 논문에서는 문서관리시스템에서 자료의 구조 정의, 처리 등을 표현할 수 있는 효과적인 데이타 언어를 정의하고 정의된 데이타 언어로 작성된 질의를 효율적으로 처리하기 위한 질의 처리기를 설계하고 구현한다. 문서관리시스템을 위한 데이타 언어가 제공해야 할 연산 및 기능은 현재 관계형 데이타베이스 관리 시스템의 표준 질의어로 사용되고 있는 SQL에 근거하여 결정한다. 또한 문서관리를 위해 필요한 Ranking 연산 및 Proximity 연산을 제공하도록 추가정의한다. 설계된 질의처리기는 질의분석 모듈, 질의 최적화 모듈 및 질의 수행 모듈 등으로 구성된다. 질의 분석 모듈은 어휘 분석, 구문 분석, 의미 분석 단계를 거쳐 파싱 트리라는 자료 구조를 생성한다. 질의 최적화 모듈은 시스템 카타로그에 기록된 데이타의 상태에 관한 정보를 이용하여 가장 효율적인 접근 계획을 수립한다. 질의 수행기는 하부 구조 연산 기능을 호출하여 결정된 접근 계획에 따라 질의를 수행한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 국내의 관계형 데이타베이스 관리 시스템의 질의처리기의 구조를 간략히 분석한다. 3장에서는 본 논문에서 구현한 질의 처리기의 연산 특성과 구조를 제시하고 4장에서 향후 연구 방향에 대하여 간략히 기술하며 결론을 맺는다.

2 관련연구

2.1 BADA DBMS의 질의처리기

전자통신연구원에서 개발한 BADA 시스템은 데이타베이스 스키마 구축과 관리, 화면폼의 설계와 화면폼을 이용한 자료의 입력 및 질의, 대화형 질의 처리기를 이용한 질의, DBMS의 각 블럭과 응용 프로그램을 호출하기 위한 메뉴의 설계와 실행, C 프로그램내에 SQL 질의어를 포함하여 프로그래밍하는 기능, 보고서의 설계와 실행, 자료를 적재하고 저장하는 유틸리티, 그리고 물리적 자료 저장 장치를 포함하는 자료 접근 방법 기능을 제공한다[5].

2.1.1 BADA DBMS의 구조

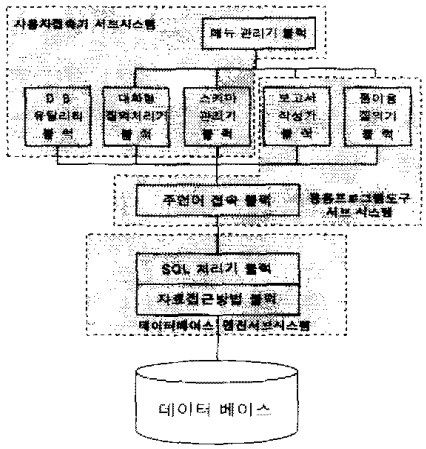
BADA 시스템은 DBMS의 사용자 관점에서 볼 때, 데이타 베이스의 설계·구축을 위한 사용자 접속기 서브 시스템, 응용프로그램 작성자를 위한 응용 프로그래밍 도구 서브 시스템, 그리고 데이타베이스 관리자를 위해 내부적으로 데이타베이스 기능을 처리하는 데이타베이스 엔진 서브 시스템으로 나뉜다[1]. (그림 1)에서 이러한 BADA 시스템의 구조를 블럭화하여 보이고 있다.

2.1.2 SQL 처리기의 구조와 기능

SQL 처리기는 TSQL(TICOM SQL) 문들로 표현되는 사용자 요구를 받아들여 이를 해석하고, 접근 권한 및 스키마의 정당성을 검사한 뒤, 최적화 단계에서 확정되는 최종 수행 계획(Query Execution Plan)에 따라 자료 접근 방법 블럭에서 제공하는 기능을 이용하여 사용자 요구에 대한 결과를 추출하는 과정으로 이루어진다[8].

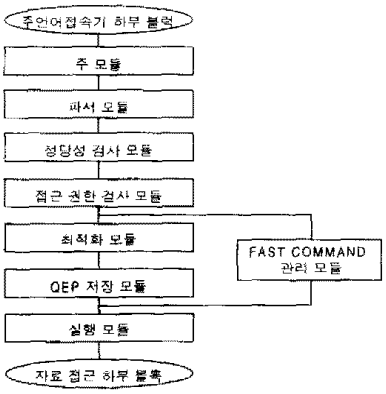
SQL 처리기의 입력 질의어인 TSQL은 ISO에서 관계형 DBMS의 표준 질의어로 채택한 SQL1을 따르고, 여기에 한글 기능을 제공한 것이다. SQL 처리기는 이웃 블럭인 주언어 접속기 블럭으로부터 TSQL문을 입력으로 받아들여 자체 처리하고, 시스템 구조항 하부

블럭의 작동 접근 방법 블럭을 이용하여 수행 결과를 주언어 접속기 블럭으로 돌려준다.



(그림 1) BADA 데이타베이스 관리 시스템 블럭 구조

SQL 처리기 블럭은 (그림 2)에 나타난 것과 같이 전체 8개의 모듈로 구성되어 있다. 그림에서 보듯 SQL 처리기는 위쪽으로는 주언어 접속기 하부 블럭과 인터페이스하고, 아래쪽으로는 자료 접근 블럭과 인터페이스한다.

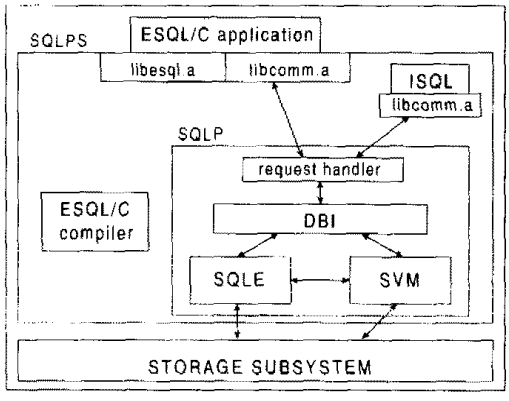


(그림 2) SQL 처리기의 모듈 구조

2.2 SQLPS(SQL Processing Subsystem)

SQLPS는 금성 소프트웨어주식회사의 위탁을 받아 한국과학기술원에서 개발한 UNIX를 기반으로한 관계형 데이타베이스 관리체계를 위한 SQL엔진이다[10]. SQLPS는 자료 저장 하부 시스템을 결합하여 완전한 관계형 데이타베이스 관리 시스템을 구축하게 된다. SQLPS는 SQL을 직접 처리하는 SQLE(SQL Engine), 스키마와 뷰를 관리하는 SVM(Schema & View Manager), 범용 프로그래밍 언어인 C 언어와 SQL을 결합하여 사용할 수 있는 환경을 제공하는 ESQL/C 처리부분, 여러 종류의 사용자 응용 프로그램을 개발할 수 있는 함수의 집합인 DBI(Database Interface), 사용자로부터 직접 SQL 문장을 입력받아서 결과를 출력해주는 응용 프로그램인 ISQL (Interactive SQL)과 같은 주요 성분요소로 이루어지고 있다. SQLPS 각각의 상호 관계를 (그림 3)에서 나타내었다.

SQLPS에서 SQL은 SQLE에서 처리되어진다. 우선, SQL문은 파서를 거치면서 Query Block Tree(QBT)가 되고, QBT는 최적화기에서 의미분석과 최적화를 거쳐 ASL(Access Specific Language)로 변환된다. 파서는 SQL문을 문법적으로 분석하여 QBT를 만드는데, QBT는 SQL문에 나타난 정보들을 잘 정의된 형식에 맞춰 정리한 자료 형태로 최적화기에서 이용하기 쉬운 형태로 질의를 표현한 것이다. QBT는 단순히 SQL문을 다루기 쉬운 형태로 바꾼 것으로 실제로 하부저장시스템의 기본적인 라이브러리를 이용하여 질의를 처리하기 위해서는, 선언적인 SQL문을 더 순차적인 질의형태로 바꾸어야 한다. 이러한 형태가 ASL이며, 최적화기는 의미분석 후에 데이타 접근에 대한 최적화를 한 후 ASL 구조로 저장한다. ASL 실행기는 ASL 블럭들을 보고 하부저장시스템의 기본적인 라이브러리 함수를 이용하여 질의를 수행한다. ASL 구조는 그 자체로서 절차적 기술이므로, ASL 실행기는 단순히 ASL block tree를 주어진 순서대로 수행한다.



(그림 3) SQLPS의 전체 구조

3. 문서관리시스템(Document Management System : DMS) 질의처리기

본 장에서는 본 논문에서 구현할 문서관리시스템의

질의처리기에 관하여 기술한다. 우선, 문서관리시스템 질의처리기의 연산 특성을 제시한 후, 이러한 연산특성에 맞는 데이타 언어를 정의하고, 구현하고자 하는 질의처리기의 구조 및 설계 내용에 대하여 설명한다.

3.1 문서관리시스템 질의처리기의 연산특성

3.1.1 문서관리시스템 질의처리기의 특징

본 논문에서 다루고자하는 문서관리시스템은 문헌 정보와 같은 데이타베이스에서 보다 효율적이고 다양한 정보 검색 및 관리를 위해 필요한 시스템이다. 또한, 본 논문에서 다루고자 하는 문서의 형태는 텍스트 데이타이고, 이와 같은 텍스트 데이타의 중심을 이루는 단어들이 이미 추출되어 데이타베이스에 저장되어 있는 상황을 가정한다. 이러한 시스템의 특징은 단 하나의 테이블만을 관리한다는 것이다. 또한 기존의 DBMS와는 달리 변경이 거의 발생되지 않고 오히려 검색을 주로 하고 있다. 따라서, 기존의 SQL에서 정의된 모든 연산을 사용한다는 것은 효율이나 비용면에서 부적절하다고 할 수 있다.

위와 같은 특징으로 미루어볼 때 문서관리시스템 질의처리기의 연산에서는 우선 조인(JOIN) 연산을 통한 질의가 발생하지 않는다. 즉, 관리하고자하는 테이블이 하나이기때문에 두 테이블간의 연산인 조인은 불필요한 연산이 된다. 특히 조인 연산은 기존의 RDBMS에서 가장 빈번히 사용되는 질의로서 다른 연산에 비해 많은 비용을 요(要)한다. 그러므로 본 문서관리시스템의 질의처리기에서는 조인 연산을 포함시키지 않음으로써 효율이나 비용측면에서 많은 이점을 얻을 수 있다.

조인 연산 외에 고려해야 할 것은 뷰(VIEW) 개념이다. 뷰란 다른 테이블로부터 유도되어지는 일종의 가상테이블로 실제 물리적으로 존재하는 것은 아니다. 즉, DBMS에서는 단지 뷰에 대한 스키마 정보만 유지하는 것이다. 이러한 뷰를 사용하는 주된 목적은 여러 테이블간의 조인을 통해 얻어지는 어떠한 정보들을 자주 질의하는 경우, 그러한 것을 뷰로 만들어 반복되는 질의 과정을 생략하기 위함이다. 그러므로 조인 연산이 사용되지 않는 문서관리시스템에서는 이러한 뷰의 사용 역시 불필요한 요소가 된다. 물론 기본 테이블만을 이용하여 뷰를 생성할 수도 있으나 성능면에서 효율적이지 못하다. 즉, '하나의 테이블'이라는 전제 조건을 지닌 문서관리시스템에서 뷰의 사용 역시 성능을 고려할 때 제외해야 한다.

또한 문서관리시스템은 기존의 데이터베이스관리시스템과는 달리 문서의 종류나 특성에 따라 검색 빈도가 많은 문서가 발생될 수 있으며, 질의를 하는 사용자 또한 자주 검색되는 문서를 더욱 중요히 여겨 먼저 검색되길 요구할 것이다. 이러한 특성으로 인하여 본 문서관리시스템에서는 해당 튜플에 대해 참조 빈도수를 유지하도록 하고, 테이블의 각 칼럼에 대해서는 가중치(Weight)를 부여하여 검색시 이를 이용한 Ranking 연산을 제공할 수 있도록 하였다. 더불어 보다 정확하고 효율적으로 질의 응답을 얻기 위하여 Proximity 연산을 고려한다. 즉, 검색하고자 하는 단어의 위치나 순서 등을 고려할 수 있도록하여 원하는 결과만을 정확히 검색할 수 있도록 한다.

3.1.2 문서관리시스템을 위한 데이타 언어

3.1.1절에서 기술한 바와 같이 문서관리시스템에서는 기존의 SQL에서 조인과 뷰를 포함하지 않는 것이 보다 효율적인 시스템을 구성할 수 있음을 제시하였다. 따라서 문서관리시스템을 위한 데이타 언어로 기존 SQL언어에서 조인과 뷰 연산을 제외한 나머지 데이타 정의어와 데이타 조작어를 사용한다. 단, 테이블의 생성이나 추가, 질의처리기의 파서 트리에 있어서 테이블을 가리키는 테이블 디스크립터 등은 포함하여 설계함으로서 향후 DBMS의 질의처리기로도 사용할 수 있도록 구현되었다. 이와같은 문서 관리 시스템을 위한 데이타 언어를 DMSQL(Document Management System Query Language)라 명명한다.

가. 데이타 정의어(DDL : Data Definition Language)

데이타 정의어란 데이타베이스 객체에 대한 정의나 표현을 제공해주는 언어로서 다음과 같이 테이블의 구조와 관련한 명령과 인덱스에 관한 명령으로 구성되어져 있다. 이들은 질의처리를 하는 과정에서 필요한 시스템 카타로그 정보를 생성하게 해준다. 이러한 명령어들의 사용방법 및 정의방법은 기존 SQL언어에서 정의한 바와 같다.

CREATE	TABLE	(테이블 생성)
ALTER	TABLE	(테이블 변경)
DROP	TABLE	(테이블 삭제)
CREATE	INDEX	(인덱스 생성)
DROP	INDEX	(인덱스 삭제)

나. 데이타 조작어(DML : Data Manipulation Language)

데이타 조작어는 각 객체들의 조작이나 처리를 해주는 언어들을 일컬으며 문서관리시스템의 질의처리기에서는 기존의 SQL에서 위에서 언급한 조인과 뷰에 관련된 것을 제외한 나머지를 구현하도록 한다. 따라서 본 논문에서 구현된 데이타 언어의 데이타 조작어는 다음과 같다.

우선 기본적인 DML문장으로 SELECT, UPDATE, DELETE, INSERT 등이 있다. 이들 문장은 질의에 있어서 가장 기본적인 작업인 갱신, 삭제, 삽입 등을 지원하며 다른 기능들을 이용하여 보다 효율적인 작업을 할 수 있도록 한다. 이와 관련하여 질의에 필요한 추가적인 함수로써 집계 함수가 있다. 집계 함수에는 COUNT, SUM, AVG 등이 포함되어 있으며, 이들 함수들과 연결하여 GROUP BY, HAVING문도 사용할 수 있도록 한다. 그리고 진보된 고급 기능으로서 LIKE나 NULL 등을 질의에 포함할 수 있으며, 다단계 중첩 질의나 부속 질의 등을 통하여 보다 자세한 정보 검색도 가능하도록 구현하였다.

다. 새로운 데이타 조작어

본 논문에서는 문서관리시스템에서의 검색시 사용자의 기대를 반영하기 위해 두가지의 연산을 새로이 제공하도록 구현하였다.

1) Ranking 연산

문서관리시스템에서의 검색시 사용자의 기대를 반영하기 위해 집계 함수로서 Ranking 연산을 구현하여 빈도수 및 가중치를 반영한 검색을 가능케 하였다. 본 질의처리기의 Ranking 연산에서는 우선 각 튜플에 대하여 검색빈도수를 유지할 수 있도록 참조 카운터(Reference counter)를 정의하여 사용자에 의해 튜플이 검색될 때마다 증가할 수 있도록 하였다. 이렇게 함으로써 검색시 검색빈도수에 따라 결과를 나타낼 수 있도록 하였다.

더불어 각각의 attribute에는 가중치를 부여할 수 있도록 하였다. 이때 사용자는 규칙(Rule)을 정의하여 해당 attribute가 입력될 때 정의된 규칙에 따라 가중치 필드도 자동으로 입력되도록 한다. 가중치 필드는 테이블 생성시 attribute 정의 마지막에 'Ranking 규칙이름'을 추가함으로써 생성된다. 이후 Ranking 연산을 이용하여 검색을 하면 정의된 규칙을 이용하여 attribute의 각 값에 대해 가중치가 적용되며 그에 따른 순서대로 정렬되어 검색되도록 구현하였다. 또한 검색시 'WITH RANK(검색 결과 수)'을 사용하여 검색결과의 수도 제한가능토록 하였다. 다음은 공학문서를 유지하는 테이블(Eng_Doc)을 생성하는 과정과 Ranking 규칙을 적용하여 검색시 검색결과의 수를 10개로 제한하는 예를 나타낸다.

- Title, Author, Year를 attribute로 하는 Eng_Doc라는 Table을 생성하라.
  (이때, 'Year' attribute는 Rule1이라는 규칙에 의해 가중치가 부여되도록 정의한다.)

```
CREATE TABLE Eng_Doc
  (Title   CHAR(20),
   Author  CHAR(10),
   Year    DATE Ranking Rule1)
```

Rule1(Year - 출판년도)

1971 - 1980	3
1981 - 1990	5
1991 - 2000	7

- 공학용 책중 computer에 관련된 책의 제목을 10개까지만 검색하라.

```
SELECT Title FROM Eng_Doc WHERE Title
LIKE '%computer%' with Rank(10);
```

* with Rank를 사용함으로써 검색결과 출력시 우선 'Year' attribute의 가중치를 적용하여 해당되는 레코드를 1차 정렬한 후 참조카운터를 이용하여 2차 정렬후 생성된 데이타에서 상위 20개만 선택하여 Result Set을 생성한다.

2) Proximity 연산

Proximity 연산은 사용자의 요구에 보다 더 부합되는 문서들을 검색해주기 위해 제공되는 연산이다. Proximity 연산에서는 질의에 나타나는 단어들의 위치, 순서 등을 고려하여 검색할 수 있다. 즉, 검색을 할 때 여러 가지의 Flag를 이용하여 조건을 구체화할 수 있고, 이를 통해 원하는 결과에 더욱 부합되는 결과만을 제공한다. Proximity 연산을 사용하는 형식은 다음의 예와 같으며, 더불어 연산에 사용하는 Flag를 다섯 가지로 정의하여 설명한다.

```
Author= "우종원"  Prox  Author= "유재수"
(Distance=0, Relation=equal, Unit=paragraph,
```

Ordered=false, **Exclusion**=false);

① Distance (>=0) - unit flag에 근거한 두 피연산자의 순서적 위치값 사이의 차를 나타내는 flag.
(예 : Distance가 0일 경우 같은 Unit내에 두 피연산자가 나타나야 한다.)

② Relation - Distance의 범위를 지정하는 flag로 다음과 같은 6가지의 option이 있다.
LessThan(LT), LessThanOrEqual(LE), Equal(EQ), GreaterThanOrEqual(GE), GreaterThan(GT), NotEqual(NE)
(예 : Distance가 1이고 Relation이 LE인 경우 Distance는 0 또는 1인 경우를 의미한다.)

③ Unit - prox 연산에 나타나는 두 피연산자가 위치하는 단위를 나타내는 flag로 두 피연산자가 반드시 독립적으로 적용되는 것이 아니라 Distance와 Relation의 조합으로 두 피연산자가 나타나는 범위를 지정하며 단위에는 Character, Word, Sentence, Paragraph, Section, Chapter가 있다.
(예 : Unit=paragraph, Relation=EQ, Distance=1이면 두 피연산자는 반드시 이웃하는 paragraph에 각각 나타나야 한다.)

④ Ordered - 두 피연산자가 prox 질의에 나타난 순서를 받아들일 것인지를 결정하는 flag.
(예 : author="우종원" prox author="유재수"이고, Ordered=true이면 "우종원"이 "유재수"보다 먼저 나타나는 문서들만 검색되고, Ordered=false이면 문서에 관계없이 "우종원"과 "유재수"가 나타나는 모든 문서들을 검색한다.)

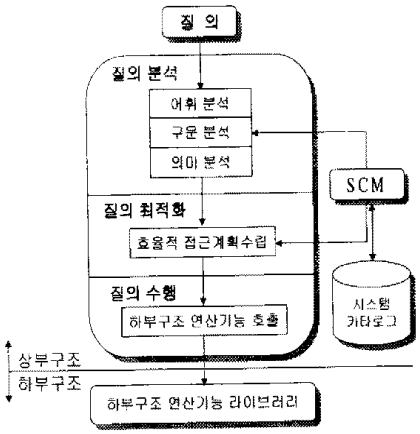
⑤ Exclusion - prox 연산을 부정하는 flag, 즉 True인 경우 해당 prox 연산에 "NOT"을 적용.
(예 : Exclusion flag가 false인 경우 prox 연산이 "동일한 paragraph내에 있는 두 author가 포함되는 문서 검색"이라면 Exclusion flag가 true인 경우 prox 연산은 "두 author가 동일한 paragraph내에 포함되어 있지 않은 문서 검색"을 나타낸다.)

3.2 질의 처리기 설계 및 구현

컴퓨터를 사용하여 대용량의 문헌정보를 효율적으로 관리하며, 정확하고 신속하게 사용자들에게 정보를 제공하기 위해서 데이타관리기법이 개발되었고, 이의 사용 요구가 급증하고 있다. 이의 결과로 문헌정보를 관리하고 사용자들이 편리하게 문헌정보에서 관련문헌을 추출할 수 있도록 하는 문서관리시스템은 업무 전산화 등의 모든 정보 처리 환경에서 중요한 위치를 차지하고 있다. 한편 이러한 문서관리시스템의 사용 요구가 급증하고 있는데도 불구하고 문서관리시스템을 위한 질의어가 제대로 정의되어 있지 않은 실정이다. 이러한 시대적 흐름에 발맞춰 문서관리시스템을 위한 데이타 언어를 3.1.2절에서 정의하였고 본 절에서는 정의된 데이타 언어를 처리할 수 있는 질의 처리기를 설계하고 구현한다. 질의처리기를 설계하고 구현하는데에 사용한 시스템 환경은 다음과 같다. 프로그램 구현을 위해 사용한 하드웨어는 SUN SPARC-5이고, 운영체제는 Solaris 2.5이다. 이용하고 있는 C 컴파일러는 gcc이고, 또한 문서 관리 시스템을 위한 질의어의 언어적인 처리를 위해 Lex와 Yacc을 사용하였다.

3.2.1 질의 처리기의 전체 구성

질의 처리기는 입력된 질의를 분석하고 질의 최적화 단계를 수행한 후, 질의 최적화단계에서 결정된 접근계획(Access Plan)에 따라 하부 구조의 기능을 호출함으로써 질의를 수행한다. 질의 처리기의 전체 구성은 (그림 4)와 같다.



(그림 4) 질의 처리기의 전체 구성

가. 질의 분석

질의 분석은 어휘 분석, 구문 분석, 의미 분석의 단계를 거쳐 파싱 트리라는 자료 구조를 생성한다. 각 단계의 역할은 다음과 같다.

1) 어휘 분석

LEX를 이용하여 사용자가 입력한 질의를 다음의 구

본 분석 단계에서 사용할 수 있도록 입력의 토큰으로 변환한다. 본 연구과제에서 정의한 데이타 언어의 예약어, 한글 또는 영어 데이타, 테이블 이름, 그리고 애트리뷰트 이름이 어휘 분석 단계에서 토큰들로 인식된다. 한글 데이타는 2바이트 완성형으로 표현된다.

2) 구문 분석

YACC를 이용하여 사용자가 입력한 질의가 BNF (backus Naur Form)로 표현된 구문에 맞는가를 검사한다. 그리고 구문 분석 과정에서 구문에 알맞은 파싱 트리를 생성한다.

3) 의미 분석

어휘 분석에서 생성된 토큰들 중에서 테이블 이름, 애트리뷰트 이름이 데이타베이스에 정의되어 있는가를 시스템 카타로그의 내용을 이용하여 분석한다. 이러한 의미 분석은 YACC를 이용한 구문 분석과 병행되어 수행된다.

나. 질의 최적화

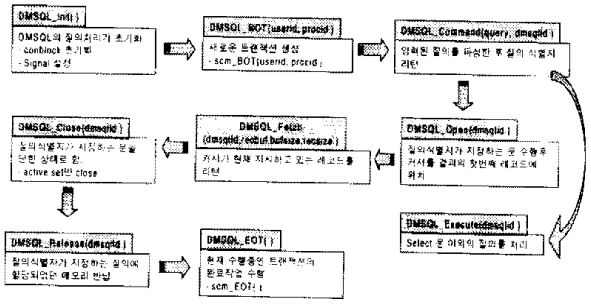
본 논문에서 설계한 데이타 언어는 비절차적 언어이기 때문에 사용자는 원하는 정보를 추출하는 방법 대신에 어떠한 정보를 원하는가 만을 기술하면 된다. 이때 질의 최적기(Query Optimizer)는 시스템 카타로그 기록된 데이타베이스의 상태에 관한 정보를 이용하여 가장 효율적인 접근 계획을 수립하고, 질의 수행기(Query Executor)는 결정된 접근 계획에 따라 질의를 수행한다. 따라서 질의 최적기의 성능은 질의 처리기의 전체적인 성능을 결정하는 가장 중요한 요인이 되므로, 본 논문에서는 효율적인 접근 계획을 작성할 수 있는 질의 최적기를 구현하였다. 구현된 질의 최적기는 경험적 지식(Heuristic)을 기초로 B+트리 인덱스의 효율적인 사용을 가능하게 한다.

다. 질의 수행

질의 수행기는 질의 최적기가 결정한 접근 계획에 따라 하부 저장 구조 연산기능을 호출함으로써 사용자 질의를 최종적으로 수행한다.

3.2.2 응용 인터페이스

문서관리시스템을 위한 질의처리기를 이용하여 DMSQL 질의를 처리하기 위해서는 DMSQL 질의처리기가 제공하는 함수를 이용하여야 한다. 이러한 함수들을 이용하여 질의를 처리하는 과정은 다음과 같다. 우선 질의가 입력된 경우 DMSQL_Init()을 수행하여 질의처리기를 초기화 한다. 이 과정에서는 질의분석후 생성될 Parsing Tree를 저장할 conblock을 초기화하며, 프로그램 수행중 발생할 수 있는 signal들과 그에 따라 수행될 함수들을 정의한다. 그다음 DMSQL_BOT()함수를 호출하여 새로운 트랜잭션을 생성하고 실제 질의를 처리하기 위한 DMSQL_Command()를 호출한다. 이 함수에서는 query를 입력받아 세 단계의 질의분석 단계를 거쳐 Parsing Tree를 생성하고, 생성된 Tree에 대하여 질의식별자를 할당한다. 입력된 질의가 SELECT 또는 INFO문인 경우에는 DMSQL_Open()을 호출하여 전달된 질의식별자를 받아 질의최적화 단계를 거쳐 질의를 수행하여 Active Result Set을 생성, 커서를 생성된 Result Set의 첫 번째 레코드에 위치하게 한다.(그 이외의 질의인 경우에는 DMSQL_Execute()함수를 이용하여 질의를 수행하게 된다.) 이후 DMS_Fetch()를 수행하여 Result Set에서 하나의 레코드를 가져오게 되며, 레코드가 여러 개인 경우 이 함수를 반복하게 된다. 작업이 끝나게 되면 DMSQL_Close()를 호출하여 질의식별자가 지정하는 문을 닫힌 상태로 하는데 이때에는 질의분석후 생성된 Parsing Tree는 그대로 존재하고 단지 생성된 Active Result Set만 닫는 것이다. 따라서 필요한 경우 DMSQL_Open()을 수행하여 다시 Result Set을 만들어 낼 수 있다. 질의에 대한 모든 작업이 끝나면 DMSQL_Release()를 호출하여 질의식별자가 지정하는 질의에 할당되었던 메모리를 반납하고 DMSQL_EOT()를 수행하여 트랜잭션을 종료함으로써 질의처리를 종료하게 된다. 이와 같이 질의처리에 대한 함수의 흐름도를 (그림 5)에서 나타냈다.

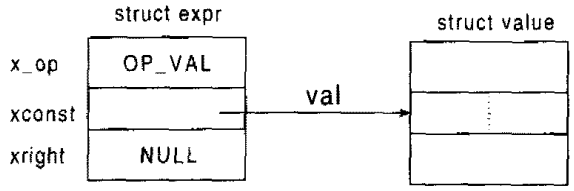


(그림 5) DMSQL Interface Function Flow

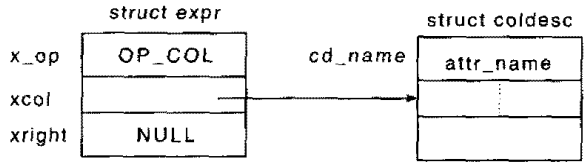
3.2.3 질의 분석시 생성되는 파싱 트리의 구조

질의 분석은 어휘 분석, 구문 분석, 의미 분석의 단

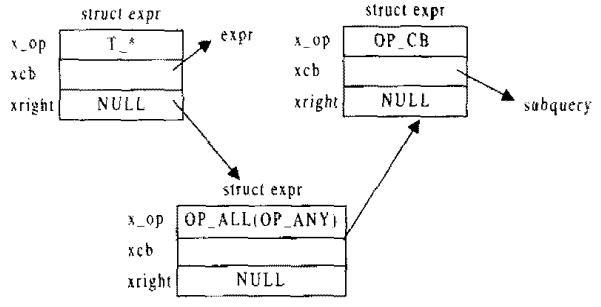
계를 거쳐 파싱 트리라는 자료 구조를 생성한다. 파싱 트리는 다음 단계인 질의 최적화 단계에서 가장 효율적인 접근 계획을 갖도록 변형된 후, 질의 수행 단계에서 하부 저장 구조 연산 기능을 호출함으로써 사용자 질의를 수행하는데 이용된다. (그림 6)에서 (그림 8)은 WHERE절의 boolean_expression을 분석할 때 생성되는 파싱 트리의 구조를 나타낸다. 즉 boolean_expression을 표현하기 위해 사용되는 각각의 구문에 대하여 그 구문을 분석할 경우 결과로서 생성되는 파싱 트리의 구조를 표현하고 있다.



(그림 6) value



(그림 7) column : table name.attr name or attr name



(그림 8) expr T_LT(T_LE, T_GT, T_GE, T_EQ, T_NE) ANY(ALL) LP subquery RP

boolean_expression은 TRUE 또는 FALSE를 최종적으로 리턴하며, 사용자에게 리턴되는 레코드는 WHERE절에서 지정한 BOOLEAN_EXPRESSION의 상태를 만족해야 한다. boolean_expression은 이진 트리로서 표현될 수 있으며, 이러한 이진 트리를 표현하기 위해 연산자, 왼쪽 피연산자, 오른쪽 피연산자의 세 부분으로 크게 구분될 수 있는 EXPR이라는 자료 구조를 정의하였다. 연산자는 EXPR 노드의 역할을 표시하며, boolean_expression의 계산시 각각의 노드는 연산자의 값에 따라 다음과 같은 방법으로 계산된다.

1) OP_VAL

상수 값을 나타내는 노드로서, 왼쪽 피연산자는 실제 값을 저장하는 value 형의 노드를 지시한다. value 형의 노드가 포함하고 있는 값을 리턴한다.

2) OP_COL

테이블내의 칼럼을 나타내는 노드로서, 왼쪽 피연산자는 실제 칼럼에 대 한 정보가 저장되어 있는 coldesc 형의 노드를 지시한다. coldesc형의 노드를 포함하고 있는 값을 리턴한다.

3) OP_PLUS, OP_MINUS, OP_PRODUCT, OP_DIVIDE

왼쪽 자식트리와 오른쪽 자식트리로부터 리턴된 값의 산술 연산을 수행한 후, 그 값을 리턴한다.

4) OP_NEGATION

왼쪽 자식트리로부터 리턴된 값의 부호를 반대로 변환한 후, 그 값을 리턴한다. 즉 양수를 음수로, 음수를 양수로 변환한다.

5) OP_SUM

왼쪽 자식트리로부터 리턴되는 값들의 합을 리턴한다.

6) OP_AVG

왼쪽 자식트리로부터 리턴되는 값들의 평균을 리턴한다.

7) OP_MIN

왼쪽 자식트리로부터 리턴되는 값들의 최소값을 리턴한다.

8) OP_MAX

왼쪽 자식트리로부터 리턴되는 값들의 최대값을 리턴한다.

9) OP_CNT

왼쪽 자식트리로부터 리턴되는 값들의 수를 리턴한다.

10) OP_NOT

왼쪽 자식트리로부터 리턴되는 진리값을 반대로 변환한 후, 그 값을 리턴한다. 즉 TRUE를 FALSE로, FALSE를 TRUE로 변환한다.

11) OP_AND, OP_OR

왼쪽 자식트리와 오른쪽 자식트리로부터 리턴된 진리값의 AND 또는 OR 연산을 수행한 후, 결정되는 진리값을 리턴한다.

12) OP_EQ, OP_GE, OP_LE, OP_GT, OP_LT

왼쪽 자식트리와 오른쪽 자식트리로부터 리턴된 값들의 비교 연산을 수행한 후, 결정되는 진리값을 리턴한다. 이때 오른쪽 자식트리의 연산자 값에 따라 계산 방법이 달라진다. OP_ALL일 경우, 오른쪽 자식트리로부터 리턴되는 모든 값들에 대해 모든 비교 연산이 TRUE이며 TRUE를 리턴한다. OP_ANY일 경우, 오른쪽 자식트리로부터 리턴되는 값들 중에서 임의의 값의 비교연산이 TRUE이면 TRUE를 리턴한다.

13) OP_NULL

왼쪽 자식트리로부터 리턴되는 값이 NULL이면, TRUE를 리턴한다.

14) OP-NOTNULL

왼쪽 자식트리로부터 리턴되는 값이 NULL이 아니면, TRUE를 리턴한다.

15) OP_IN

왼쪽 자식트리로부터 리턴되는 값이 오른쪽 자식트리로부터 리턴되는 값들에 포함되면, TRUE를 리턴한다.

16) OP_NOTIN

왼쪽 자식트리로부터 리턴되는 값이 오른쪽 자식트리로부터 리턴되는 값들에 포함되지 않으면, TRUE를 리턴한다.

17) OP_EXIST

왼쪽 자식트리로부터 리턴되는 값이 존재하면, TRUE를 리턴한다.

18) OP_CB

EXPR 트리에 conblock형의 노드를 연결하는데 사용한다.

SELECT 질의에 대해 파싱 트리가 구성되는 방법은 다음과 같다.

① 사용자 질의에 대해 conblock 구조를 할당하고, SELECT 질의임을 표시하기 위해 cb_selflag는 SQ_SELECT라는 값을 갖는다.

② tabdesc 구조의 td_name은 해당 테이블의 이름을 저장한다. 그 테이블 각각의 애트리뷰트에 대해 하나의 coldesc 구조를 할당하고 모든 coldesc 구조들을 리스트로서 연결 한 후, tabdesc 구조의 td_cols가 그 리스트의 처음을 지시한다.

③ SELECT 절에 명시된 애트리뷰트들로 구성되는 임시 테이블을 가정하여, 그 테이블에 대해 하나의 tabdesc 구조를 할당한다. conblock 구조의 cb_outtab이 그 tabdesc 구조를 지시한다.

④ FROM 절에 명시된 각각의 테이블에 대해 하나의 tabdesc 구조를 할당하고, 모든 tabdesc 구조들을 리스트로서 연결한다. conblock 구조의 cb_tables가 그 리스트의 처음을 지시한다.

⑤ WHERE 절에 명시된 boolean_expression은 앞에서 설명된 방법에 의해 이진 트리로 구성된다. conblock 구조의 cb_where가 그 이진 트리의 루트를 지시한다.

⑥ 질의에 명시된 모든 함수는 하나의 애트리뷰트로 간주된다. 함수 애트리뷰트와 GROUP절에 명시된 애트리뷰트로 구성되는 임시 테이블을 가정하여, 그 테이블에 대해 하나의 tabdesc 구조를 할당한다. conblock 구조의 cb_grouptab이 그 tabdesc 구조를 지시한다.

3.2.4 질의 최적화

질의 최적화 과정은 하나의 질의를 수행하기 위하여 사용할 수 있는 여러 가지 자료 접근 방법 중 가장 효과적으로 빠른 시간내에 수행할 수 있는 접근 계획을 작성하는 과정으로, 필터 팩터 계산, 접근 경로 선택, 접근 계획 생성의 과정을 거친다[9].

가. conjunct 분류

CNF 형태로 만들어진 WHERE절의 각 conjunct를 filterable과 non-filterable conjunct로 구분하여 각각 리스트를 만든다. filterable conjunct는 하부저장 시스템에서 커서를 읽을 때 사용할 수 있는 conjunct들로 비교 프레디키트와 NULL 프레디키트가 속한다. non-filterable conjunct는 하부저장 시스템에서 처리할 수 없는 프레디키트들로 DBLP에서 처리해야 할 LIKE, IN, UNIQUE, OVERLAP 프레디키트들이다. filterable conjunct와 non-filterable conjunct는 각 테이블 단위로 목록이 만들어진다. 여기에서 여러 테이블이 한 conjunct 내에 포함되어 있으면 그 conjunct를 각 테이

블에 모두 포함시킨다.

나. 사용가능 접근 경로 선택

conjunct를 분류한 다음, filterable conjunct에서 사용할 수 있는 접근 경로를 찾는다. 이때 접근 경로에 영향을 주는 요소는 다음과 같다.

- 테이블에 있는 페이지와 행의 수
- 질의의 형태
- 사용 가능한 인덱스
- 정렬(sort) 여부

접근 경로로 사용할 수 있는 접근 방법은 크게 테이블 스캔과 인덱스 스캔으로 나눌 수 있다. 테이블 스캔은 테이블을 처음부터 끝까지 순서적으로 읽어 가는 것을 의미하며, 인덱스가 없거나 또는 테이블 전체를 읽어야 하는 경우 등에서 사용되는 방법이다. 테이블 스캔은 기본적으로 제공되는 접근 방법으로 항상 접근 경로 선택의 고려 대상에 포함된다.

다. 필터 팩터 계산

사용 가능한 접근 경로를 선택한 다음, 각 프레디키트의 필터 팩터를 계산한다. 필터 팩터 계산 과정은 질의어의 조건문에 있는 프레디키트를 적용하였을 때 검색되는 튜플의 수를 미리 계산하는 과정으로 접근 비용을 계산할 때 사용한다.

필터 팩터는 각 프레디키트별로 다른 값을 가지게 되며 계산되는 값과 기본 값이 있다. 팩터로 계산되는 값은 0보다 작거나 1보다 큰 값이 될 수 없다. 부질의어를 포함하는 프레디키트들은 대부분 질의어 변환 과정을 거쳐 모두 부질의어가 없는 질의로 변환되기 때문에 필터 팩터를 계산할 필요가 없으나, 질의어 변환을 할 수 없는 특수한 경우는 예외이다.

라. 질의어 블럭간의 실행 순서 결정

각 프레디키트의 필터 팩터를 계산한 후 질의어 블럭간의 실행 순서를 결정하게 된다. 질의어 블럭은 하나의 SELECT ... FROM ... WHERE 문으로 구성되며, 질의어 표현식은 여러 질의어 블럭으로 구성될 수 있다. 즉, UNION, EXCEPT, INTERSECT 등으로 구성되는 질의어인 경우와 질의어 변환시 임시 테이블을 만들어야 하는 경우, 그리고 질의어 변환 과정에서 변환을 할 수 없는 특수한 경우에 부 질의어를 그대로 남겨두어 처리하는데 이때에도 부질의어가 하나의 질의어 블럭을 이루게 된다. 따라서 여러개의 질의어 블럭간의 실행 순서 결정이 필요하다.

마. 최소 접근 비용 경로 선택

질의어 블럭의 가능한 접근 경로가 선택된 후 각 접근 경로별로 접근 비용을 계산한다. 이때 각 접근 방법에 따라 접근 비용 공식을 이용하여 계산하게 된다. 접근 비용을 계산하기 위해 고려해야할 요소는 입출력 비용과 CPU 비용, 그리고 입출력 비용과 CPU 비용 사이의 weighting factor이다. 접근 비용이 계산되면, 접근 계획을 생성하기 위해서는 최소 비용이 드는 접근 경로를 선택한다.

바. 접근 계획 생성

접근 계획은 질의 최적화 과정의 최종 결과로 만들어지는 것으로, 최소 비용 접근 경로를 선택한 다음, 그 경로를 따라 실제 실행에 필요한 접근 계획을 만든다. 접근 계획은 트리 형태의 자료 구조로 만들어지며 실행에 필요한 모든 정보를 가지고 있다.

접근 계획에 포함되는 정보는 다음과 같다.

- 관계 연산자 및 그 실행 순서
- 각 테이블 및 칼럼 정보

3.2.5 SCM(System catalog management)

하부저장시스템은 사용자의 실세계 관점의 자료 모형을 갖지 않고 단지 컴퓨터 시스템의 관점인 디스크 볼륨, 화일, 레코드의 개념을 제공할 뿐이다. 따라서 사용자로 하여금 실세계를 보다 편리하게 표현하고 이를 관리, 처리할 수 있도록 하기 위해서 문서관리시스템은 보다 상위의 자료 모형을 제공해야 한다. 문서관리시스템의 상부 구조인 질의어 처리기가 사용자에게 단지 하나의 테이블을 제공하기 때문에 이를 하부 구조인 트랜잭션 처리 시스템에 사상시킬 수 있는 시스템 목록 관리자(SCM)가 필요하다.

가. 자료 모형의 사상

본 문서관리시스템의 자료 모형에서 문헌 정보들은 단지 하나의 테이블만으로 나타내지고, 이 테이블은 같은 형식의 튜플을 갖는데, 이들 튜플은 여러 속성(attribute)의 연결로 이루어진다.

트랜잭션 처리 시스템은 하나의 볼륨에 한개의 화일과 이에 대한 여러 개의 인덱스를 제공한다. 그러고

관계의 화일은 서로 같이의 레코드 집합으로 이루어진다. 본 문서관리시스템의 자료 모형에서 테이블은 여러 튜플의 집합으로 이루어진다. 이는 트랜잭션 처리 시스템의 화일로 자연스럽게 사상 될 수 있다.

그러나, 트랜잭션 처리 시스템은 레코드에 어떠한 형식도 제공하지 않는다. 따라서 문서관리시스템의 자료 모형의 튜플을 사상하기 위해서는 적절한 목록 화일이 필요하다. SCM은 튜플의 속성을 트랜잭션 처리 시스템에 사상시키기 위해 목록 화일의 하나인 속성목록(attribute catalog)을 관리한다. 이외에도 상부 구조인 질의어 처리기나 혹은 트랜잭션을 위하여 SCM은 데이타에 대한 시스템 정보를 관리한다. 이의 예로는 테이블 목록(relation catalog), 인덱스 목록(index catalog), 그리고 별칭 목록(synomym catalog) 등 여러 가지가 있다.

나. 목록 화일의 구조

이 절에서는 설계, 구현된 SCM이 제공하는 3가지 목록 화일의 구조를 기술한다.

1) 테이블 목록 화일

테이블 목록 화일은 문서관리시스템에서 관리하는 하나의 테이블 정보를 유지하며, 트랜잭션 처리 시스템에서의 화일 이름은 TBLCAT.SYS이다. 각 테이블에 대하여 관리되는 정보는 다음과 같다.

TblName : 테이블의 이름을 저장
TblID : 테이블의 식별자로서 트랜잭션 시스템에서 관리되는 화일의 식별자를 저장
Type : 테이블의 종류(REGULAR, CATALOG, TEMPORARY 등)
Owner : 테이블의 소유자 식별자로서 운영체제에서 관리되는 사용자의 식별자를 유지
Perm : 테이블의 연산 허용에 관한 정보 (소유자 및 비소유자에 대해 기록, 판독 허용 성도 표시)
NumPages : 이 테이블에 할당된 페이지 개수를 저장
Card : 현재 테이블에 저장된 튜플의 개수를 표시
NumIndex : 이 테이블에 대하여 만들어진 B-트리 인덱스의 개수 표시
NumHash : 이 테이블에 대하여 만들어진 확장 해쉬 구조 인덱스의 개수 표시
Length : 이 테이블에 저장되는 튜플의 길이를 바이트 수로 저장
NumAttr : 이 테이블에 저장되는 튜플이 갖는 속성의 개수 표시
Created : 테이블의 생성 시각을 저장

2) 속성 목록 화일

속성 목록 화일은 각 테이블의 튜플 형식을 기술하는 화일로 트랜잭션 처리 시스템에서 ATTRCAT.SYS라는 이름으로 관리된다. ATTRCAT.SYS에는 다음과 같은 정보를 저장된다.

AttrName : 속성 이름을 저장
TblID : 이 속성이 속한 테이블의 식별자를 저장
AttrNum : 이 속성이 튜플에서 몇 번째 속성인가를 표시
AttrDesc : 속성의 기술로서 튜플에서의 시작 위치, 길이 그리고 자료의 형으로 이루어짐.

3) 인덱스 목록 화일

테이블의 여러 개의 속성에 대하여 B-트리 인덱스가 형성될 수 있는데 이때, 각 인덱스의 구조 및 정보를 인덱스 목록 화일(INDEXCAT.SYS)에 저장한다. 다음은 SCM이 제공하는 인덱스 목록의 구조를 나타낸 것이다.

IndexNo : 한 테이블에 대하여 인덱스가 여러개 생성 가능, 각각의 일련번호가 저장.
KeyDesc : 인덱스를 생성하는데 이용된 키의 정보 표시
TblID : 이 인덱스가 어느 테이블에 대하여 생성되었는지 표시. (해당 테이블의 식별자를 저장)
Owner : 인덱스를 생성한 사용자의 식별자를 저장
Perm : 이 인덱스의 비소유자에 대한 사용 허용 정도 표시
Numpages : 인덱스가 차지하고 있는 페이지 개수를 저장
Card : 인덱스에 저장된 인덱스 레코드의 개수 표시
Created : 인덱스가 생성된 시각을 저장

다. 구현

① scm_createdb(DBName, Title, DBID, NumExt, ExtSize)

DBName 이름으로 갖는 디스크의 한 볼륨을 사용자가 사용할 수 있도록 데이타베이스로 만들어준다. 여기서 Title은 이 데이타베이스에 주는 사용자 개념의 이름이고 DBID는 이 데이타베이스를 다른 데이타베이스와 구별하기 위한 식별자이다. NumExt는 데이타베이스가 차지하는 Extent의 개수이고 ExtSize는 한 Extent가 몇 개의 page로 이루어져 있는지를 나타낸다. 트랜잭션 처리 시스템을 통하여 데이타베이스를 생성한 다음 테이블 목록 화일, 속성 목록 화일 그리고 인덱스 목록 화일등을 생성한다.

② scm_opendb(DBName)

트랜잭션 처리 시스템을 이용하여 DBName으로 지정된 데이타베이스의 정보를 주기억 장치에 읽어들인 후, 이 데이타베이스의 모든 목록 화일을 연다.

③ scm_closedb()

현재 사용중인 데이타베이스의 모든 목록 화일을 닫은 다음, 이 데이타베이스의 변경된 모든 정보를 디스크에 기록하고 데이타베이스를 닫는다.

④ scm_createtbl(TblName, Attr, Type)

현재 사용중인 데이타 베이스에 TblName을 이름으로 갖는 관계를 생성한다. Attr은 생성될 관계의 속성 정보이고 Type은 이 관계가 REGULAR인지, TEMPORARY인지를 나타낸다.

⑤ scm_destroyrel(TblName)

현재 사용중인 데이타베이스에서 TblName을 이름으로 갖는 관계를 지운다.

⑥ scm_renametbl(New, Old)

현재 사용중인 데이타베이스에서 Old를 이름으로 갖는 테이블 이름을 바꾼다.

⑦ scm_createbtree(TblName, BtreeNo, keyDesc, primary, Cluster)

현재 사용중인 데이타베이스에서 Tblname을 이름으로 갖는 테이블에 대하여 B-트리 인덱스를 형성한다. 여기서 BtreeNo는 형성하고자 하는 인덱스의 일련 번호이고, keyDesc는 키의 정보이다. 그리고, Primary가 TRUE이면 키의 중복을 허락하지 않는 인덱스를 형성하게 되고, Cluster가 TRUE이면 자료 화일도 주어진 키의 순서대로 정렬된다.

⑧ scm_destroybtree(Tblname, BtreeNo)

현재 사용중인 데이타베이스에서 TblName을 이름으로 갖는 테이블에 대한 인덱스 중에서 일련번호가 BtreeNo인 인덱스를 삭제한다.

⑨ scm_sortinto(TblName, IntoTblName, KeyDesc, Suffix)

현재 사용중인 데이타베이스에서 Tblname을 이름으로 갖는 테이블을 KeyDesc에 의해 서술되는 키에 따라 정렬하여 IntoTblName이 가리키는 이름의 화일로 저장하여 준다. 이때 Suffix는 정렬시 사용할 임시 화일 이름의 일련번호이다.

⑩ scm_updstat(TblName)

TblName으로 지정되는 관계의 통계를 변경한다. 이때 해당 관계는 반드시 REGULAR형 이어야 한다.

⑪ scm_updstatall()

REGULAR형인 테이블의 통계를 변경한다.

⑫ scm_updcatstat()

CATALOG형인 테이블의 통계를 변경한다.

3.3 구현된 질의처리기의 고찰

본 논문에서는 문서관리시스템을 위한 질의처리기를 제안하였고, 이러한 질의처리기에서 요구되는 특징을 분석하여 문서관리시스템에 맞는 질의처리기를 설계·구현하였다. 본 절에서는 이렇게 구현된 질의처리기를 기존의 관계형 데이타베이스시스템의 질의처리기와 비교하여 그 특징을 살펴본다. 또한 현재 상용화되어있는 전자 문서관리시스템의 질의처리기와도 비교하여 보도록 한다.

3.3.1 기존 관계형데이타베이스 시스템의 질의처리기와의 비교

기존의 관계형 데이타베이스 시스템은 일반적으로 여러 개의 테이블을 만들고, 그들간의 relationship을 유지하도록 하고 있다. 이를 기반으로 여러가지 기본 DML과 함께 뷰, 조인 등의 연산을 제공할 수 있도록 질의처리기가 구현되어있다. 특히 뷰와 조인 연산은 대개 두 개 이상의 테이블을 이용하여 검색을 이루는 연산이기에 프로그램이 복잡하며, 그로 인해 다른 어떤 연산보다도 높은 비용을 요구한다. 그러나 본 논문에서 대상으로 하는 문서관리시스템에서는 단지 하나의 테이블만을 관리하는 특징을 지니고 있다. 또한 문서관리시스템에서는 갱신보다는 검색 연산이 주로 이루어지며, 문서의 종류나 검색 빈도수 등에 따라 우선 검색을 필요로 하는 경우가 발생한다. 그러나 기존 데이타베이스 시스템의 질의처리기는 이러한 것을 고려한 연산을 제공하지 못하고 있다.

본 논문에서 구현한 질의처리기는 이러한 기존의 관계형데이타베이스 시스템의 질의처리기가 갖는 단점을 해결하기 위해 우선 많은 비용을 요하는 연산인 뷰와 조인 연산을 제거하였다. 그렇게 함으로써 기존 관계형데이타베이스 시스템보다 간결한 질의처리기를 구성하여 효율과 비용면에서 우수함을 얻을 수 있었다. 둘째, 검색이 주가 되는 문서관리시스템의 요구를 반영하여 기존 시스템에서 제공되지 않았던 Ranking 연산을 제공할 수 있도록 하였다. Ranking 연산에서는 검색 빈도수, 문서의 중요도에 따른 가중치를 반영하여 사용자 질의시 중요한 데이타를 우선 보여줄 수 있도록 하였다. 또한 Proximity 연산을 구현하여 원하는 결과를 보다 정확하고 쉽게 검색할 수 있도록 하였다. Proximity 연산은 검색하고자 하는 단어의 위치나 순서까지도 반영할 수 있기에 사용자가 원하는 결과에 가장 접근하는 데이타만 Filtering하여 얻을 수 있다.

3.3.2 전자 문서관리시스템과의 비교

현재 여러 가지 전자문서관리시스템이 상용화되어 시판되고 있다. 이러한 전자문서관리시스템은 편리한 사용자 인터페이스를 제공하고 있으며, 보안이나 사용자 관리 등에서 여러 가지 서비스를 제공하고 있다. 그러나 이러한 시스템의 질의처리기에서는 본 논문의 질의처리기에서 제공하는 비교연산자, COUNT나 SUM과 같은 집계함수(Aggregate Function), GROUP BY와 같은 grouping 등을 제공하지 못하고 단지 기존 정보검색시스템에서 사용하는 제한된 질의만을 제공한다. 따라서 사용자가 임의로 다양한 형태의 질의를 구성할 수 없으며, 이로 인해 구체적인 질의가 어려워져 많은 양의 검색결과를 제공하는 단점을 갖는다. 또한 검색 결과에 대해서도 해당 시스템에서 제공되는 제한된 형태로 보여줄 수 밖에 없다.

그러나 본 논문에서 구현한 질의처리기에서는 기존 관계형데이타베이스 시스템의 질의처리기를 수용하여 여러 비교연산자나 집계함수, 중첩질의 등 사용자가 다양한 질의를 구성할 수 있도록 하였다. 또한 질의 구성시 검색결과의 형태도 필요한 Format으로 선택하여 원하는 형태의 결과를 얻을 수 있도록 하였다. 즉, 일부 항목만 제한하여 결과로서 나타내거나, 집계함수 등을 사용하여 새로운 항목을 검색결과로서 만들어 낼 수 있도록 함으로써 사용자의 다양한 요구를 수용할 수 있도록 하였다.

4. 결 론

본 논문에서는 문헌 정보를 관리하는 문서관리시스템의 데이타 언어를 기존관계형 데이타베이스 관리 시스템의 SQL 언어에 기반하여 정의하였고 이를 처리할 수 있는 질의 처리기를 설계하고 구현하였다. 문서관리시스템을 위한 질의 처리기는 질의분석기, 질의 최적화기, 질의 수행기 및 시스템 카타로그 관리자로 이루어져 있다. 이렇게 구현된 문서관리시스템의 질의처리기는 기존 정보검색시스템 및 문서관리시스템의 질의 처리기를 대체할 수 있는 효과를 얻을 수 있다. 그리고, 이를 기반으로 국제 수준의 S/W 판매 경쟁력을 확보할 수 있을 것이다. 또한, 문서관리시스템의 핵심 기술개발로 인해 문서관리시스템 개발시의 비용 절감 효과도 기대할 수 있을 것이다. 향후에는 본 논문에서 구현된 문서 관리 시스템을 위한 질의 처리기를 하부 저장 시스템과 통합하여 DMSQL의 처리 결과의 분석과 시험을 수행하고자 한다.

참 고 문 헌

[1] 한국전자통신연구소 컴퓨터 시스템 연구실, "데이타베이스 관리 체계 서브시스템 설계서", 한국전자통신연구소, 1990.

[2] 한국전자통신연구소 컴퓨터 시스템 연구실, "SQL 처리기 블록 설계서", 한국전자통신연구소, 1990.

[3] 오석균 외 4인, "TDBMS에서의 FAST COMMAND 지원", '90 동계 데이타베이스 학술세미나논문집, pp.136-144, 1990. 2.

[4] 이애영 외 5인, "TDBMS의 사용자 접속기 개발", '90 동계 데이타베이스 학술세미나논문집, pp.145-157, 1990. 2.

[5] 김명준, 임기욱, "관계 데이타베이스 관리 시스템 설계 개발", 대한전자공학회 전자계산연구회 컴퓨터기술, 제7권, 제1호, pp.127-135, 1990. 12.

[6] 손지수 외 3인, "스키마 관리 도구 설계 및 구현", '91 동계 데이타베이스 학술대회논문집, 제5권, 제1호, pp.116-127, 1991. 2.

[7] 박진원 외 12인, "TICOM 시스템 설계서", 한국전자통신연구소 행정전산망주전산기개발본부, 1990. 11.

[8] 오석균 외 5인, "TDBMS SQL 처리기 개발 전략", 한국전자통신연구소 행정전산망주전산기개발본부.

1989. 12.

[9] 이윤홍 외 4인, "바다 II 데이타베이스 언어 처리기의 설계 및 구현", 한국정보과학회, Vol.20, No.2, 1993

[10] 한국과학기술원, "UNIX를 기반으로한 관계형 데이타베이스 관리체계를 위한 SQL엔진 개발", 금성소프트웨어주식회사, 1993. 9.

[11] 김성한 외 3인, "UNIX 시스템 툴 잘쓰기", PC 어드밴스, pp.548-612, 1996.

[12] 충북대학교, "데이타관리시스템을 위한 질의처리기 원형 개발 최종보고서", 연구개발정보센터, 1997. 12.

[13] R. Lorie and J. Nilson, "An Access Specification Language for a Relational Database System," IBM Journal of Research and Development, Vol.23. No.3, 1979.

[14] W. Kim, "On Optimizing an SQL-like Nested Query," ACM Trans. Database Systems, Vol.7, No.3, 1982.

[15] R. Ganski and H. Wong, "Optimization of Nested SQL Queries Revisited," Proc. Int'l Conf. ACM SIGMOD, 1987.

[16] S. Ceri and G. Gottlob, "Translating SQL into Relational Algebra : Optimization, Semantics, and Equivalence of SQL Queries," IEEE Trans. Software Eng., 1985.

[17] J. Melton and A. R. Simon, "Understanding the new SQL : a complete guide," Morgan Kaufmann Publishers, 1993.

[18] P. Selinger et al., "Access Path Selection in a Relational Database management System," Proc. Int'l Conf. ACM SIGMOD, 1979.

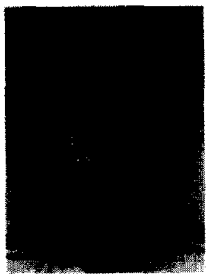
우 종 원

e-mail : sunsuso@engine.chungbuk.ac.kr

1997년 충북대학교 정보통신공학과 졸업(학사)

1999년 충북대학교 정보통신공학과 졸업(공학석사)

관심분야 : 데이터베이스시스템, 암호학



윤 승 현

e-mail : yunsh@pretty.chungbuk.ac.kr

1998년 충북대학교 정보통신공학과 졸업(학사)

1999년 충북대학교 정보통신공학과 (석사과정)

관심분야 : 멀티미디어 데이터베이스시스템, 분산객체컴퓨터

유 재 수

e-mail : yjs@cbucc.chungbuk.ac.kr

1989년 전북대학교 컴퓨터공학과 졸업(학사)

1991년 한국과학기술원 전산학과 졸업(공학석사)

1995년 한국과학기술원 전산학과 졸업(공학박사)

1995년 3월~1996년 8월 목포대학교 전산통계학과 전임강사

1996년 9월~현재 충북대학교 전기전자공학부 조교수

관심분야 : 데이터베이스시스템, 멀티미디어 데이터베이스시스템, 정보검색, 분산객체컴퓨터