

Adabas

Glossary

Version 8.2.3

May 2011

This document applies to Adabas Version 8.2.3.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1971-2011 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their licensors.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <http://documentation.softwareag.com/legal/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". This document is part of the product documentation, located at <http://documentation.softwareag.com/legal/> and/or in the root installation directory of the licensed product(s).

Table of Contents

| | |
|----------------|----|
| Glossary | 1 |
| A | 1 |
| B | 3 |
| C | 4 |
| D | 5 |
| E | 6 |
| F | 6 |
| G | 9 |
| I | 10 |
| L | 11 |
| M | 12 |
| N | 12 |
| O | 13 |
| P | 14 |
| Q | 15 |
| R | 15 |
| S | 16 |
| T | 18 |
| U | 19 |
| V | 20 |
| W | 21 |
| Index | 23 |

Glossary

A

| | |
|---------------------------------|--|
| ABD list | <p>An ABD list is a file containing a list of pointers to the Adabas buffer descriptions to be used for an Adabas direct call. ABD lists can only be used if the direct call is made using the ACBX direct call interface.</p> <p>For complete information about ABD lists, read <i>ABD Lists</i>, in the <i>Adabas Command Reference Guide</i>.</p> |
| ACB interface direct call | <p>The <i>ACB direct call interface</i> is the classic direct call interface, used for Adabas releases prior to Adabas version 8. Direct calls in this format require the use of the classic Adabas control block (ACB). If you have been using releases of Adabas prior to Adabas 8, the direct calls used by your applications use the ACB direct call interface.</p> <p>For information about Adabas direct calls, read <i>Calling Adabas</i>, in the <i>Adabas Command Reference Guide</i>.</p> |
| ACBX interface direct call | <p>The <i>ACBX direct call interface</i> is the extended direct call interface, used for Adabas releases starting with Adabas 8. Direct calls in this format require the use of the extended Adabas control block (ACBX). If you have purchased and installed Adabas 8 (or later), you can use this format of direct call in your applications. Otherwise, you cannot.</p> <p>For information about Adabas direct calls, read <i>Calling Adabas</i>, in the <i>Adabas Command Reference Guide</i>.</p> |
| Adabas | <p>Adabas, the <u>ad</u>aptable data <u>base</u>, is a high-performance, multithreaded, database management system for mainframe platforms where database performance is a critical factor. It is interoperable, scalable, and portable across multiple, heterogeneous platforms including mainframe, midrange, and PC.</p> |
| Adabas buffer description (ABD) | <p>An <i>Adabas buffer description (ABD)</i> describes a buffer segment used in a direct call. Specifically, it identifies the buffer type (format, record, multifetch, search, value, ISN, performance, or user), location,</p> |

size, and other pertinent information. The buffer itself or a pointer to the buffer may be included in the ABD. ABDs may only be used if buffers are required for an Adabas direct call that is made using the **ACBX direct call interface** provided with Adabas 8 (or later); they cannot be used if buffers are required for an Adabas direct call made using the **ACB direct call interface**.

For complete information about the structure of ABDs, read *Adabas Buffer Descriptions (ABDs)*, in the *Adabas Command Reference Guide*. For information about Adabas direct calls, read *Calling Adabas*, in the *Adabas Command Reference Guide*.

- Adabas direct access method (ADAM) The Adabas direct access method (ADAM) facility permits the retrieval of records directly from Data Storage without accessing the *inverted lists*. The Data Storage block number in which a record is located is calculated using a randomizing algorithm based on the ADAM key of the record. The use of ADAM is completely transparent to application programs and query and report writer facilities. Read *Random Access Using the Adabas Direct Access Method (ADAM)*, in *Adabas Concepts and Facilities Manual*, for more information.
- Adabas session The *Adabas session* starts when the nucleus is invoked and ends when the nucleus is terminated. An Adabas nucleus is invoked using job control specific to a particular operating system that contains Adabas startup, or ADARUN, parameters.
- Adabas trigger driver Part of the Adabas nucleus that has overall control of the Adabas triggers and stored procedures facility. It detects trigger and stored procedure requests and initializes the Natural trigger driver to execute them.
- Adalink A generic term for that part of the Adabas API (application program interface) that is specific to a particular teleprocessing (TP) monitor. The Adabas API is used to link application programs to Adabas. The actual module name depends on the TP monitor being used; for example, the module name for linking to a batch or TSO program is ADALNK and for CICS, the module name is ADALNC.. The term *Adalink* refers to the module appropriate for the given environment. The terms *Adalink(s)* and *ADALNK(s)* are synonyms.
- ADARUN The ADARUN control statement defines and starts the Adabas operating environment. The ADARUN control statement also starts Adabas utilities.

ADARUN:

- loads the ADAIOR module, which performs all database I/O and other operating-system-dependent functions;
- interprets the ADARUN parameter statements; then loads and modifies the appropriate Adabas nucleus or utility modules according to the ADARUN parameter settings; and
- transfers control to Adabas.

The ADARUN statement, normally a series of entries each specifying one or more ADARUN parameter settings, is specified in the DDCARD (z/OS or BS2000) or VSE CARD data set.

address converter

Adabas stores each database record in a Data Storage block identified by a relative Adabas block number (RABN). Each record's RABN is kept in a table called the address converter. The address converters, one for each database file, are stored in the Associator. Address converter entries are in ISN order (that is, the first entry tells the RABN location of data for ISN 1, the 15th entry holds the RABN location of data for ISN 15, and so on).

If spanned records are used, a secondary address converter is used to map the secondary ISNs to the RABNs of the Data Storage blocks where the secondary records are stored.

address space

The storage area assigned to a program task/work unit. In z/OS, an address space is a region; in VSE, a partition; and in BS2000, a task. In this documentation, the term *region* is used as a synonym for *partition* and *task*.

anchor file

In an expanded file, the anchor file is the file with the lowest ISN range. The anchor file's file number is the number assigned the entire expanded file.

For more information, read *Linking Physical Files in a Single Logical File* and *Expanded Files in Adabas DBA Tasks Manual*.

asynchronous trigger

An *asynchronous* trigger executes independently of the initiating Adabas command. Adabas command processing for the user continues uninterrupted while the triggered procedure executes as a separate process. The triggered procedure may execute after the user has already received the response from the initiating command.

Contrast this trigger type with a *synchronous trigger*.

B

base file When dealing with large object (LB) fields, Adabas stores the LB field values in a separate file, called a *LOB file*, that is tightly associated with the file containing the LB fields, which is called the *base file*. Behind the scenes, Adabas will store LB field values (except for very short ones) in the LOB file, but this is transparent to your application. Commands in application programs should always be directed against the base file; application programs need neither know nor care about the existence of a LOB file.

For more information, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

binary occurrence counter (BOC) The *binary occurrence counter* (BOC) stores the number of occurrences or multiple-value (MU) and periodic group (PE) fields in a file.

BUB The block of unreadable blocks. It is contained in the primary ASSO RABN 2 and the mirror ASSO RABN 9 for the primary ASSO, DATA, and WORK; in the primary ASSO RABN 9 and mirror ASSO RABN 2 for the mirror ASSO, DATA, and WORK; and the primary and the mirror PLOGn RABN 1 for PLOGn.

buffer flush Associator and Data Storage blocks in the buffer pool that have been updated since the last *buffer flush* have write flags set *on*. When the buffer is flushed, these blocks are written to Associator and Data Storage data sets, respectively. The flushed blocks remain in the buffer pool with their write flags set *off*.

A buffer flush can be synchronous or asynchronous. Update commands cannot be selected during a synchronous flush, but can be selected during an asynchronous flush.

C

CICS execution unit CICS applications that use Adabas services require an *Adabas CICS execution unit* to function.

Effective with Adabas 8.2 and later versions, the Adabas CICS execution unit is comprised of:

- the Adabas CICS stub, ADACICS
- an Adabas CICS names module, ACINAMES
- one or more Adabas task-related user exits (TRUEs), ADACICT
- a globals table associated with the stub module and the TRUE.

command ID

The command ID is an automatically generated or user-specified nonblank, nonzero value that:

- Prevents repetitive format buffer decoding by acting as an internal ID for decoded record formats;
- Tags ISN lists generated by the S_x command for later access, and saves ISN list overflow.

It is specified in Adabas control block fields (ACBCID or ACBXCID). For more information about command IDs, read *Command, Format, and Global Format IDs*, in the *Adabas Command Reference Guide*.

communicator

A routine for communicating between operating systems, making remote targets accessible. Entire Net-work is a communicator.

compressed records

Adabas retains data records in compressed form, using several different compression options. Data compression significantly reduces the amount of storage required by the data in a database. It also permits transmission of more information per physical transfer, resulting in greater I/O efficiency.

For complete information on data compression in Adabas, read *Compression*, in *Adabas Concepts and Facilities Manual*.

D

data compression

Data compression significantly reduces the amount of storage required. It also permits the transmission of more information per physical transfer, resulting in greater I/O efficiency.

Adabas retains data records in compressed form. Several compression options are supported. For complete information, read *Compression*, in *Adabas Concepts and Facilities Manual*.

database

In Adabas, a *database* is a group of related files.

A *physical database* identified by its database ID number (DBID) is defined with Adabas utilities. A *single physical database* is one set of Associator and Data Storage data sets identified by a single DBID. An Adabas nucleus running in an address space allows access to the physical files in the physical database.

database administrator (DBA)

Controls and manages the database resources. Tasks include defining database distribution, assigning a structure and resources, creating and maintaining programming and operation standards, ensuring high performance, resolving user problems, defining and teaching

user training, controlling database access and security, and planning for growth and the integration of new database resource applications and system upgrades. Also known as the *database analyst*.

descriptor

A descriptor is a search key. A *unique descriptor* has a different (i.e., unique) value for each record in the file. Entries are made in the Associator's inverted list for descriptor fields, adding disk space and processing overhead requirements.

Any field can be used within a selection criterion. When a field that is used extensively as a search criterion is defined as a descriptor (key), the selection process is considerably faster since Adabas is able to access the descriptor's values directly from the inverted list without reading any records from Data Storage.

A descriptor field can be used as a sort key in a search command, as a way of controlling a logical sequential read process (ascending or descending values), or as the basis for file coupling.

A portion of a field may be defined as a *subdescriptor*; combinations of fields or portions thereof may be defined as a *superdescriptor*; a user-supplied algorithm may be the basis of a *hyperdescriptor* or a *collation descriptor*; and a *sounds-like* encoding algorithm may be the basis of a *phonetic descriptor*, which may be customized for specific language requirements. For more information, see *Adabas Design in the Concepts and Facilities*.

E**elementary field**

An *elementary field* has only one value per record.

expanded file

To reduce the number of files accessed, Adabas allows you to link multiple physical files containing records of the same format together as a single logical file. This file structure is called an *expanded file*, and the physical files comprising it are the component files. An expanded file can comprise up to 128 component files, each with a unique range of logical ISNs. An expanded file cannot exceed 4,294,967,294 records.

For more information, read *Linking Physical Files in a Single Logical File* and *Expanded Files* in *Adabas DBA Tasks Manual*.

F**field**

In Adabas, a *field* is the smallest logical unit of information (e.g., current salary) that may be defined and referenced by the user.

Adabas supports four field types:

| | Single Value per Record | Multiple Values per Record |
|-----------------|-------------------------|----------------------------|
| Single Field | Elementary | MU |
| Multiple Fields | Group | PE |

The two basic field types are elementary and multiple-value. An *elementary* field has only one value per record. A *multiple-value* (MU) field can have up to about 65,534 values, or occurrences, in a single record. Each multiple-value field has a *binary occurrence counter* (BOC) that stores the number of occurrences.

When two or more consecutive fields in the FDT are frequently accessed together, you can reference them by defining a *group* field. Other than its level and Adabas short name, a group field has no attributes defined. It immediately precedes its member fields in the FDT. A higher field *level* number is used to assign the member fields to the group field. Adabas supports up to seven field levels.

A *periodic* (PE) group field defines consecutive fields in the FDT that repeat together in a record. Like the members of a normal group field, PE members immediately follow the PE group field, have a higher level number than the PE field, and can be accessed both individually and as a group. Each PE has a BOC that stores the number of occurrences. A periodic group may be repeated up to about 65,534 times per record and may contain one or more multiple-value fields.

The actual limit to the number of occurrences of MU fields and PE groups is derived from the maximum data storage record length (the ADALOD MAXRECL parameter), which defaults to the size of the data storage block minus 4.

Field definition table (FDT) A table that defines each file's record structure and content. There is one FDT for each database file. FDTs, stored in the Associator's fixed area, have three parts: the first is a list of the file's fields in physical record order, the second part is a *quick index* to the records in the first part, and the third part defines the files sub/superfields and sub-/super-/hyper- and phonetic descriptors.

A portion of a field (*subfield*) or any combination of fields (*superfield*) may be defined as an elementary field. Subfields and superfields may be used for read operations only. They may only be changed by updating the original fields.

- file** In Adabas, a *file* is a group of related records that have the same format (with some exceptions; see *Adabas Design* in the *Concepts and Facilities*).
- The disk storage space allocated to a single Adabas database is segmented into *logical* Adabas files. A certain part of the overall space within the database is allocated to each logical file. When this space is filled with records from that file, Adabas automatically allocates more space to the file from the common free space pool. This dynamic space allocation, together with the dynamic recovery of released space, allows Adabas databases to run without intervention for long periods of time.
- A *physical* Adabas file contains database records. Each physical file is identified by a file number. The number of physical files (and physical file numbers) per physical database is limited to 5000 or one less than the ASSOR1 block size, whichever is lower.
- An *expanded file* is a logical file comprising physical files in one or more locations. The physical files have the same field definition table (FDT), but non-overlapping ISN ranges. The data content of at least one field (the field value criterion) determines the physical file in which a data record is located.
- A *multiclient file* is an Adabas file with records accessible through an owner ID. Only records identified by the same individual or group owner ID can be accessed or updated by the related user. This allows the file to be maintained as a single Adabas file, but to be used as multiple logical files (each record group belonging to an owner ID is a *logical file*). *Super owner IDs* allow access to all records in the file.
- file control blocks (FCBs)** The control block Adabas assigns to each file in a database. There is one FCB per file. The FCB identifies the file's physical location and associated RABNs. The FCBs are part of the Associator.
- format ID** The format ID is an ID that is assigned a decoded format buffer. The format buffer is stored once for each user and can then be referenced by its format ID and reused. Individual format IDs are stored once for each user and used only by that user. The format buffers are identified by a combination of unique format ID and terminal ID.
- Contrast this with *global format IDs* which store a decoded format buffer once globally and share it with many users running on the same Adabas nucleus.

The format ID is specified in Adabas control block fields (ACBADD5 or ACBXADD5). For more information about format IDs, read *Command, Format, and Global Format IDs*, in the *Adabas Command Reference Guide*.

G

| | |
|-------------------------------|---|
| general control blocks (GCBs) | The two control blocks Adabas assigns to each database. The GCBs identify the database and where it resides. The GCBs are part of the Associator. |
| generation | <p>Information is stored on the Recovery log (RLOG) by generation, the logical unit used for recovery. A <i>generation</i> includes all activity between consecutive operations of:</p> <ul style="list-style-type: none"> ■ ADASAV SAVE/RESTORE (database), ■ RESTORE GCB, or ■ SAVE DELTA/RESTORE DELTA (database). <p>For more information, read <i>Generation: The Unit of Recovery</i> in <i>Adabas Utilities Manual</i>.</p> |
| global format ID | <p>A global format ID is an ID that is assigned a decoded format buffer. The format buffer is stored once globally and can then be referenced by its format ID and reused. Global format IDs are stored once globally and can be used by any user running on the same Adabas nucleus. The format buffers are identified by a unique format ID.</p> <p>Contrast this with <i>format IDs</i> which store a decoded format buffer once for each user and are used only by that user.</p> <p>The global format ID is specified in Adabas control block fields (ACBADD5 or ACBXADD5). For more information about format IDs, read <i>Command, Format, and Global Format IDs</i>, in the <i>Adabas Command Reference Guide</i>.</p> |
| global parameters | Some ADARUN parameters are <i>global parameters</i> ; that is, they must have the same values for all nuclei in a cluster. In addition, some of the global parameters are set at session initialization and cannot be changed. |
| global transaction | A <i>global transaction</i> is a unit of work that involves changes to resources under the control of more than one database operating in one or more operating system images. |

I

| | |
|--------------------------------|---|
| ID | An abbreviation of <i>target ID</i> , a unique identifier used for directing Adabas calls to their targets. |
| ID table | A reference data list maintained for all active targets within the boundaries of one operating system. The ID table is located in commonly addressable storage. |
| IIBS | The <i>isolated ID bit string</i> , a 256-bit (32-byte) string contained in the ID table header. Each bit corresponds in ascending order to a logical ID. If the bit has the value "1", the corresponding ID is isolated. |
| I/O buffer | <p>The Adabas <i>I/O buffer</i> area, which can be resized for each Adabas session, contains the most frequently used data and data relationships; it helps to minimize physical input/output (I/O) activity and thus saves computer time. It is loaded into main memory at startup, along with the Adabas nucleus.</p> <p>The buffer contains blocks read from the database and blocks to be written to the database:</p> <p>For blocks read from the database, a <i>buffer algorithm</i> ensures that the most frequently accessed blocks stay in memory. When a block from the database is needed, the buffer content is checked to determine if the block is already in memory, thus avoiding unnecessary reads.</p> <p>Multiple updates are accumulated in a block before it is written (flushed) to external storage.</p> |
| internal sequence number (ISN) | <p>Every Adabas record is assigned an internal sequence number (ISN) to identify the record. Each record keeps its original ISN, regardless of where it is located.</p> <p>Records in a physical database file have four-byte ISNs ranging from MINISN to MAXISN. In replicated files, a record has the same ISN in all file copies. In partitioned files, the ISN ranges are non-overlapping for each physical file.</p> |
| inverted list | A list, organized by descriptor value rather than by ISN which is used to resolve Adabas search commands and read records in logical sequence. It is built and maintained for each field in an Adabas file that is designated as a key field or <i>descriptor</i> . The list comprises the normal index (NI) and as many as 14 upper indexes (UI). |
| isolated ID | The ID of an isolated target, which can be specified by the user as a logical ID. An isolated ID must be greater than zero and less than |

256. The isolated ID is interpreted as a physical ID for addressing the target.

isolated target

A target called directly by a user.

L

large object field (LB field) A large object field (LB field) is an alphanumeric field that can have a theoretical size of up to 2 GB. Such fields can be used, for example, to store documents (for example, HTML, XML, Microsoft Word, or PDF documents), pictures (for example JPG or BMP files), or other data conglomerates in single fields in the database.

For more information, read *Large Object Option LB*, in *Adabas Concepts and Facilities Manual* and *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

LOB file Adabas stores large object field values in a separate file, called a *LOB file*, that is tightly associated with the file containing the LB fields, which is called the *base file*. Behind the scenes, Adabas will store LB field values (except for very short ones) in the LOB file, but this is transparent to your application. Commands in application programs should always be directed against the base file; application programs need neither know nor care about the existence of a LOB file.

For more information, read *Large Object (LB) Files and Fields*, in *Adabas DBA Tasks Manual*.

LOB group The combination of a *base file* and its associated *LOB file* is referred to as a *LOB group*.

logical ID A user's identifier of targets to which a message is directed. It must be greater than "0" and less than "256" (either explicitly or implicitly, the content of the first byte of ACBFNR is a logical ID).

logical duplication The process of permanently storing information that is implied by the contents of other records. For example, if a credit control routine needs the sum of all invoices for its customers, the information can be quickly accessed if totals have been maintained permanently in customer records, rather than reading and totaling the relevant invoices with each request, since that process might involve random access of a large number of records.

logical transaction The smallest unit of work (as defined by the user) that must be performed in its entirety to ensure that the information contained in the database is logically consistent.

A logical transaction may comprise one or more Adabas commands that together perform the database read/update required to complete a logical unit of work. A logical transaction begins with the first command that places a record in hold status and ends when an ET (end transaction), BT (back out transaction), CL (close), or OP (open) command is issued for the same user.

logically deleted fields Fields that have been logically removed from a database file using the ADADBS DELFN utility function. Logically deleted fields have been removed from the FDT for the file, but the data for the fields is retained in the database.

M

MIRTAB The mirror table, which indicates the status of primary RABNs. It is contained in the primary and the mirror ASSO RABN 7 for ASSO, DATA, and WORK; and the primary and the mirror PLOG n RABN 1 for PLOG n .


multiple-value (MU) field A *multiple-value* (MU) field can have multiple values. The number of occurrences of each MU field in a file can be 191 or up to about 65,534 values, or occurrences, in a single record. The actual limit is derived from a number of factors. For more information on the actual limit of MU and PE group fields, read *MU and PE Options and Field Types*, in *Adabas Concepts and Facilities Manual*. Each MU field has a *binary occurrence counter* (BOC) that stores the number of occurrences.



Note: The use of more than 191 MU or PE groups in a record must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS MUPEX and MUPECOUNT parameters.

N

Natural trigger driver The Natural nucleus component that runs as a batch Natural environment. The Natural trigger driver is responsible for the actual execution of both triggers and stored procedures. Operating in conjunction with the Adabas trigger driver, it handles the interprocess


| | |
|---------------------------|---|
| | communications between the Adabas nucleus and the Natural subsystem that executes the procedure. |
| non-DB target | A target that is not an Adabas nucleus. Access and X-COM are non-DB targets. |
| non-participating trigger | <p>A <i>non-participating</i> trigger can be synchronous or asynchronous. It has a user ID that is different from that of the Adabas user queue element (UQE) that identifies the initiating command; thus, it does not participate in the initiating command's transaction logic.</p> <p>Contrast this trigger type with a <i>participating trigger</i>.</p> |
| normal index (NI) | An index for a particular descriptor in the <i>inverted list</i> . The normal index has an entry for each descriptor value. The entry contains the value itself, the number of records in which the value occurs, and the ISNs of those records. |
| nucleus | <p>The <i>nucleus</i> is a set of programs that drive Adabas, coordinate all work, and translate user program statements into Adabas commands. All programs access Adabas files through the nucleus. All database activities such as data access and update are managed by the Adabas nucleus. In most cases, a single nucleus is used to manage a single physical database.</p> <p>Adabas Parallel Services makes it possible to run a cluster of up to 31 Adabas nuclei on a single operating system against a single database. Adabas Cluster Services supports the IBM parallel sysplex environment making it possible to run a cluster of up to 32 Adabas nuclei on multiple operating systems set up as a sysplex. Again the cluster runs against a single database.</p> <p> Note: See <i>Optional Extensions</i> in the <i>Concepts and Facilities</i> for information about running multiple nuclei against a single physical database under a single operating system image (ADASMP) or under multiple z/OS images (Adaplex+).</p> |

O

| | |
|-------------------|---|
| operator commands | <p>Adabas operator commands are entered during an Adabas session or during utility operation to:</p> <ul style="list-style-type: none">■ terminate an Adabas or user session;■ display nucleus or utility information;■ log commands into CLOG; |
|-------------------|---|

- change Adabas operating parameters or conditions.

P

| | |
|---------------------------|---|
| participating trigger | <p>A <i>participating</i> trigger must be synchronous. Such a trigger results in the execution of a procedure that is assigned the same user ID as the Adabas command that caused the participating trigger to be fired; thus, it can participate fully in the logic of the transaction.</p> <p>Contrast this trigger type with a <i>non-participating trigger</i>.</p> |
| periodic (PE) group field | <p>A <i>periodic (PE) group field</i> defines consecutive fields in the FDT that repeat together in a record. Like the members of a non-periodic group field, PE members immediately follow the PE group field, have a higher level number than the PE field, and can be accessed both individually and as a group. Each PE has a <i>binary occurrence counter</i> (BOC) that stores the number of occurrences.</p> <p>A periodic group may be repeated 191 or up to about 65,534 times per record and may contain one or more <i>MU fields</i>. Occurrences or values that are not used require no storage space. The actual limit is derived from a number of factors. For more information on the actual limit of MU and PE group fields, read <i>MU and PE Options and Field Types</i>, in <i>Adabas Concepts and Facilities Manual</i>.</p> <p> Note: The use of more than 191 MU or PE fields in a file must be explicitly allowed for a file (it is not allowed by default). This is accomplished using the ADADBS MUPEX function or the ADACMP COMPRESS MUPEX and MUPECOUNT parameters.</p> |
| physical ID | <p>The identifier of a target. It must be greater than "0" and less than "65,536". A database ID (DBID) is a physical ID.</p> |
| post-command trigger | <p>A <i>post-command</i> (or "post-") trigger executes after the initiating Adabas command is processed by the Adabas nucleus. The triggered procedure executes only if the return code for the command from the Adabas nucleus is zero. If the return code is non-zero, no checking for triggers is done, and processing continues in the normal way. For a successfully executed command, the command is checked for any triggers before processing of the command completes.</p> <p>Contrast this trigger type with a <i>pre-command trigger</i>.</p> |

| | |
|---------------------|--|
| post-trigger queue | Contains the procedures waiting to be executed for the post-command triggers that have been fired. For more information, read <i>post-command trigger</i> . |
| pre-command trigger | <p>A <i>pre-command</i> (or "pre-") trigger is executed before the initiating Adabas command is processed by the Adabas nucleus. Before an Adabas command is executed, a check is made to verify whether a trigger should be fired.</p> <p>Contrast this trigger type with a <i>post-command trigger</i>.</p> |
| pre-trigger queue | Contains the procedures waiting to be executed for the pre-command triggers that have been fired. For more information, read <i>pre-command trigger</i> . |
| primary ISN | The ISN assigned the primary record in a spanned record . |
| primary record | <p>Adabas version 8 (or later) introduces the concept of spanned records. The first physical record in a spanned record is called the <i>primary record</i> and contains the beginning of the compressed record. A spanned record is comprised of one primary record and one or more secondary records.</p> <p>For complete information about spanned records, read <i>Spanned Record Support</i> in <i>Adabas Concepts and Facilities Manual</i>.</p> |
| procedure | A <i>procedure</i> is a Natural subprogram that is written and tested using standard Natural facilities. The same types of parameters are passed to the subprogram whether it is a <i>trigger</i> or a <i>stored procedure</i> . |
| pseudo-cylinder | The logical cylinder on a fixed-block-addressed (FBA) device that has no actual DASD cylinder. |

Q

| | |
|-------------------------|--|
| quasi-reentrant program | A program that may alter its code between calls to TP monitor functions. When a monitor function is invoked, all user data must be saved in a special work area obtained from the TP monitor system. |
|-------------------------|--|

R

| | |
|-------------------------------------|---|
| RABN (relative Adabas block number) | The basis of Adabas storage addressing. Adabas divides Data, Associator, and Work disk space into device-dependent logical blocks. The blocks in each of the three areas are numbered consecutively in ascending sequence beginning with RABN 1. The data blocks themselves as well as their addresses are referred to throughout |
|-------------------------------------|---|

Software AG publications as "RABNs". In other words, the sentence, "Adabas assigns RABNs 1-10 to the Associator" means ten Adabas storage blocks, numbered "1" through "10", are assigned to the Associator, whereas "Adabas assigns 50 RABNs to the Associator" means 50 blocks of storage with unspecified RABN numbers are assigned.

| | |
|---------------|---|
| record | In Adabas, a <i>record</i> is a collection of related fields that make up a complete unit of information (e.g., all the payroll data for a single employee). |
| record buffer | The portion of the calling program's parameter area, called the user buffer, that contains the data transferred during Adabas read, search, and update operations. When reading data field definitions, Adabas also returns the field definition information in the record buffer. |
| region | Collectively refers to storage space allocated to user jobs by z/OS, VSE, and BS2000 operating systems. |
| reset | A flag bit is said to be reset when it contains "0". |
| restart point | The <i>restart point</i> is the latest point of execution that can be restored from the Temp data set. |
| router | A central routine for communication within the boundaries of one operating system. The routine is called by users with Adalink routines, and by targets with ADAMPM. The router's main purpose is to transfer information between the Adalink and Adabas. The router also maintains the ID table. BS2000 environments divide router functions among Adalink or other Adabas functions. The Adabas SVCs in z/OS and VSE are examples of routers. |

S

| | |
|------------------|---|
| secondary ISN | The ISN assigned a secondary record in a spanned record . |
| secondary record | <p>Adabas version 8 (or later) introduces the concept of spanned records. The remaining physical records after the <i>primary record</i> in a spanned record are called <i>secondary records</i> and contain the rest of the data of the logical record. A spanned record is comprised of one primary record and one or more secondary records.</p> <p>For complete information about spanned records, read <i>Spanned Record Support</i> in <i>Adabas Concepts and Facilities Manual</i>.</p> |
| service | A processor of Adabas calls and issuer of replies. An Adabas nucleus is an example of a service. See also <i>target</i> . |

| | |
|---------------------|---|
| session | <p>A <i>user session</i> is a sequence of Adabas calls optionally starting with an OP command and ending with a CL command. A <i>user</i> is either a batch mode program or a person using a terminal. The uniqueness of each user is assured by the user ID, a machine, an address space, and a terminal ID.</p> <p>An <i>Adabas session</i> starts when Adabas is activated and continues until Adabas is terminated. During this time, the Adabas nucleus creates a sequence of protection entries in exact historical sequence reflecting all modifications made in the database. The sequence of protection entries is written to the Work data set (part 1) and to a protection log in blocks. Each block contains the nucleus session number, a unique block number, and a time stamp.</p> |
| set | <p>A flag bit is said to be set when it contains "1".</p> |
| spanned records | <p>Adabas version 8 (or later) introduces the concept of <i>spanned records</i>. In the database, the logical record is split into a number of physical records, each part fitting into a single Data Storage (DS) block. The resulting physical records are each assigned individual ISNs. Spanned records are comprised of one <i>primary record</i> and one or more <i>secondary records</i>.</p> <p>For complete information about spanned records, read <i>Spanned Record Support</i> in <i>Adabas Concepts and Facilities Manual</i>.</p> |
| stored procedure | <p>A <i>stored procedure</i> is a procedure executed by Adabas, but invoked directly or manually by an application.</p> |
| STPLNK | <p>The stored procedure link routine used to invoke a stored procedure request.</p> |
| STPRBE | <p>The record buffer extraction routine, which is called by a procedure and used to retrieve the parameters passed by the calling routine.</p> |
| STPUTRAK | <p>A routine that tracks every request to invoke a procedure and every procedure invoked. You can use this routine to write trace messages or audit triggers processing for each subsystem. A default STPUTRAK routine is supplied with the Adabas triggers and stored procedures facility.</p> |
| subtask | <p>A task that is spawned from a parent task.</p> |
| synchronous trigger | <p>A <i>synchronous</i> trigger requires a suspension of Adabas command processing for that user. The initiating Adabas command is suspended until the triggered procedure completes execution. It is possible that the results of the procedure will affect the Adabas command.</p> |

Contrast this trigger type with an *asynchronous trigger*.

system field

A system field is a field in an Adabas file whose value is automatically set by the Adabas nucleus when records are inserted or updated in the file. Optionally, some system field values are only set when records are inserted. System fields are fields that store information such as the job name of the user making the update, the eight-byte user ID of the user, the session ID of the user, or the time at which the update was made.

For more information, read *System Fields in Adabas Concepts and Facilities Manual*.

T

target

A receiver of Adabas calls. A target maintains a command queue, and communicates with routers using ADAMP. A target is also classified as a *service*. The Adabas nucleus is a target.

thread

Adabas provides multithreaded processing to maximize throughput. If I/O activity suspends command processing in an active thread, Adabas automatically switches to another thread. The user may set the number of 8-kilobyte threads to be used for an Adabas session up to a maximum of 250.

thread time

The elapsed time that was needed by the nucleus to process the command. This does *not* include the times when the Adabas thread executing the command was waiting on Adabas I/O operations or other resources; it *does* include the times when the thread was waiting for a processor so it could execute the code. The time is measured in 1/4096 microsecond units and is in binary format.

The Adabas thread time is reported in the ACBXCMDT (command time) field of the Adabas call.

transaction

Adabas data protection, recovery, and user restart is based on the concept of a *logical transaction*: the smallest unit of work (as defined by the user) that must be performed in its entirety to ensure that the information contained in the database is logically consistent.

A logical transaction may comprise one or more Adabas commands that together perform the database read/update required to complete a logical unit of work. A logical transaction begins with the first command that places a record in hold status and ends when an ET (end transaction), BT (back out transaction), CL (close), or OP (open) command is issued for the same user.

The ET command must be issued at the end of each logical transaction. Successful execution of an ET command ensures that all the updates performed during the transaction are physically applied to the database, regardless of subsequent user or Adabas session interruption.

Updates performed during transactions for which ET commands are not successfully executed are backed out, either manually by issuing the BT command or automatically by the Autobackout routine (see *Using Adabas in the Concepts and Facilities*).

| | |
|------------------------------|---|
| translator | A process that converts a logical ID of a user's Adabas call into a corresponding physical ID for a target. |
| trigger | A procedure that is executed automatically by Adabas when a specified set of criteria is met. The set of criteria is determined for each Adabas command sent to the DBMS and is based on the target file number and optionally the command type or field (or both). The command type refers to the commands FIND, READ, STORE, UPDATE, and DELETE. The field must be in the corresponding format buffer of the command. For information about specific types of triggers, read <i>pre-command trigger</i> , <i>post-command trigger</i> , <i>asynchronous trigger</i> , <i>synchronous trigger</i> , <i>participating trigger</i> , and <i>non-participating trigger</i> . |
| trigger definition | The name and attributes of the procedure to be executed, and the selection criteria that compose the triggering event (Adabas command type, file name or number, and field name). |
| Trigger Maintenance Facility | A Natural application accessible from Adabas Online System (AOS). A system of menus allows you to create and maintain trigger definitions, define the Adabas triggers profile, and monitor and to some extent control trigger activity within the nucleus. |
| trigger table | A buffer in the Adabas nucleus that holds the trigger definitions. |
| two-phase commit processing | <i>Two-phase commit processing</i> ensures commercial transaction integrity by securing or rejecting transactions as a whole across separately managed resources. |

U

| | |
|------------------|--|
| upper index (UI) | An index in the inverted list automatically created by Adabas as required to manage the indexes on the next lower level. Upper indexes are used by Adabas to increase search efficiency. The first level UI, |
|------------------|--|

like the NI it manages, contains entries for only one descriptor in each index block. All other UI levels contain entries for all descriptors in each index block. UIs require a minimal amount of space: two blocks is the minimum.

- user** A batch or online application program that generates Adabas calls and uses an Adalink for communication.
- user program** A batch or online application program that generates Adabas calls and uses an Adalink for communication.
- user session** A *user session* is either a batch mode program or a person using a terminal. A user session can occur only during an *Adabas session*; that is, when the Adabas nucleus is active. It is a sequence of Adabas calls optionally starting with an open user session (OP) command and ending with a close user session (CL) command.
- utility session** A *utility session* is executed in batch, or online using the Adabas Online System. Some utilities require the Adabas nucleus to be active; others do not. ADARUN startup parameters are also used for executing utilities.

V

- version** A product version is identified by the first two digits of the versioning number. Software AG distinguishes between major and minor versions according to the amount of functionality or technology added to the product. All other digits indicate correction levels.
- In the product documentation, the notations *vrs*, *vr*, or simply *v* are often used as placeholders for the current product version, for example, in data set or module names.

| Placeholder | Meaning | Definition |
|-------------|---------|--|
| <i>v</i> | version | Major Version The first digit of the product version number indicates major architecture and functionality implementation or enhancement that adds value to the product. |
| <i>r</i> | release | Minor Version The second digit of the version number indicates new or enhanced functionality that adds value to the product. |

| Placeholder | Meaning | Definition |
|-------------|--------------------------|--|
| s | system maintenance level | <p>Correction Level</p> <p>Correction levels contain error corrections only, without new functionality, including documentation of all modifications and repairs.</p> <p>In case it is necessary to include functional changes into a correction level, an exception handling process ensures that corresponding quality assurance activities are triggered. These functional changes are documented. The main goal is to avoid impacts when you install such a correction level.</p> <p>The third number of an Adabas version denotes the system maintenance level.</p> <p>On certain platforms supported by Adabas, additional levels may exist, such as update package, patch level, service pack and hot fix.</p> |

vrs

See *version*.

W

work load balancing

Adabas Parallel Services attempts to balance the work load evenly across the cluster nuclei based on the number of users and the number of commands processed by each of the nuclei.

Index

(see database administrator (DBA)) (see Field definition table (FDT)) (see internal sequence number (ISN)) (see relative Adabas block number (RABN))

A

Adabas
 definition of, 1
Adabas session
 definition of, 17
Adalink
 definition of, 2
ADARUN
 definition of, 2
address converter
 definition of, 3
address space
 definition, 3

B

binary occurrence counter
 definition of, 7
buffer flush
 definition of, 4
 from I/O buffer, 10

C

cluster of nuclei, 13
commands
 operator, 13

D

data compression
 definition of, 5
database
 definition of, 5
 definition of physical, 5
 definition of single physical, 5
database administrator (DBA)
 definition of, 5
DBA, 5
descriptor
 definition of, 6
 value to order inverted list, 10

E

expanded files
 definition of, 8

F

FDT, 7
Field definition table (FDT)
 definition of, 7
field type
 elementary, 7
 group, 7
 multiple value, 7
 periodic group, 7
fields
 definition of, 6
 levels, 7
files
 definition of, 8
 physical, 8

G

global transaction
 definition of, 9
group
 field type, 7

I

I/O buffer
 algorithm for, 10
 definition of, 10
internal sequence number (ISN)
 definition of, 10
ISN, 10

M

multiclient files
 definition of, 8
multiple-value fields
 definition of, 7

N

nucleus

definition of, 13
nucleus cluster, 13

O

operator commands, 13

P

periodic groups
definition of, 7
procedure
definition of, 15

R

RABN, 15
record buffer
definition, 16
records
definition of, 16
region
definition of, 16
relative Adabas block number (RABN)
definition of, 15
router
definition, 16

S

service
definition, 16
session
Adabas, 17
user, 17
stored procedure
definition, 17
STPRBE routine
definition, 17
STPUTRAK routine
definition, 17
subfield, 7
superfield, 7
system fields
definition, 18

T

target
definition, 18
thread
definition of, 18
thread time
definition of, 18
transaction
definition of, 18
translator
definition of, 19
trigger
asynchronous type, 3
definition, 19
non-participating type, 13
participating type, 14

post-command type, 14
pre-command type, 15
synchronous type, 17
Trigger Maintenance Facility, 19
trigger table, 19

U

user program
definition of, 20
user session
definition of, 17