ADABAS TEXT RETRIEVAL (MVS/BS2000)

REFERENCE MANUAL

TRS - 211 - 030

Manual Order Number: TRS - 211 - 030

This document is applicable to ADABAS TEXT RETRIEVAL Version 2.1 and to all subsequent releases. Specifications contained herein are subject to change and these changes will be reported in subsequent update series or new editions.

Readers' comments are welcomed. A form for this purpose is provided at the back of this publication. If this form has been removed, comments may be directed to:

SOFTWARE AG
Documentation
Uhlandstrasse 12
6100 Darmstadt
Federal Republic of Germany

Telefax: 06151 - 92 - 2610

or your nearest affiliate.

TABLE OF CONTENTS

4. FILE STRUCTURE

5. TOKENIZATION AND KEYWORD DEFINITION

6. THE NATURAL TEXT RETRIEVAL INTERFACE

7. SAMPLE APPLICATION

CHAPTER 1

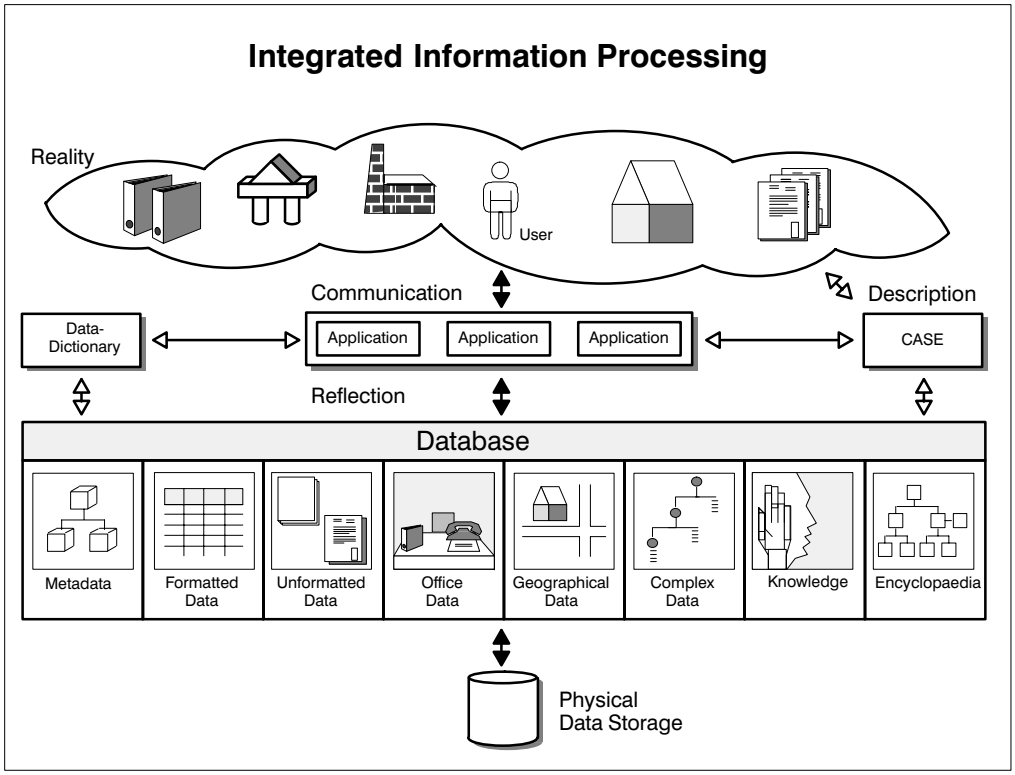
INTRODUCTION

INTRODUCTION

In industrialized societies, where the quantity of available information is increasing exponentially, a major priority has become effective information management and distribution. Database technology provides perhaps the only means of managing information of such vast proportations.

Traditional information processing has been performed almost exclusively on *formatted data* — data having a specified type and length. Advances in software and hardware technology, however, have made possible the storage and retrieval of textual information as well. The ability to process unformatted data makes it possible to extract needed information quickly from a large text data bases.

The demand for efficient text retrieval is large and growing rapidly in almost all industries. For example, textual information is found in great quantities in such fields as publishing, library archiving, law and technical documentation. All of the necessary data retrieval and data management services needed to create a comprehensive and truly integrated information processing environment are provided by ADABAS TEXT RETRIEVAL together with other Software AG products.



ADABAS TEXT RETRIEVAL Overview

ADABAS TEXT RETRIEVAL is the heart of Software AG's text retrieval architecture. It offers the full range of functionality expected of powerful information retrieval systems. Applications which access both formatted and unformatted data simultaneously can be developed using ADABAS TEXT RETRIEVAL. Other Software AG products which apply this architecture include:

- NATURAL DOCUMENT MANAGEMENT - a complete document management system;
- CON-NECT DOCUMENT RETRIEVAL - a optional extension to the functionality of Software AG's office information system CON-NECT.

Since ADABAS TEXT RETRIEVAL is an extension of Software AG's database management system ADABAS, it inherits such advantages as high-performance data compression, on to restart, automatic recovery and 24-hour operation.

ADABAS TEXT RETRIEVAL manages the index information and not the content of the data. This means that document contents can be stored at any location (ADABAS, sequential files, CD-ROM, PC, etc.).

ADABAS TEXT RETRIEVAL can be used via its call interface from inside NATURAL or any third generation language such as COBOL or PL/1.

ADABAS TEXT RETRIEVAL Functionality

Text can be designated either as formatted or unformatted depending on your requirements. Unformatted text is referred to in the remainder of this manual as free-text chapters.

Free-text chapters are subject to a process called inversion which creates the information necessary to retrieve a text based on content. Any of three different *inversion* methods can be used:

- Full-text inversion;
- Thesaurus-controlled inversion;
- Inversion using stopword lists.

Numerous functions and operators are available for flexible retrieval:

- Word searches
- Word truncation:
  - Right truncation;
  - Left truncation;
  - Left and right truncation:
  - Middle truncation.
- Phonetic searches;
- Synonym searches;
- Integration of thesaurus relations:
  - Broader terms;
  - Narrower terms;
  - Synonyms.
- Proximity operators for searching adjacent terms, near terms, terms in a sentence and/or paragraph;
- Relational operators;
- Boolean operators;
- Structure-independent search (any combination of free-text chapters and formatted fields);
- References to previous queries (refinement);
- Sorting in ascending and descending order;
- Highlighting of found items.

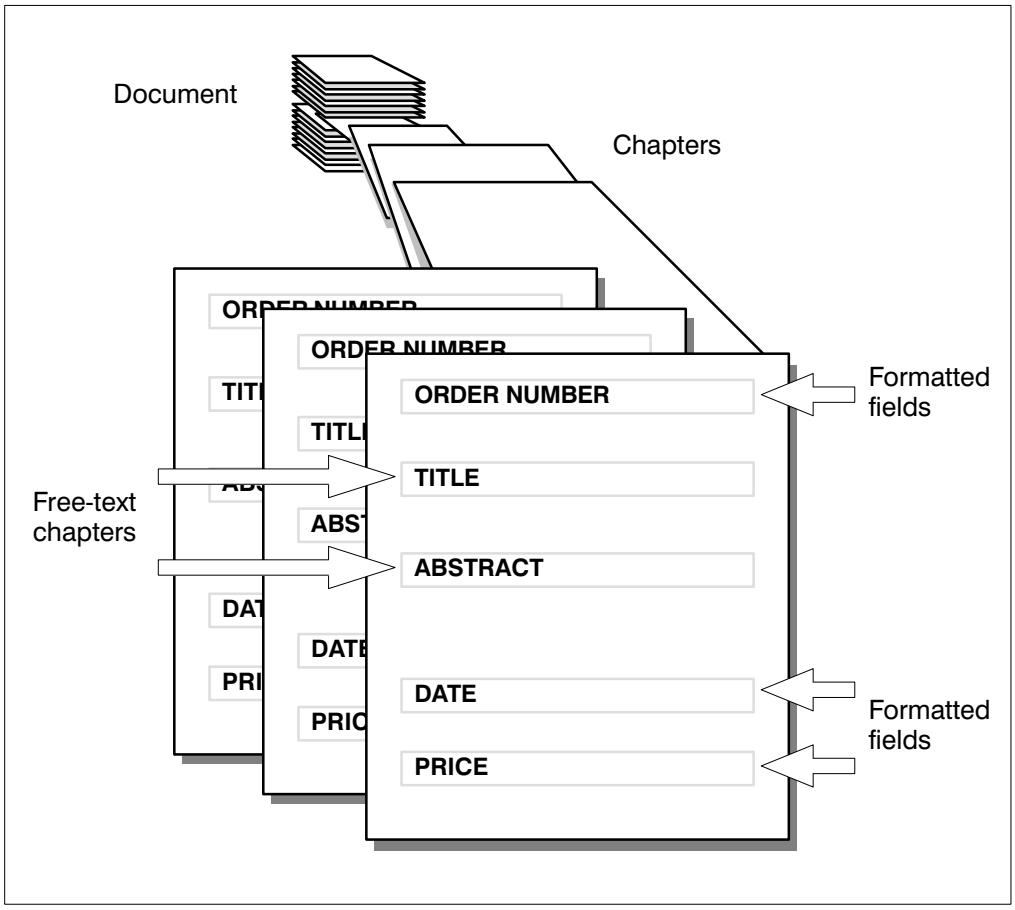
ADABAS TEXT RETRIEVAL Terminology

Defined below are the most important and frequently used terms in ADABAS TEXT RETRIEVAL.

Document

Documents consists of chapters (sometimes referred to as categories) which are equivalent to fields in the relational database model. Chapters can either be designated as free-text chapters, which are managed by ADABAS TEXT RETRIEVAL, or as formatted fields in accordance with the database system.

Free-text chapters can be separated into paragraphs and sentences. This allows you to issue queries which search individual sentences and paragraphs.



Tokenization

Tokenization is the process by which ADABAS TEXT RETRIEVAL identifies substrings of an entered text. The process consists of three major steps:

- Identification of a character as being a valid character defined in the ADABAS TEXT RETRIEVAL character table;
- Making the process of tokenization dependent on the context of the occurrence of characters; this part is defined by the appropriate algorithms implemented in a specifically designed macro assembler language.
- Translation of the word detected by the previous parts of the process.

Inversion

Inversion is the process which creates the necessary document index entries for the contents of free-text chapters in the document. ADABAS TEXT RETRIEVAL supports three inversion methods:

- Full-text inversion;
- Inversion using a controlled thesaurus;
- Inversion by ignoring words in the stopword list.

You can choose one of the inversion methods for each free-text chapter.

CHAPTER 2

ADABAS TEXT RETRIEVAL CALLS

ADABAS TEXT RETRIEVAL CALLS

ADABAS TEXT RETRIEVAL provides calls which perform the following functions:

- Set up ADABAS TEXT RETRIEVAL sessions;
- Invert free-text chapters;
- Retrieve text and information;
- Browse through ISN sets;
- Highlight search terms.

This chapter explains what each call is designed to carry out and the parameters of each call. The names of dynamic parameters are appear in uppercase capital letters. All other parameters appear in italics.

Alphabetical Listing

The following table lists all available ADABAS TEXT RETRIEVAL calls in alphabetical order:

Topical Listing

The following table provides a cross reference of ADABAS TEXT RETRIEVAL calls according to function:

ADD

Description

The ADD call creates the document index entries for the contents of a free-text chapter within a document. This process is called document inversion.

Before the ADD call can be executed, the free-text chapter to which the entered text belongs must have been established.

The free-text chapter is established by a BC or DYP call which provides the name of the ADABAS hyperdescriptor associated with the free-text chapter in question, as the value of the TEXT parameter.

The ADD call stores entries in the document index; it does not store the document text.

Example

See page 7 - 15.

Syntax

```
CALL 'TRS' 'ADD' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Document-ID	alphanumeric	variable	input
Source-Text Length	binary	4 bytes	input
Source Text	alphanumeric	variable	input
Document-ISN	binary	4 bytes	output
End-of-Text Indicator	alphanumeric	6 bytes	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Document-ID

A unique Document-ID must be provided. There are two different ways of providing a Document-ID depending on the setting of the dynamic parameter DOCID of the DYP call:

- If the DOCID parameter contains the name of a formatted field (ADABAS descriptor), the *Document-ID* parameter must contain a unique value for the formatted field in question. If a record containing the specified value of *Document-ID* already exists, the ADABAS ISN of this record is used in the document index, otherwise an ADABAS record containing only the value for the formatted field in question is added to the document file.
- If the DOCID parameter contains the value '##', the *Document-ID* parameter must contain the ADABAS ISN of a record on the document file. An ADABAS record with the specified ISN must already exist on the document file.

Source-Text Length

The length of the text in bytes, as contained in the parameter *Source Text*.

Source Text

The text to be inverted.

Document ISN

Document ISN

The Document ISN reflects either of the following ISNs:

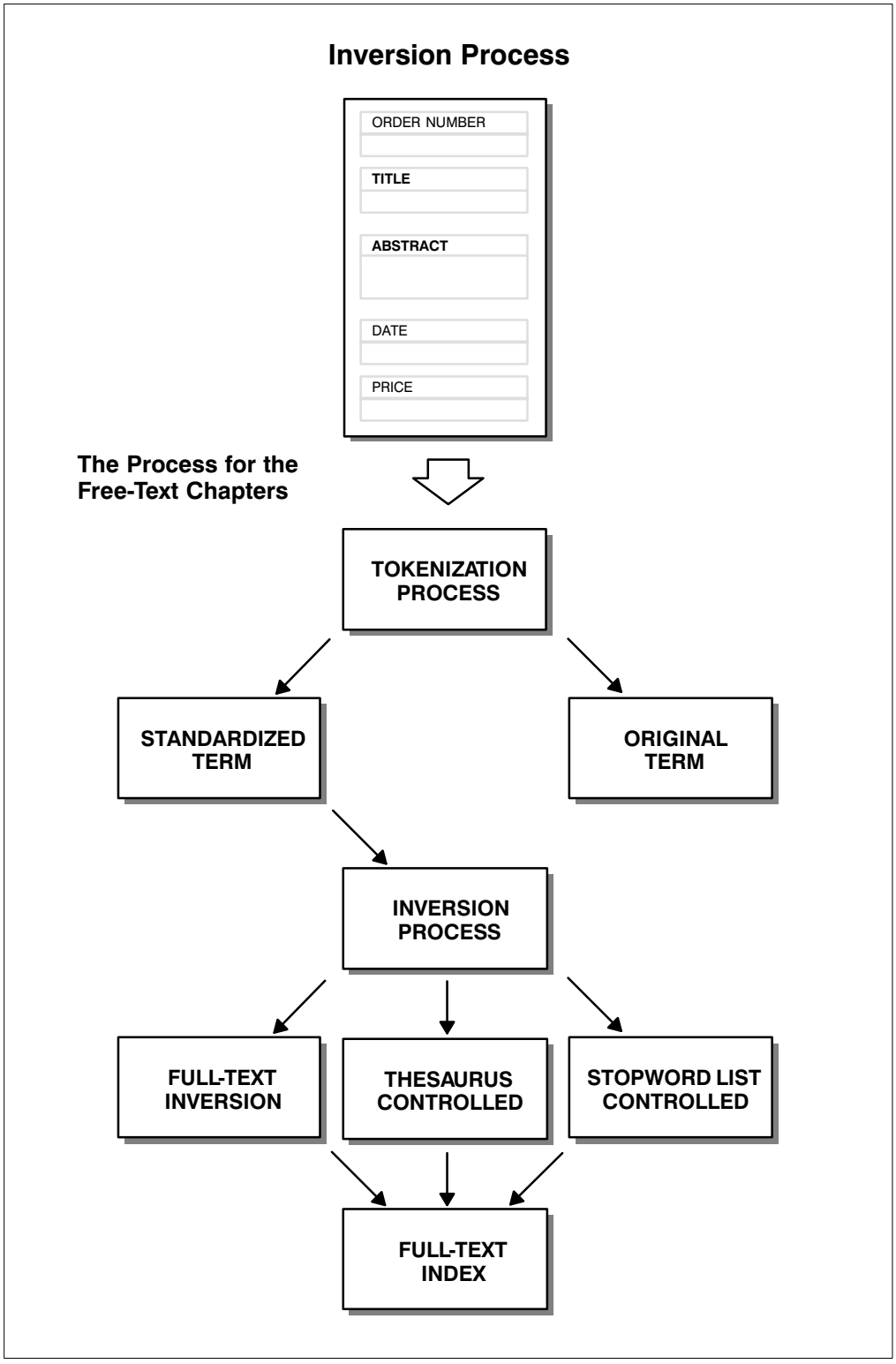
- the ISN entered in the Document-ID parameter;
- the ISN of the record containing the value of the Document-ID parameter.

End-of-Text Indicator

Free-text chapters can be inverted as one text string or divided into several parts. The *End-of-Text Indicator* parameter must contain either of the following values:

LAST	For the last part of a free-text chapter.
NOLAST	In all remaining cases.

The inversion process for the contents of a free-text chapter can generally be executed in one step. However, for the inversion of long texts it may be necessary to execute the inversion in multiple steps, because intermediate ADABAS end transactions may be required in order to prevent an ADABAS Hold Queue overflow.


Inversion Process
ORDER NUMBER
TITLE
ABSTRACT
DATE
PRICE
The Process for the Free-Text Chapters
TOKENIZATION PROCESS
STANDARDIZED TERM
ORIGINAL TERM
INVERSION PROCESS
FULL-TEXT INVERSION
THESAURUS CONTROLLED
STOPWORD LIST CONTROLLED
FULL-TEXT INDEX

BC

Description

The BC call opens an ADABAS TEXT RETRIEVAL session. This call is mandatory and must be invoked once at the beginning of each session.

Example

See page 7 - 6.

Syntax

```
Call 'TRS' 'BC' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Size of Buffer	binary	4 bytes	in/output
Save Area	alphanumeric	100 bytes	output
Dynamic Parameters	alphanumeric	variable	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Size of Buffer

The size of the ADABAS TEXT RETRIEVAL buffer.

Minimum Size	18 K
Recommended Size	48 K

If ADABAS TEXT RETRIEVAL is invoked by NATURAL, the *Size of Buffer* becomes an output parameter reflecting the value of the NATURAL TSIZE parameter as defined in the NATURAL parameter module.

Save Area

The name of the storage area used by ADABAS TEXT RETRIEVAL as a temporary save area.

Dynamic Parameters

Any of the following dynamic paramters can be specified in a BC call:

AUTOASP	**DSFNR**	**MAXDPRO**	**SETCHAR**	**WORDLEN**
DEFOPER	**ERRPRE**	**MAXVSET**	**TEXT**	
DFNR	**ICBUF**	**PASSWORD**	**TRUNCHAR**	
DOCID	**INDEX**	**SEARCHLB**	**VFNR**	

For details of the dynamic parameters and their possible values, refer to the section entitled **Dynamic Parameters for the DYP and BC Calls** on page 2 - 30.

CL

Description

The CL call closes an ADABAS TEXT RETRIEVAL session and releases all resources.

Example

See page 7 - 8.

Syntax

```
CALL 'TRS' 'CL' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Save Area	alphanumeric	100 bytes	output

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Save Area

This parameter refers to the storage area used by ADABAS TEXT RETRIEVAL for a temporary save area.

DDS

Description

The DDS call deletes the document index entries for a specific free-text chapter within a document.

Important

Before the DDS call can be executed, the free-text chapter to which the entered text belongs must have been established.

The free-text chapter is established by a BC or DYP call which provides the name of the ADABAS hyperdescriptor associated with the free-text chapter in question, as the value of the TEXT parameter.

The DDS call deletes entries in the document index; it does not delete the document text.

Example

See page 7 - 17.

Syntax

```
CALL 'TRS' 'DDS' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Document-ID	alphanumeric	variable	input
Delete Option	alphanumeric	3 bytes	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Document-ID

A unique Document-ID must be provided. There are two different ways of providing a Document-ID depending on the setting of the DOCID parameter of the DYP call:

- If the DOCID parameter contains the name of a formatted field (ADABAS descriptor), the *Document-ID* parameter must contain a unique value for the formatted field in question.
- If the DOCID parameter contains the value '##', the *Document-ID* parameter must contain the ADABAS ISN of a record on the document file.

Delete Option

There is only one possible delete option:

SUM	Deletes the document index entries for the current free-text chapter.

Note: If more than one free-text chapter exists for a document whose index entries are to be deleted, the document index entries for each chapter must be deleted separately using repeated pairs of DYP and DDS calls.

DSL

Description

The DSL call defines search labels on the document index. These labels allow direct referencing of both formatted fields and free-text chapters in queries.

For free-text chapters, the same search label can be entered for more than one ADABAS hyperdescriptor name, thus making up a global search label which enables the user to address multiple free-text chapters with one search label in a query.

Example

See page 7 - 7.

Syntax

```
CALL 'TRS' 'DSL' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Search Labels	alphanumeric	variable	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Search Labels

A search label can be assigned to an ADABAS field. This can either be a formatted field (ADABAS descriptor), or a free-text chapter (ADABAS hyperdescriptor).

The definition of search labels consists of a character string containing one or more entries separated by commas and ending with a period.

To define a search label, the name of the formatted field (ADABAS descriptor), or the free-text chapter (ADABAS hyperdescriptor) must be entered followed by an equal sign and the search label of the specific field. For upward compatibility, the name of the hyperdescriptor must be entered twice.

Example:

```
'F1=DATE,Y1Y1=TITLE, Y2Y2=ABSTRACT, Y1Y1=ALL, Y2Y2=ALL.'
```

'F1'	Formatted field
'Y1'	Free-text chapter
'Y2'	Free-text chapter

DYP

Description

The DYP call enables users to define dynamic parameters or redefine any specified in the BC call at the start of a session or in previous DYP calls. For example, the call enables users to handle more than one free-text chapter within a document by changing the TEXT parameter.

Example

See pages 7 - 15, 7 - 17, and 7 - 37.

Syntax

```
CALL 'TRS' 'DYP' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Dynamic Parameters	alphanumeric	variable	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A “Messages and Codes”.

Dynamic Parameters

Any of the following dynamic paramters can be specified in a DYP call:

AUTOASP	**DSFNR**	**MAXDPRO**	**SETCHAR**	**WORDLEN**
DEFOPER	**ERRPRE**	**MAXVSET**	**TEXT**	
DFNR	**ICBUF**	**PASSWORD**	**TRUNCHAR**	
DOCID	**INDEX**	**SEARCHLB**	**VFNR**	

For details of the dynamic parameters and their possible values, refer to the section entitled **Dynamic Parameters for the DYP and BC Calls** on page 2 - 30.

EISE

Description

Each query executed by ADABAS TEXT RETRIEVAL results in an ADABAS ISN set. This set can be referenced by EISE, EISG and EISS calls. After an EISS call and any number of EISG calls have been used, the EISE call must be used to conclude browsing through an ISN set.

Example

See page 7 - 33.

Syntax

```
CALL 'TRS' 'EISE' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Command-ID	binary	4 bytes	input
Set Type	alphanumeric	1 byte	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Command-ID

The ADABAS Command-ID of the ISN set to be referenced, as created by the QR call. It is identical to the *Command-ID* output parameter of the QR call which generated the ISN set and must therefore be set to the same value.

Set Type

The type of ISN set for which browsing is to be terminated. One of the following values must be specified:

'D'	Document ISN set
'V'	Vocabulary ISN set

The value of the *Set Type* parameter must be identical to the value of the *Type* parameter of the QR call.

EISG

Description

Each query executed by ADABAS TEXT RETRIEVAL results in an ADABAS ISN set. This set can be referenced by EISE, EISG and EISS calls.

The EISG call is used to browse through an ISN set created by a QR call. The sequence of one or more EISG calls must be preceded by an EISS call and concluded by an EISE call.

Example

See page 7 - 30.

Syntax

CALL 'TRS' 'EISG' *parameters*

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Command-ID	binary	4 bytes	input
Set Type	alphanumeric	1 byte	input
Quantity	binary	4 bytes	input
Position	binary	4 bytes	input
ISN	binary	4 bytes	output

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Command-ID

The ADABAS Command-ID of the ISN set to be referenced, as created by the QR call. It is identical to the *Command-ID* output parameter of the QR call which generated the ISN set and must therefore be set to the same value.

Set Type

The type of ISN set for which browsing is to be executed. One of the following values must be specified:

'D'	Document ISN set
'V'	Vocabulary ISN set

The value of the *Set Type* parameter must be identical to the value of the *Type* parameter of the QR call.

Quantity

The number of ISNs in the set generated by the QR call. The value of the *Quantity* parameter must be identical to the *Quantity* parameter of the QR call.

Position

The position of the requested *ISN* within the ISN set as generated by the QR call.

ISN

The ISN within the position as indicated by the *Position* parameter in the ISN set is returned by the EISG call.

EISS

Description

Each query executed by ADABAS TEXT RETRIEVAL results in an ADABAS ISN set. This set can be referenced by EISE, EISG and EISS calls.

The EISS call starts browsing through an ISN set created by a QR call. The EISS call must be performed once before each sequence of EISG calls used to browse through an ISN set.

Example

See page 7 - 29.

Syntax

CALL 'TRS' 'EISS' *parameters*

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Command-ID	binary	4 bytes	input
Set Type	alphanumeric	1 byte	input
Quantity	binary	4 bytes	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Command-ID

The ADABAS Command-ID of the ISN set to be referenced, as created by the QR call. It is identical to the *Command-ID* output parameter of the QR call which generated the ISN set and must therefore be set to the same value.

Set Type

This parameter defines the type of ISN set for which browsing is to be started. One of the following values must be specified:

'D'	Document ISN set
'V'	Vocabulary ISN set

The value of the *Set Type* parameter must be identical to the value of the *Type* parameter of the QR call.

Quantity

The number of ISNs in the set generated by the QR call. The value of the *Quantity* parameter must be identical to the *Quantity* parameter of the QR call.

HIGH

Description

The HIGH call is used to mark those words in a document which have been found for a given query.

The HIGH call marks the beginning and ending of the words to be highlighted by two specific characters. In NATURAL these characters can be used for dynamic highlighting by means of the dynamic attribute feature (DY).

Important

Before the HIGH call can be executed, the free-text chapter to which the entered text belongs must have been established.

The free-text chapter is established by a BC or DYP call which provides the name of the ADABAS hyperdescriptor associated with the free-text chapter in question, as the value of the TEXT parameter.

Example

See page 7 - 36.

Syntax

```
CALL 'TRS' 'HIGH' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Document-ID	alphanumeric	variable	input
Query Name	alphanumeric	8 bytes	input
Input Text	alphanumeric	variable	input
Output Text	alphanumeric	variable	output
Text Length	binary	4 byte	input
Prefix	alphanumeric	1 byte	input
Suffix	alphanumeric	1 byte	input
Cursor	binary	4 bytes	in-/output

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A “Messages and Codes”.

Document-ID

A unique Document-ID must be provided. There are two different ways of providing a Document-ID depending on the setting of the DOCID parameter of the DYP call:

- If the DOCID parameter (DYP call) contains the name of a formatted field (ADABAS descriptor), the *Document-ID* parameter must contain a unique value for the formatted field in question.
- If the DOCID parameter (DYP call) contains the value '**##**', the *Document-ID* parameter must contain the ADABAS ISN of a record on the document file representing the document in question.

Query Name

The name of the query as defined in the *Query Name* parameter of the QR call. The words will be highlighted according to the selection criteria specified in this query.

Input Text

The source text which contains the words to be highlighted. Blanks should be provided at the beginning and at the end of the source text to host the assigned prefix and suffix characters if necessary.

Output Text

The source text including the assigned prefix and suffix characters.

Text Length

The length of the source text expressed in bytes.

Prefix

The special character indicating the beginning of a word to be highlighted.

Recommended character: '**<**'

Suffix

The special character indicating the end of a word to be highlighted.

Recommended character: **'>'**

Cursor

For each free-text chapter of a document, this parameter has to be set to zero at the beginning of the highlighting process and must not be changed for the remainder of the process.

QR

Description

The QR call is used to retrieve text and information from free-text chapters and formatted fields. This process is called information retrieval.

Example

See pages 7 - 21, 7 - 23, 7 - 24, 7 - 29, 7 - 42, 7 - 47, 7 - 48.

Syntax

```
CALL 'TRS' 'QR' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Query	alphanumeric	variable	input
Query Length	binary	4 byte	input
Query Name	alphanumeric	8 bytes	input
Disp Error	binary	4 byte	output
Length Error	binary	4 byte	output
Default Mode	alphanumeric	1 byte	input
Command-ID	binary	4 bytes	output
Quantity	binary	4 bytes	output
Type	alphanumeric	1 byte	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A “Messages and Codes”.

Query

The query. Its syntax is described in the chapter 3 “Query Syntax”.

Query Length

The length of the current query expressed in bytes.

Query Name

The name of the current query. Subsequent queries can use this name to refer to the results of the current query. Two different types of query names are possible depending on the value of the *Type* parameter:

Type	Query Name	Range of *nnn*
D	DOCS0nnn	001 - 999
V	WRDS0nnn	001 - 999

Disp Error

If an error is detected when the syntax of the query is checked, this parameter will contain the displacement of the erroneous term within the query.

Length Error

If an error is detected when the syntax of the query is checked, this parameter will contain the length of the erroneous term within the query.

Default Mode

The default selection mode. One of the following letters must be specified as the default selection mode:

Letter	Selection Mode
=	PRECISE
A	ASPECT
G	GROUP
P	PHONETIC
R	ROOT
S	SYN
X	SYR

Selection modes are explained in chapter 3 **Query Syntax**.

Command-ID

The ADABAS Command-ID of the ADABAS ISN set created by the QR call.

This parameter serves as input to the EISS, EISG, EISE and RET calls.

Quantity

The number of ISNs contained in the ISN set created by the ADABAS TEXT RETRIEVAL QR call.

Type

The type of retrieval to be executed by the QR call. There are two possible values:

'D'	Document retrieval
'V'	Vocabulary retrieval

RET

Description

The RET call is used to enter an ADABAS Command-ID created by a QR call into the NATURAL Retain-Table.

Example

See page 7 - 47.

Syntax

```
CALL 'TRS' 'RET' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Command-ID	binary	4 bytes	input
Quantity	binary	4 bytes	input
Retain-Name	alphanumeric	32 bytes	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Command-ID

The ADABAS Command-ID of the ISN set created by a QR call to be entered in the NATURAL Retain-Table.

Quantity

The number of ISNs contained in the ISN set created by a QR call.

Retain-Name

The name of the NATURAL Retain set. This name can be used in subsequent NATURAL FIND statements for further retrieval.

RQR

Description

The RQR call is used to release all results of a specific query.

A query is automatically released when another QR call with the same Query Name is executed.

Example

See pags 7 - 21, 7 - 23, 7 - 24, 7 - 46.

Syntax

```
CALL 'TRS' 'RQR' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Query Name	alphanumeric	8 bytes	input

Return Code

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Query Name

The name of the query to be released, as assigned in the QR call.

RULE

Description

The Rule call is used to define the rules for document inversion.

Syntax

```
CALL 'TRS' 'RULE' parameters
```

Required Parameters	Format	Length	In/Output
Return Code	binary	4 bytes	output
Option	alphanumeric	7 bytes	input
Max Words	binary	4 bytes	input
Aspects	alphanumeric	variable	input
Marks	alphanumeric	variable	input

The return code is the message delivered at the end of processing which indicates whether an error has occurred. A zero code indicates the normal end of processing. Other codes are explained in Appendix A **Messages and Codes**.

Option

This parameter defines the type of inversion to be executed. The options are:

Option	Characteristics
'FULL'	Full text inversion of all words contained in the document to be inverted. This option is generally the default setting.
'EXCLUDE'	Inversion of all words not belonging to one of the aspects as defined in the *Aspect* parameter.
'INCLUDE'	Inversion only of those words belonging to aspects as defined in the *Aspect* parameter.
'MARKED'	Inversion only of those words which are tagged with special markers as defined in the *Marks* parameter.

Max Words

The maximum number of words to be inverted per document. This parameter is only maintained for upward compatibility. If ADABAS hyperdescriptors are used for document indexing then this parameter is of no relevance.

Aspects

A list of the aspects to be used for the inversion of a document. Individual aspects on the list must be separated by commas and the list must end with a period.

Marks

A list of predefined character strings used in the document to be inverted. If the mark used refers to a predefined aspect, the latter can be entered after the mark and separated from it by the equals (=) symbol. Marks must be separated by commas and the list concluded with a period.

Dynamic Parameters for the DYP and BC Calls

The following dynamic parameters can be specified by a BC or DYP call.

Parameter	Explanation
AUTOASP	Automatically creates an aspect for each new word entered in the vocabulary file.
DEFOPER	Specifies the default operator to be used in case none is explicitly mentioned between two terms. Possible values are: AND, OR, NOT, ADJ, INPAR, INSEN, NEAR Default setting: ADJ
DFNR	Document File Number. The Document File Number can be specified either as it stands, or alternatively together with the ADABAS Database ID in question. In the latter case, Database ID and file number must be enclosed in parentheses and separated by a comma. Example: **'DFNR=39'** **'DFNR=(1,39)'** Note: The DFNR parameter is mandatory
DOCID	ADABAS field name of the Document-ID in the document file. The value **'##'** indicates that the ISN of the document file is to be used as document identification. Default setting: **'DA'**
DSFNR	Document Index File Number. The Document Index File Number can be specified either as it stands, or alternatively together with the ADABAS Database ID in question. In the latter case, Database ID and file number must be enclosed in parenthesis and separated by a comma. Example: **'DSFNR=39'** **'DSNR=(1,39)'** Note: The DSFNR parameter is mandatory
ERRPRE	A constant value to be added to all ADABAS TEXT RETRIEVAL return codes. Default setting: 0 (zero)
ICBUF	Number of in-core buffers (512 bytes each) used for the vocabulary. Default setting: 0 (zero)

Parameter	Explanation
INDEX	This parameter indicates which proximity indices are to be maintained by ADABAS TEXT RETRIEVAL. The following values can be entered: **'INDEX=(WORD)'** only word positions are maintained. **'INDEX=(WORD,SENTENCE)'** word and sentence positions are maintained. **'INDEX=(WORD,PARAGRAPH)'** word and paragraph positions are maintained. If this parameter is omitted, ADABAS TEXT RETRIEVAL will maintain word, sentence and paragraph positions.
MAXDPRO	Specifies the maximum number of selected documents on which a proximity search is to be performed. Default setting: 200
MAXVSET	Specifies the maximum number of words for a search. Default setting: 2000
PASSWORD	An ADABAS password can be supplied.
SEARCHLB	Specifies the default search label. The names of free-text chapters can be entered to a maximum of 20. If this parameter is omitted it takes the current value of the TEXT parameter. Example: **'SEARCHLB=(Y1Y1,Y2Y2,Y3Y3)'**
SETCHAR	Specifies the prefix character used to identify the result of previous queries. Default setting: NONE Recommended setting: **#**
TEXT	The name of the ADABAS hyperdescriptor of the current free-text chapter. This name is used for subsequent ADD and DDS calls. For compatibility reasons regarding previous versions of ADABAS TEXT RETRIEVAL, the name of the hyperdescriptor must be entered twice. If the SEARCHLB parameter is omitted, the value of the TEXT parameter is used as the default search label. Default setting: **'Y1Y1'**
TRUNCHAR	The character to be used as word truncation indicator. Default setting: **'*'**

Parameter	Explanation
VFNR	Vocabulary File Number. The Vocabulary File Number can be specified either as it stands, or alternatively together with the ADABAS Database ID in question. In the latter case, Database ID and file number must be enclosed in parenthesis and separated by a comma. Example: **'VFNR=38'** **'VFNR=(1,38)'** The VFNR parameter is mandatory
WORDLEN	The value of this parameter indicates the word length to be used by ADABAS TEXT RETRIEVAL. Maximum word length: 64. Default setting: 32

The dynamic parameters form a character string of one or more parameter entries and must be separated by commas and ended by a period. Each parameter must be coded as follows:

'Parameter=Value'

Example:

'VFNR=38,DFNR=39,DSFNR=39,TEXT=Y2Y2.'

CHAPTER 3

QUERY SYNTAX

QUERY SYNTAX

Search Labels

A search label is an alphanumeric identifier up to eight characters long used to refer to an element name in a search query. For example, the label “ABS” could be used to refer to the element “ABSTRACT” in search queries. This method of abbreviation saves keystrokes in entering search queries.

Search labels are defined using the DSL call (see page 2 - 11).

Queries in formatted fields must be preceded by a search label.

Queries to free text chapters can be preceded by a search label, but do not have to be if the search label required is the same as that last specified in the TEXT or SEARCHLB parameter of the last BC or DYP call. Thus, a search label for a free text chapter remains valid until another search label has been chosen to replace it. For example:

ABS ADABAS

The occurrences of the term ADABAS within the free text chapter ABSTRACT will be retrieved.

Search Mode Parameters

The search mode indicates the method to be used when retrieving inverted words.

The search mode precedes the search term in a query. If not search mode is specified, then the mode most previously specified is used. The following syntax must be used:

```
search-label  search-mode  search-term
```

When no selection mode is specified in the QR call which initiates the query, the default selection mode is used. The default selection mode is defined by the value of the Default Selection Mode parameter of the QR call in question.

search-label	The name of the search label for the element concerned
search-term	The term to be used as the basis for retrieval
search-mode	One of the following parameters

Parameter*	**Mode**
=	PRECISE
PHONETIC	PHONETIC
SYN	SYNONYM
ROOT	ROOT
SYR	SYNONYM/ROOT
ASPECT	ASPECT
GROUP*n*	GROUP

The options are explained in more detail below.

PRECISE Mode

The default mode. Searches are performed on the basis of spelling alone. Input must be identical to that contained in the document.

PHONETIC Mode

All words are retrieved which have the same phonetic value. This feature is designed for searches in German language, but it can also be used with some success for English language. For example, a PHONETIC search for the name “Mayer” will also retrieve the names “Maier” and “Meyer”.

SYNONYM Mode

If search terms are have synonyms defined for them in a thesaurus, then all documents containing the search term and its synonyms are found. The selection mode SYNONYM is based on the information stored previously in the SYNONYM field (V8) of the vocabulary file.

ROOT Mode

The ROOT search mode selects those documents which contain any of the search term's previously defined roots. This search mode is based on the information stored previously in the ROOT field (V4) of the vocabulary file.

SYNONYM/ROOT Mode

The SYR selection mode selects those documents which contain the words specified in the query, and/or any of their previously defined synonyms, and/or any of their previously defined roots. This search mode is based on the information stored previously in the SYNONYM field (V8) and/or ROOT field (V4) of the vocabulary file.

ASPECT Mode

All words are retrieved which are narrower terms for the search term. The number of hierarchical levels between the search term and the terms found is irrelevant. For example, an ASPECT search for the term "fiction" would retrieve the words "poem", "epic", "sonnet", "ballad", "haiku", "novel", "story", etc. The selection mode ASPECT is based on the information stored previously in the ASPECT field (V5) of the vocabulary file.

GROUP Mode

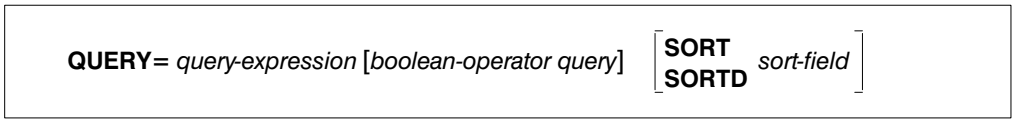
All words are retrieved which are narrower terms occurring *n* levels lower in the thesaurus hierarchy than the search term. Where *n* is omitted the depth is equal to 1.

For example, a GROUP search for the term "fiction" would retrieve the words "poem", "novel", "story", etc., but NOT the narrower terms ("sonnet", "epic", "ballad", "haiku", etc.) for these.

The selection mode GROUP is based on the information stored previously in the ASPECT field (V5) of the vocabulary file.

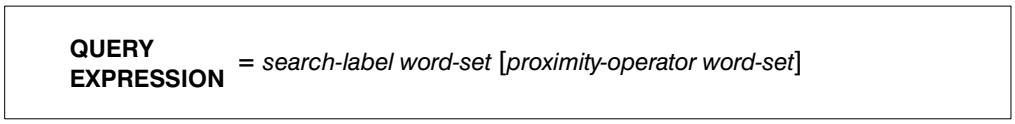
Query Syntax Diagrams

The syntax diagram below shows the various elements of a search query. Expressions in square brackets are optional.

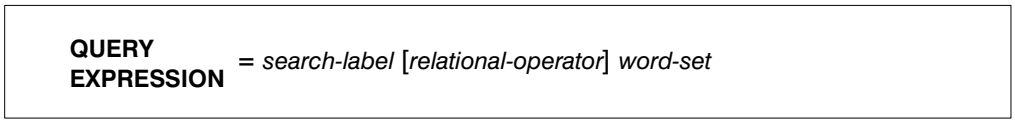


The content of *query-expression* varies depending on whether you reference word-inverted elements, formatted-inverted elements, or search numbers. The different possibilities are listed below.

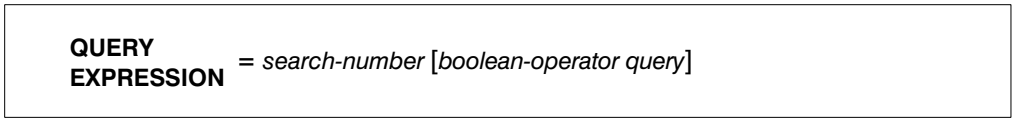
Word-inverted Elements (free-text chapters)



Formatted-inverted Elements



Search Numbers



Search Query Language

Case

Search queries may be entered in either upper or lower case.

Blanks

In searching formatted-inverted text elements, search terms which contain blanks or non-TRS characters must be enclosed in quotes.

In searching word-inverted text elements, blanks between words are interpreted as ADJ or NEAR operators, depending on parameters set during NDM installation. (see the section entitled **Proximity Operators** below).

Reserved Words

Search labels, global labels, and operators are reserved words within NDM and therefore cannot be used directly in search queries. To apply a reserved word in a search query, it must be surrounded by quotes.

Truncation

Truncation enables the retrieval of documents containing word segments or derivatives. Three types of truncation are available: left, right and middle. The character used to indicate truncation is specified by your administrator during installation. The following examples illustrate the three truncation options applying an asterisk (“*”) as truncation character.

Note: Word truncation is not possible for formatted-inverted elements.

Right Truncation

Right truncation is used to retrieve documents which contain words beginning with a specified string of letters. Thus the query

```
ABSTRACT kilo*
```

retrieves all words beginning with the string “kilo”, such as “kilogram” and “kilowatt”.

Left Truncation

Left truncation is used to retrieve documents which contain words ending with a specified string of letters. Thus the query

```
ABSTRACT *gram
```

retrieves all words ending with the string “gram”, such as “kilogram” and “program”.

Left and Right Truncation

Combined right and left truncation is used to retrieve documents which have words containing a specified string of letters. Thus the query

```
ABSTRACT *ra*
```

retrieves words such as “gram”, “kilogram”, “hurrah”, “arab”, etc.

Note: Combined use of left and right truncation is only possible if explicitly defined during NDM installation. Contact your administrator to determine if this function is available for your installation.

Middle Truncation

Middle truncation is used to retrieve documents which contain words beginning and ending with a specified string of letters. Thus the query

```
ABSTRACT hypo*mia
```

retrieves all words beginning with the string “hypo” and ending with the string “mia”, such as “hypothermia” and “hypoglycaemia”. It is not possible to specify how many letters are to be truncated.

Word Set

In NDM syntax, a word set is simply one or more or search terms or number strings forming part of a query expression.

Operators

Evaluation Order of Operators

When several different operators are used in the same search query, the order of evaluation is determined on a the basis of predefined priority. The evaluation priorities, from highest to lowest, are the following:

1. Expressions enclosed in parentheses;
2. Proximity operators (ADJ, NEAR, INSEN, INPAR);
3. AND;
4. NOT;
5. OR.

Boolean Operators

Boolean operators are used to join query expressions of the same or different types. The following Boolean operators may be used:

- AND ;
- NOT;
- OR .

The AND Operator

The AND operator is used to select documents based on the commonality of two query expressions. For example, the query

```
ABSTRACT strawberries AND cream
```

retrieves all documents in which the words “strawberries” and “cream” both occur in the element with the ABSTRACT search label.

The AND operator can be used more than once in a single query. For example, the query

```
ABSTRACT gin AND vermouth AND olive
```

retrieves all documents in which all the words are contained.

The OR Operator

The OR operator is used to retrieve documents in which any one of the terms specified occur. This is especially useful for retrieving related concepts. For example, the query

```
ABSTRACT sugar OR sweetener
```

retrieves all documents in which either the word “sugar” or the word “sweetener” occurs in the element with the ABSTRACT search label.

Like the AND operator, the OR operator can be used more than once within a single query. You can also replace the OR in search queries with a comma.

```
ABSTRACT sugar, sweetener
```

The NOT Operator

The NOT operator is used to retrieve documents which contain one specified term and which do not contain another. It can be used only following a query expression. For example, the query

```
ABSTRACT sweetener NOT honey
```

retrieves those documents in which the element with the ABSTRACT search label contains the word “sweetener”, but not the word “honey”. The following example, however, is invalid:

```
ABSTRACT NOT honey
```

Relational Operators

Relational operators are used to reference alphanumeric and numeric formatted-inverted elements.

The following relational operators may be used:

- BETWEEN *n*,*n*
- EQ *n* (“equal to”)
- GE *n* (“greater than or equal to”)
- GT *n* (“greater than”)
- LE *n* (“less than or equal to”)
- LT *n* (“less than”)

where “*n*” in each case is an obligatory value which depends on the element in question.

The relational operator can be omitted in which case the default is ‘EQ’.

The operators are described in more detail below.

The BETWEEN Operator

The BETWEEN operator is used to retrieve documents containing any one of a range of values. For example, the query

```
DATE BETWEEN 19870101,19871231
```

retrieves all documents in which the date given in the element with the DATE search label is 1987.

The EQ Operator

The EQ (“equal to”) operator is used to retrieve documents containing a single, precise value. For example, the query

```
NUMBER EQ 109
```

retrieves the document in which the value for the NUMBER search label is 109.

The same result is also be achieved by omitting the operator:

```
NUMBER 109
```

The GE Operator

The GE (“greater than or equal to”) operator is used to retrieve documents containing a value greater than or equal to the specified value. For example, the query

```
NUMBER GE 109
```

retrieves all documents in which the value for the NUMBER search label is 109 or greater.

The GT Operator

The GT (“greater than”) operator is used to retrieve documents containing a value larger than the specified value. For example, the query

```
NUMBER GT 109
```

retrieves all documents in which the value for the NUMBER search label is greater than 109.

The LE Operator

The LE (“less than or equal to”) operator is used to retrieve documents containing a value less than or equal to the specified value. For example, the query

```
NUMBER LE 109
```

retrieves all documents in which the value for the NUMBER search label is less than or equal to 109.

The LT Operator

The LT (“less than”) operator is used to retrieve documents containing a value smaller than the one specified. For example, the query

```
NUMBER LT 109
```

retrieves all documents in which the NUMBER search label is less than 109.

Proximity Operators

Proximity operators specify retrieval based on relative word position within a text. They can only be used to search word-inverted elements.

The following proximity operators are available:

- ADJn;
- NEARn;
- INPARn;
- INSENn.

where “n” is an optional integer value.

Although extremely powerful, proximity operators may cause performance problems and for this reason your NDM administrator may have disabled them for your database. If allowed, they should only be used if you are searching relatively small document sets.

The ADJ Operator

The ADJ operator is used to retrieve documents in which words appear within the distance from one another specified by the “n” parameter, and in the order of entry. The ADJ operator used alone is equivalent to “ADJ1”. For example, the query

```
ABSTRACT Moon ADJ River
```

selects all documents in which the element with the “ABSTRACT” search label contains the word string “Moon River”.

Where the operator is used with the “n” parameter, NDM retrieves all documents in which the distance between the words specified is equal to or less than the parameter value given.

Thus, for example, the query

```
ABSTRACT Tennessee ADJ3 Authority
```

selects all documents in which the word “Authority” occurs three or less words after the word “Tennessee”, e.g. “Tennessee River Valley Authority” and “Tennessee Highways Authority.”

The NEAR Operator

The NEAR operator is used to retrieve documents in words appear near each other, irrelevant of their order. For example the query

```
ABSTRACT recycled NEAR paper
```

selects all documents in which the element with the ABSTRACT search label contains the word pairs “recycled paper” and “paper recycled”.

The NEAR operator can be modified by adding a numerical delimiter. Thus, the query

```
ABSTRACT recycled NEAR4 paper
```

selects all documents containing the word groups “recycled paper”, “paper is recycled” and “paper will be recycled” and “recycled different kinds of paper”.

The INSEN Operator

The INSEN operator is used to retrieve documents in which the words specified occur within the same sentence and in which the individual sentences have been marked with delimiting characters. For example, the query

```
ABSTRACT coffee INSEN columbia
```

selects all documents in which the word “coffee” occurs in the same sentence of the element with the ABSTRACT search label as the word “columbia”.

The INSEN operator can be modified by adding a numerical delimiter. Thus the query

```
ABSTRACT coffee INSEN4 columbia
```

selects all documents in which the word “coffee” occurs within four sentences of the word “columbia”.

The INPAR Operator

The INPAR operator is used to retrieve documents in which the words specified occur within the same paragraph and in which the individual paragraphs have been marked with delimiting characters. For example, the query

```
ABSTRACT coffee INPAR columbia
```

retrieves all documents in which the word “coffee” occurs in the same paragraph as the word “columbia”.

The INPAR operator can be modified by adding a numerical delimiter. Thus, the query

```
ABSTRACT banana INPAR4 guatemala
```

retrieves all documents in which the word “banana” occurs within four paragraphs of the word “guatemala”.

Search Numbers

Search numbers are the numbers allocated to each query you issue during a session. They can be used in subsequent queries to reference the set of documents already retrieved. To distinguish them from normal numbers in queries, they must be given a prefix (e.g. #1). The prefix is specified by your administator during installation.

The search number consits of the current value of the SETCHAR parameter and the number of the query whose results are to be referenced.

An example of a query using a search number is:

#1 AND AUTHOR DICKENS

This would retrieve all documents in query set 1 with the author Dickens.

The SORT and SORTD Functions

Using the SORT and SORTD functions, you can sort a document set in ascending or descending order.

The syntax of the sort function is as follows:

SORT *search-label*

SORTD *search-label*

SORT	Specifies a sort in order of ascending magnitude.
SORTD	Specifies a sort in order of descending magnitude
search-label	Specifies the formatted-inverted element to be used as the sort criterion.

For example, the query

```
ABSTRACT sugar SORT DATE
```

retrieves all documents in which the word “sugar” occurs in the element with the ABSTRACT search label and sorts them according to the value given in the element which has search label “DATE”, starting with the oldest document.

As many as three sort criteria can be specified in a search query, for example:

```
SORT NUMBER  SORT PUBLISH  SORT ACC - NO
```

CHAPTER 4

FILE STRUCTURE

FILE STRUCTURE

The information in ADABAS TEXT RETRIEVAL is stored in three logical ADABAS files which can be stored in one or more physical ADABAS files. The three logical ADABAS files are:

- Document file (DFNR);
- Vocabulary file (VFNR);
- Document index file (DSFNR).

The document file must contain user-defined formatted fields which are to be used for retrieval operations. The ADABAS ISNs of the document file will make up the resulting ISN sets for retrieval operations.

The vocabulary file contains the word index and thesaurus of ADABAS TEXT RETRIEVAL.

The document index file contains the indices of all free-text chapters.

Important:

Because the ADABAS ISN's of all files mentioned above are used by ADABAS TEXT RETRIEVAL for free-text indexing, the ADABAS ISNs must not be deleted or changed in any way. If any of these files is reloaded, the USERISN parameter must be specified.

An Example of the Index Structure of ADABAS TEXT RETRIEVAL.

The following diagrams show how the internal index structure of ADABAS TEXT RETRIEVAL functions by way of a simple example, and how ADABAS TEXT RETRIEVAL indices are used when a retrieval operation is carried out.

For example, in Figure 1 the two documents DOC1 and DOC2 are allocated the ISNs 678 and 679 in the document file.

Document File

ISN	DOCID	TEXT
678	DOC1	this is a computer
679	DOC2	this is another computer

Figure 1

During the inversion process, each word of a document is entered on the vocabulary file (see Figure 2). The vocabulary file, similar to a dictionary, contains a *single* entry of each word known to the system no matter how frequently the word occurs in the documents. The words known to the system are entered in the V1 field. Each word entered receives an ADABAS ISN unique to that word. This unique ADABAS ISN is called a word ISN.

Thereafter, the name of the free-text chapter to which the information belongs and the document ID of the document in question are entered in the D1 field on the document index file (see Figure 3). The document ISN of the document to which the information belongs is entered in the D0 field. The word ISNs representing the words of which the document consists are entered in the multiple field D3.

The ADABAS hyperdescriptor technique enables ADABAS TEXT RETRIEVAL to make up an inverted list which relates the word ISNs of a document to the document ISN of the document in question, although this document ISN is part of the data of the document index file and not the original ISN as assigned by ADABAS.

During a retrieval operation (see Figure 4) ADABAS TEXT RETRIEVAL searches the vocabulary file for the ISN of the word sought. In the example on the following pages, the word “computer” is sought. If the word cannot be found, the result of the query will be zero; otherwise ADABAS TEXT RETRIEVAL uses the word ISN (in this case 4) to find all documents containing the word sought on the document index file. In this case, the documents with the document ISNs 678 and 679 are in the document file.

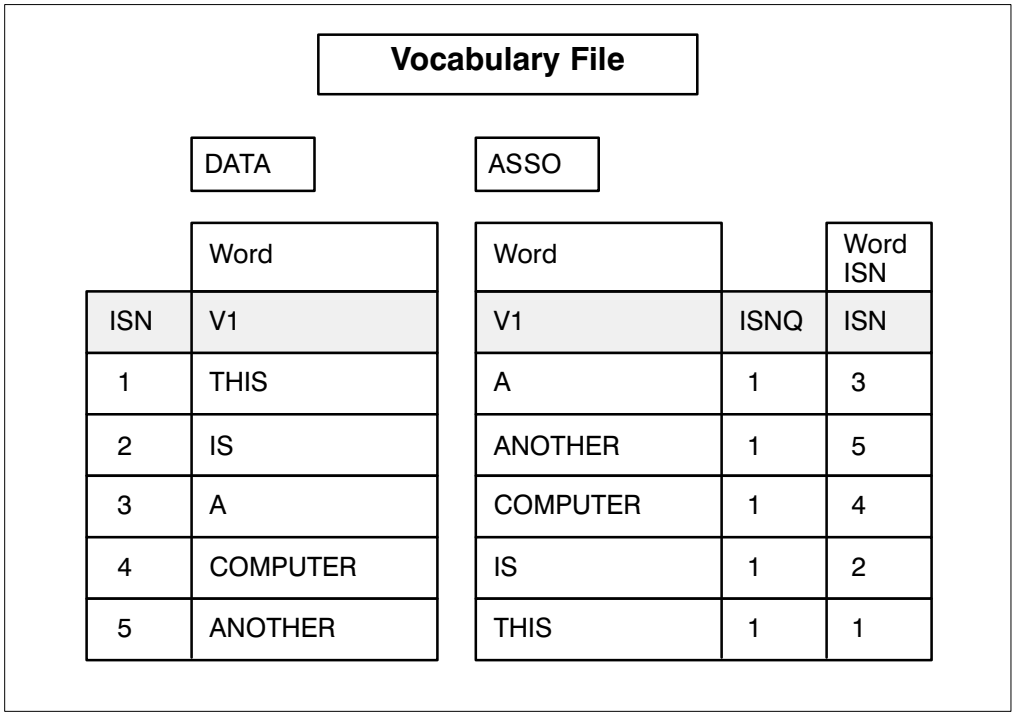


Figure 2

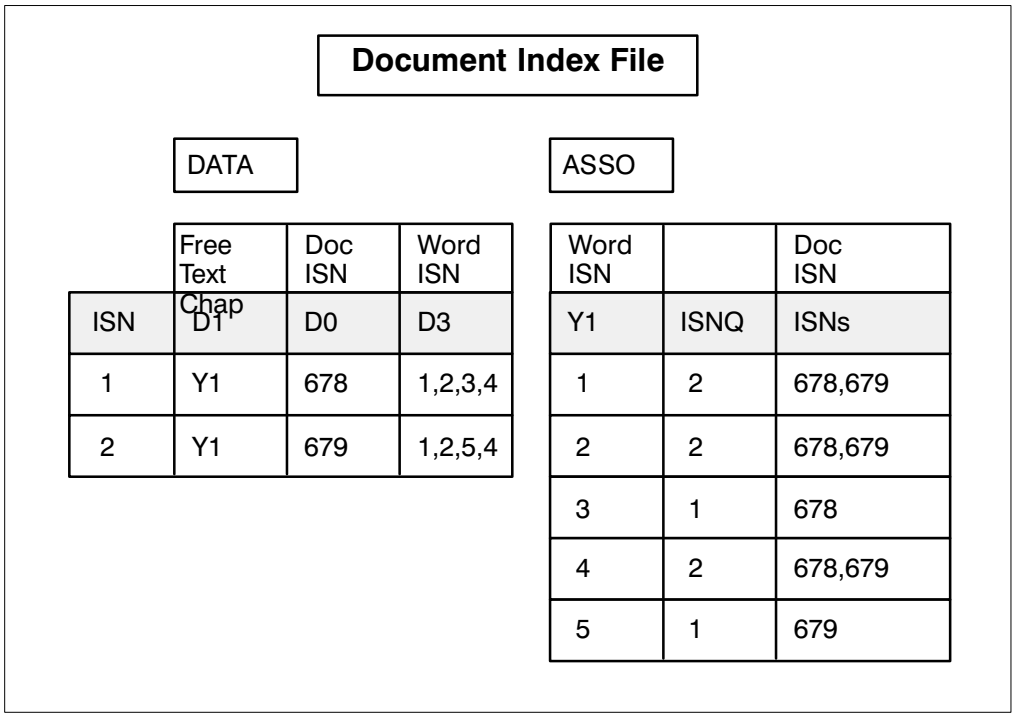


Figure 3

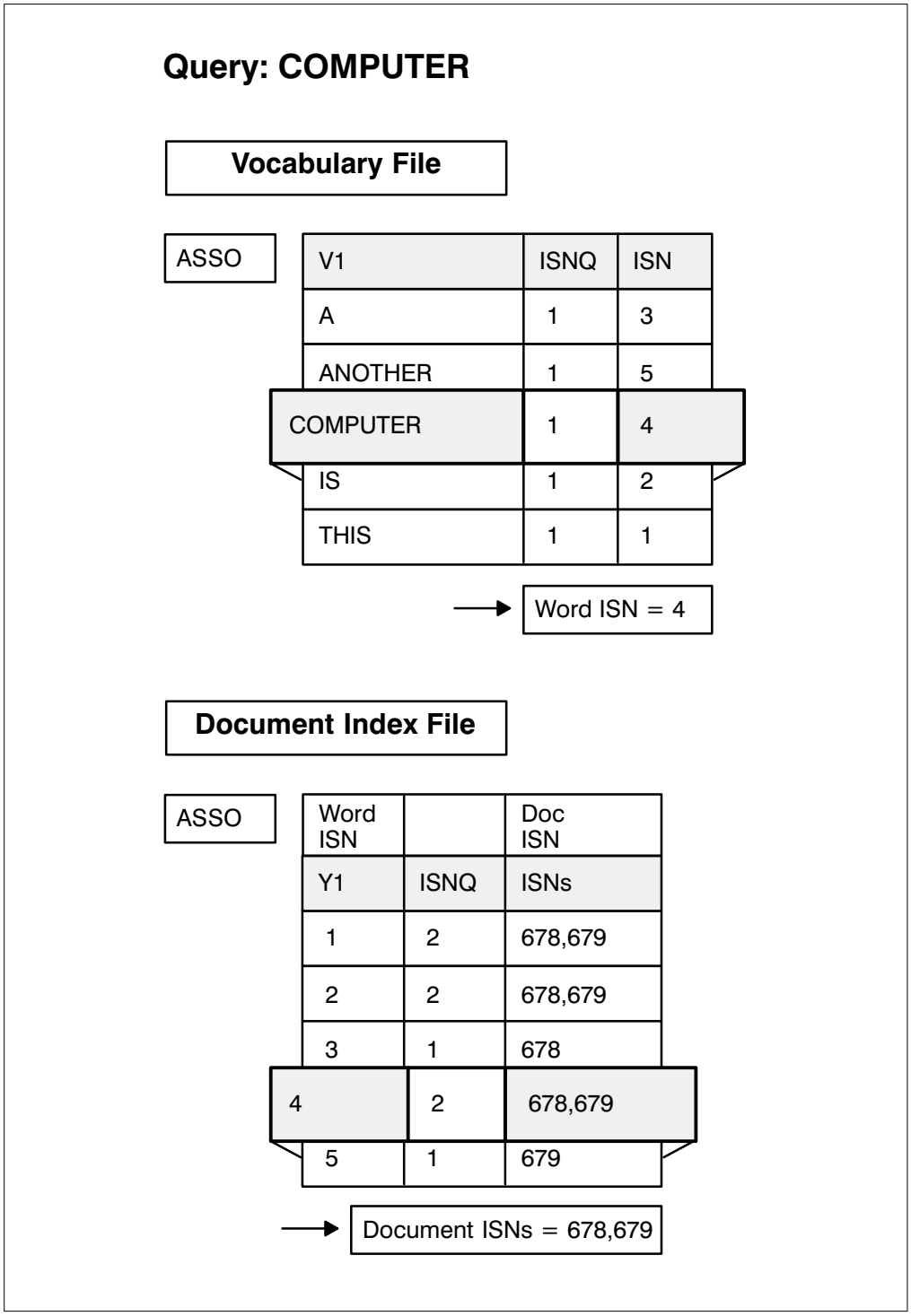

Query: COMPUTER
Vocabulary File
ASSO
V1
ISNQ
ISN
A
1
3
ANOTHER
1
5
COMPUTER
1
4
IS
1
2
THIS
1
1
Word ISN = 4
Document Index File
ASSO
Word ISN
Doc ISN
Y1
ISNQ
ISNs
1
2
678,679
2
2
678,679
3
1
678
4
2
678,679
5
1
679
Document ISNs = 678,679

Figure 4

Document File

ADABAS TEXT RETRIEVAL uses the ADABAS ISNs of the document file to identify the documents. These ADABAS ISNs form the result of any query entered in the system in the form of ADABAS ISN sets. The respective ADABAS ISN is either identified by the value of a unique formatted field on the document file or entered directly as a parameter to the ADD or DDS call (see the DOCID parameter in the DYP call, page 2 - 30).

In order to perform queries accessing the contents of the free-text chapters together with user defined formatted fields all relevant formatted fields must be contained in the document file.

Vocabulary File

The vocabulary file contains:

- The vocabulary (word index) of ADABAS TEXT RETRIEVAL;
- The ADABAS TEXT RETRIEVAL thesaurus.

It consists of the following fields:

Field	Description
FNDEF='01,V1,32,A,DE,NU'	(WORD) The field V1 contains the standardized word as encountered by ADABAS TEXT RETRIEVAL during the inversion process. A typical case of standardization is the translation of all characters (letters) contained in a word to upper case.
HYPDE='12,V2,32,A,NU,MU=V1'	(WORD-DOUBLE) The hyperdescriptor V2 contains the internal index values for the execution of double truncation (*string*). If double truncation is not required, the hyperdescriptor V2 can be omitted.
FNDEF='01,V3,32,A,DE,NU'	(WORD-REVERSE) The field V3 contains the internal index values for the execution of left and middle truncation.
FNDEF='01,V4,32,A,DE,MU,NU'	(ROOT) The field V4 contains user defined roots of the word contained in the V1 field; it supports the execution of the ROOT or SYR selection modes.

Field	Description
FNDEF='01,V5,32,A,DE,MU,NU'	(ASPECT) The V5 field contains user defined broader terms (ASPECTS) of the word contained in the V1 field: it supports the execution of the ASPECT and GROUPn selection modes.
PHONDE='V6(V1)'	(PHONETIC) The Phonetic descriptor V6 is built on the basis of the word contained in the V1 field. It supports the execution of the PHONETIC selection mode.
FNDEF='01,V8,32,A,DE,MU,NU'	(SYNONYM) The occurrences of the multiple value field V8 make up a synonym ring containing those user defined words which are of equal meaning; they support the execution of SYN or SYR selection modes.
FNDEF='01,V9,32,A,NU'	(ORIGINAL) The field V9 contains the original nonstandard form of the word as contained in the V1 field and as encountered by ADABAS TEXT RETRIEVAL during the inversion process at its first occurrence in any text entered into the system. If the original non-standardized form of the word is not required, the field V9 can be omitted.

Note: The length of all the fields above corresponds to the word length inside ADABAS TEXT RETRIEVAL. It is specified by the WORDLEN parameter of the BC or DYP call. It must not exceed 64 bytes. The default word length is 32 bytes.

Document Index File

The document index file contains the internal document index created by ADABAS TEXT RETRIEVAL during the inversion process.

It consists of the following fields:

Field	Description
FNDEF='01,D0,4,B,NU'	Document ISN
FNDEF='01,D1,32,A,NU,DE'	Name of free-text chapter and document
FNDEF='01,D2,4,B,NU,DE'	Segment number
FNDEF='01,D3,3,B,MU,NU'	Word index
FNDEF='01,D4,2,B,MU,NU'	Paragraph positions
FNDEF='01,D5,2,B,MU,NU'	Sentence positions
FNDEF='01,D6,2,B,MU,NU'	Word positions
FNDEF='01,D8,6,B,NU'	ISN and position of last word in previous segment

For each user-defined free-text chapter a hyperdescriptor in the following form must be added to the document index file:

Field	Description
HYPDE='12,Y1,3,B,MU,NU=D0,D1,D3'	This hyperdescriptor definition should be used for applications which either do not use proximity search at all, or only very occasionally. In this case the field D8 must be omitted from the document index file.
HYPDE='12,Y1,6,B,MU,NU=D0,D1,D3,D6,D8'	This hyperdescriptor definition should be used if fast operation for the ADJ1 and NEAR1 operators is required.

CHAPTER 5

TOKENIZATION AND KEYWORD DEFINITION

TOKENIZATION AND KEYWORD DEFINITION

Tokenization is the process by which ADABAS TEXT RETRIEVAL identifies substrings of a text as words. Tokenization occurs whenever the content of a free-text chapter is inverted (ADD call) or a query is processed by the system (QR call).

The process consists of three major parts:

1. The identification of valid characters. Character definitions are made in the ADABAS TEXT RETRIEVAL character table.
2. The possibility of making the process of tokenization dependent on the context of the occurrence of characters; this part is defined by the appropriate algorithms implemented in a specifically designed macro Assembler language.
3. The translation of the word detected by the previous parts of the process (above); the translation is defined by an appropriate translation table.

All parts of the tokenization process can be found in the module TRSSCT. Several standard implementations of TRSSCT are parts of the ADABAS TEXT RETRIEVAL installation libraries. You can use an existing module as is or modify a module to your requirements.

Versions of TRSSCT

The following different standard versions of TRSSCT are currently available:

Version	Explanation
TRSSCT	Standard implementation to support international character set including the German ‘Umlaute’.
TRSSCT1	The same as TRSSCT but including also the suppression of dashes (X’60’) and the substitution of the German ‘Umlaute’ by their appropriate transcriptions (e.g. Ä = AE etc.).
TRSSCTS	The same as TRSSCT but to be used in BS2000 environments.
TRSSCT1S	The same as TRSSCT1 but to be used in BS2000 environments.

Character Definition

Characters can be grouped into different user-defined classes. The definition of the appropriate user-defined classes is done by the TRSMLT macro.

Typical character classes could be, for example, ALPH, NUMBER, DOT etc. Each possible relevant character value is assigned to one or more classes by the TRSMT macro. The character definition part of TRSMT must contain the CSECT/ENTRY definitions TRSSCTQC and TRSSCTTC.

Character Recognition

The tokenization logic is implemented with the TRSMSL macro. The tokenization part of TRSSCT must contain the CSECT/ENTRY definitions TRSSCTQL and TRSSCTTL.

Character Translation

The character translation table containing a translated character value for all relevant characters must be defined for all possible 256 EBCDIC characters. The character translation part of TRSSCT must contain the CSECT/ENTRY definitions TRSSCTQT and TRSSCTTT.

TRSSCT Macros

TRSMLT

The TRSMLT macro is used to define all classes of characters which will be used by ADABAS TEXT RETRIEVAL (i.e. ALPH, NUM, etc.).

The TRSMLT macro has the parameter TYPES:

```
TRSMLT TYPES=(class1,class2,class3...)
```

where **class1, class2, class3** denote the user defined classes for character definition. The TRSSCT module must contain one call to TRSMLT only.

TRSMT

The TRSMT macro is used to assign specific characters to one or more of the classes already defined by the TRSMLT macro.

The TRSMT macro has the two parameters CHAR and CLASS:

```
TRSMT CHAR=(c1,c2,c3...), CLASS=class1
```

where **c1, c2, c3...** denote the characters to be assigned to the class in question.

The input of the different characters can either be in character or hexadecimal format. If the input is in hexadecimal format it must start with the letter X and the values must be enclosed in quotation marks (e.g. X'4B'). If the input is in character format the values should be enclosed in quote marks (e.g. 'A').

It is not necessary to define all of the 256 possible EBCDIC characters by the TRSMT macro. Those which are not defined will be regarded as delimiters (usually blanks).

The TRSSCT module can contain as many calls of TRSMT as necessary for the definition of all relevant characters.

TRSMSL

The TRSMSL macro is used to define those actions which are to be taken by ADABAS TEXT RETRIEVAL when the tokenization process comes across a character belonging to a specific class. The TRSMSL macro has the following syntax.

```
TRSMSL  IF=class,GOTO=label,
        ACTION=action, IFMODE=mode,
        IFID=id, TYPE=type,CHAR=c1
```

The parameters of the TRSMSL macro take the following values:

Parameter	Parameter Value
IF	**class:** Denotes the class of characters for which the TRSMSL macro is to be executed. If the parameter IF is omitted, the TRSMSL macro is executed.
GOTO	**label:** Denotes the label in TRSSCT from which program execution is to be continued after the the execution of the current TRSMSL macro. The parameter GOTO can be omitted.
ACTION	**action:** Denotes the action to be taken by TRSSCT for the current character. The parameter ACTION can be omitted. There are six different possible values for the parameter ACTION: **ACCEPT** Character is to be accepted. **SKIP** Charactr is to be ignored. **TRUNCATE** Current character string is to be truncated, thus forming a word. **REPLACE** Character is to be replaced by the character contained in the value **c1** of the parameter CHAR. **INSERT** The character contained in the value c1 of the parameter CHAR is to be inserted in the current character string.
IFMODE	**mode:** Denotes whether a specific action is to be taken during the text inversion process or the query syntax analysis only. The value 'T' denotes that the action is to be taken during the text inversion process and the value 'Q' denotes that the action is to be taken during the query syntax analysis only. The parameter IFMODE can be omitted.
IFID	**id:** One-byte alphanumeric character denoting that a specific action is only to be executed for a specific tokenization process as assigned to certain free text chapters or formatted fields by the DSL call. The parameter IFID can be omitted.
CHAR	**c1:** Denotes a one byte character used together with the actions REPLACE or INSERT. If the value of the parameter ACTION equals REPLACE or INSERT, a value for the parameter CHAR must be supplied; in all other cases the parameter CHAR is ignored.

An Example of TRSSCT

```
TRSSCT   CSECT                                    TRS00010
*                                                 TRS00020
*      CHARACTER RECOGNITION                      TRS00030
*                                                 TRS00040
TRSSCTQL CSECT                                    TRS00050
         ENTRY TRSSCTTL                           TRS00060
TRSSCTTL DS    0H                                 TRS00070
         COPY TRSEQU                              TRS00080
         TRSMLT TYPES=(ALPH,NUMBER,SPEC,DOT,BLANK,SEP,COMM,APOS,EQ,REL) TRS00090
START    TRSMSL IF=SPEC,GOTO=SPEC1                TRS00100
         TRSMSL IF=APOS,GOTO=APOS1                TRS00110
         TRSMSL IF=ALPH,GOTO=SA,ACTION=ACCEPT     TRS00120
         TRSMSL IF=NUMBER,GOTO=SN,ACTION=ACCEPT   TRS00130
         TRSMSL IF=DOT,GOTO=DOT1                  TRS00140
         TRSMSL ACTION=SKIP,GOTO=START            TRS00150
SA       TRSMSL IF=SEP,GOTO=SA,ACTION=SKIP        TRS00160
         TRSMSL IF=ALPH,GOTO=SA,ACTION=ACCEPT     TRS00170
         TRSMSL IF=NUMBER,GOTO=SA,ACTION=ACCEPT   TRS00180
         TRSMSL IF=DOT,GOTO=SA1,ACTION=ACCEPT     TRS00190
         TRSMSL ACTION=TRUNCATE                   TRS00200
SA1      TRSMSL IF=ALPH,GOTO=SA,ACTION=ACCEPT     TRS00210
         TRSMSL IF=NUMBER,GOTO=SA,ACTION=ACCEPT   TRS00220
         TRSMSL ACTION=BACK                       TRS00230
         TRSMSL ACTION=TRUNCATE                   TRS00240
SN       TRSMSL IF=NUMBER,GOTO=SN,ACTION=ACCEPT   TRS00250
         TRSMSL IF=SEP,GOTO=SA,ACTION=SKIP        TRS00260
         TRSMSL IF=ALPH,GOTO=SA,ACTION=ACCEPT     TRS00270
         TRSMSL IF=DOT,GOTO=SN1,ACTION=ACCEPT     TRS00280
         TRSMSL ACTION=TRUNCATE,TYPE=NUMER        TRS00290
SN1      TRSMSL IF=ALPH,GOTO=SA,ACTION=ACCEPT     TRS00300
         TRSMSL IF=NUMBER,GOTO=SN,ACTION=ACCEPT   TRS00310
         TRSMSL ACTION=BACK                       TRS00320
         TRSMSL ACTION=TRUNCATE,TYPE=NUMER        TRS00330
DOT1     TRSMSL IFMODE=Q,GOTO=DOT2                TRS00340
         TRSMSL ACTION=ACCEPT                     TRS00350
         TRSMSL ACTION=TRUNCATE                   TRS00360
DOT2     TRSMSL ACTION=SKIP,GOTO=START            TRS00370
SPEC1    TRSMSL IFMODE=Q,GOTO=SPEC2               TRS00380
         TRSMSL ACTION=SKIP,GOTO=START            TRS00390
SPEC2    TRSMSL IF=REL,GOTO=SPEC3                 TRS00400
         TRSMSL ACTION=ACCEPT                     TRS00410
         TRSMSL ACTION=TRUNCATE                   TRS00420
SPEC3    TRSMSL ACTION=ACCEPT                     TRS00430
         TRSMSL IF=EQ,ACTION=ACCEPT               TRS00440
         TRSMSL IF=REL,ACTION=ACCEPT              TRS00450
         TRSMSL ACTION=TRUNCATE                   TRS00460
APOS1    TRSMSL IFMODE=Q,GOTO=APOS1A              TRS00470
         TRSMSL ACTION=SKIP,GOTO=START            TRS00480
APOS1A   TRSMSL ACTION=SKIP                       TRS00490
APOS1B   TRSMSL IF=APOS,ACTION=SKIP,GOTO=APOS2    TRS00500
         TRSMSL ACTION=ACCEPT,GOTO=APOS1B         TRS00510
APOS2    TRSMSL ACTION=TRUNCATE,TYPE=STRING       TRS00520
         LTORG                                    TRS00530
*                                                 TRS00540
*      CHARACTER DEFINITION                       TRS00550
*                                                 TRS00560
TRSSCTQC CSECT                                    TRS00570
         ENTRY TRSSCTTC                           TRS00580
```

```
TRSSCTTC DS    0H                                          TRS00590
      TRSMT  CHAR=(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P),CLASS=ALPH      TRS00600
      TRSMT  CHAR=(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p),CLASS=ALPH      TRS00610
      TRSMT  CHAR=(Q,R,S,T,V,U,W,X,Y,Z),CLASS=ALPH                  TRS00620
      TRSMT  CHAR=(q,r,s,t,v,u,w,x,y,z),CLASS=ALPH                  TRS00630
      TRSMT  CHAR=('$','#','&&','*'),CLASS=ALPH                     TRS00640
      TRSMT  CHAR=(1,2,3,4,5,6,7,8,9,0),CLASS=NUMBER                TRS00650
      TRSMT  CHAR=('-'),CLASS=SEP                                   TRS00660
      TRSMT  CHAR=('!','?','.'),CLASS=DOT                           TRS00670
      TRSMT  CHAR=(' '),CLASS=BLANK                                 TRS00680
      TRSMT  CHAR=('('),CLASS=SPEC                                  TRS00690
      TRSMT  CHAR=(')'),CLASS=SPEC                                  TRS00700
      TRSMT  CHAR=('='),CLASS=SPEC+EQ                               TRS00710
      TRSMT  CHAR=('<','>'),CLASS=SPEC+REL                          TRS00720
      TRSMT  CHAR=(','),CLASS=SPEC+COMM                             TRS00730
      TRSMT  CHAR=(''''),CLASS=APOS                                 TRS00740
      TRSMT  CHAR=(X'4A',X'E0',X'5A'),CLASS=ALPH                    TRS00750
      TRSMT  CHAR=(X'C0',X'6A',X'D0'),CLASS=ALPH                    TRS00760
      TRSMT  CHAR=(X'A1'),CLASS=ALPH                                TRS00770
      LTORG                                                         TRS00780
*                                                                   TRS00790
*      CHARACTER TRANSLATION                                        TRS00800
*                                                                   TRS00810
TRSSCTQT CSECT                                                      TRS00820
      ENTRY TRSSCTTT                                                TRS00830
TRSSCTTT DS    0H                                                   TRS00840
TRTAB    DC    256AL1(*-TRTAB)                                      TRS00850
      ORG   TRTAB+C'a'                                              TRS00860
      DC    C'ABCDEFGHI'                                            TRS00870
      ORG   TRTAB+C'j'                                              TRS00880
      DC    C'JKLMNOPQR'                                            TRS00890
      ORG   TRTAB+C's'                                              TRS00900
      DC    C'STUVWXYZ'                                             TRS00910
      ORG   TRTAB+X'C0'                                             TRS00920
      DC    X'4A'                                                   TRS00930
      ORG   TRTAB+X'6A'                                             TRS00940
      DC    X'E0'                                                   TRS00950
      ORG   TRTAB+X'D0'                                             TRS00960
      DC    X'5A'                                                   TRS00970
      ORG                                                           TRS00980
      END                                                           TRS00990
```

Keyword Definition

The system keywords of ADABAS TEXT RETRIEVAL are defined in the module TRSTEXT.

You can change but should under no circumstances remove any keyword entries made by the macro TRSMKEY. You can assign additional values to all keywords by the macro TRSMALT.

An Example of TRSTEXT

```
TRS     TITLE 'TEXT RETRIEVAL SPECIAL TEXTS. MODIFIABLE SECTION.'     TRS00010
TRSTEXT  CSECT                                                        TRS00020
*********************************************************************       TRS00030
*                                   *      TRS00040
*   THE FOLLOWING TEXTS MAY BE MODIFIED, BUT NOT ELIMINATED.  *      TRS00050
*                                   *      TRS00060
*********************************************************************       TRS00070
TXTESEN  TRSMKEY  '. '              END OF SENTENCE           TRS00080
      TRSMALT  '? '                          TRS00090
      TRSMALT  '! '                          TRS00100
TXTEPAR  TRSMKEY  '$$ '             END OF PARAGRAPH          TRS00110
TXTFEQU  TRSMKEY  '='               EQUAL                  TRS00120
TXTFSYN  TRSMKEY  'SYN'               SYNONYMS               TRS00130
TXTFSYR  TRSMKEY  'SYR'               SYN-ROOT               TRS00140
TXTFPHO  TRSMKEY  'PHONETIC'          PHONETIC               TRS00150
TXTFCAT  TRSMKEY  'ASPECT'            ASPECT                 TRS00160
      TRSMALT  'CATEGORY'           CATEGORY               TRS00170
TXTFGRP  TRSMKEY  'GROUP'             GROUP                  TRS00180
TXTFROT  TRSMKEY  'ROOT'              ROOT                   TRS00190
TXTFFA1  TRSMKEY  'FUNC1'             USER FUNCTION 1           TRS00200
TXTFFA2  TRSMKEY  'FUNC2'             USER FUNCTION 2           TRS00210
TXTFFA3  TRSMKEY  'FUNC3'             USER FUNCTION 3           TRS00220
TXTFFA4  TRSMKEY  'FUNC4'             USER FUNCTION 4           TRS00230
TXTFFA5  TRSMKEY  'FUNC5'             USER FUNCTION 5           TRS00230
TXTFFA6  TRSMKEY  'FUNC6'             USER FUNCTION 6           TRS00230
TXTFFA7  TRSMKEY  'FUNC7'             USER FUNCTION 7           TRS00230
TXTFFA8  TRSMKEY  'FUNC8'             USER FUNCTION 8           TRS00230
TXTFFA9  TRSMKEY  'FUNC9'             USER FUNCTION 9           TRS00230
TXTFFA0  TRSMKEY  'FUNC0'             USER FUNCTION 0           TRS00230
TXTFTHNT TRSMKEY  'DOWN'
TXTFTHBT TRSMKEY  'UP'
TXTFTHSY TRSMKEY  'SY'
TXTOLEP  TRSMKEY  '('               LEFT PARENTHSIS           TRS00240
TXTORIP  TRSMKEY  ')'               RIGHT PARENTHSIS          TRS00250
TXTMCOM  TRSMKEY  ','                 COMMA                  TRS00260
TXTSORA  TRSMKEY  'SORT'              SORT BY                TRS00270
TXTSORD  TRSMKEY  'SORTD'             SORT BY                TRS00280
TXTOOR   TRSMKEY  'OR'                OR                  TRS00290
TXTOAND  TRSMKEY  'AND'               AND                    TRS00300
TXTONOT  TRSMKEY  'NOT'               NOT                    TRS00310
TXTOADJ  TRSMKEY  'ADJ'               ADJ                 TRS00320
TXTONEA  TRSMKEY  'NEAR'              NEAR                   TRS00330
TXTOISEN TRSMKEY  'INSEN'             SENTENCE               TRS00340
TXTOIPAR TRSMKEY  'INPAR'             PARAGRAPH              TRS00350
TXTRLEQ  TRSMKEY  'EQ'              EQUAL                  TRS00360
```

```
      TRSMALT '='                          TRS00370
TXTRLGT TRSMKEY 'GT'          GREATER      TRS00380
      TRSMALT '>'                          TRS00390
      TRSMALT '>>'                         TRS00400
TXTRLLT TRSMKEY 'LT'          LESS THAN    TRS00410
      TRSMALT '<'                          TRS00420
      TRSMALT '<<'                         TRS00430
TXTRLGE TRSMKEY 'GE'          GREATER EQUAL TRS00440
      TRSMALT '>='                         TRS00450
TXTRLLE TRSMKEY 'LE'          LESS OR EQUAL TRS00460
      TRSMALT '<='                         TRS00470
TXTRLBT TRSMKEY 'BETWEEN'     BETWEEN      TRS00480
      TRSMALT '<>'                         TRS00490
      TRSMALT '><'                         TRS00500
      TRSMFIN                              TRS00510
      END                                  TRS00520
```

CHAPTER 6

NATURAL TEXT RETRIEVAL INTERFACE

THE NATURAL TEXT RETRIEVAL INTERFACE

The NATURAL text retrieval interface enables ADABAS TEXT RETRIEVAL to be used with the aid of normal NATURAL statements.

The following NATURAL statements are supported by the NATURAL interface:

- STORE
- FIND
- DELETE

When the interface is to be used, the document file and the document index file must be the same physical file. The DB ID and the file number of the vocabulary file are entered to the NATURAL session either by a NTFILE macro or the LFILE parameter; the logical file ID to be used is 238.

The relation between a field containing text and the hyperdescriptor containing its free text index is indicated by entering the ADABAS field name of the field containing the text as first entry in the list of the parent fields of the appropriate hyperdescriptor.

Example:

```
FNDEF='01,T1,72,A,MU,NU'
HYPDE='12,Y1,3,B,MU,NU=T1,D0,D1,D3'
```

The hyperdescriptor Y1 will contain the document index values generated during the inversion of the text entered in the field T1.

Use of the interface requires that a logical DBID be established by means of the NTDB macro. In NATURAL 2.1 it must be of type USER. In NATURAL 2.2 it must be of type TRS. This logical DBID must be contained in the DDMs used in the NATURAL programs. The physical DBID will be taken either from the NTFILE macro or the LFILE parameter with the logical ID 238.

The NATURAL STORE Statement

If a STORE statement is executed which is directed to the interface, the interface examines whether one of the fields, contained in the current view is to be free text inverted in accordance with the definition of the document indices (hyperdescriptor).

If such fields are found, the free text inversion is immediately executed.

The WITH NUMBER clause of the NATURAL STORE statement is used to support the inversion of long documents.

If the ISN provided in the WITH NUMBER clause points to a previously stored document index entry, the index of this record is extended by the index entries made up during the inversion of the text contained in the record currently stored.

The NATURAL DELETE Statement

When deleting a record all corresponding document index entries are also deleted.

The NATURAL FIND Statement

The NATURAL FIND statement can be used to execute retrieval operations on previously inverted texts. All sub-clauses of the NATURAL FIND statement are supported.

The NATURAL name of the relevant hyperdescriptor must be used to enter selection criteria. Any type of query can be entered as a value for a hyperdescriptor.

Setting Dynamic Parameters

When using the interface, it is also possible to set TRS dynamic parameters (TRS parameters are explained in Chapter 2, **ADABAS TEXT RETRIEVAL Calls**). This is done using the NATURAL PROCESS statement.

The DDM to be used in the NATURAL PROCESS statement is called TRS - SYSTEM and is contained in the ADABAS TEXT RETRIEVAL INPL dataset. Before issuing a PROCESS call, the database ID in the DDM must be adjusted to the logical database ID addressing the interface and the file number to the physical file number of the vocabulary file in question.

The names of the variables representing the different parameters correspond to the names of the parameters themselves. The values of the required parameters must be supplied according to the syntax of the PROCESS statement.

Example:

```
PROCESS TRS - SYSTEM USING SETCHAR='*', WORDLEN=64
```

In addition to the dynamic parameters described in chapter 4, there are two other parameters to be used in connection with the interface:

HIGH	Turns highlighting on or off. Possible values: ON or OFF. Default value: ON
HIGHCHAR	Hexadecimal values for the indication of the beginning and end of the string to be highlighted. Four characters must be supplied. The first two characters indicate the beginning of the string to be highlighted. The last two characters indicate the end of the string to be highlighted. Default: 4C6E (‘<>’).

Example:

```
PROCESS TRS - SYSTEM USING HIGH='ON', HIGHCHAR='FEFF'
```

Scrolling Through RETAIN Sets

The NATURAL text retrieval interface offers an additional feature for scrolling through NATURAL RETAIN sets. It allows forward and backward scrolling and positioning within a set.

To use this feature, two fields must be added to the DDM of the file in question. One field denotes the RETAIN set requested; the other field specifies the start position for scrolling. The fields must have the following format:

L	DB	NAME	F	LENGTH	D
1	Z0	SET	A	32	D
1	Z1	POSITION	N	8	D

These fields need not be physically present on the ADABAS file.

Example

```
FIND NUMBER
     DOCUMENT FILE   TEXT='COMPUTER'
                     RETAIN AS 'LIST1'

FIND DOCUMENT FILE   SET='LIST1'
                     AND POSITION=1 THRU 10

DISPLAY ...

END FIND
```

The first ten objects contained in set LIST1 are presented in ascending ISN sequence. If backward scrolling is required, the values for the POSITION parameter must be reversed (POSITION= 10 THRU 1).

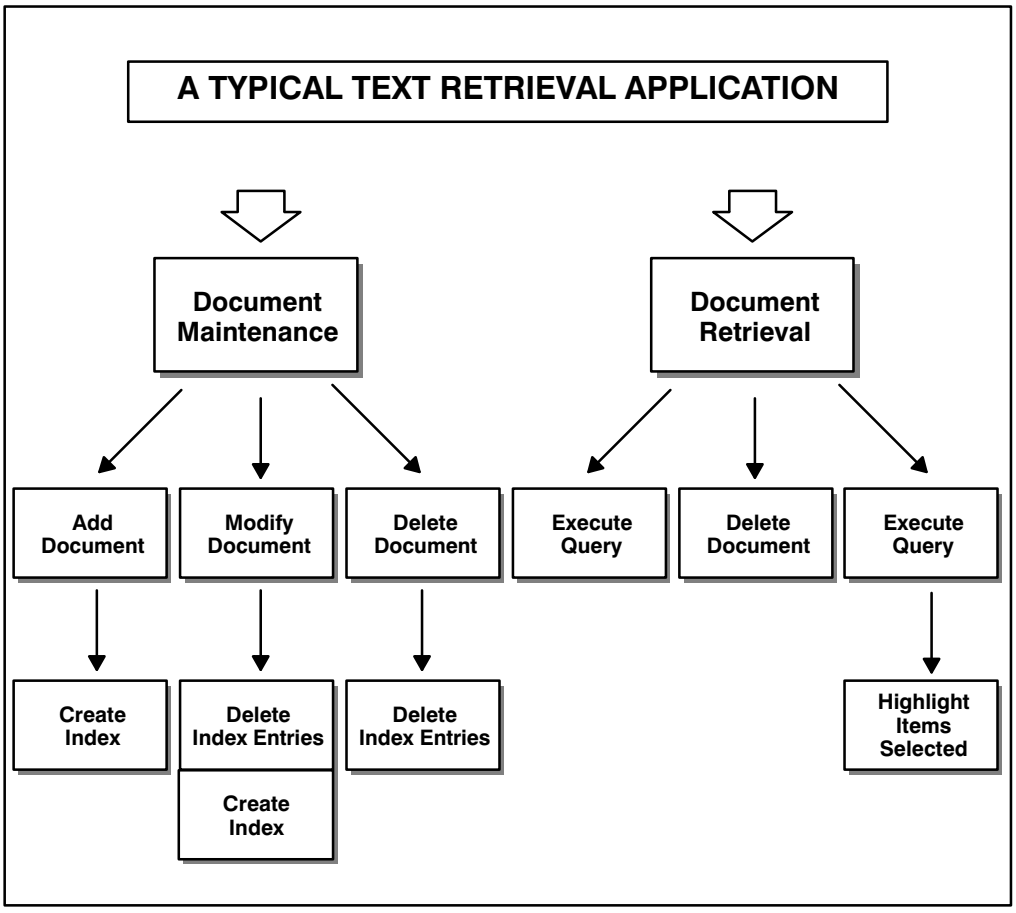
CHAPTER 7

SAMPLE APPLICATION

SAMPLE APPLICATION

This chapter demonstrates the use of ADABAS TEXT RETRIEVAL calls in the context of a sample application. This sample application is written in NATURAL and is contained as a NATURAL INPL on the ADABAS TEXT RETRIEVAL installation tape.

A retrieval application typically consists of the functions illustrated below:

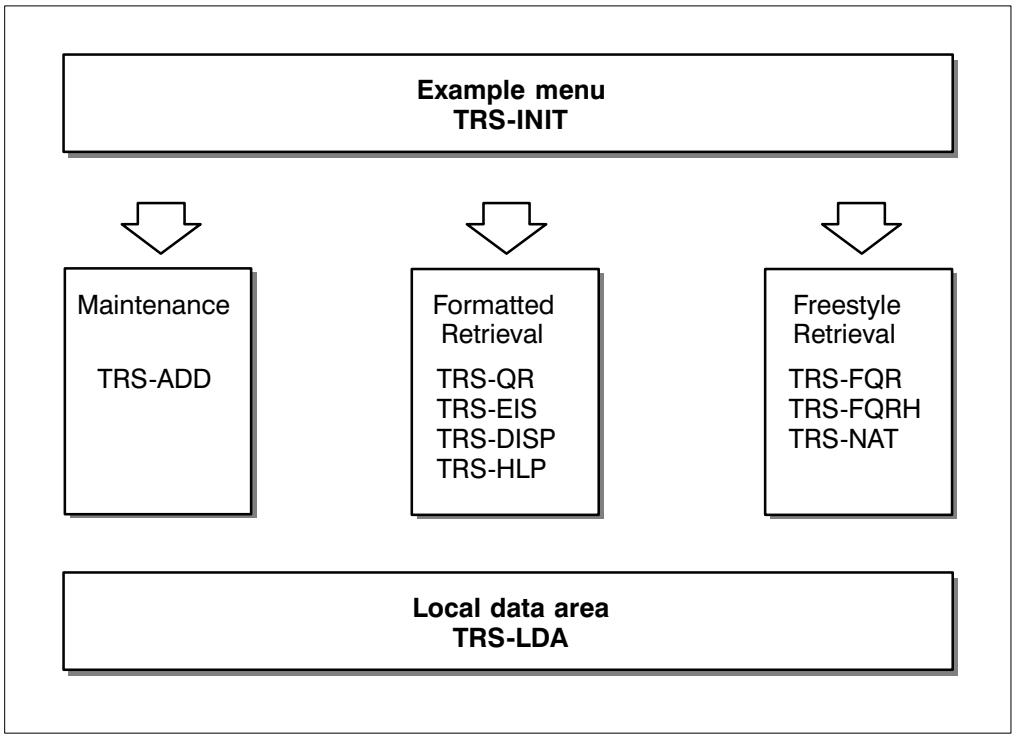


The sample application explained in this chapter is a small text retrieval system used to retrieve information on Software AG's product documentation. It enables the maintenance and retrieval of documents.

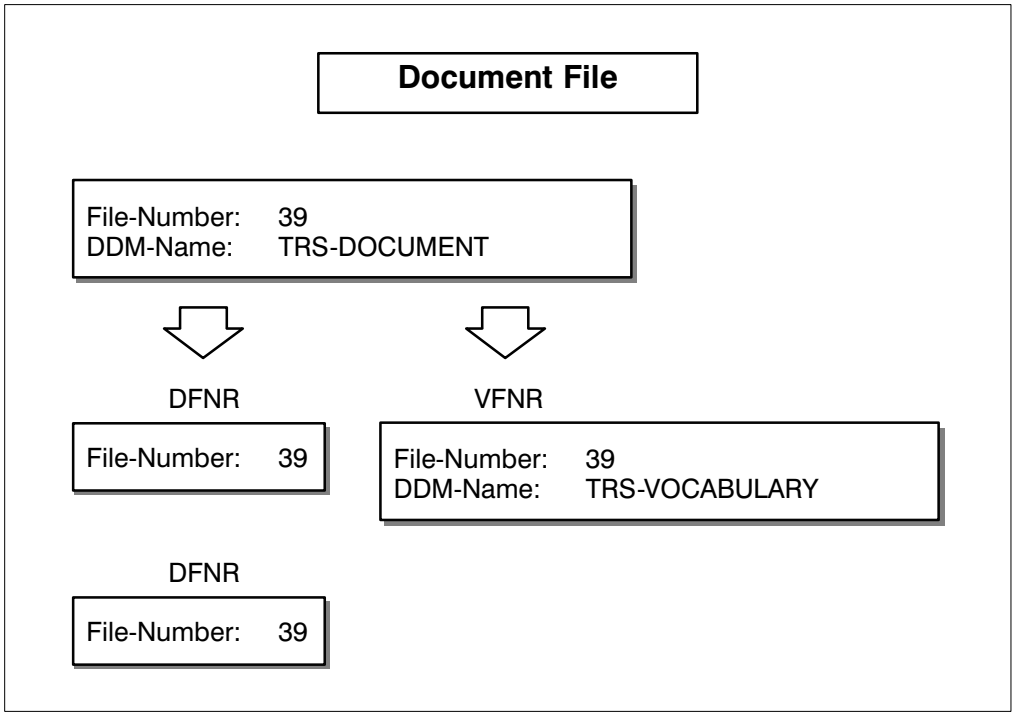
The following main functions are implemented:

- **Maintenance.** Store, update, delete documents;
- **Formatted retrieval.** Query, select, display documents;
- **Freestyle retrieval.** Query, display, overview.

The following figure provides an overview of the sample application:



The following diagram provides an overview of the sample application file structure:



To implement the sample application, you must adjust the physical file numbers to the values chosen by you when installing the sample application files in the ADABAS database:

- Change the file numbers of the DDMs TRS - DOCUMENT and TRS - VOCABULARY;
- Change the file numbers used for the ADABAS TEXT RETRIEVAL BC call in the NATURAL program TRS - INIT (statement number 0330 , page 7 - 6);
- Recatalog all NATURAL objects with the NATURAL CATALL command.

The Calls Used in the Sample Application

The following table alphabetically lists the calls used in the sample application:

Call	Primary Application	Contained in Sample Programs
ADD	Invert a document	**TRS - ADD**
BC	Start an ADABAS TEXT RETRIEVAL session	**TRS - INIT**
CL	Close an ADABAS TEXT RETRIEVAL session	**TRS - INIT**
DDS	Delete document index entries	**TRS - ADD**
DSL	Define search label	**TRS - INIT**
DYP	Change dynamic parameters	**TRS - ADD** **TRS - DISP**
EISE	End browsing through an ISN - Set	**TRS - EIS**
EISG	Start browsing through an ISN - Set	**TRS - EIS**
EISS	Browse through an ISN - Set	**TRS - EIS**
HIGH	Highlight a document	**TRS - DISP**
QR	Execute a query	**TRS - HLP** **TRS - FQR** **TRS - QR**
RET	Create a NATURAL Retain - Set	**TRS - FQR**
RQR	Release a query	**TRS - FQR** **TRS - QR**

Initialize ADABAS TEXT RETRIEVAL Session

TRS - INIT

The program TRS - INIT initializes an ADABAS TEXT RETRIEVAL session and invokes the required function (maintenance, retrieval, freestyle retrieval). The program is broken down into the following functional segments.

- Initialize program;
- Set to lower case;
- Initialize the ADABAS TEXT RETRIEVAL Session (BC call);
- Define Search Labels (DSL call);
- Display Menu and Invoke Selected Functions;
- Close the ADABAS TEXT RETRIEVAL Session (CL call).

Program Start Up

```
0010   ************************************************************************
0020 *                                                                      *
0030 *  ADABAS TEXT RETRIEVAL  Example Application                          *
0040 *                                                                      *
0050 *  Object   :  TRS-INIT                                               *
0060 *  Type     :  Program                                                 *
0070 *  Function :  Initialize TRS session                                 *
0080 *  Author   :  Software AG                                            *
0090 *                                                                      *
0100   ************************************************************************
0110 *
0120 DEFINE DATA LOCAL USING TRS-LDA
0130          LOCAL
0140 1 SEL (A1)
0150 END-DEFINE
0160 *
```

Use Lowercase letters

```
0170 * ------------------------------------------------------------------- *
0180 *  Set lower case                                                     *
0190 * ------------------------------------------------------------------- *
0200 SET CONTROL 'L'
0210 *
```

Initialize the ADABAS TEXT RETRIEVAL Session

Note: It is recommended that you execute a CL call prior to a BC call to ensure that the existing session is closed.

The BC call in line 0330 is used to initialize an ADABAS TEXT RETRIEVAL session.

The TRS.DYP - PARM parameter contains all relevant ADABAS TEXT RETRIEVAL start-up parameters for that specific session.

```
0220 * ------------------------------------------------------------------------ *
0230 *  Initialize TRS session                                  *
0240 * ------------------------------------------------------------------------ *
0250 *
0260 *
0270 CALL 'TRS' 'CL' TRS.RC TRS.SAVE
0280 *
0290 *
0300 MOVE 'VFNR=38,DFNR=39,DSFNR=39,DOCID=DA,SETCHAR=#.' TO TRS.DYP-PARM
0310 *
0320 *
0330 CALL 'TRS' 'BC' TRS.RC TRS.SIZE TRS.SAVE TRS.DYP-PARM
0340 *
0350 *
0360 IF TRS.RC NE 0
0370    MOVE TRS.RC TO TRS.RC1
0380 *
0390 *
0400    CALL 'TRS' 'CL' TRS.RC TRS.SAVE
0410 *
0420 *
0430    WRITE  'Error in TRS-BC:' TRS.RC1
0440    STOP
0450 END-IF
0460 *
```

Define Search Labels

In order to use logical names for the retrieval of free-text chapters and formatted fields in the document structure, search labels are assigned to these fields using the DSL call (see line 0530). The parameter TRS.DSL - PARM must contain all necessary search label definitions.

The logical name #TITLE is assigned to the free-text chapter hyperdescriptor Y1Y1 and the logical name #ABSTRACT to Y2Y2 (see line 0500).

The logical name #DATE is assigned to the formatted field represented by the ADABAS descriptor DB and the logical name #ORDER to DA (see line 0500).

These logical names can be used for processing queries which access the requested fields in the document structure.

```
0470 * ------------------------------------------------------------------ *
0480 *  Define Search-Labels                                              *
0490 * ------------------------------------------------------------------ *
0500 MOVE 'Y1Y1=#TITLE,Y2Y2=#ABSTRACT,DB=#DATE,DA=#ORDER.' TO TRS.DSL-PARM
0510 *
0520 *
0530 CALL 'TRS' 'DSL' TRS.RC TRS.DSL-PARM
0540 *
0550 *
0560 IF TRS.RC NE 0
0570    MOVE TRS.RC TO TRS.RC1
0580    CALL 'TRS' 'CL' TRS.RC TRS.SAVE
0590    WRITE  'Error in TRS-DSL:' TRS.RC1
0600    STOP
0610 END-IF
0620 *
0630 SET KEY ALL
0640 SET KEY PF3 NAMED 'Quit'
0650 *
```

Display Menu and Invoke Selected Functions

A menu is displayed for selecting a desired function. The following functions are available:

A Document Maintenance (store, update, delete);

R Formatted Retrieval;

F Freestyle Retrieval.

```
0660 * ------------------------------------------------------------------- *
0670 *  Display Menu                                  *
0680 * ------------------------------------------------------------------- *
0690 REPEAT
0700  INPUT USING MAP 'TRS-MENU'
0710  IF *PF-KEY = 'PF3'
0720    ESCAPE BOTTOM
0730  END-IF
0740  DECIDE ON FIRST VALUE SEL
0750    VALUE 'A'
0760      FETCH 'TRS-ADD'
0770    VALUE 'R'
0780      FETCH 'TRS-QR'
0790    VALUE 'F'
0800      FETCH 'TRS-FQR'
0810    VALUE '.'
0820      ESCAPE BOTTOM
0830    NONE
0840      REINPUT 'Invalid function code.'
0850  END-DECIDE
0860 END-REPEAT
0870 *
```

Close the ADABAS TEXT RETRIEVAL Session

The CL call in line 0930 closes the ADABAS TEXT RETRIEVAL session.

```
0880 * ------------------------------------------------------------------- *
0890 *  Close TRS session                             *
0900 * ------------------------------------------------------------------- *
0910 *
0920 *
0930 CALL 'TRS' 'CL' TRS.RC TRS.SAVE
0940 *
0950 *
0960 SET CONTROL 'U'
0970 *
0980 END
```

Document Maintenance and Retrieval

TRS - ADD

The program TRS - ADD enables the user to perform the following functions.

- Select a document for update/delete;
- Store a document;
- Update a document;
- Invert a document (ADD call);
- Delete a document;
- Delete document index entries (DDS call).

Program Start Up

```
0010   ************************************************************************
0020 *                                                                      *
0030 *  ADABAS TEXT RETRIEVAL  Example Application                          *
0040 *                                                                      *
0050 *  Object   :  TRS-ADD                                                *
0060 *  Type     :  Program                                                 *
0070 *  Function :  Store, update, delete and invert document              *
0080 *  Author   :  Software AG                                            *
0090 *                                                                      *
0100   ************************************************************************
0110 *
0120 DEFINE DATA LOCAL USING TRS-LDA         /* TRS Parameter
0130 *
0140 LOCAL
0150 *
0160 1 MAP1                          /* Fields in Map
0170   2 ORDER    (A16)
0180   2 DATE     (N8)
0190   2 PRICE    (N3)
0200   2 TITLE    (A70)
0210   2 ABSTRACT (A70/12)
0220 *
0230 1 DOCUMENT VIEW OF TRS-DOCUMENT         /* Document View
0240   2 ORDER    (A16)
0250   2 DATE
0260   2 PRICE    (N3)
0270   2 TITLE
0280   2 ABSTRACT (12)
0290 *
0300 1 #ORDER-OLD (A16)                  /* Work Fields
0310 1 #MSG       (A72)
0320 *
0330 END-DEFINE
0340 *
```

Set Keys

```
0350 * ------------------------------------------------------------------ *
0360 * Set keys
0370 * ------------------------------------------------------------------ *
0380 SET KEY ALL
0390 SET KEY PF2 NAMED 'Clear'
0400 SET KEY PF3 NAMED 'Quit'
0410 SET KEY PF4 NAMED 'Update'
0420 SET KEY PF5 NAMED 'Delete'
0430 SET KEY PF6 NAMED 'Store'
0440 SET KEY PF8 NAMED 'Next '
0450 SET KEY PF10 NAMED 'Query'
0460 *
0470 *
0480 R1.                          /* Main Loop
0490 REPEAT
0500 *
0510 *
0520   INPUT WITH TEXT #MSG USING MAP 'TRS-ADDM'
0530 *
```

Terminate

```
0540 * ------------------------------------------------------------------ *
0550 * Quit Function
0560 * ------------------------------------------------------------------ *
0570   IF *PF-KEY = 'PF3'
0580     ESCAPE BOTTOM
0590   END-IF
0600 *
```

Invoke Formatted Retrieval

```
0610 * ------------------------------------------------------------------ *
0620 * Invoke Query
0630 * ------------------------------------------------------------------ *
0640   IF *PF-KEY = 'PF10'
0650     FETCH 'TRS-QR'
0660   END-IF
0670 *
```

Clear Screen

```
0680 * ------------------------------------------------------------------ *
0690 * Clear screen
0700 * ------------------------------------------------------------------ *
0710   IF *PF-KEY = 'PF2'
0720     RESET MAP1
0730     MOVE 'Input data and press enter.' TO #MSG
0740     ESCAPE TOP
0750   END-IF
0760 *
```

Next Document

```
0770 * ------------------------------------------------------------------ *
0780 * Next document
0790 * ------------------------------------------------------------------ *
0800   IF *PF-KEY = 'PF8'
0810     READ DOCUMENT BY ORDER = MAP1.ORDER
0820       IF MAP1.ORDER NE DOCUMENT.ORDER
0830         ESCAPE BOTTOM
0840       END-IF
0850     END-READ
0860     MOVE DOCUMENT.ORDER TO MAP1.ORDER
0870   END-IF
0880 *
0890 *
0900   IF MAP1.ORDER = ' '
0910     REINPUT 'Please enter Order Nr.'
0920   END-IF
0930 *
```

Select Document for STORE/UPDATE/DELETE

```
0940 * ------------------------------------------------------------------- *
0950 * Select document
0960 * ------------------------------------------------------------------- *
0970   IF *PF-KEY = 'ENTR' OR *PF-KEY = 'PF8'
0980     IF MAP1.ORDER = #ORDER-OLD
0990       ESCAPE TOP
01000      END-IF
01010 *
01020      F1.
01030      FIND (1) DOCUMENT WITH ORDER = MAP1.ORDER
01040        MOVE BY NAME DOCUMENT TO MAP1
01050        MOVE MAP1.ORDER TO #ORDER-OLD
01060      END-FIND
01070 *
01080      IF *NUMBER (F1.) > 0
01090        COMPRESS 'Order-Nr.:' MAP1.ORDER 'found.' INTO #MSG
01100      ELSE
01110        COMPRESS 'Order-Nr.:' MAP1.ORDER 'not found.' INTO #MSG
01120      END-IF
01130      ESCAPE TOP
01140 *
01150    END-IF
01160 *
```

Store Document

In this program segment, the document is stored in the document file and inverted by ADABAS TEXT RETRIEVAL. The document is inverted using the subroutine SR - INVERT specified in line 1300. This subroutine begins in line 2020.

```
01170 * ------------------------------------------------------------------- *
01180 * Store document
01190 * ------------------------------------------------------------------- *
01200   IF *PF-KEY = 'PF6'
01210      FIND (1) DOCUMENT WITH ORDER = MAP1.ORDER
01220        IF *NUMBER > 0
01230           COMPRESS 'Document' MAP1.ORDER 'already exists.' INTO #MSG
01240           REINPUT #MSG MARK *MAP1.ORDER
01250        END-IF
01260      END-FIND
01270      MOVE BY NAME MAP1 TO DOCUMENT
01280      STORE DOCUMENT
01290 *
01300      PERFORM SR-INVERT              /* Invoke Inversion
01310 *
01320      END TRANSACTION
01330      COMPRESS 'Order-Nr.:' MAP1.ORDER 'successfuly added.' INTO #MSG
01340      RESET #ORDER-OLD
01350    END-IF
01360 *
```

Update Document

The following program segment updates the document in the document file and re-inverts the document. The document is inverted using the subroutine SR - INVERT specified in line 1540. This subroutine begins in line 2020.

```
01370 * ------------------------------------------------------------------- *
01380 * Update document
01390 * ------------------------------------------------------------------- *
01400  IF *PF-KEY = 'PF4'
01410    IF MAP1.ORDER NE #ORDER-OLD
01420      REINPUT 'No change of Order-Nr. allowed. for update'
01430        MARK *MAP1.ORDER
01440    END-IF
01450    IF MAP1.DATE NE MASK(YYYYMMDD)
01460      REINPUT 'Please correct date.' MARK *MAP1.DATE
01470    END-IF
01480    FIND (1) DOCUMENT WITH ORDER = MAP1.ORDER
01490      MOVE BY NAME MAP1 TO DOCUMENT
01500 *
01510      UPDATE                    /* Invoke Inversion
01520 *
01530      RESET #ORDER-OLD
01540      PERFORM SR-INVERT
01550      END OF TRANSACTION
01560      COMPRESS 'Order-Nr.:' MAP1.ORDER 'successfuly updated' INTO #MSG
01570    END-FIND
01580 *
01590  END-IF
01600 *
```

Delete Document

The documents are deleted from the document file. The selection of the relevant input entries is carried out in subroutine SR - DELETE specified in line 1830. This subroutine begins in line 2570.

```
01610 * ------------------------------------------------------------------- *
01620 * Delete document
01630 * ------------------------------------------------------------------- *
01640  IF *PF-KEY = 'PF5'
01650    F2.
01660    FIND DOCUMENT ORDER = MAP1.ORDER
01670      SET CONTROL 'WFL70C7B10/10'
01680      INPUT 'PLease retype Order-Nr.:'  #ORDER-OLD (AD=T'_')
01690      SET CONTROL 'WB'
01700      IF #ORDER-OLD NE MAP1.ORDER
01710        MOVE 'No record deleted.' TO #MSG
01720        ESCAPE BOTTOM
01730      END-IF
01740      DELETE
01750    END-FIND
01760 *
01770    IF *NUMBER (F2.) = 0
01780      BACKOUT TRANSACTION
01790      COMPRESS 'Document' MAP1.ORDER 'doesn"t exists.' INTO #MSG
01800      REINPUT #MSG MARK *MAP1.ORDER
01810    END-IF
01820 *
01830    PERFORM SR-DELETE              /* Invoke Delete Index
01840 *
01850    RESET MAP1
01860    END OF TRANSACTION
01870    COMPRESS 'Order-Nr.:' MAP1.ORDER 'successfuly deleted.' INTO #MSG
01880    RESET #ORDER-OLD
01890    ESCAPE TOP
01900  END-IF
01910 *
01920 *
01930 END-REPEAT
```

Invert Document

The SR-INVERT subroutine inverts the contents of the two free-text chapters TITLE and ABSTRACT.

Before the inversion by the ADD call, the TEXT parameter must be set to the name of the ADABAS hyperdescriptor (Y1) which represents the free-text chapter TITLE in the document file. This is carried out by the DYP call in line 2120..

The ADD call is then executed in line 2180 which performs the inversion. Within the ADD call:

- The field MAP1.ORDER contains the current value of the document ID;
- The field TRS.ALEN contains the length of the text to be inverted, in this case 72 bytes;
- The field MAP1.TITLE(1) contains the one and only text line of the free-text chapter TITLE;
- The field TRS.DISN will contain the ISN assigned to the DFNR record by TRS;
- The constant, LAST, indicates that there are no subsequent parts of the free-text chapter to be inverted.

The procedure for inverting the chapter ABSTRACT is identical to that for TITLE above (see lines 2370 and 2430).

Bearing in mind that the name of the ADABAS hyperdescriptor for ABSTRACT is Y2, the possible length of the free-text can be up to 864 bytes.

```
01940   ************************************************************************
01950   ************************************************************************
01960 *** S u b r o u t i n e s ********************************************
01970   ************************************************************************
01980   ************************************************************************
01990 *
02000 *
02010   ************************************************************************
02020 DEFINE SUBROUTINE SR-INVERT    /* Inversion Process for Document
02030   ************************************************************************
02040 *
02050 * ------------------------------------------------------------------ *
02060 * TRS-ADD   =====  Inversion for Document Chapter - TITLE
02070 * ------------------------------------------------------------------ *
02080 *
02090   MOVE 'TEXT=Y1Y1.' TO TRS.DYP-PARM
02100 *
02110 *
02120   CALL 'TRS' 'DYP' TRS.RC TRS.DYP-PARM
02130 *
02140 *
02150   MOVE 68 TO TRS.ALEN
02160 *
02170 *
02180   CALL 'TRS' 'ADD' TRS.RC MAP1.ORDER TRS.ALEN MAP1.TITLE
02190                TRS.DISN 'LAST  '
02200 *
```

```
02210 *
02220  IF TRS.RC NE 0
02230    MOVE TRS.RC TO TRS.RC1
02240    BACKOUT TRANSACTION
02250    COMPRESS 'Error in TRS-ADD (Title) =>' TRS.RC1 INTO #MSG
02260    REINPUT #MSG
02270  END-IF
02280 *
02290 *
02300 * ------------------------------------------------------------------ *
02310 * TRS-ADD    =====   Inversion for Document Chapter - ABSTRACT
02320 * ------------------------------------------------------------------ *
02330 *
02340  MOVE 'TEXT=Y2Y2.' TO TRS.DYP-PARM
02350 *
02360 *
02370  CALL 'TRS' 'DYP' TRS.RC TRS.DYP-PARM
02380 *
02390 *
02400  MOVE 816 TO TRS.ALEN
02410 *
02420 *
02430  CALL 'TRS' 'ADD' TRS.RC MAP1.ORDER TRS.ALEN MAP1.ABSTRACT(1)
02440               TRS.DISN 'LAST  '
02450 *
02460 *
02470  IF TRS.RC NE 0
02480    MOVE TRS.RC TO TRS.RC1
02490    BACKOUT TRANSACTION
02500    COMPRESS 'Error in TRS-ADD (Abstract) =>' TRS.RC1 INTO #MSG
02510    REINPUT #MSG
02520  END-IF
02530 *
02540 END-SUBROUTINE
02550 *
02560 *
```

Delete Document Index Entries

Within the SR - DELETE subroutine, the DDS call is invoked in order to remove index entries from the document index file (DSFNR).

The index entries for the free-text chapters TITLE and ABSTRACT must be deleted separately.

The deletion process takes place for ABSTRACT in lines 2670 and 2700 and for TITLE in lines 2860 and 2890. Prior to the invocation of the DDS call, the TEXT parameter must be set to the name of the ADABAS hyperdescriptor representing the free-text chapter in question (see lines 2640 and 2830).

The field MAP1.ORDER contains the value of the document ID.

```
02570   ************************************************************************
02580 DEFINE SUBROUTINE SR-DELETE    /* Delete Document Index
02590   ************************************************************************
02600 *
02610 * ------------------------------------------------------------------ *
02620 * TRS-DDS   =====   Delete Index for Document Chapter - TITLE
02630 * ------------------------------------------------------------------ *
02640   MOVE 'TEXT=Y1Y1.' TO TRS.DYP-PARM
02650 *
02660 *
02670   CALL 'TRS' 'DYP' TRS.RC TRS.DYP-PARM
02680 *
02690 *
02700   CALL 'TRS' 'DDS' TRS.RC MAP1.ORDER 'SUM'
02710 *
02720 *
02730   IF TRS.RC NE 0
02740     MOVE TRS.RC TO TRS.RC1
02750     BACKOUT TRANSACTION
02760     COMPRESS 'Error in TRS-DDS (Title) =>' TRS.RC1 INTO #MSG
02770     REINPUT #MSG
02780   END-IF
02790 *
```

```
02800 * ------------------------------------------------------------------ *
02810 * TRS-DDS    =====   Delete Index for Document Chapter - ABSTRACT
02820 * ------------------------------------------------------------------ *
02830   MOVE 'TEXT=Y2Y2.' TO TRS.DYP-PARM
02840 *
02850 *
02860   CALL 'TRS' 'DYP' TRS.RC TRS.DYP-PARM
02870 *
02880 *
02890   CALL 'TRS' 'DDS' TRS.RC MAP1.ORDER 'SUM'
02900 *
02910 *
02920   IF TRS.RC NE 0
02930     MOVE TRS.RC TO TRS.RC1
02940     BACKOUT TRANSACTION
02950     COMPRESS 'Error in TRS-DDS (Abstract) =>' TRS.RC1 INTO #MSG
02960     REINPUT #MSG
02970   END-IF
02980 *
02990 END-SUBROUTINE
03000 *
03010 *
03020 *
03030 *
03040 FETCH 'MENU'
03050 END
```

Formatted Retrieval

TRS - QR

The TRS-QR program represents the main retrieval part of the demonstration application. The program enables the user to enter specific queries for each of the categories in the document. The program returns the respective results and totals them by linking all relevant queries with the boolean operator “AND.” The program is broken up into the following subsections:

- Execute Query for the categories Order, Title, Abstract, Date (QR call);
- Produce Final Result (QR call);
- Invoke Overview.

Program Start Up

```
0010   ************************************************************************
0020 *                                          *
0030 *  ADABAS TEXT RETRIEVAL  Example Application                  *
0040 *                                          *
0050 *  Object   : TRS-QR                              *
0060 *  Type     : Program                            *
0070 *  Function :  Retrieval                          *
0080 *  Author   :  Software AG                          *
0090 *                                          *
0100   ************************************************************************
0110*
0120DEFINE DATA LOCAL USING TRS-LDA        /* TRS Parameter
0130*
0140LOCAL
0150*
01601 #MAP                        /* Fields in Map
0170  2 #ORDER(A60)
0180  2 #ORDER-R(N7)
0190  2 #TITLE(A60)
0200  2 #TITLE-R(N7)
0210  2 #ABSTRACT(A60)
0220  2 #ABSTRACT-R(N7)
0230  2 #DATE(A60)
0240  2 #DATE-R(N7)
0250  2 #RESULT(N7)
0260*
02701 #MSG(A72)
0280*
0290END-DEFINE
0300*
```

Set Keys

```
0310* ------------------------------------------------------------------- *
0320*  Set keys
0330* ------------------------------------------------------------------- *
0340SET KEY ALL
0350SET KEY PF2 NAMED 'Clear'
0360SET KEY PF3 NAMED 'Quit'
0370SET KEY PF6 NAMED 'Over'
0380SET KEY PF10 NAMED 'Add'
0390*
0400REPEAT
0410*
0420  INPUT USING MAP 'TRS-QRM'
0430*
```

Terminate

```
0440* ------------------------------------------------------------------- *
0450*  Escape to menu
0460* ------------------------------------------------------------------- *
0470  IF *PF-KEY = 'PF3'
0480    ESCAPE BOTTOM
0490  END-IF
0500*
```

Invoke Document Processing

```
0510* ------------------------------------------------------------------- *
0520*  Invoke Add
0530* ------------------------------------------------------------------- *
0540  IF *PF-KEY = 'PF10'
0550    FETCH 'TRS-ADD'
0560  END-IF
0570*
```

Clear Screen

Clear Screen and release queries.

For the release of the queries a RQR call is used (see lines 0620 and 0650).

```
0580* ------------------------------------------------------------------- *
0590*  Clear the screen
0600* ------------------------------------------------------------------- *
0610  IF *PF-KEY = 'PF2'
0620*
0630*
0640    CALL 'TRS' 'RQR' TRS.RC 'DOCS0002'
0650*
0660*
0670    CALL 'TRS' 'RQR' TRS.RC 'DOCS0003'
0680*
0690*
0700    RESET #MAP
0710    ESCAPE TOP
0720  END-IF
0730*
```

Execute Query for ORDER

```
0740* ------------------------------------------------------------------- *
0750*  TRS Query   =====    ORDER-NUMBER
0760* ------------------------------------------------------------------- *
0770  RESET TRS.QUERY-G #ORDER-R #TITLE-R #ABSTRACT-R #DATE-R #RESULT
0780*
0790  IF #ORDER NOT EQ ' '
0800*
0810    MOVE 'DOCS0001' TO TRS.NAME
0820    COMPRESS '#ORDER' #ORDER TO TRS.QUERY
0830*
0840*
0850    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
0860                TRS.MODE TRS.CID TRS.QTY TRS.TYPE
0870*
0880*
0890    IF TRS.RC NE 0
0900      MOVE TRS.RC TO TRS.RC1
0910      COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
0920      REINPUT #MSG MARK *#ORDER
0930    END-IF
0940*
0950    MOVE TRS.QTY TO #ORDER-R
0960    MOVE '#1' TO TRS.QUERY-G
0970*
0980  END-IF
0990*
1000*
```

Execute Query for TITLE

The query entered by the user for the formatted field TITLE is executed by the QR call in line 1100).

The search label entered by the user and the query expression are combined with the parameter TRS.QUERY for the QR call (see lines 1060 - 1070). The constant 'DOCS0002' constitutes the query name in TRS.NAME. This will be used to reference back to this specific query later on in the program.

In the QR call:

- The TRS.QLEN parameter indicates the length of the query. In the example, it is 80 bytes;
- The parameters TRS.DERR and TRS.LERR contains information on syntax errors detected in the query;
- The constant "D" is chosen for the TRS.MODE parameter in order to indicate that a document retrieval must be carried out;
- The TRS.CID parameter contains the ADABAS command ID which identifies the ADABAS ISN list resulting from the query;
- The TRS.QTY parameter contains the number of selected documents. This parameter is shown to the user as the first result;
- The constant value '=', in the TRS.TYPE parameter, chooses the default selection mode to be PRECISE.

In the field TRS.QUERY-G of line 1220, the constant entry '#2' indicates that the result of this query will be incorporated into the final query (for determining the final result).

```
1010* ------------------------------------------------------------------ *
1020*  TRS Query  =====   TITLE
1030* ------------------------------------------------------------------ *
1040  IF #TITLE NOT EQ ' '
1050*
1060    MOVE 'DOCS0002' TO TRS.NAME
1070    COMPRESS '#TITLE' #TITLE TO TRS.QUERY
1080*
1090*
1100    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
1110                 TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1120*
1130*
1140    IF TRS.RC NE 0
1150      MOVE TRS.RC TO TRS.RC1
1160      COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
1170      REINPUT #MSG MARK *#TITLE
1180    END-IF
1190*
1200    MOVE TRS.QTY TO #TITLE-R
1210    IF TRS.QUERY-G EQ ' '
1220      MOVE '#2' TO TRS.QUERY-G
1230    ELSE
1240      COMPRESS TRS.QUERY-G 'AND #2' TO TRS.QUERY-G
1250    END-IF
1260*
1270   ELSE
1280*
1290*
1300    CALL 'TRS' 'RQR' TRS.RC 'DOCS0002'
1310*
1320*
1330  END-IF
1340*
1350*
```

Execute Query for ABSTRACT and DATE

A QR call is used to select documents according to the queries entered by the user for the fields ABSTRACT and DATE. Note that the QR call in lines 1450 and 1460 is identical to that used for TITLE above.

```
1360* ------------------------------------------------------------------- *
1370*  TRS Query  =====   ABSTRACT
1380* ------------------------------------------------------------------- *
1390  IF #ABSTRACT NOT EQ ' '
1400*
1410    MOVE 'DOCS0003' TO TRS.NAME
1420    COMPRESS '#ABSTRACT' #ABSTRACT TO TRS.QUERY
1430*
1440*
1450    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
1460                 TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1470*
1480*
1490    IF TRS.RC NE 0
1500      MOVE TRS.RC TO TRS.RC1
1510      COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
1520      REINPUT #MSG MARK *#ABSTRACT
1530    END-IF
1540*
1550    MOVE TRS.QTY TO #ABSTRACT-R
1560    IF TRS.QUERY-G EQ ' '
1570      MOVE '#3' TO TRS.QUERY-G
1580    ELSE
1590      COMPRESS TRS.QUERY-G 'AND #3' TO TRS.QUERY-G
1600    END-IF
1610*
1620   ELSE
1630*
1640*
1650    CALL 'TRS' 'RQR' TRS.RC 'DOCS0003'
1660*
1670*
1680  END-IF
1690*
1700*
```

```
1710* ------------------------------------------------------------------ *
1720*  TRS Query  =====   DATE
1730* ------------------------------------------------------------------ *
1740  IF #DATE NOT EQ ' '
1750*
1760    MOVE 'DOCS0004' TO TRS.NAME
1770    COMPRESS '#DATE' #DATE TO TRS.QUERY
1780*
1790*
1800    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
1810                 TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1820*
1830*
1840    IF TRS.RC NE 0
1850      MOVE TRS.RC TO TRS.RC1
1860      COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
1870      REINPUT #MSG MARK *#DATE
1880    END-IF
1890*
1900    MOVE TRS.QTY TO #DATE-R
1910    IF TRS.QUERY-G EQ ' '
1920      MOVE '#4' TO TRS.QUERY-G
1930    ELSE
1940      COMPRESS TRS.QUERY-G 'AND #4' TO TRS.QUERY-G
1950    END-IF
1960*
1970  END-IF
1980*
1990*
```

Produce Final Result

A QR call is executed in line 2090 to produce the final total result based on the queries which have already been executed.

This final result is stored under the name DOCS0011 in line 2050.

```
2000* ------------------------------------------------------------------ *
2010*  TRS Query  =====   Total
2020* ------------------------------------------------------------------ *
2030  IF TRS.QUERY-G NOT EQ ' '
2040*
2050    MOVE 'DOCS0011'  TO TRS.NAME
2060    MOVE TRS.QUERY-G TO TRS.QUERY
2070*
2080*
2090    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
2100                 TRS.MODE TRS.CID TRS.QTY TRS.TYPE
2110*
2120*
2130    IF TRS.RC NE 0
2140      MOVE TRS.RC TO TRS.RC1
2150      COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
2160      REINPUT #MSG
2170    END-IF
2180*
2190    MOVE TRS.QTY TO #RESULT
2200*
2210  END-IF
2220*
```

Invoke Overview

The program TRS-EIS is invoked to create an overview of all documents selected by the final query.

```
2230* ------------------------------------------------------------------- *
2240*  Invoke Overview
2250* ------------------------------------------------------------------- *
2260  IF *PF-KEY = 'PF6'
2270    IF #RESULT = 0
2280      REINPUT 'No final result build.'
2290    ELSE
2300      FETCH RETURN 'TRS-EIS'
2310    END-IF
2320  END-IF
2330*
2340END-REPEAT
2350*
2360FETCH 'MENU'
2370*
2380*
2390END
```

Overview of Selected Documents

TRS - EIS

The program TRS - EIS creates an overview of the selected documents using the EISS, EISG and EISE calls. The user can then select documents to be displayed by marking them with “X”. The program is divided into the following subsections:

- Resume Query (QR call);
- Create Overview (EISG call);
- Page Up;
- Page Down;
- Select Item for Display.

Program Start Up

```
0010************************************************************************
0020*                                                                      *
0030*  ADABAS TEXT RETRIEVAL  Example Application                          *
0040*                                                                      *
0050*  Object   :  TRS-EIS                                                 *
0060*  Type     :  Program                                                 *
0070*  Function :  Retrieval overview                                      *
0080*  Author   :  Software AG                                             *
0090*                                                                      *
0100************************************************************************
0110*
0120DEFINE DATA LOCAL USING TRS-LDA        /* TRS-Parameter
0130*
0140LOCAL
0150*
0160 1 DOCUMENT VIEW OF TRS-DOCUMENT       /* Document View
0170   2 ORDER
0180   2 DATE
0190   2 TITLE
0200*
0210 1 MAP1                          /* Fields in Map
0220   2 LINE(A72/16)
0230   2 MARK(A1/16)
0240   2 MARK-CV(C/16)
0250*
02601 MSG(A72)
0270*
02801 J(N2)                          /* Work Fields
02901 K(N2)
03001 I(N7)
03101 P-FROM(N7)
03201 P-THRU(N7)
03301 ORD(A16/16)
0340*
03501 LINE1(A72)
03601 REDEFINE LINE1
0370  2 ORDER(A11)
0380  2 FILLER1 (A1)
0390  2 DATE(N8)
0400  2 FILLER2 (A1)
0410  2 LINE-TEXT(A51)
0420*
0430END-DEFINE
0440*
```

Set Keys

```
0450* ------------------------------------------------------------------ *
0460*  Set keys
0470* ------------------------------------------------------------------ *
0480SET KEY ALL
0490SET KEY PF3 NAMED 'Quit'
0500SET KEY PF6 NAMED 'Disp'
0510SET KEY PF7 NAMED 'Back'
0520SET KEY PF8 NAMED 'For'
0530*
```

Resume Query

A QR call in line 0610 is executed in order to resume the result of the final query by the program TRS - QR and to sort the selected documents according to the formatted field ORDER (see line 0570).

The EISS call in line 0740 is used to start browsing through an ISN set created by a QR call.

```
0540* ------------------------------------------------------------------ *
0550*  Resume Queries
0560* ------------------------------------------------------------------ *
0570MOVE '#11 SORT #ORDER' TO TRS.QUERY
0580MOVE 'DOCS0012' TO TRS.NAME
0590*
0600*
0610CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
0620             TRS.MODE TRS.CID TRS.QTY TRS.TYPE
0630*
0640*
0650IF TRS.RC NE 0
0660   MOVE TRS.RC TO TRS.RC1
0670   WRITE 'INTERNAL ERROR'
0680   STOP
0690END-IF
0700*
0710MOVE TRS.QTY TO TRS.QTY1
0720*
0730*
0740CALL 'TRS' 'EISS' TRS.RC TRS.CID TRS.TYPE TRS.QTY
0750*
0760*
0770MOVE 1 TO P-FROM
0780*
```

Create Overview

An overview is of the selected documents is created. These documents are fetched via ISNs which have been provided by the EISG call in line 1000. This call contains the following parameters:

- The TRS.CID parameter contains the ADABAS Command ID assigned to the query previously executed.
- The constant “D” is assigned to the parameter TRS.TYPE to indicate that a document selection was executed.
- The TRS.QTY parameter contains the number of documents selected by the previous QR call;
- The TRS.POS parameter must contain the relative position of the requested ISN inside the ADABAS ISN set. In this case, the user variable “I” is used to browse through the selected documents;
- The parameter TRS.ISN will contain the ADABAS ISN selected by the EISG call.

The document represented by the ISN provided by the EISG call is then accessed by a GET command in line 1050.

```
0790* ------------------------------------------------------------------- *
0800*  Create Overview
0810* ------------------------------------------------------------------- *
0820REPEAT
0830*
0840                        /* Calculate Posistion in Set
0850  MOVE P-FROM TO P-THRU
0860  ADD 15 TO P-THRU
0870  IF P-THRU GT TRS.QTY1
0880    MOVE TRS.QTY1 TO P-THRU
0890  END-IF
0900*
0910  RESET MAP1 K
0920  MOVE (AD=NP) TO MARK-CV(*)
0930*
0940*
0950  FOR I = P-FROM TO P-THRU
0960*
0970    MOVE I TO TRS.POS
0980*
0990*
1000    CALL 'TRS' 'EISG' TRS.RC TRS.CID TRS.TYPE TRS.QTY TRS.POS TRS.ISN
1010*
1020*
1030    MOVE TRS.ISN TO TRS.ISN1
1040*
1050    GET DOCUMENT TRS.ISN1
1060*
1070       ADD 1 TO K
1080       MOVE DOCUMENT.ORDER TO LINE1.ORDER
1090       MOVE DOCUMENT.ORDER TO ORD(K)
1100       MOVE DOCUMENT.DATE  TO LINE1.DATE
1110       MOVE TITLE          TO LINE-TEXT
1120       MOVE LINE1 TO LINE(K)
1130       RESET MARK-CV(K)
1140*
1150  END-FOR
1160*
1170*
1180  INPUT WITH TEXT MSG USING MAP 'TRS-EISM'
1190*
1200*
1210  RESET MSG
1220*
```

Escape

```
1230* ------------------------------------------------------------------- *
1240*  Escape
1250* ------------------------------------------------------------------- *
1260  IF *PF-KEY = 'PF3'
1270    ESCAPE BOTTOM
1280  END-IF
1290*
```

Page Up

Scroll back through the set of documents.

```
1300* ------------------------------------------------------------------ *
1310*  Page-Up
1320* ------------------------------------------------------------------ *
1330  IF *PF-KEY = 'PF7'
1340     SUBTRACT 16 FROM P-FROM
1350     IF P-FROM LT 1
1360        MOVE 'This is the first page.' TO MSG
1370        MOVE 1 TO P-FROM
1380     END-IF
1390  END-IF
1400*
```

Page Down

Scroll forward through the set of documents.

```
1410* ------------------------------------------------------------------ *
1420*  Page-Down
1430* ------------------------------------------------------------------ *
1440  IF *PF-KEY = 'PF8'
1450     ADD 16 TO P-FROM
1460     IF P-FROM GT TRS.QTY1
1470        MOVE 'This is the last page.' TO MSG
1480        SUBTRACT 16 FROM P-FROM
1490     END-IF
1500  END-IF
1510*
```

Select Item for Display

The subprogram TRS-DISP is invoked in line 1650 to display a document (see line 1560). The document - iol (ORDER) is passed as parameter to the subprogram.

At the end, an EISE call is issued in line 1960 to end browsing.

```
1520* ------------------------------------------------------------------- *
1530*  Select item for display
1540* ------------------------------------------------------------------- *
1550  IF *PF-KEY = 'ENTR' OR *PF-KEY = 'PF6'
1560*
1570    IF *PF-KEY = 'PF6'
1580      MOVE ALL 'X' TO MARK(1:K)
1590    END-IF
1600*
1610    FOR J = 1 TO 16
1620      IF MAP1.MARK (J) NE ' '
1630        MOVE LINE(J) TO LINE1
1640*
1650        CALLNAT 'TRS-DISP' ORD(J) MSG
1660*
1670        IF *PF-KEY = 'PF3'
1680          ESCAPE BOTTOM
1690        END-IF
1700*
1710        IF *PF-KEY = 'PF6'
1720          MOVE 'This is the first page.' TO MSG
1730          RESET J
1740        END-IF
1750*
1760        IF *PF-KEY = 'PF8' AND J = K
1770          MOVE 'This is the last page.' TO MSG
1780          SUBTRACT 1 FROM J
1790        END-IF
1800*
1810        IF *PF-KEY = 'PF7'
1820          SUBTRACT 2 FROM J
1830          IF J LT 0
1840            MOVE 'This is the first page.' TO MSG
1850            RESET J
1860          END-IF
1870        END-IF
1880*
1890      END-IF
1900    END-FOR
1910  END-IF
1920*
1930END-REPEAT
1940*
1950*
1960CALL 'TRS' 'EISE' TRS.RC TRS.CID TRS.TYPE
1970*
1980*
1990END
```

Document Display

TRS - DISP

The program TRS - DISP displays a single document selected from the document overview provided by the program TRS - EIS. The HIGH call is used to mark the words which fulfill the search criterion specified in the QR calls. The program is divided into the following major subsections:

- Find document;
- Highlight document (HIGH call);
- Display documents.

Program Start Up

```
0010************************************************************************
0020*                                        *
0030*  ADABAS TEXT RETRIEVAL  Example Application                  *
0040*                                        *
0050*  Object   :  TRS-DISP                          *
0060*  Type     :  Subprogram                        *
0070*  Function :  Display selected documents                  *
0080*  Author   :  Software AG                        *
0090*                                        *
0100************************************************************************
0110*
0120DEFINE DATA PARAMETER
0130*
01401 PARA
0150  2 ORDER (A16)
0160  2 MSG   (A72)
0170*
0180LOCAL USING TRS-LDA
0190LOCAL
0200*
02101 MAP1
0220  2 ORDER(A16)
0230  2 PRICE(N3)
0240  2 DATE(N8)
0250  2 TITLE1(A70)
0260  2 ABSTRACT1(A70/12)
0270*
02801 DOCUMENT VIEW OF TRS-DOCUMENT
0290  2 ORDER
0300  2 DATE
0310  2 PRICE
0320  2 TITLE
0330  2 C*ABSTRACT
0340  2 ABSTRACT (12)
03501 I(N2)
0360*
0370END-DEFINE
0380*
0390REPEAT
0400*
0410  RESET MAP1
0420*
```

Find Document

This procedure collects the document information for display and highlighting.

```
0430* ------------------------------------------------------------------- *
0440* Find Document
0450* ------------------------------------------------------------------- *
0460  FIND DOCUMENT WITH ORDER = PARA.ORDER
0470*
0480    MOVE DOCUMENT.ORDER TO MAP1.ORDER
0490    MOVE DOCUMENT.PRICE TO MAP1.PRICE
0500    MOVE DOCUMENT.DATE TO MAP1.DATE
0510    MOVE 70 TO TRS.HLEN
0520    MOVE DOCUMENT.TITLE    TO TRS.HTEXT1
0530    MOVE 0  TO TRS.CURSOR
0540*
```

Highlight Document

A HIGH call is issued in order to highlight the term or terms specified in a query. Prior to highlighting, a DYP call (see line 0600 and 0880) must be executed in order to set the TEXT parameter to the name of the ADABAS hyperdescriptor representing the free-text chapter in question.

In the HIGH call, the DOCUMENT.ORDER parameter contains the document ID. This process is also carried out for the document abstract (see lines 0830 - 1020).

The constant DOCS0002 is the name of the query (issued in the program TRS - QR) for which the highlighting is to be performed. The input parameter TRS.HTEXT contains the source text to be highlighted (in this case the title). By use of the NATURAL dynamic attribute parameter (DY), the TRS.HTEXT2 output parameter, which contains the source text including prefixes and suffixes created by the HIGH call, can be used for immediate physical highlighting.

The constants '<' and '>' represent the suffix and prefix to be used to mark the words fulfilling the search criteria. Highlighting is achieved using the NATURAL dynamic attribute facility.

The TRS.CURSOR parameter is an internal variable which controls the highlighting process. Prior to the start of the highlighting process, it must be set to zero. It must not be changed during intermediate processing (see line 0740).

```
0550*  ------------------------------------------------------------------ *
0560*  TRS High   =====  Highlight Document Chapter - TITLE
0570*  ------------------------------------------------------------------ *
0580*
0590*
0600   CALL 'TRS' 'DYP'  TRS.RC 'TEXT=Y1Y1.'
0610*
0620*
0630   CALL 'TRS' 'HIGH' TRS.RC DOCUMENT.ORDER 'DOCS0002'
0640                TRS.HTEXT TRS.HTEXT2 TRS.HLEN '<' '>' TRS.CURSOR
0650*
0660*
0670   IF NOT (TRS.RC = 0 OR = 6)
0680     MOVE TRS.RC TO TRS.RC1
0690     INPUT(AD=OIL) 'ERROR IN TRS-HIGH :' TRS.RC1
0700     STOP
0710   END-IF
0720*
0730   MOVE TRS.HTEXT2 TO MAP1.TITLE1
0740   MOVE 0  TO TRS.CURSOR
0750*
0760   FOR I = 1 TO C*DOCUMENT.ABSTRACT
0770     IF I > 12
0780       ESCAPE BOTTOM
0790     END-IF
0800     MOVE 70 TO TRS.HLEN
0810     MOVE DOCUMENT.ABSTRACT(I) TO TRS.HTEXT1
0820*

0830*    ------------------------------------------------------------- *
0840*    TRS High   =====  Highlight Document Chapter - ABSTRACT
0850*    ------------------------------------------------------------- *
0860*
0870*
0880     CALL 'TRS' 'DYP'  TRS.RC 'TEXT=Y2Y2.'
0890*
0900*
0910     CALL 'TRS' 'HIGH' TRS.RC DOCUMENT.ORDER 'DOCS0003' TRS.HTEXT
0920                  TRS.HTEXT2 TRS.HLEN '<' '>' TRS.CURSOR
0930*
0940*
0950     IF NOT (TRS.RC = 0 OR = 6)
0960       MOVE TRS.RC TO TRS.RC1
0970       INPUT(AD=OIL) 'ERROR IN TRS-HIGH :' TRS.RC1
0980       STOP
0990     END-IF
1000     MOVE TRS.HTEXT2 TO MAP1.ABSTRACT1(I)
1010   END-FOR
1020  END-FIND
1030*
```

Display Documents

Display the documents with highlighting.

Note: The NATURAL dynamic attribute feature is used to highlight the words marked by the HIGH call.

```
1040* ------------------------------------------------------------------- *
1050* Display Highighted Document
1060* ------------------------------------------------------------------- *
1070  INPUT WITH TEXT MSG
1080        USING MAP 'TRS-DISM'
1090*
1100  IF *PF-KEY = 'PF3' OR *PF-KEY = 'PF6' OR
1110     *PF-KEY = 'PF7' OR *PF-KEY = 'PF8'
1120    RESET MSG
1130    ESCAPE ROUTINE
1140  END-IF
1150*
1160END-REPEAT
1170*
1180END
```

Index Display

TRS - HLP

The help routine TRS - HLP shows the values of the formatted fields and the free-text chapters. For formatted fields DATE and ORDER, a simple histogram is used. For the free-text chapters, logical reads on the “word fields” of the vocabulary and subsequent QR calls are used to obtain the number of documents where the words occur.

The program is divided into the following subsections:

- Display Available Values for ORDER;
- Display Available Values for DATE;
- Display Vocabulary for TITLE or ABSTRACT;
- Show Screen.

Program Start Up

```
0010************************************************************************
0020*                                          *
0030*  ADABAS TEXT RETRIEVAL  Example Application                  *
0040*                                          *
0050*  Object   : TRS-HLP                            *
0060*  Type     : Helproutine                        *
0070*  Function :  Display category index                    *
0080*  Author   : Software AG                         *
0090*                                          *
0100************************************************************************
0110*
0120DEFINE DATA PARAMETER
0130*
01401 PARA
0150  2 FIELD (A65)
0160  2 VALUE (A60)
0170  2 REDEFINE VALUE
0180    3 DATE (N4)
0190*
0200LOCAL USING TRS-LDA
0210*
0220LOCAL
02301 VOC VIEW OF TRS-VOCABULARY
0240  2 WORD
0250  2 ORIGINAL-WORD
02601 DOC1 VIEW OF TRS-DOCUMENT
0270  2 ORDER
02801 DOC2 VIEW OF TRS-DOCUMENT
0290  2 DATE
03001 MAP1
0310  2 CAT  (A10)
0320  2 MARK (A1/10)
0330  2 WORD (A30/10)
0340  2 QTY  (N5/10)
0350  2 CV   (C/10)
0360*
03701 #I (N2)
03801 #J (N2)
03901 #K (A1)
04001 #D (N4)
04101 #A (A60)
04201 #V (A34)
0430END-DEFINE
0440*
0450SET   KEY PF3
0460MOVE (AD=PN) TO CV(*)
0470*
0480*
0490IF NOT(PARA.FIELD = '#TITLE' OR = '#ABSTRACT' )
0500*
```

Display Available Values for ORDER

A HISTOGRAM statement is used in line 550 to determine the values present for the formatted field ORDER.

```
0510* ------------------------------------------------------------------- *
0520*  Display available values for Order                 *
0530* ------------------------------------------------------------------- *
0540  IF PARA.FIELD = '#ORDER'
0550    HISTOGRAM DOC1 FOR ORDER STARTING FROM PARA.VALUE
0560      ADD 1 TO #I
0570      MOVE DOC1.ORDER  TO MAP1.WORD(#I)
0580      MOVE *NUMBER     TO MAP1.QTY(#I)
0590      RESET CV(#I)
0600      IF #I GE 10
0610        PERFORM SR-SCREEN
0620        IF *PF-KEY = 'PF3'
0630          ESCAPE ROUTINE
0640        END-IF
0650      END-IF
0660    END-HISTOGRAM
0670  END-IF
```

Display Available Values for DATE

A HISTOGRAM statement is used in line 770 to determine the values present for the formatted field DATE.

```
0680* ------------------------------------------------------------------- *
0690*  Display available values for Date                   *
0700* ------------------------------------------------------------------- *
0710  IF PARA.FIELD = '#DATE'
0720    IF PARA.DATE NE MASK(YYMM)
0730      MOVE PARA.DATE TO #D
0740    END-IF
0750    RESET PARA.VALUE
0760*
0770    HISTOGRAM DOC2 FOR DATE STARTING FROM #D
0780      ADD 1 TO #I
0790      MOVE DOC2.DATE  TO MAP1.WORD(#I)
0800      MOVE *NUMBER    TO MAP1.QTY(#I)
0810      RESET CV(#I)
0820      IF #I GE 10
0830        PERFORM SR-SCREEN
0840        IF *PF-KEY = 'PF3'
0850          ESCAPE ROUTINE
0860        END-IF
0870      END-IF
0880    END-HISTOGRAM
0890*
0900  END-IF
0910ELSE
0920*
```

Display Vocabulary for TITLE or ABSTRACT

The vocabulary file is read in line 1010 and a QR call is issued in line 1080 to determine how often the queried word is found in the contents of the relevant free-text chapter. If the number of documents selected by this query is greater than zero, the word becomes part of the display. The "original word" representing the non-standard word is used for the display to show the word in lower/upper case and special characters (see line 1190).

```
0930* ------------------------------------------------------------------- *
0940*  DISPLAY VOCABULARY FOR TITLE OR ABSTRACT                           *
0950* ------------------------------------------------------------------- *
0960  MOVE PARA.VALUE TO #A
0970  IF PARA.VALUE < H'81'
0980    MOVE H'81' TO #A
0990  END-IF
1000*
1010  READ VOC BY WORD = #A
1020*
1030    COMPRESS '"' VOC.WORD '"' TO #V LEAVING NO
1040    COMPRESS PARA.FIELD #V TO TRS.QUERY
1050    MOVE 'DOCS0020'            TO TRS.NAME
1060*
1070*
1080    CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
1090                TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1100*
1110*
1120    IF TRS.RC NE 0
1130      MOVE TRS.RC TO TRS.RC1
1140      INPUT (AD=OIL) 'ERROR IN TRS.QR =>' TRS.RC1
1150    END-IF
1160*
1170    IF TRS.QTY > 0
1180      ADD 1 TO #I
1190      MOVE VOC.ORIGINAL-WORD TO MAP1.WORD(#I)
1200      MOVE TRS.QTY TO MAP1.QTY(#I)
1210      RESET CV(#I)
1220    END-IF
1230*
1240    IF #I GE 10
1250      PERFORM SR-SCREEN
1260      IF *PF-KEY = 'PF3'
1270        ESCAPE ROUTINE
1280      END-IF
1290    END-IF
1300  END-READ
1310END-IF
1320IF #I > 0
1330  PERFORM SR-SCREEN
1340END-IF
1350*
```

Show Screen

The selected values are displayed to screen and can be selected for inclusion in queries.

```
1360************************************************************************
1370DEFINE SUBROUTINE SR-SCREEN    /* Display Screen
1380************************************************************************
1390*
1400MOVE PARA.FIELD TO MAP1.CAT
1410*
1420INPUT USING MAP 'TRS-HLPM'
1430*
1440IF *PF-KEY = 'PF3'
1450  ESCAPE ROUTINE
1460END-IF
1470*
1480* ---------------------------------------------------------------------- *
1490* Check if words are marked
1500* ---------------------------------------------------------------------- *
1510FOR #J = 1 TO #I
1520  IF MAP1.MARK(#J) NE ' '
1530    IF PARA.VALUE NE ' '
1540      MOVE ',' TO #K
1550    END-IF
1560    COMPRESS PARA.VALUE #K MAP1.WORD(#J) INTO PARA.VALUE LEAVING NO
1570    IF NOT (PARA.FIELD = '#TITLE' OR = '#ABSTRACT') AND
1580      PARA.VALUE NE ' '
1590      ESCAPE BOTTOM
1600    END-IF
1610  END-IF
1620END-FOR
1630*
1640RESET MAP1 #I #J
1650MOVE (AD=PN) TO CV(*)
1660*
1670END-SUBROUTINE
1680END
```

Freestyle Retrieval

TRS - FQR

The program TRS - FQR enables the user to enter queries which conform to ADABAS TEXT RETRIEVAL query syntax. The program is divided into the following logical units.

- Delete queries (RQR);
- Copy query;
- Create NATURAL retained set (RET call);
- Execute query (QR call).

Program Start Up

```
0010************************************************************************
0020*                                          *
0030*  ADABAS TEXT RETRIEVAL  Example Application              *
0040*                                          *
0050*  Object   :  TRS-FQR                          *
0060*  Type     :  Program                        *
0070*  Function :  Freestyle retrieval                 *
0080*  Author   :  Software AG                      *
0090*                                          *
0100************************************************************************
0110*
0120DEFINE DATA LOCAL USING TRS-LDA
0130*
0140LOCAL
01501 #CHAPTER(A10)
01601 #QUERY   (A62)
01701 #NR (N2/10)
01801 #MARK(A1/10)
01901 #RESULT(N5/10)
02001 #PQR(A62/10)
02101 #MSG   (A60)
02201 #I     (N4)
02301 #J     (N4)
02401 #SETID (A32) INIT<'TRSSET'>
02501 #QNAM  (A8)
02601 REDEFINE #QNAM
0270  2 HEADER (A4)
0280  2 COUNT  (N4)
02901 CV1    (C/10)
0300*
0310END-DEFINE
0320*
```

Set Keys

```
0330* ------------------------------------------------------------------- *
0340*  Set keys
0350* ------------------------------------------------------------------- *
0360SET KEY ALL
0370SET KEY PF2 NAMED 'Delete'
0380SET KEY PF4 NAMED 'Copy '
0390SET KEY PF3 NAMED 'Quit'
0400SET KEY PF6 NAMED 'Over'
0410*
0420MOVE (AD=PN) TO CV1(*)
0430*
0440REPEAT
0450*
0460  RESET #MARK(*)
0470*
0480  INPUT WITH TEXT #MSG USING MAP 'TRS-FQRM'
0490*
```

Terminate

```
0500* ------------------------------------------------------------------- *
0510*  Escape to menu
0520* ------------------------------------------------------------------- *
0530  IF *PF-KEY = 'PF3'
0540    ESCAPE BOTTOM
0550  END-IF
0560*
```

Delete Queries

The user deletes previously executed queries by marking them with “D” and pressing PF2. The queries are released by executing a RQR call as seen in line 0660.

```
0570* ------------------------------------------------------------------- *
0580*  Delete Queries
0590* ------------------------------------------------------------------- *
0600  IF *PF-KEY = 'PF2'
0610    FOR #I = 1 TO 10
0620      IF #MARK(#I) EQ  'D'
0630        MOVE #I TO #QNAM.COUNT
0640*
0650*
0660        CALL 'TRS' 'RQR' TRS.RC #QNAM
0670*
0680*
0690        RESET #RESULT(#I) #NR(#I) #PQR(#I) #MARK(#I)
0700        MOVE (AD=PN) TO CV1(#I)
0710      END-IF
0720    END-FOR
0730    ESCAPE TOP
0740  END-IF
0750*
```

Copy Query

A previously executed query can be copied to make up a new query which can be modified and executed.

```
0760* ------------------------------------------------------------------- *
0770*  Copy Query
0780* ------------------------------------------------------------------- *
0790  IF *PF-KEY = 'PF4'
0800    FOR #I = 1 TO 10
0810      IF #MARK(#I) EQ  'C'
0820        MOVE #PQR(#I) TO #QUERY
0830        RESET #MARK (*)
0840        ESCAPE BOTTOM
0850      END-IF
0860    END-FOR
0870    ESCAPE TOP
0880  END-IF
0890*
```

Create NATURAL Retained Set and Invoke Overview

After resuming the query, the RET call is used in line 1070 to create a NATURAL retain set. This NATURAL retain set is used in the program TRS - NAT to create an overview of selected documents.

Before executing the RET call , the query in question is resumed by the use of the QR call in line 1030.

```
0900* ------------------------------------------------------------------- *
0910*  Create NATURAL Retain Set and Invoke Overview
0920* ------------------------------------------------------------------- *
0930  IF *PF-KEY = 'PF6'
0940    FOR #I = 1 TO 10
0950      IF #MARK(#I) = 'X'
0960        IF #RESULT (#I) = 0
0970          REINPUT 'Query result is zero'
0980        END-IF
0990        MOVE 'DOCS0011' TO TRS.NAME
1000        COMPRESS '#' #NR(#I) INTO TRS.QUERY LEAVING NO
1010*
1020*
1030        CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME
1040             TRS.DERR TRS.LERR TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1050*
1060*
1070        CALL 'TRS' 'RET' TRS.RC TRS.CID TRS.QTY #SETID
1080*
1090*
1100        FETCH RETURN 'TRS-NAT'
1110      END-IF
1120    END-FOR
1130    RESET #MARK(*)
1140    ESCAPE TOP
1150  END-IF
1160*
```

Execute Query

A QR call (see line 1400) is executed to perform the query. The query and its results are put on the stack of executed queries.

```
1170* ------------------------------------------------------------------- *
1180*  TRS Queries
1190* ------------------------------------------------------------------- *
1200  IF #QUERY = ' '
1210    REINPUT 'No query specified.'
1220  END-IF
1230*
1240  FOR #I = 1 TO 10
1250     IF #PQR(#I) = ' '
1260       ESCAPE BOTTOM   /* Check for empty slot
1270     END-IF
1280  END-FOR
1290  IF #I = 11
1300    REINPUT
1310     'Stack is full ! delete queries by marking with "D" and PF2'
1320  END-IF
1330*
1340  MOVE 'DOCS'    TO #QNAM.HEADER
1350  MOVE #I        TO #QNAM.COUNT
1360  MOVE #QNAM     TO TRS.NAME
1370  COMPRESS #CHAPTER #QUERY TO TRS.QUERY
1380*
1390*
1400  CALL 'TRS' 'QR' TRS.RC TRS.QUERY TRS.QLEN TRS.NAME TRS.DERR TRS.LERR
1410               TRS.MODE TRS.CID TRS.QTY TRS.TYPE
1420*
1430*
1440  MOVE (AD=I)    TO CV1(#I)
1450  MOVE TRS.QUERY TO #PQR(#I)
1460  MOVE TRS.QTY   TO #RESULT(#I)
1470  MOVE #I        TO #NR(#I)
1480*
1490  IF TRS.RC NE 0
1500    MOVE TRS.RC TO TRS.RC1
1510    COMPRESS 'Error in TRS.QR =>' TRS.RC1 TO #MSG
1520  END-IF
1530  RESET  #QUERY
1540*
1550END-REPEAT
1560*
1570FETCH 'MENU'
1580END
```

Access NATURAL RETAIN Set

TRS - NAT

The program TRS - NAT displays the documents that are represented by a NATURAL retain set created by a RET call in the retrieval program TRS - FQR.

It shows how to integrate ADABAS TEXT RETRIEVAL resulting sets into the processing of retain sets created by simple NATURAL Find commands.

Program Start Up

```
0010************************************************************************
0020*                                                                      *
0030*  ADABAS TEXT RETRIEVAL  Example Application                          *
0040*                                                                      *
0050*  Object   :  TRS-NAT                                                 *
0060*  Type     :  Program                                                 *
0070*  FUNCTION :  Access NATURAL SET created by TRS RET-CALL              *
0080*  Author   :  Software AG                                             *
0090*                                                                      *
0100************************************************************************
0110DEFINE DATA LOCAL
01201 DOC VIEW OF TRS-DOCUMENT
0130  2 ORDER
0140  2 TITLE
0150END-DEFINE
0160*
0170FORMAT PS=20
0180FORMAT KD=ON
0190SET KEY ALL
0200SET KEY PF3 NAMED 'Quit'
0210*
```

Retrieve NATURAL Retain Set

The NATURAL retain set TRSSET, created by the RET call in program TRS - FQR, is accessed in line 0250. In line 0380, the set is released.

```
0220* ------------------------------------------------------------------- *
0230*  Retrieve NATURAL Retain Set
0240* ------------------------------------------------------------------- *
0250FIND DOC WITH 'TRSSET'
0260*
0270  DISPLAY ORDER TITLE    (AL=60)
0280  AT END OF PAGE
0290    INPUT NO ERASE 1/50 ' '
0300    IF *PF-KEY = 'PF3'
0310      ESCAPE BOTTOM
0320    END-IF
0330  END-ENDPAGE
0340END-FIND
0350*
0360*
0370NEWPAGE
0380RELEASE SET
0390END
```

APPENDIX A — MESSAGES AND CODES

Return Code 1

Explanation: Invalid number of parameters.

Return Code 2

Explanation: Storage allocation failed.

Action: Increase TSIZE parameter.

Return Code 3

Explanation:
- TSIZE not found.
- Invalid TRS call.
- Invalid order of TRS call.
- Invalid *type* parameter for QR call

Action: Check *Type* parameter.
- Invalid *Default Mode* parameter for QR call

Action: Check *Default Mode* parameter.

Return Code 5

Explanation: Invalid order of EIS calls.

Action: Execute an EISS call before executing an EISG call.

Return Code 6

Explanation: Referenced query not found.

Action: Check reference for previous query.

Return Code 7

Explanation: Invalid request for vocabulary query.

Return Code 8

Explanation: Invalid number of operands in formatted fields.

Action: For each relational operator with the exception of BETWEEN, one operand is required; for the BETWEEN operation two operands are required.

Return Code 9

Explanation: The proximity operation is overburdened. Before a proximity operation is performed the result of an AND operation is checked and if it is greater than the value of the MAXDPRO parameter the response code will be displayed.

Action: Increase the MAXDPRO parameter.

Return Code 10

Explanation: Invalid truncation operation action.

Action: Check the use of truncation.

Return Code 11

Explanation: Too many terms in ASPECT operation.

Return Code 12

Explanation: Internal error; index destroyed

Return Code 13

Explanation: Maximum number of words per document exceeded.

Return Code 14

Explanation: Invalid parameters in BC or DYP call.

Action: Check parameters in BC or DYP call.

Return Code 16

Explanation: Invalid parameters in DSL call.

Action: Check parameters for DSL call.

Return Code 19

Explanation: Invalid parameters in RULE call.

Action: Check all parameters for RULE call.

Return Code 20

Explanation: Too many words in one query.

Action: Increase MAXVSET parameter.

Return Code 30

Explanation: Using LOADER=YES and vocabulary ISN is in use.

Action: Do not use STARTISN or change its value.

Return Code 40

Explanation: Using LOADER=YES sort error.

Action: Check sort messages.

Syntax Errors

Return Code 101

Explanation: Operator followed by another operator.

Return Code 102

Explanation: An operand must appear before a comma.

Return Code 104

Explanation: Operator followed by right parenthesis.

Return Code 105

Explanation: Operator as last token in query.

Return Code 106

Explanation: Operand not found after function.

Return Code 107

Explanation: Function followed by another function.

Return Code 108

Explanation: Comma appears after function.

Return Code 109

Explanation: Left parenthesis appears after function.

Return Code 110

Explanation: Right parenthesis appears after function.

Return Code 111

Explanation: Function as last token in query.

Return Code 112

Explanation: Comma followed by function.

Return Code 113

Explanation: Comma followed by comma.

Return Code 114

Explanation: Comma followed by left parenthesis.

Return Code 115

Explanation: Comma followed by right parenthesis.

Return Code 116

Explanation: Comma as last token in query.

Return Code 117

Explanation: Operand followed by another operand.

Return Code 118

Explanation: Operand followed by left parenthesis.

Return Code 119

Explanation: Left parenthesis followed by operator.

Return Code 120

Explanation: Left parenthesis followed by comma.

Return Code 121

Explanation: Left parenthesis is followed by right parenthesis

Return Code 122

Explanation: Left parenthesis is last token in query.

Return Code 123

Explanation: Right parenthesis followed by comma.

Return Code 124

Explanation: Right parenthesis followed by operand.

Return Code 125

Explanation: Right parenthesis followed by left parenthesis.

Return Code 126

Explanation: Comma as first token in query.

Return Code 127

Explanation: Right parenthesis as first token.

Return Code 128

Explanation: Empty query.

Return Code 130

Explanation: Reference number followed by function.

Return Code 131

Explanation: Reference number followed by comma.

Return Code 132

Explanation: Reference number followed by operand.

Return Code 133

Explanation: Reference number followed by left parenthesis.

Return Code 135

Explanation: Function followed by reference number.

Return Code 136

Explanation: Comma followed by reference number.

Return Code 137

Explanation: Operand followed by reference number.

Return Code 138

Explanation: Right parenthesis followed by reference number.

Return Code 139

Explanation: Reference number followed by reference number.

Return Code 140

Explanation: Operator as first token in query; unmatched parenthesis.

Return Code 141

Explanation: Operator after formatted field.

Return Code 142

Explanation: Function after formatted field.

Return Code 143

Explanation: Comma after formatted field.

Return Code 144

Explanation: Left parenthesis after formatted field.

Return Code 145

Explanation: Right parenthesis after formatted field.

Return Code 146

Explanation: Value is expected after formatted field.

Return Code 148

Explanation: Formatted field after formatted field.

Return Code 149

Explanation: Operator after relation operator is invalid.

Return Code 150

Explanation: Function after relation operator is invalid.

Return Code 151

Explanation: Comma after relation operator is invalid.

Return Code 152

Explanation: Left parenthesis after operator is invalid.

Return Code 153

Explanation: Right parenthesis after operator is invalid.

Return Code 154

Explanation: Value is expected after relation operator.

Return Code 155

Explanation: Formatted field is not expected after relational operator.

Return Code 156

Explanation: Relation operator after relation operator.

Return Code 157

Explanation: Relation operator after operator.

Return Code 158

Explanation: Formatted field after function.

Return Code 159

Explanation: Relation operator after function.

Return Code 160

Explanation: Formatted field after comma.

Return Code 161

Explanation: Relation operator after comma.

Return Code 162

Explanation: Relation operator not after formatted field.

Return Code 163

Explanation: Relation operator after left parenthesis.

Return Code 164

Explanation: Formatted field after right parenthesis.

Return Code 165

Explanation: Relation operator after right parenthesis.

Return Code 166

Explanation: Relation operator not preceded by formatted field.

Return Code 167

Explanation: Formatted field after reference number.

Return Code 168

Explanation: Relation operator after reference number.

Return Code 169

Explanation: Formatted field after operand.

Return Code 195

Explanation: Too many tokens for formatted field.

Return Code 196

Explanation: Non-numeric token for numeric formatted field.

Return Code 198

Explanation: Invalid formatted field for SORT or SORTD.

Return Code 199

Explanation: Internal error during syntax checking.

APPENDIX B — APPLYING ADABAS TEXT RETRIEVAL IN A NON-NATURAL ENVIRONMENT

Most of the ADABAS TEXT RETRIEVAL functions can be used in any programming environment which supports standard linkage conventions.

If ADABAS TEXT RETRIEVAL is to be used in a non-NATURAL environment the call link conventions are slightly different from those described in chapter 1:

- instead of passing the requested ADABAS TEXT RETRIEVAL function as first parameter the character string “TRS” plus the function name make up the name of the program to be called.

Example:

CALL TRSQR or **CALL TRSADD**

- as additional first parameter to each call, the address of an area to be used by ADABAS TEXT RETRIEVAL internally must be passed by the calling program. The size of the storage area must be supplied in the *Size of Buffer* parameter of the TRSBC call; for instance an address in an Assembler program; a pointer variable in a PL1 program etc.

This storage area is initialised in the TRSBC call. It must be passed unchanged to every subsequent ADABAS TEXT RETRIEVAL call.

INDEX

- F -

- G -

- H -

- I -

- K -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- V -

- W -