ETLARS－Ⅱ

한글정보검색시스팀의 개발에 관한 연구

A Study on the Development of ETLARS－Ⅱ Hangeul Information Retrieval System

박계숙, 이용준*

초 록

ETLARS-Ⅱ 시스팀은 기개발되어 운영해온 ETLARS 시스팀의 기능을 대폭 확장, 개선하여 개발한 시스팀으로, 다수의 Database를 통합 운영할 수 있음은 물론 1Byte 영문정보와 2Byte 한글정보를 모두 처리할 수 있는 한글정보검색 시스팀이다.

본 고에서는 ETLARS-Ⅱ 한글정보검색시스팀의 온라인 검색프로그램의 개발에 관한 내용을 기술하였다.

ABSTRACT

ETLARS-Ⅱ system is a Hangeul information retrieval system, which can operate multiple bibliographic databases and process 2 byte Hangeul data as well as 1 byte English data.

This paper describes the design of database structure and the implementation of online retrieval program.

* 한국전자통신연구소 정보관리실
접수일자 1988. 4. 25

Ⅰ. 시스템 개요

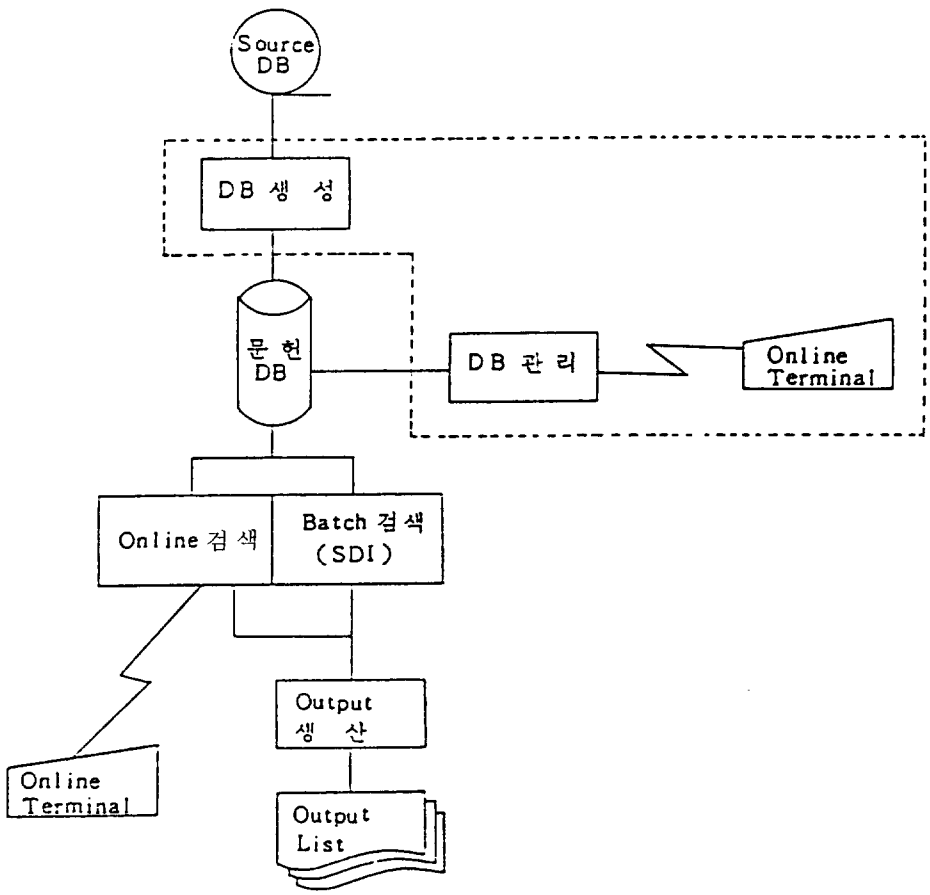
ETLARS-Ⅱ 시스템은 기개발되어 운영해 온 ETLARS 시스템의 기능을 대폭 확장 개선하여 개발한 시스템으로, 다수의 Database를 통합 운영할 수 있음은 물론 1 Byte 영문정보와 2 Byte 한글정보를 모두 처리할 수 있는 한글정보검색 시스템이다.

ETLARS-Ⅱ 시스템은 당 연구소가 보유하고 있는 IBM 4311-M12 컴퓨터를 사용하여 개발하였으며, 개발도구로써는 〈표1〉과 같이 기존의 파일시스템(예, ISAM, VSAM)보다 DBMS가 유리하다는 판단에 따라 IBM/VS DBMS를 이용하였다. 또한 정보검색프로그램은 알고리듬이 매우 복잡하고 스트링 처리 등 다양한 기능을 필요로 하므로 프로그램의 개발은 COBOL이 아닌 PL/1 언어를 사용하였으며 구조화 프로그래밍(Structured Programming)기법을 이용하였다.

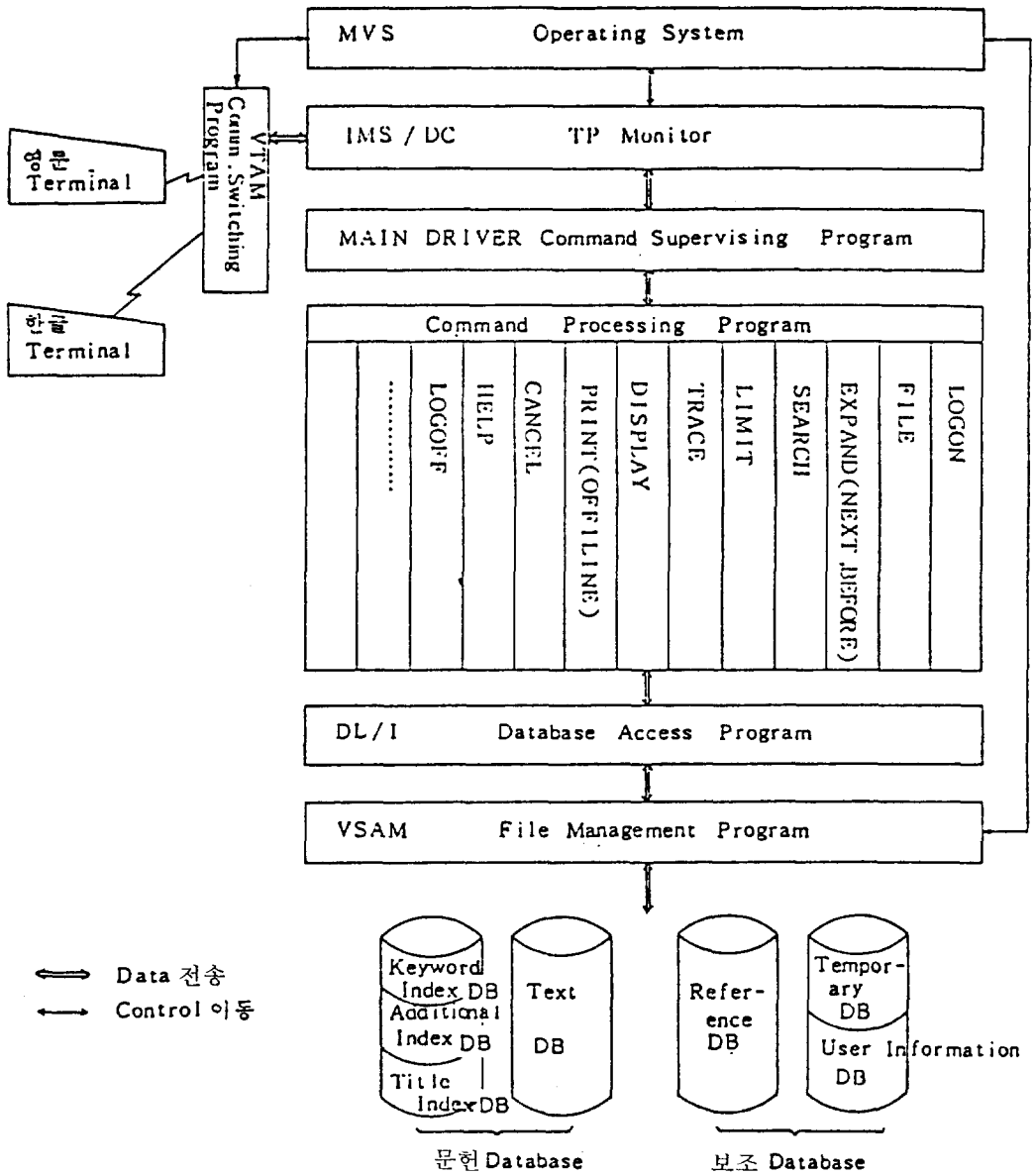
〈표1〉 **File 시스템과 DBMS의 비교**

	장 점	단 점
File 시스템 (예 : VASM)	1. DBMS에 관한 교육과 이용 절차가 필오없다.	1. 여러개의 독립된 File들을 별도로 유지해야 한다. 2. File 구조가 변경되면 응용프로그램을 대폭 수정해야 한다. 3. 온라인 대화식 처리를 위해서 별도의 TP-Monitor (예 : CITS)가 설치되어야 한다.
DBMS (예 : IMS/VS)	1. File간의 관계를 정의함으로서 여러개의 File을 1개의 물리적인 Database로 통합할 수 있다. 2. Data의 독립성에 의해서 기존의 Database 구조에 영향을 주지 않고 Database의 성능향상이 가능하다. 3. 온라인 대화식 처리를 위해서 DBMS자체가 갖고 있는 TP-Monitor 기능(예 : IMS/VS DC)을 그대로 이용할수 있다.	1. 상용 DBMS는 주로 구조적인 정보를 처리하기 위해 설계되었으므로 정보검색시스템의 도구로서는 부적합하다. 2. DBMS는 Recovery, Concurrency같은 기능을 복잡하게 수행해야 하므로, 온라인 처리시에 응답시간(Response Time)이 늦어질 수 있다.

ETLARS-Ⅱ 시스템은 (그림 1)에서 보는 바와 같이 Database의 생성 및 관리를 위한 프로그램과 온라인 검색프로그램으로 구성된다. 온라인 검색은 Command-driven 방식에 의하여 수행되며 배치 검색은 BTS(Batch Terminal Simulator)라는 온라인 Test Tool을 이용하여 기존 온라인 검색프로그램을 Simulation함으로써 처리된다. (그림 2)는 온라인 검색프로그램의 개발환경 및 세부구조를 보여주는 것으로, 본 고에서는 ETLARS-Ⅱ 시스템의 온라인 검색프로그램의 개발에 관한 내용을 기술하였다.



(그림 1) ETLARS-Ⅱ 시스템의 구성도



(그림 2) ETLARS-Ⅱ 온라인 검색프로그램의 세부구조

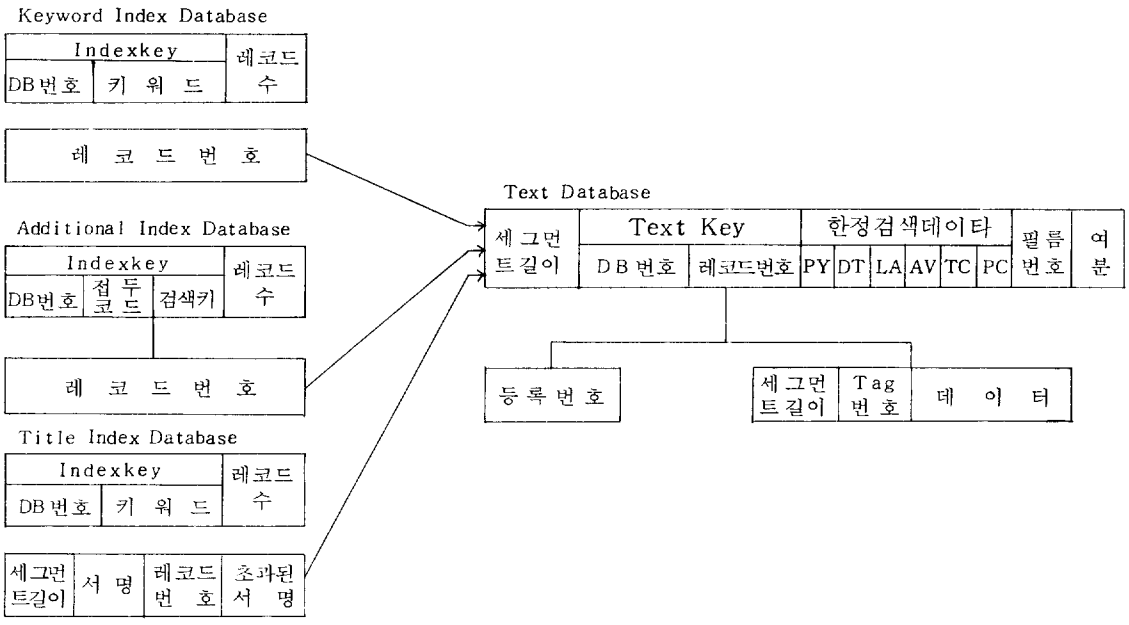
Ⅱ. Database 구조

1. 개 요

현존하는 대부분의 문헌정보검색시스템의 File 구조는 실제로 문헌정보를 저장하는 문헌 File과 한개 이상의 Inverted Index File로 구성된다. Inverted Index File은 색인어들(예: 키워드, 저자명, 분류코드 등)과 각 색인어에 관련된 문헌 레코드 번호들의 집합으로 구성되며 문헌 레코드 번호는 문헌 File에 접근하기 위한 고유 Key로서 사용된다.

Inverted Index File 방식은 문헌정보 검색에 주로 이용되는 Boolean Operator (AND, OR, NOT)를 처리하기 위한 File 구조로서 매우 적합하다.
즉 다량의 문헌을 검색할때 문헌 File 자체를 접근하는 것이 아니라, 먼저 Inverted Index File에 접근해서 Index에 저장된 문헌 레코드번호 만을 가지고 Boolean Operator를 수행한 후, 그 결과로 얻어진 문헌 레코드 번호로 문헌 File로 부터 요구된 문헌을 검색함으로서, 보다 빠른 시간내에 검색을 수행할 수 있다. 이러한 이유로 인해 ETLARS-Ⅱ 시스템은 Inverted Index File 방식을 기본 구조로 채택하였으며, IMS/VS의 물리적 계층 Database로 구현하였다.

ETLARS-Ⅱ 시스템의 Database 종류로는 (그림 2)에서 본 바와 같이 문헌 Database와 보조 Database가 있다. 문헌 Database는 문헌저장을 위한 TEXT Database, Inverted Index를 위한 KEYWORD INDEX Database, ADDITINAL INDEX Database, TITLE INDEX Database의 4개 Database로 구성된다. (그림 3)은 문



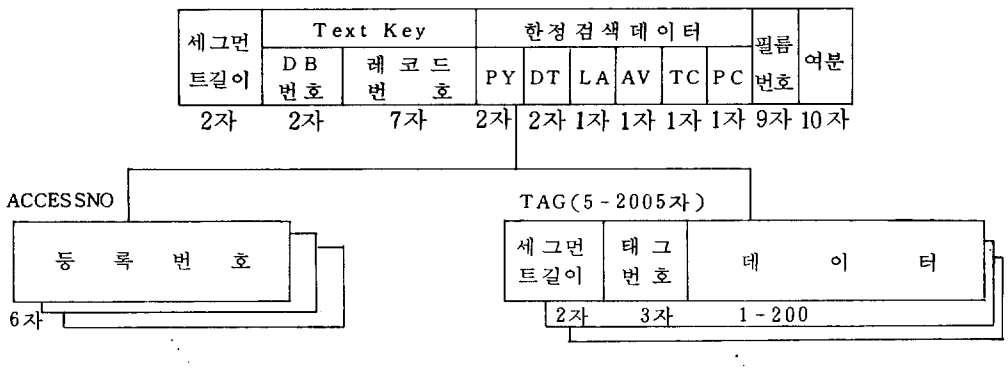
(그림 3) 문헌 Database의 구성도

헌 Database를 구성하는 각 Database간의 상호연관 관계를 보여주는 것으로, 이용자가 검색 명령어를 통해 키워드나 저자명 등의 검색 Key를 입력하면 시스팀은 우선 해당 검색 Key를 INDEX Database에서 찾아 1차적으로 관련 문헌집합(문헌 레코드번호의 집합)을 형성한 후 이용자가 문헌의 실제내용(서지사항 및 초록)을 출력하도록 요구하면 TEXT Database를 찾아가 해당문헌 레코드들을 출력시키게 된다. 시스팀은 입력된 검색 Key의 종류에 따라, 즉 입력된 검색 Key에 어떤 접두코드(Prefix Code)가 사용되었는가에 따라 접근대상 INDEX Database를 결정하게 된다. 즉 검색 Key에 접두코드가 없는 경우는 키워드로 간주하여 KEYWORD INDEX Database를 접근하며, 접두코드가 AU=, CC=, PN=, DN= 인 경우는 Additional INDEX Database를, 접두코드가 TI= 인경우는 TITLE INDEX Database를 접근하게 된다. 보조 Database는 문헌 Database의 이용을 돕거나 검색처리를 위한 Database로, 코드화된 검색 Key(분류코드)의 이용을 돕기 위한 REFERENCE Database, 검색과정에서 생겨나는 중간결과들을 저장하기 위한 TEMPORARY Database, 그리고 이용자에 관한 정보를 저장하기 위한 USER INFORMATION Database가있다.

2. 문헌 Database

(1) TEXT Database

TEXT Database는 실제 문헌이 저장되어 있는 Database로 Database 구조는 (그림 4)와 같다.



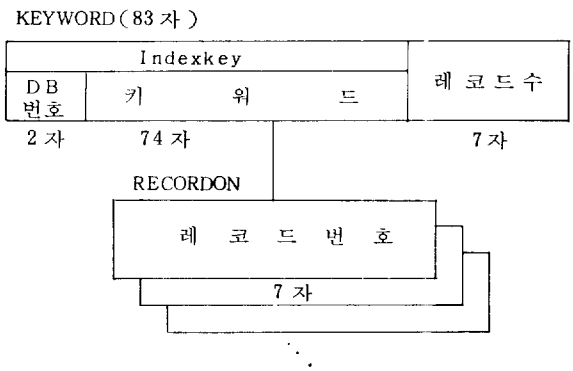
(그림 4) TEXT Database의 구조

LEADER 세그먼트는 가변장(Variable Length) 세그먼트로 세그먼트의 첫 2 Byte에 세그먼트의 길이가 저장된다. TEXT KEY는 LEADER 세그먼트의 Key 필드로서 문헌 Database의 종류를 나타내는 Database 번호와 문헌 Database의 레코드 번호로 구성된다. 한정검색 데이터 요소는 Database 마다 상이하므로, Database 번호에 의해서 한정검색 데이터의 길이가 결정된다. ACCESS NO 세그먼트에는 BOOKS Database와 DOCUMENT Database의 문헌등록번호가 수록되며 TAG 세그먼트는 문헌 레코드의 가변장 필드 데이터를 저장하기 위해서 사용된다.

Database 처리 방식으로는 HDAM(Hierachcal Direct Access Method)을 채택하였으며 따라서 Hashing 루틴을 이용하여 해당 문헌 레코드를 직접 찾아낼 수 있게 하였다.

(2) KEYWORD INDEX Database

TEXT Database의 키워드 필드로 부터 추출된 키워드들이 순차적으로 저장되어 있는 Database로, (그림 5)와 같이 키워드가 수록되는 Root 세그먼트와 각 키워드와 관련된 TEXT Database의 문헌 레코드 번호들이 수록되는 Child 세그먼트로 구성된다.

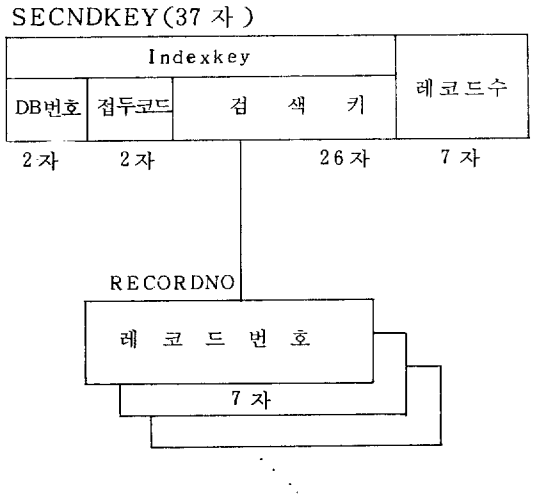


(그림 5) **KEYWORD INDEX Database의 구조**

Database 처리 방식으로는 HISAM(Hierarchical Indexed Sequential Access Method)을 채택하였다.

(3) ADDITIONAL INDEX Database

TEXT Database로 부터 추출된 키워드와 서명을 제외한 모든 검색 Key들이 저장되어 있는 Database로 (그림6)과 같이 검색 Key가 수록되는 Root 세그먼트와 각 검색 Key와 관련된 TEXT Database의 문헌 레코드 번호들이 수록되는 Child 세그먼트로 구성된다.



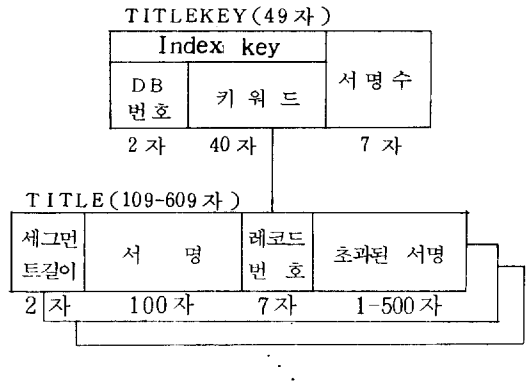
(그림6) ADDITIONAL INDEX Database의 구조

INDEX KEY내의 접두코드는 검색 Key의 종류를 구분하기 위한 것으로, 각 검색 Key는 접두코드별로, 즉 검색 Key의 종류별로 모이게 된다.

Database 처리 방식으로는 HISAM을 채택하였다.

(4) TITLE INDEX Database

TITLE INDEX Database는 TEXT Database로 부터 추출된 서명이 수록되어 있는 Database로, 서명에 대한 접근점(Access point)을 증가시키기 위해 (그림7)과 같이 KWOC(Keyword-Out-Of-Context) Index로 구성 하였다.



(그림7) TITLE INDEX Database의 구조

Root 세그먼트 서명내 키워드와 동일한 키워드를 갖는 서명갯수가 수록되며 Child 세그먼트에는 문헌의 서명 전체와 각 서명과 관련된 TEXT Database의 문헌 레코드 번호가 수록 된다. 따라서 이용자는 서명검색을 하기 위해 서명전체를 모르더라도 서명내 임의의 키워드를 선정하여 입력함으로서 원하는 문헌의 서명을 찾을 수 있다. Database 처리 방식으로는 HIDAM을 채택하였다.

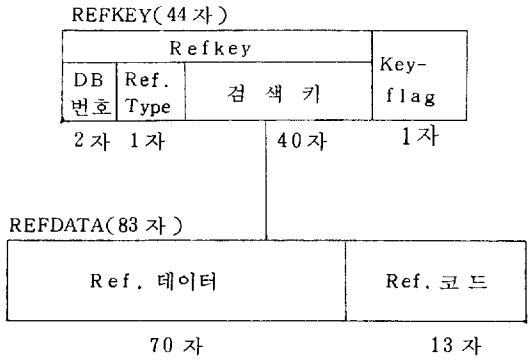
3. 보조 Database

(1) REFERENCE Database

REFERENCE Database는 TEXT Database에 대한 Index Database가 아니라 이용자들이 분류코드와 같은 코드화된 검색 Key를 확인할 수 있도록 마련해 놓은 단순한 참조 Database이다.

REFERENCE Database의 구조는 (그림8)과 같이 검색 Key가 수록되어 있는 Root 세그먼트와 실제 이용자에게 출력시키게 될 참조내용인 분류코드와 주제어가 수록되어 있는

Child 세그먼트로 구성된다.

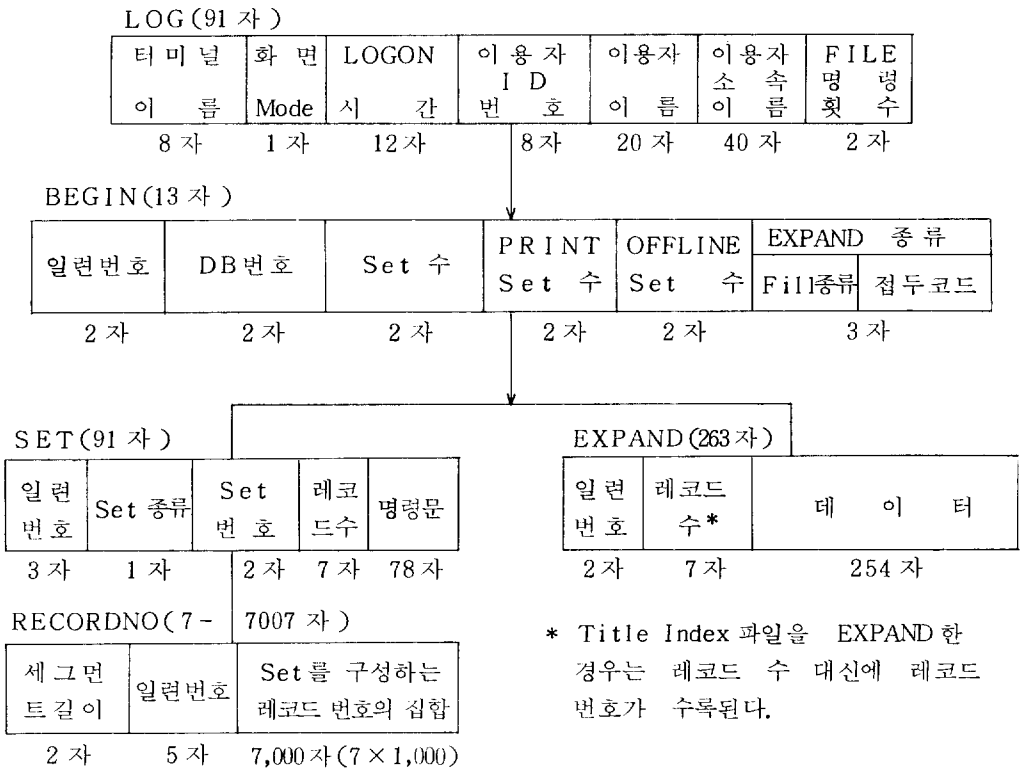


(그림8) **RERFERENCE Database의 구조**

Database 처리 방식으로는 HISAM을 채택하였다.

(2) TEMPORARY Database

검색도중에 만들어지는 모든 정보(이용자에 관한 정보, Set, 검색결과정보)를 저장하는 Database로 LOGON 할때 터미널 마다 1개씩 생기며 LOGOFF시에는 TEMPORARY Database내의 모든 정보를 USER INFORMATION Database로 옮긴후 삭제된다. TEMPORARY Database의 구조는 (그림9)와 같다.



(그림9) **TEMPORARY Database의 구조**

또한 TEMPORARY Database의 특정 레코드에는 HELP 명령을 위한 Tutorial이 (그림 10)과 같은 형태로 저장된다. Database 처리 방식으로는 HDAM을 채택하였다.

LOG

* HHELP	

* 영문 터미널이면 '*AHELP' * 한글 터미널이면 '*HHELP'를 저장

BEGIN

일련번호	

* 일련번호 1개마다 HELP OPTION MENU의 하나의 번호에 해당되는 Tutorial을 저장

SET

일련번호	

* 일련번호 1개마다 터미널 화면의 1 Line에 해당되는 Tutorial을 저장

(그림10) HELP Tutorial의 저장 형태

(3) USER INFORMATION Database

검색 시스템 이용자들에 관한 정보를 저장하는 일종의 History Database로, 등록된 이용자 ID 번호 마다 1레코드씩 만들어지며 이용자가 검색 시스템을 이용할 때 마다 그 정보가 계속적으로 누적된다.

USER INFORMATION Database의 구조는 (그림 11)과 같으며, 누적된 정보는 시스템 이용 상황을 분석하거나 시스템 이용료를 계산하는데 사용된다.

Database 처리 방식으로는 HDAM을 채택하였다.

USER(113자)

이용자 ID번호	DB Map *	이용횟수
8자	99자	6자

* 이용자의 Database 사용가능 여부를 Bit map으로 표시함.

LOG(93자)

일련번호	화면 Mode	LOG ON 시간	LOG OFF 시간	이용자 이름	이용자 소속부서	FILE 명령 횟수
6자	1자	12자	12자	20자	40자	2자

BEGIN(10자)

일련번호	DB번호	Set 수	PRINT Set 수	OFFLINE Set 수
2자	2자	2자	2자	2자

SET(91자)

일련번호	Set 종류	Set 번호	레코드수	명령문
3자	1자	2자	7자	78자

RECORDNO(705자)

일련번호	Set를 구성하는 레코드 번호 집합
5자	700자(7 × 100)

(그림 11) USER INFORMATION Database의 구조

Ⅲ. 온라인 검색 프로그램의 개발

1. 검색명령어 설계

(1) 명령어의 종류 및 기능

ETLARS-Ⅱ 시스템에서는 효율적인 검색을 위해 COMMAND-driven 방식을 채택하였다. COMMAND-driven 방식은 검색명령어를 이용하여 문헌검색을 실시하는 방식으로, 이용자가 명령어의 사용법을 익혀야만 하는 단점이 있으나 대규모 문헌 Database를 검색하는데 있어서 그 간략함과 융통성 때문에 많이 이용되고 있다.

ETLARS-Ⅱ 시스템에서 사용하게 될 검색명령어의 종류 및 기능은 (표2)와 같다.

〈표2〉 검색명령어의 종류 및 기능

명 령 어	기 능
HELP (또는 H)	· 시스템 이용에 도움을 필요로 할때 사용
FILE (또는 F)	· 검색대상 파일을 선정하는데 사용
EXPAND (또는 E)	· 검색하고자 하는 단어가 시스템에서 이용 가능한지를 확인하는데 사용
SEARCH (또는 S)	· 검색하고자 하는 주제에 관한 문헌이 검색 파일내에 몇건이나 존재하는지를 알아보는 데 사용
LIMIT (또는 L)	· SEARCH명령결과 검색된 문헌수를 제한조건을 적용하여 한정시키는데 사용
TRACE (또는 T)	· 검색도중 지금까지 수행한 검색과정 추적하는데 사용
DISPLAY (또는 D)	· 검색결과 문헌이 실질적인 내용(서지사항 및 초록)을 CRT화면에 디스플레이 하는데 사용
PRINT (또는 P)	· 검색결과 문헌 리스트를 온라인 프린트하는데 사용
OFFLINE (또는 O)	· 검색결과 문헌 리스트를 오프라인 프린트하는데 사용
LOGOFF	· 시스템 이용을 종료할때 사용

(2) 명령어의 Syntax

검색명령어의 Syntax를 BNF(Backus Naur Form)로 표현하면 아래와 같다.〈 〉는 Non Terminal을 나타낸다.

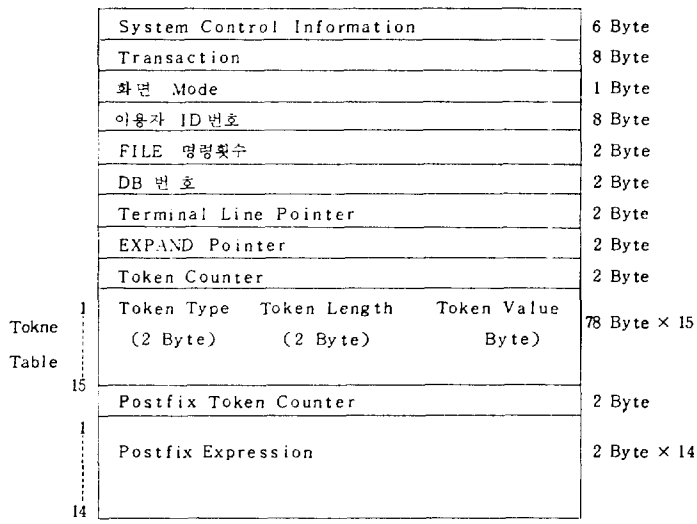
```
〈Command〉              : : = 〈operation 1〉 | 〈operation 2〉 | 〈expression〉
〈Operation 1〉          : : = 〈FILE  operation〉 | 〈NEXT operation〉 | 〈BEFORE operation〉
                              〈TRACE operation〉 | 〈HELP operation〉 | 〈LOGOFF operation〉
〈FILE operation〉       : : = FILE | F
〈NEXT operation〉       : : = NEXT | N
〈BEFORE operation〉     : : = BEFORE | B
〈TRACE operation〉      : : = TRACE | T
〈HELP operation〉       : : = HELP | H
〈LOGOFF operation〉     : : = LOGOFF
〈operand 2〉            : : = 〈EXPAND operation〉 | 〈SEARCH operation〉 | 〈LIMIT operation〉
                              〈DISPLAY operation〉 | 〈PRINT operation〉 | 〈OFFLINE operation〉
                              〈CANCEL operation〉 | 〈MICROFILM operation〉
〈EXPAND operation〉     : : = EXPAND | E
〈SEARCH operation〉     : : = SEARCH | S
〈LIMIT operation〉      : : = LIMIT | L
〈DISPLAY operation〉    : : = DISPLAY | D
〈PRINT operation〉      : : = PRINT | P
〈OFFLINE operation〉    : : = OFFLINE | O
〈CANCEL operation〉     : : = CANCEL | C
〈MICROFILM operation〉  : : = MICROFILM | M
〈Expression〉           : : = 〈expression〉 | 〈Binary operator〉 〈expression〉 |
                              (〈Expression〉) | 〈operand〉
〈BINARY operator〉      : : = 〈AND operator〉 | 〈OR operator〉 | 〈NOT operator〉 | :
〈AND operator〉         : : = AND | *
〈OR operator〉          : : = OR | +
〈NOT operator〉         : : = NOT | #
〈operand〉              : : = 〈keyword〉 | 〈Additional key〉 | 〈Title key〉 | 〈reference key〉
                              〈LIMIT〉 | 〈SET〉 | 〈E〉
〈keyword〉              : : = id
〈Additional key〉       : : = AU=id | CC=id | PN=id | DN=id
〈Title key〉            : : = TI=id
〈Reference key〉        : : = SC=id | SN=id
〈LIMIT〉                : : = PY=id | DT=id | LA=id | AV=id | TC=id | PC=id
〈SET〉                  : : = S digit | E digit | T digit | P digit | O digit
〈E〉                    : : = E digit
```

2. 검색명령어 Processor의 구현

(1) 개 요

검색명령어 Processor는 IMS/DC의 Conversation & Program to Program Communication 기능을 이용하여 이용자가 대화식 검색을 할 수 있도록 개발하였다. 명령어 Processor는 (그림 2)에서 본바와 같이 1개의 Main Driver 프로그램과 12개의 독립된 명령어처리 프로그램으로 구성된다. 이들 13개의 프로그램은 모두 IMS/DC의 통제를 받는 Conversational MPP로서 SPA(Scratch-Pad Area)를 공동으로 이용하여 SPA와 입력 메시지를 상호전달한다.

SPA는 대화식 처리를 하기 위해 IMS/DC 시스템에서 할당하는 영역으로 주로 대화중에 처리된 중간결과를 저장한다. 입력메시지는 이용자가 터미널로 부터 직접 입력시킨 데이터와 Main Driver로 부터 전달된 메시지가 있다. SPA의 구조는 그림(12)와 같다.



1. System Control Information: IMS/VS의 Control Area
2. Transaction Code : Conversation Program의 Transaction Code
3. 화면 Mode : 영문화면이면 'A', 한글화면이면 'H'
4. 이용자 ID 번호 : 이용자의 계정번호 또는 Password
5. FILE 명령횟수 : FILE 명령을 사용해서 선택한 DB 번호
6. DB 번호 : FILE 명령을 사용해서 선택한 DB 번호
7. Terminal Line Pointer : 메시지를 출력할 터미널의 Line 위치
8. EXPAND Pointer : EXPAND 명령에 의해 생성된 Index Entry중 NEXT나 BEFORE 명령에 의해 출력될 Index Entry의 위치

(그림 12) SPA의 구 조

이용자가 터미널에서 입력시킨 모든 명령문은 일단 MAIN DRIVER 프로그램에서 그 구문(Syntax)이 분석된다. 이 과정에서 명령문이 구문상 에러이면 에러메시지가 터미널로 전달되며 구문 분석에 이상이 없으면 생성된 모든 Token은 SPA내의 Token Table에 저장된다. 구문 분석이 끝나면 SPA와 입력메시지는 프로그램 스위칭에 의해 해당되는 명령어 처리 프로그램으로 전달된다.

프로그램 스위칭에는 직접적으로 MAIN DRIVER의 메시지가 전달되어 처리되는 Immediate 스위칭과 이용자가 입력한 메시지와 SPA가 전달되어 간접적으로 처리되는 Differed 스위칭이 있다.

이와 같이 각 명령어 처리 프로그램은 MAIN DRIVER 프로그램의 통제를 받아 동작하지만 각기 독자적인 기능을 분담하여 별도의 IMS/VS 영역에서 수행되므로 많은 이용자들이 동시에 여러 터미널을 통해 검색을 하더라도 지연 시간을 줄여서 검색의 응답시간을 향상시킬 수 있다.

(2) 화면설계

ETLARS-Ⅱ 시스템은 영문데이터와 한글데이터를 모두 검색해야 하므로 검색명령문을 입력시키거나 그 결과를 볼 수 있는 터미널의 화면도 영문화면과 한글화면이 모두 제공되어야 한다. ETLARS-Ⅱ 시스템의 화면은 Initial 화면, LOGON화면, Command 화면, File Option Menu화면, Display화면, LOGOFF화면, HELP Option Menu 화면의 7개 화면으로 구성되며 이중 LOGON 화면, Command화면, Display 화면의 3화면은 영문화면과 한글화면의 2가지 형태가 있고 나머지 화면은 영문화면만 존재한다.

온라인 처리를 위해서는 터미널 화면의 I/O 형태를 설계하는 것이 중요하며 ETLARS-Ⅱ 시스템은 IBM에서 제공하는 화면 설계용 Utility인 MFS(Message Format Service)를 사용하였다. MFS가 제공하는 기능으로는 I/O 화면의 형태정의, 터미널의 Attribute(Cursor의 위치지정, 문자의 명암조절, Alarm, Protection) 제어, 2 Byte 한글데이터의 I/O등이 있다.

영문, 한글 화면의 처리는 MFS의 Multiple Logical Page라는 기능을 이용하였다. 이 기능은 화면 설계시에 서로 다른 형태의 여러 화면을 설계하여 단일 명칭의 구룹으로 저장하였다가 필요에 따라 그 중에서 한개의 화면을 임의로 선택할 수 있는 기능이다.
이때 화면선택은 프로그램에서 특정 화면을 지정함으로서 이루어진다. 이 기능을 이용하여 영문화면과 한글화면을 프로그램에서 필요에 따라 선택하여 적절히 사용함으로서 영문데이터베이스와 한글데이터베이스의 검색을 단일 프로그램에서 선택할 수 있도록 하였다.

또한 ETLARS-Ⅱ 시스템은 Menu 방식이 아닌 Command 방식이므로 Menu 방식과 달리 터미널 화면상의 I/O 위치가 일정하지 않을 뿐만 아니라 IBM의 SNA(System Network Architecture)통신 규격의 터미널 제어방식은 Screen Control 방식이므로 ETLARS-Ⅱ 시스템의 화면 제어를 위해 MFS의 기능 중 프로그램내에서 터미널을 제어하는 기능을 이용하여 화면의 I/O를 제어하였다. 즉 SPA내의 Terminal Line Pointer에 저장되어 있는 Cursor의 현재 위치와 출력 메시지의 Line 수를 가지고 화면의 I/O 위치를 제어하였다. 예를 들어 터미널 상에 출력가능한

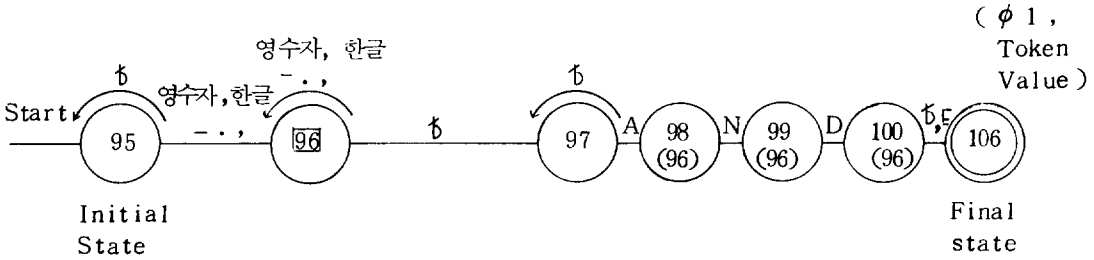
Line 수가 20이라 하고 현재의 Cursor 위치가 5, 출력메시지의 Line수가 4라고 하면 프로그램에서는 5,6,7,8 Line에 출력메시지, 9 Line에 OK./ 메시지를 내보내고 10 Line에 Cursor를 위치시키며 SPA의 Terminal Line Pointer를 10으로 변경시킨다. 또한, 만일 현재의 Cursor 위치와 출력메시지의 Line 수를 더한 값이 20 Line이 넘으면 새로운 화면에 메시지를 출력시킨다.

(3) MAIN DRIVER 프로그램

구문분석(Syntax Analysis)을 위해서는 검색 명령문에서 Token들을 분리해내야 한다. 이를 위해 각 Token들의 Format을 (표3)과 같이 설계하였다.

Token Name	Token Type	Token Length	Token Value
FILE(F)			ϕ1
EXPAND(E)			ϕ2
NEXT(N)			ϕ3
BEFORE(B)			ϕ4
SEARCH(S)			ϕ5
LIMIT(L)			ϕ6
TRACE(T)			ϕ7
DISPLAY(D)			ϕ8
PRINT(P)			ϕ9
OFFLINE(O)			1ϕ
CANCEL(C)			11
HELP(H)			12
LOGOFF			13
MICROFILM(M)			14
keyword	≠1		
AU=	ϕ2		
CC=	ϕ3		
PN=	ϕ4		
DN=	ϕ5		
TI=	ϕ6		
SC=	ϕ7		
SN=	ϕ8		
PY=	21		
DT=	22		
LA=	23		
AV=	24		
TC=	25		
PC=	26		
set. no	41		
E. no	42		
T. no	43		
P. no	44		
O. no	45		
AND (*)	51		
OR(-)	52		
NOT(#)	53		
:	54		
(	55		
)	56		
/	57		

Token을 인식하기 위하여 Finite Automata를 설계하였으며, Finite Automata를 위한 Transition Table은 행이 State이고 열이 입력문자인 2차원 Array로 구현하였다. Finite Automata의 예는 (그림 13)과 같다.



(그림 13) Finite Automata 예

Finite Automata에 의해 인식된 Token들은 Bit Map을 이용하여 유효한지 아닌지를 조사한후, SPA내의 Token Table에 저장된다. Bit Map은 Database에 따라 유효한 Token을 검사하기 위한 Bit Map과 검색명령어에 따라 유효한 Token을 검사하기 위한 Bit Map의 두가지 종류가 사용되었다.

(그림 14)는 Database별 Token Bit Map으로, "1"은 사용가능, "0"은 사용불가능을 표시한다.

Operand / DB	key word	AU=	CC=	PN= DN=	Ti=	SC=	SN=	PY=	TC=	DT=	LA=	AV=	PC=
TELECOM	1	1	1	φ	φ	1	φ	1	1	1	1	1	φ
Books	1	1	1	φ	1	1	φ	1	φ	1	1	φ	φ
Doucument	1	1	1	1	1	1	1	1	φ	1	φ	φ	1
NTIS	1	1	φ	φ	φ	φ	φ	1	φ	φ	φ	φ	φ
Patent	1	1	1	φ	φ	1	φ	1	φ	φ	φ	φ	φ
Journal List	φ	φ	φ	φ	1	φ	φ	φ	φ	φ	φ	φ	φ

(그림 14) Database별 Token Bit Map

나. Syntax Analysis

Lexical Analysis에서 얻어진 Token을 가지고 구문분석을 하기 위하여 기본적인 Bottom-up Parsing 기법인 Operator Precedence Parsing을 사용하였다. 또한 Parsing을 하면서 SEARCH 명령문의 Infix Expression을 Postfix Expression으로 변환하여 후에 SEARCH 프로그램에서 편리하게

사용할 수 있도록 하였다. 예를 들면 Operand 1 * Operand 2 + Operator 3의 Infix Expression은 Operand 3 Operand 1 Operand * +의 Postfix Expression으로 변환된다. 변환된 Postfix Expression은 SPA에 저장되며 이때 저장되는 내용은 Token Table 내의 Token 위치이다.

Operator의 처리 우선 순위는 : # * +의 순서이다.

Operator Precedence Parsing을 위해서 Operator Precedence Relation Table을 만들었으며 Parsing Algorithm은 PASCAL과 유사한 Pseudo Code로 정의하였다.

(4) SEARCH 프로그램

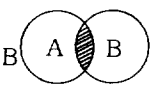
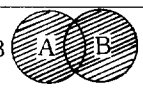
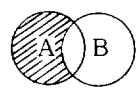
SEARCH 프로그램은 실질적으로 검색을 수행하는 프로그램에서 이용자가 입력시킨 SEARCH 명령문을 수행하여 Set 정보를 생성한후 이를 TEMPORARY Database에 저장한다.

가. 검색 Operator

SEARCH 명령문에서는 Boolean Operator (AND, OR, NOT)를 이용하여 둘 이상의 주제개념을 자유로이 조합할 수 있으며 그 외에도 여러가지 편리한 검색 Operator가 마련되어 있다.

검색 Operator의 종류 및 기능은 (표 4)와 같다.

〈표 4〉 검색 Operator의 종류 및 기능

기 능	설 명
A N D (또는 *)	· 주제개념 A와 주제개념 B가 모두 존재하는 문헌을 검색한다. A * B 
O R (또는 +)	· 주제개념 A나 주제개념 B가 하나라도 존재하는 문헌이면 검색한다. A + B 
N O T (또는 #)	· 주제개념 A를 포함하는 문헌중 주제개념 B를 포함하는 문헌은 제외한다. A # B  * NOT Operator는 본의 아니게 관련 문헌을 누락시킬 우려가 있으므로 주의하여 사용해야 한다.
:	· 범위검색 (Range Search) 부호로, E Ref, NO, T, Ref, NO, Set NO., 출판년도를 여러개의 OR로 조합하는 대신에 사용한다. 최대 50개 요소를 조합할 수 있다. 예) S E1:E3 S S1:S2 L S4 AND PY=80:PY=84 D T1:T30/2 P T1:T50/3
?	· 절단 (Truncation) 부호로, ?앞의 단어 어간과 일치하는 모든 단어 (최대 50개)를 검색하여 OR로 조합한다. 완전한 단어형태를 모를때 사용하면 효과적이다. * SEARCH 명령에서만 사용할 수 있다.

나. 중간 Code의 생성

SEARCH 프로그램은 MAIN DRIVER 프로그램에서 전달된 SPA내의 Postfix Expression을 연산하기 위하여 중간 Code를 생성시킨다. 예를 들면 Operand 3 Operand 1 Operand 2 * +의 Postfix Expression은 T1= Operand 1 * Operand 2, T2 = T1 + Operand 3로 평가될 수 있다. 중간 Code의 표현을 위한 중간 Code Table 구조는 (그림 15)와 같은 2차원 Array(Quadruple) 이다.

	Operator (1 Byte)	Operand 1		Operand 2		Result	
		Type (1Byte)	Value (3Byte)	Type (1Byte)	Value (3Byte)	Type (1Byte)	Value (3Byte)
1							
2							
⋮							
500							

※ Operator

종류	표 현	내 용
*	Binary	· AND기능을 가지며 Operand 1과 Operand 2가 있어야 한다. 예) * A B Result
+	Binary	· OR기능을 가지며 Operand 1과 Operand 2가 있어야 한다.
#	Binary	· NOT기능을 가지며 Operand 1과 Operand 2가 있어야 한다.
=	Unary	· 단지 Set를 만들어 주는 기능을 가지며 Operand 1만 있다. 또는 중간결과를 단지 Temporary Area로 옮기는 기능도 한다. 예) S Computer →= Computer Result

※ Operand,Result

종 류	Type	Value
Keyword	K	· Token Table 내의 Token 위치
AU =	K	· 〃
CC =	K	· 〃
PN =	K	· 〃
DN =	K	· 〃
Set No.	S	· Set No.
E Ref.No	E	· E Ref. No.
Temporary No.	T	· Temporary Area의 No.
Truncated Key	?	· Key Table No. (1 Byte)와 Table내의 Key 위치(2 Byte)로 구성

(그림 15) 중간 Code Table 구조

검색 Operator는 한 명령문 내에서 최대 5개까지 사용할 수 있으며 (그림 16)과 같은 별도의 Key Table를 만들어서 처리하였다.

범위검색 (:)과 절단검색 (?)은 모두 OR (+) Operator로 변환하여 처리하였으며, 이때 처리 가능한 요소는 최대 50개이다. 절단

※ 5개의 Table 존재

	Token Table 내의 Truncation Token 위치 (2 Byte)
	Truncation의 최종결과가 저장될 Temporary Area의 No.(2 Byte)
1	키워드 또는 저자명 (74 Byte)
2	
3	
:	
50	

(그림 16) 절단검색 처리를 위한 Key Table 구조

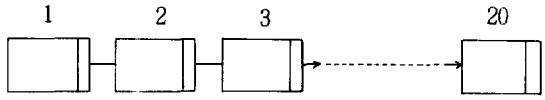
다. 중간 Code의 연산

생성된 중간 Code Table에 따라 순차적으로 중간 Code를 연산한다.

이때 연산된 중간결과를 저장하기 위해서는 별도의 Temporary Area가 필요한데 이를 위하여 Available List와 Temporary Area Table을 설계하였다.

이 2개의 Area는 처리속도를 빠르게 하기 위하여 프로그램내에 할당하였다.

Available List는 연산된 중간결과를 저장하기 위하여 사용되는 Area로 (그림 17) 과 같이 20개의 Node로 구성된 Linked List 구조이며 각 Node는 데이터 부분과 Pointer 부분으로 구성된다. 각 데이터 부분은 1,000개의 레코드 번호를 저장할 수 있는 7 Byte × 1,000=7,000 Byte의 크기이며 Pointer 부분은 다음 Node를 가리킨다. 이와 같이 Linked List 구조를 사용한 이유는 연산도중에 Temporary Area들이 중간결과를 저장하기 위하여 Node를 필요로 할때 즉시 Node를 제공하고 Temporary Area가 더 이상 필요없을 때는 이 Temporary Area에 할당된 Node들을 Available List로 회수하는데 있어서 편리하기 때문이다.



(그림 17) Available List 구조

Temporary Area Table은 연산도중에 생기는 Tempoary Area에 관한 정보를 저장하며 T1부터 T99까지 최대 99개의 Temporary Area를 처리할 수 있다.

Table에 저장되는 정보의 내용은 (그림18)과 같다.

	Tag	마지막 Node의 레코드번호수	Start Node Pointer
1			
2			
:			
99			

(그림 18) Temporary Area Table 구조

1. Tag:Temporary Area에 Node가 할당되었는지 여부를 나타내는 부분으로 Tag가 0이면 할당이 안되었고 1이면 할당되었음을 나타낸다.

2. 마지막 Node의 레코드번호수 :할당된 Temporary Area의 마지막 Node에 저장된 레코드 번호의 갯수를 나타낸다.
3. Start Node Pointer:해당되는 Temporary Area의 시작위치이다.
   즉 Temporary Area는 Available List의 Node들로 구성된 Linked List로서 각 Temporary Area를 구성하고 있는 Node List의 첫번째 Node를 지정한다.

이상의 2개의 데이터 구조를 이용한 연산과정은 다음과 같이 순차적으로 수행한다.

1. 중간 Code Table의 Rusult 부분이 Temporary Area이면 Available List의 Node를 1개 할당하고 Set이면 Temporary 파일에 새로운 Set를 저장할 수 있는 준비를 한다.
2. Operator (AND, OR, NOT)에 따라 해당되는 연산을 하며 연산된 결과를 Temporary Area에 저장한다. 이때 저장을 위한 Node가 더 필요할 때마다 Available List에서 Node를 얻어서 Temporary Area에 계속 연결시켜 준다.
   만일 Available List의 Node가 더 이상 남아있지 않으면 중간 Code Table을 조사하여 필요없는 Temporary Area를 찾아서 Available List로 복귀시켜 재사용할 수 있도록 한다.
3. 1과 2과정을 중간 Code Table이 다 처리될 때까지 반복한다.
4. 최종적으로 새로운 Set를 만들어 Temporary 파일에 저장한다.

(5) 기타 프로그램

가. LOGON 프로그램

이용자가 ETLARS-Ⅱ 시스팀을 이용할 수 있도록 TEMPORARY Database의 LOG 세그먼트를 생성하며, 이용자가 입력한 이용자정보(이용자 ID번호, 이용자이름, 이용자소속부서)를 LOG 세그먼트에 저장한다.

나. FILE 프로그램

이용자가 File Option Menu 화면에서 선택한 Database를 이용할 수 있도록 TEMPORARY Database의 BEGIN 세그먼트를 생성하며, 선택한 Database 번호를 BEGIN 세그먼트에 저장한다.

다. EXPAND 프로그램

EXPAND 명령문에서 지정한 검색 Key에 해당되는 Index Database를 조사하여, 검색 Key와 관련된 Index Term을 순차적으로 50개 찾아내어 TEMPORARY Database의 EXPAND 세그먼트를 생성한 후 처음 10개의 세그먼트를 디스플레이 한다.

NEXT 명령의 경우에는 EXPAND 명령에 의해 생성된 50개의 EXPAND 세그먼트중 디스플레이된 세그먼트의 최종번호를 SPA에서 찾아내어, 최종 세그먼트의 다음 세그먼트로 부터 10개의 세그먼트를 디스플레이 한다.
만일 50개의 세그먼트가 이미 디스플레이 된 경우에는 프로그램내의 EXPAND Procedure를 재수행 한다. BEFORE 명령은 EXPAND 세그먼트중 디스플레이된 세그먼트의 처음 번호를 SPA에서 찾아내어, 처음 세그먼트의 이전 10개 세그먼트를 디스플레이 한다.

라. LIMIT 프로그램

LIMIT명령이 분석된 SPA내의 Token Table을 참조하여 DL/I Call의 SSA(Search Segment Argument)를 생성한다. SSA를 생성할 때 AND, OR Operator를 처리하는 과정을 예를 들면 SI AND (LA = E OR LA = J)의 조건

식을 (SI AND LA=E) OR (SI AND LA=J) 형태로 변환시킨 후 DL/I 문법에 의해 SSA를 만든다. SSA를 생성한 후, TEMPORARY Database의 해당 SET 세그먼트에 속한 레코드 번호들을 1개씩 차례로 SSA에 대입하여 TEXT Database에 대한 DL/I (Get Unique)을 수행한다.

DL/I Call이 정상 수행된 레코드 번호들은 LIMIT명령의 조건을 만족하므로 이 레코드 번호들을 모아서 새로운 SET 세그먼트를 생성한다.

마. TRACE 프로그램

FILE명령을 사용하여 Database를 선택한 후, SEARCH, LIMIT, PRINT, OFFLINE 명령에 의해 생성된 모든 SET 세그먼트를 디스플레이 한다.

바. DISPLAY 프로그램

DISPLAY명령에서 지정한 Set에 해당되는 TEMPORARY Database의 세그먼트에 속한 레코드 번호들중, 명령문에서 지정한 출력범위의 레코드 번호들을 가지고 TEXT Database의 TAG 세그먼트를 읽어서 출력형식에 따라 Edit하여 디스플레이 한다.

사. PRINT OFFLINE 프로그램

PRINT명령 또는 OFFLINE 명령에서 지정한 Set에 해당되는 TEMPORARY Database의 SET 세그먼트에 속한 레코드 번호들중, 명령문에서 지정한 출력범위의 레코드 번호들을 가지고 TEMPORARY Database의 새로운 SET 세그먼트를 생성한다.

아. CANCEL 프로그램

PRINT 명령 또는 OFFLINE 명령에 의해 생성된 TEMPORARY Database의 SET 세그먼트를 삭제한다.

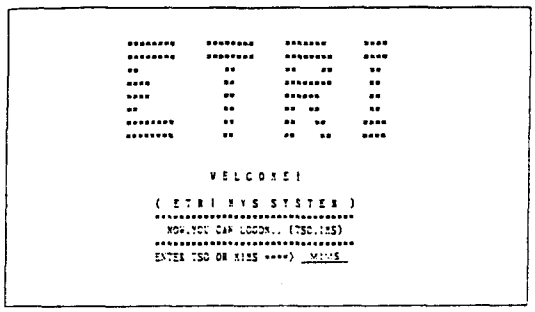
자. HELP 프로그램

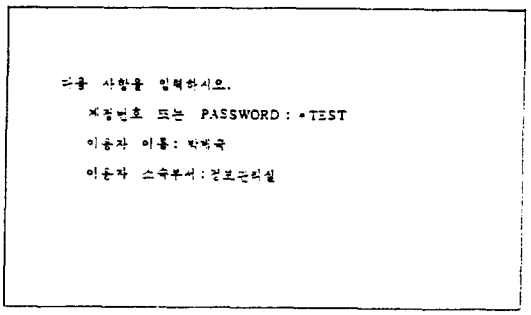
HELP 명령이 들어오면 HELP Option Menu를 디스플레이 한다. 이용자가 Menu에서 알고 싶은 사항을 선택하면 TEMPORARY Database에 저장된 안내정보를 디스플레이 한다.

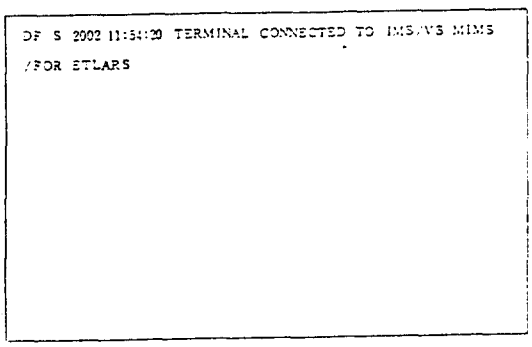
차. LOGOFF 프로그램

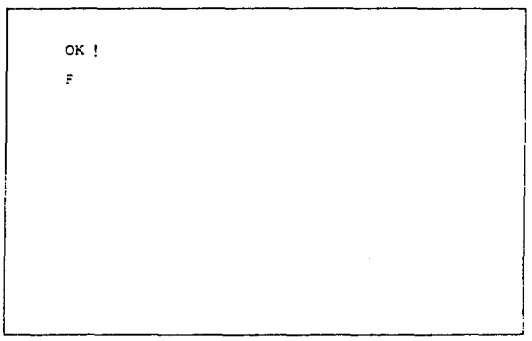
TEMPORARY Database내에 저장된 모든 검색 정보를 USER INFORMATION Database로 옮긴 후, TEMPORARY Database를 삭제한다. 또한 검색도중 PRINT 명령 또는 OFFLINE 명령에 의해 생성된 USER INFORMATION Database의 SET 세그먼트에 속한 레코드 번호들을 가지고 TEXT Database를 읽어서 출력형식에 따라 Edit하여 Local 프린터 또는 Host 프린터로 출력시킨다.

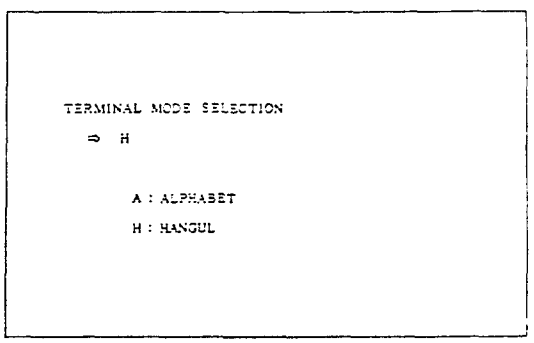
Ⅳ. 온라인 검색과정

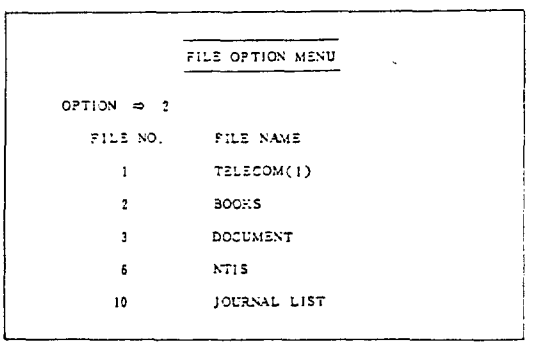

WELCOME!
(ETRI NYS SYSTEM)
NOW,YOU CAN LOGON.. (TSO,IMS)
ENTER TSO OR IMS ****) MIMS


다음 사항을 입력하시오.
*정번호 또는 PASSWORD : *TEST
이용자 이름: 박백국
이용자 소속부서 : 정보관리실


DF S 2002 11:54:20 TERMINAL CONNECTED TO IMS/VS MIMS
/FOR ETLARS


OK !
F


TERMINAL MODE SELECTION
⇨ H
A : ALPHABET
H : HANGUL


FILE OPTION MENU
OPTION ⇨ 2
FILE NO. FILE NAME
1 TELECOM(1)
2 BOOKS
3 DOCUMENT
6 NTIS
10 JOURNAL LIST

```
OK !
E T1=데이터베이스
REF   TITLE
T1    국내 및 해외 한국인 과학 기술 인력 데이터베이스 개발에 관한 연구
T2    데이터베이스 개론
T3    데이터베이스 대상 추량
T4    데이터베이스 드입과 설계
T5    데이터베이스 매니지먼트: 계획 개발 관리의 핀드
T6    데이터베이스 백서
T7    데이터베이스 시스템
T8    데이터베이스 으론
T9    데이터베이스의 고드 이용
T10   데이터베이스의 선택 방법 : 데이터베이스 관리 시스템의 선정과 도입
```

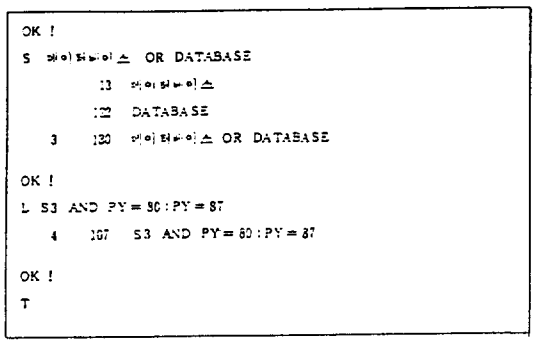
```
OK !
E 데이터
REF  ITEMS     INDEX TERM
E1       41    데이터
E2        1    데이터 분석
E3        3    데이터 통신
E4        1    데이터망
E5        1    데이터뱅크
E6       13    데이터베이스
E7        1    데이터베이스론
E8        1    데이터베이스화
E9        1    데이터록
E10       1    데이터킹
```

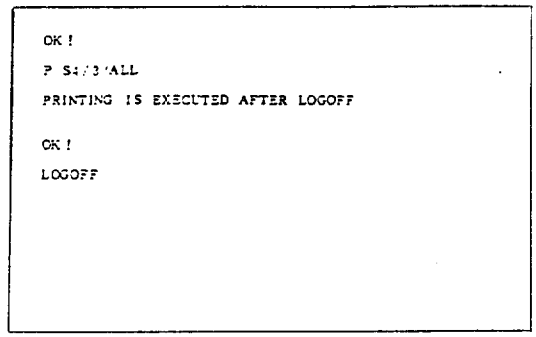
```
OK !
D T5:8
```

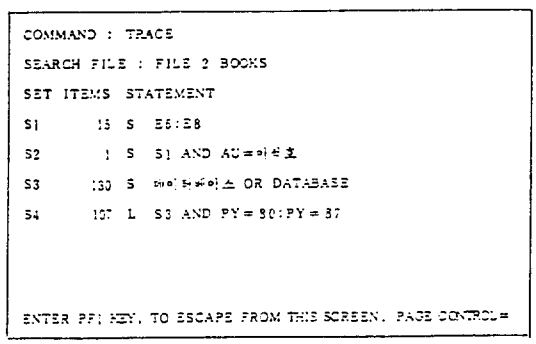
```
OK !
S E6:E8
          15   E6:E8
    1     15   E6:E8
OK !
S S1 AND AU=이석호
          15   S1
           1   AU=이석호
    2      1   S1 AND AU=이석호
OK !
D S2/3
```

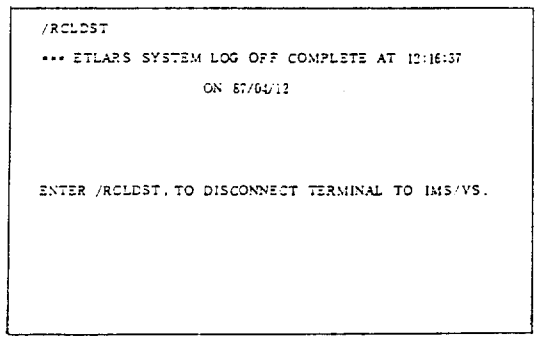
```
COMMAND:DISPLAY
도    제 : 데이터베이스 백서
저    자 : 일본, 통상산업성 ; 데이터베이스진흥센터
서지사항 : 동경, 데이터베이스진흥센터, 1985
          403p.  26㎝
          LA:J   CC:R/QA76.9.D3/일45ㄷ
색 인 어 : DATABASE
내    용 : 일본의 데이터베이스 동향 ; 서책 ; 선진 주요국의 데이터베이스
          동향 ; 데이터베이스 기술 ; 퍼스컴 네트워크 ; 사내 데이터베이스와
          산업계 데이터베이스 ; 등

ENTER PF1 KEY, TO ESCAPE FROM THIS SCREEN. PAGE CONTROL=
```

```
COMMAND:DISPLAY
도    제 : 데이터베이스론
저    자 : 이석호
서지사항 : 서울, 경학사, 1985
          429p.  23㎝
          LA:K   CC:QA76.9.D3/ 이 52ㄷ
색 인 어 : DATABASE:DBMS
내    용 : 데이터베이스 관리 시스템 ; 데이터베이스 시스템 구성 ; 데이터 드렙 ;
          네트워크 데이터베이스 시스템, 등

ENTER PF1 KEY, TO ESCAPE FROM THIS SCREEN, PAGE CONTROL=
```









V. 결 어

본 고에서는 ETLARS-Ⅱ 한글정보검색시스템의 온라인 검색프로그램의 개발에 관한 내용을 기술하였다. ETLARS-Ⅱ 시스템은 대규모 Databank용 정보검색시스템으로서 국내에서 자체 개발하여 운영하고 있는 극히 소수의 시스템들 중의 하나로 그 기술수준을 자랑할만 하다.

ETLARS-Ⅱ 시스템은 연구소내의 LAN이 개통됨에 따라 소내 이용자들이 연구실 어디에서나 이용할 수 있게 되었으며 점차적으로는 체신부를 비롯한 국내전자통신 관련기관과 네트워크를 연결하여 전자통신 정보유통망을 구축해 나갈 계획이다. 그러나 ETLARS-Ⅱ 시스템이 명실공히 전자통신 전문 Databank로서의 기능을 발휘하기 위해서는 Database의 질적 및 양적 확충은 물론 이를 위한 Database 제작 전문인력의 양성 및 컴퓨터 환경의 개선 등이 추진되어야 할 것이다.

참 고 문 헌

1. 컴퓨터시스템 운영사업. 대전, 한국전자통신연구소, 1984.
2. 정보검색서비스 이용안내. 대전, 한국전자통신연구소, 1987.
3. Aho, A., V., Sethi, R. & Ullman, J. D. Compilers; principles, techniques and tools. Reading, Addison-Wesley, 1986.
4. Date, C. J. An introduction to database systems. 3rd ed. Vol. 1. Reading, Addison-Wesley, 1981.
5. Guide to Dialog searching. Palo Alto, Dialog Information Services, Inc., 1985.
6. Henry, W. M. et al. Online searching; an introduction. London, Butterworths, 1980.
7. Horowitz, E. & Shani, S, Fundamentals of data structures in PASCAL. Rockville, Computer Science Press, 1984.
8. Hughes, J. PL/I structured programming. New Yo k. John Wiley & Sons, 1979.
9. IBM manual.
10. Lancaster, F. W. Information Retrieval system. 2nd ed. New York, John Wiley & Sons, 1979.
11. Salton, G. Introduction to modern information retrieval. New York, McGraw-Hill, 1983.