Using Adabas

Generally, Adabas uses between 10% and 50% of the data processing resources (disk storage, CPU time, elapsed processing time) used by other database management systems. Since fewer hardware facilities are used, more can be accomplished with less. Very large online applications using several terabytes of data have been successfully implemented, with thousands of terminal workstations, and with the response times and the cost of much smaller systems.

This chapter covers the following topics:

- Accessing a Database from Programs
- Maintaining Database Integrity

Accessing a Database from Programs

Adabas access is field-oriented: user programs access and retrieve only the fields they need. User program statements invoke Adabas search and retrieval operations automatically.

- Direct Call Interface
- Complex Searches
- Access Methods
- Using Triggers and Stored Procedures
- Universal Encoding Support (UES)

Direct Call Interface

Adabas provides a powerful and flexible set of direct call commands for performing database operations. Adabas direct call commands provide a direct interface to the Adabas database when Natural or another fourth-generation database language is not being used.

The commands can be categorized by function:

- database query
- read (Data Storage or Associator)
- database modification
- logical transaction processing
- special commands

Database Query Commands (Sx)

Database query commands (S1/S4, S2, S5)) search for and return the ISNs of specified records or record groups according to specified search criteria. Other commands in this category (S8, S9) sort the resulting ISN lists in preparation for later operations.

The ISN lists resulting from any S*x* command may be saved on the Adabas Work data set for later retrieval during the user session.

In most cases, these commands do not actually read the database; ISNs are read directly from the Associator's inverted lists. Options allow the ISN's record to be placed in hold status to prevent it from being updated by other programs until the record is released; if desired, additional field values contained in the first ISN's record can be read from Data Storage.

Read Commands (Lx)

The L1 through L6 commands are used to read actual records from Data Storage. Depending on the specified command and its options, records are read individually

- in the sequence in which they are stored;
- in the order of an ISN list created by one of the database query commands; or
- in logical sequence according to a user-specified descriptor.

Read *Sequential Access* for more information about sequential access methods.

A hold option allows the database records to be locked until released by a separate command or at transaction end.

The L9 and LF commands read information directly from the Associator inverted lists or field definition tables (FDTs), returning either the inverted list values for a specified descriptor or the field definitions for a specified file in the database.

Database Modification Commands (A1, E1, N1/N2)

Database modification commands (A1, E1, and N1/N2) change, delete, or add database records and update the related Associator lists accordingly. ISNs can be assigned to new records either by the user or by Adabas.

The inverted lists and the address converter are automatically maintained by Adabas. When the user supplies a new value for a descriptor, either in a new record or as part of a record update, Adabas performs all necessary maintenance of the inverted lists. When a new record is added or a record is deleted, the address converter is appropriately updated. These operations are completely transparent to the user.

Logical Transaction Control Commands (ET/BT)

An Adabas logical transaction defines the logical start (BT) and end (ET) of the database operation being performed. If the user operation or Adabas itself terminates abnormally, these commands provide the capability for restarting a user, beginning with the last unsuccessfully processed transaction. ET/BT commands

- define the transaction start and end;
- restore pretransaction conditions if a situation occurs that prevents successful completion of the transaction; and
- read program-specified user data written during the transaction sequence.

Programs that use these commands are called ET logic programs. Although not required, Software AG recommends that you use ET logic. Read *Transaction Logic* for more information.

Special Commands

Special commands perform many of the housekeeping functions required for maintaining the Adabas database environment. Commands in this group

- open (OP) and close (CL) a user session (but do not control a transaction);
- write data protection information, user data, and checkpoints (C1, C3, C5); and
- set (HI) and release (RI) record hold status.

In addition, the RC command releases one or more command IDs currently assigned to a user, or deletes one or all global format IDs.

The RE command reads user data previously stored in an Adabas system file by C3, CL, or ET commands.

Complex Searches

In many large database systems, the time required to process complex searches is often excessive. Adabas efficiently solves this problem. Many Adabas applications are currently in use with up to 150 complex selection criteria created dynamically. Data is retrieved immediately from files with more than 50 million records.

Multifile Searching

It is often necessary to perform a multifile search in order to resolve an inquiry. Multifile searching can be accomplished using multiple search commands, physically coupled files, or soft file-coupling. Read *Coupled Files* for information about coupled files.

Multiple search commands may be used in which a value retrieved in one command is used as the search value for the next command. This process is not restricted to two files.

Multi-index Searching

Fuzzy matching (i.e., retrieving data based on *similar* rather than on *exact* search criteria) can be implemented using hyperdescriptors. Hyperdescriptors allow multiple virtual indexes, meaning that several different search index entries can be made for a single data field. Read *Hyperdescriptor* for information about hyperdescriptors.

Access Methods

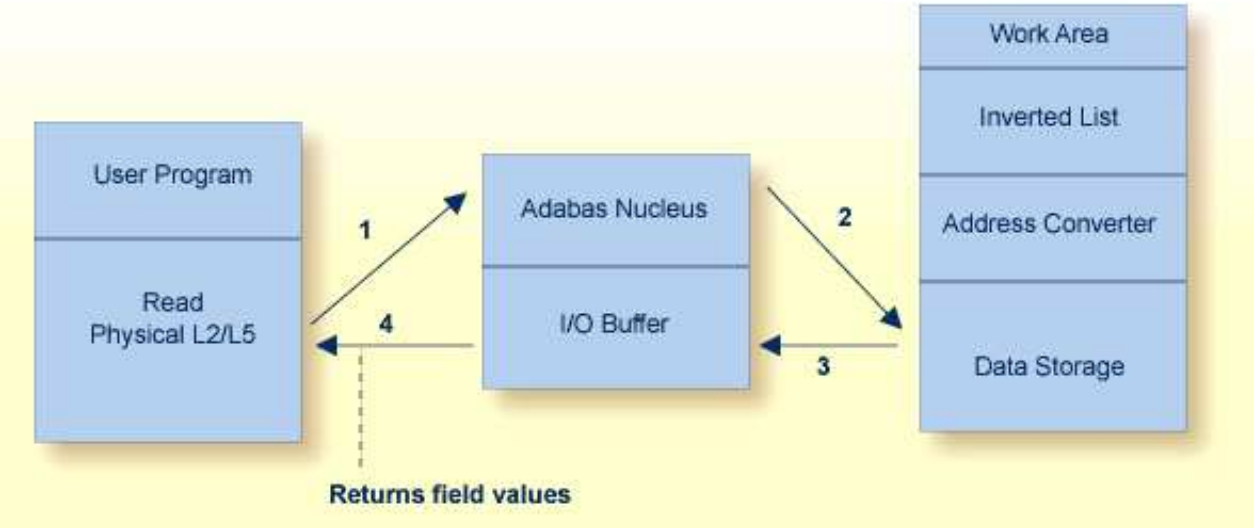
Adabas supports both sequential and random access methods. Different calls use different Adabas access paths and components; the most efficient method depends on the kind of information you want and the number of records you need to retrieve.

- Sequential Access
- Random Access
- Random Access Using the Adabas Direct Access Method (ADAM)

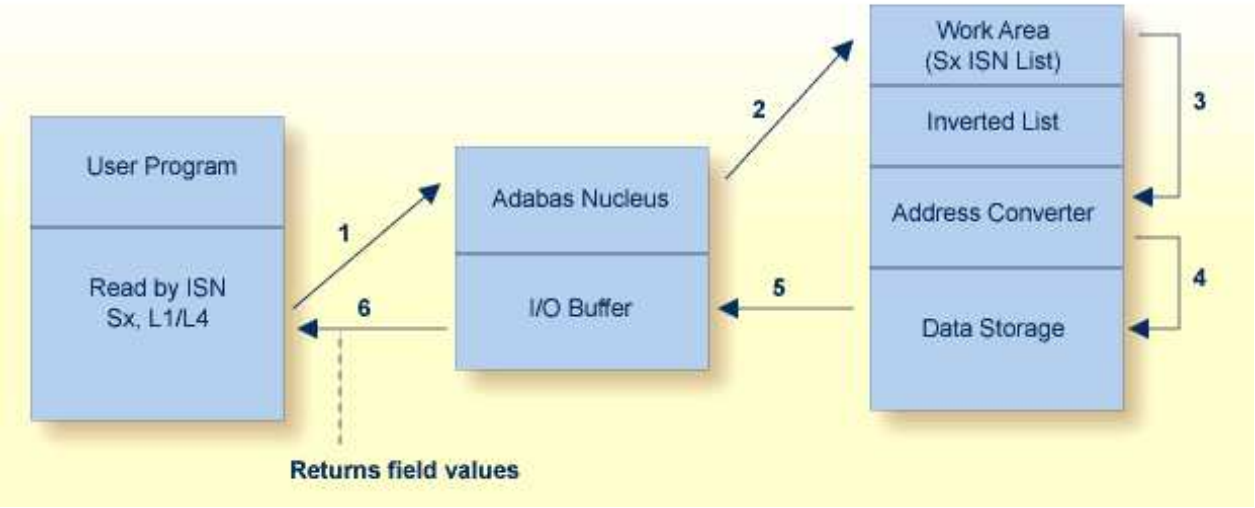
Sequential Access

Physical sequence retrieves every record in a file in the order the records are stored in Data Storage. You can limit the fields within each record for which values are to be returned. You can also specify a starting ISN: the sequential pass then begins at the first record physically located after the record identified by the specified ISN.

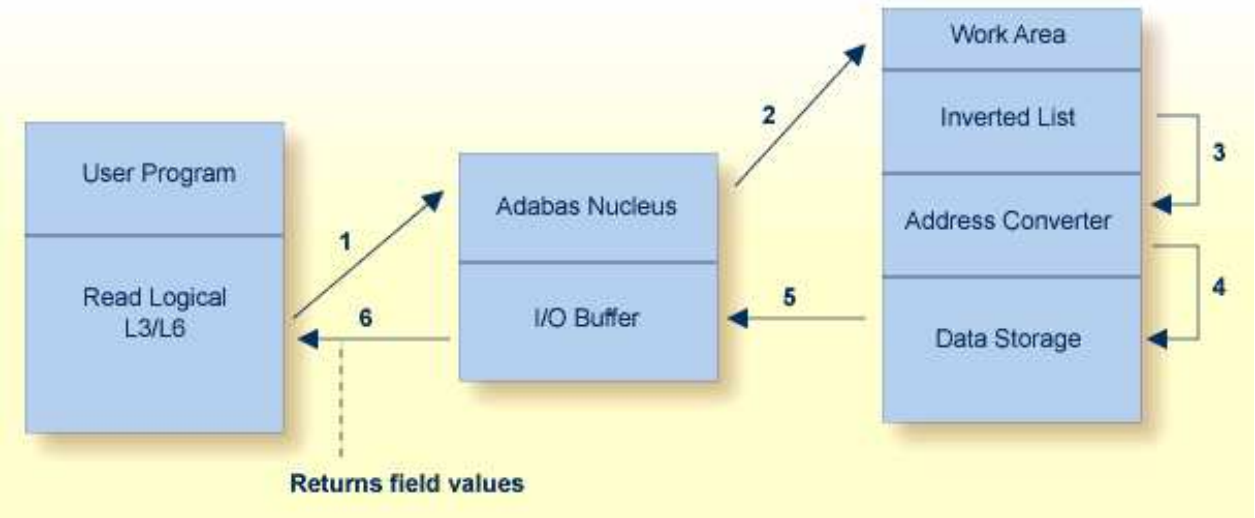
Adabas bypasses the Associator and goes directly to Data Storage, reading the first data block (or the first record following a specified ISN) and continuing in consecutive sequence until the last block is read. Physical sequence is the fastest way to process a large volume of records.



ISN sequence retrieves records in ISN order. Adabas uses database query commands (S*x*) to build and sort ISN lists, which can then be read using L1/L4 commands with the GET NEXT option. When reading, Adabas uses the address converter to find the RABNs of each ISN in the list and then reads and returns the records from Data Storage.

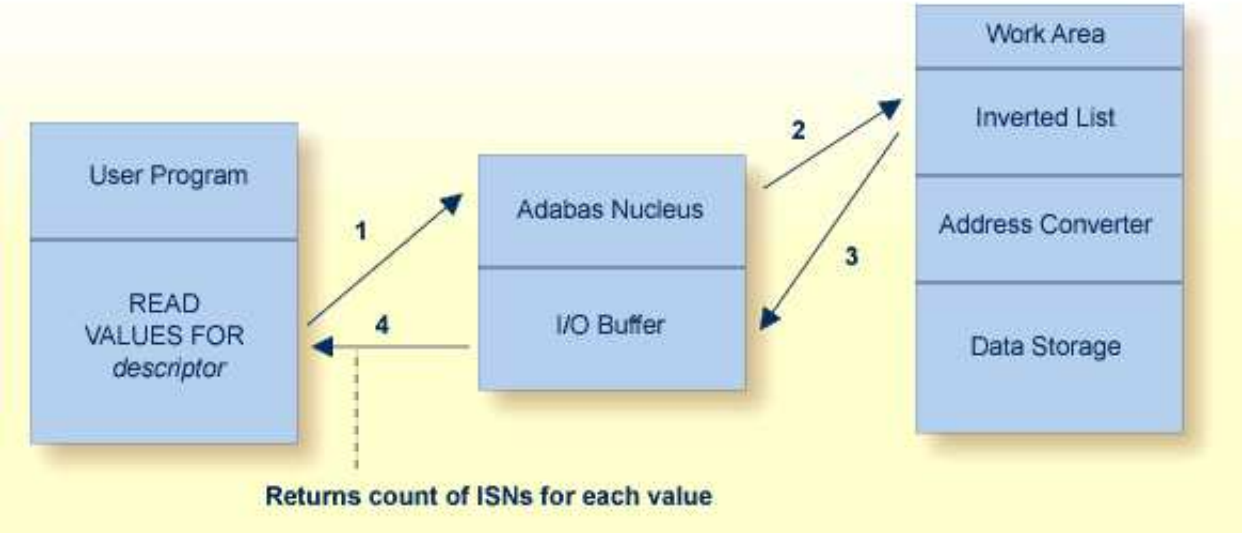


Logical sequence retrieves records by descriptor value. Adabas finds the value(s) in the inverted list, uses the address converter to find the RABNs of the ISNs related to the value, and retrieves the records from Data Storage.



Reading in logical sequence retrieves all the records related to a single value or a range of values of the specified descriptor. It returns the records sorted in ascending/descending order by descriptor value, and in ascending/descending ISN order within each value. You can specify a starting and ending value. You can also specify the field(s) for which values are returned. Read logical is useful when you want the returned records sorted on a particular field.

Adabas provides a special read command (L9) to determine the range of values present for a descriptor, and the number of records that contain each value. Such a retrieval is called a histogram. The L9 command does not require any access to data records, only to the inverted lists stored in the Associator.



Random Access

Adabas uses the S1/S2/S4 commands to select records that satisfy a *search criterion*: the count of the records found and a list of their ISNs is returned. The S1/S4 commands return the ISNs in ascending sequence; the S2 command allows you to specify a sort sequence for the returned ISNs.

The search criterion may comprise

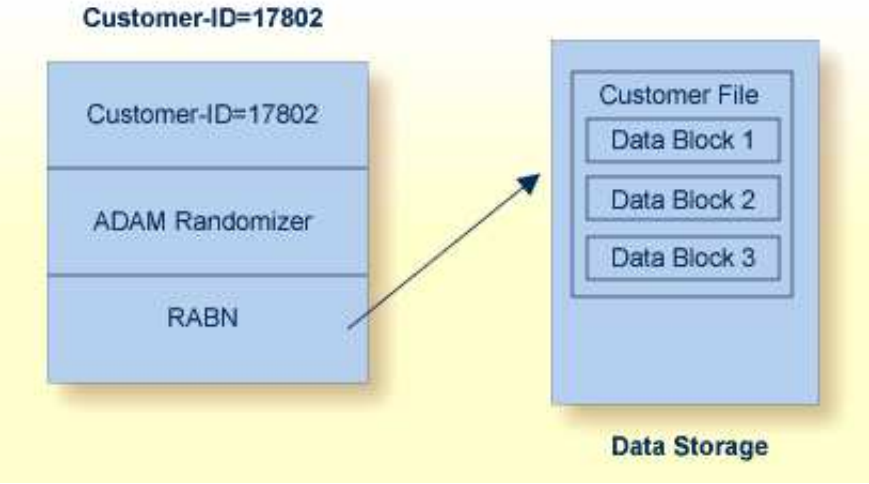
- one or more fields in a single file;
- fields contained in two or more physically coupled files; or
- search, read, and internal list matching based on the soft coupling feature.

A search criterion may contain one or more fields that are not defined as descriptors. If nondescriptors are used, Adabas performs read operations to determine which records to return to the user. If only descriptors are used within the search criterion, Adabas resolves the query by using the Associator inverted lists; no read operation is required.

Random Access Using the Adabas Direct Access Method (ADAM)

Note:
The ADAM feature is available on mainframe platforms only.

The Adabas direct access method (ADAM) improves random access performance on a particular descriptor field in a file. ADAM uses the field value to compute the relative block address (RABN) for record storage. The ADAM descriptor may also be used like any other descriptor within a selection criterion, and for logical sequential processing as well. This option may be selected for any given file when the file is loaded into the database.



Using Triggers and Stored Procedures

The Adabas triggers and stored procedures facility is an integral part of Adabas; however, Natural (read *Natural Application Development Environment*) is required to use the facility. The online Trigger Maintenance facility can be accessed using the Adabas Online System add-on product.

A procedure is a Natural subprogram that is written and tested using standard Natural facilities.

- A *trigger* is a procedure that is executed automatically by Adabas when a specified set of criteria is met. The set of criteria is determined for each command sent to Adabas and is based on the target file number and optionally the command type and/or field. The command type refers to the find, read, store, update, and delete commands. The field must be in the corresponding format buffer of the command.
- A *stored procedure* is executed by Adabas, but is invoked directly by a special user call from any of a number of applications that use it. Storing programs that are used by multiple clients in an Adabas file on the server reduces the amount of data traffic to and from the server.

The same types of parameters are passed to the subprogram whether it is a trigger or a stored procedure.

The Adabas facility for triggers and stored procedures allows you to implement and maintain both types of procedures. It resides within Adabas and provides an extension to an application. It can be used to

- implement various security and auditing features for an application; and
- provide a consistent, central environment where data can be verified or manipulated, either manually by the application or automatically by Adabas when triggers are defined.

Universal Encoding Support (UES)

Adabas provides facilities for converting data bidirectionally between different architectures. This makes it possible to support access to the mainframe database through the TCP/IP protocol from web-based applications or from PC-based applications such as Natural for Windows.

A database can use these UES facilities if activated with ADADEF.

Adabas performs character data conversion for fields with alphanumeric (A) and wide-character (W) formats. It supports a wide range of character sets or code pages, including Unicode, double-byte character sets (DBCS), and multiple-byte characters sets (MBCS).

- Alphanumeric fields are extended to support wide-character data by defining encoding keys on both the database and file levels: the file level encoding takes precedence over the database encoding. The encoding specifies the format in which the data is to be stored. It is also used as the default format in which data is exchanged with a local user. The file encoding must belong to the EBCDIC group.
- Wide-character fields are similar to alphanumeric fields in that encoding defaults are defined on both the database and file levels: the file encoding takes precedence over the database encoding. Unicode is the default encoding.

In addition, Adabas supports the conversion of numeric data, such as low-order-byte-first or IEEE floating point numbers.

At the client application, the Adalink code determines the default architecture (for example, ASCII, byte-swap, and IEEE floating point). The client application can override this default by specifying different encodings or architecture using the session open (OP command).

A number of utilities provide for special encoding and architecture settings.

To ensure round-trip compatibility between architectures and encodings, a file encoding should be chosen that has the same or a superset of characters versus the user encoding. For example, US-EBCDIC and ISO-8850-1 have the same character set.

Collation descriptors may be defined for alphanumeric or wide-character format fields. The descriptor values are obtained algorithmically by calling a collation exit, for example to produce a culturally correct, sorted key (that is, dictionary order).

Maintaining Database Integrity

Adabas provides facilities to ensure the logical consistency of data in a competitive updating environment and when a user or Adabas session is interrupted.

Facilities are available for both online and traditional batch update. For online, transaction-oriented processing, Adabas ensures that the database is free of incomplete transactions. For batch mode updating, Adabas ensures restart in the event of failure by writing checkpoints and backing out/regenerating updates.

- Transaction Logic
- Distributed Transaction Processing
- Competitive Updating
- Timeout Controls
- Backout, Recovery, and Restart

- Extended Error Handling and Message Buffering

Transaction Logic

Adabas data protection, recovery, and user restart are 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 OP (open) or RE (read ET data) commands can be used to retrieve user restart data stored by the C3, CL, or ET command. This data is also written to the Adabas data protection log with each checkpoint written by the transaction and can be read using the ADASEL utility.

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.

Distributed Transaction Processing

Adabas incorporates nucleus functions to support the execution of global database transactions in distributed environments; that is, across multiple local or remote databases and/or system images in parallel.

A two-phase commit protocol ensures that all database management systems (DBMSs) that participate in processing the global transaction (that is, resource managers or RMs) either commit or roll back their local parts of a transaction as a whole. In the first phase, the coordinating component (a transaction manager or TM) prepares all involved RMs for the commit. Only when the first phase is successful does the TM instruct the RMs to commit (second phase).

Adabas functions in this scenario as an RM. Adabas Transaction Manager, an Adabas add-on product, functions as the coordinator within operating system images and, with the help of Entire Net-Work, across system images.

The new protocol also integrates Adabas with other DBMSs. It is transparent to existing application systems and to Natural.

Included in Adabas is a CICS-controlled interface that conforms to the CICS Resource Manager Interface (RMI). It issues the appropriate Adabas commands to coordinate with the two-phase commit protocol.

Competitive Updating

Competitive updating is in effect when two or more users (in multiuser mode) are updating the same Adabas file(s). The Adabas facilities used to ensure data integrity in a competitive updating environment include the following features:

record hold/release, avoidance of resource deadlock, exclusive control updating, and shared hold updating.

- **Record Hold and Release:** The Adabas record hold facility ensures that a record will not be updated by more than one user at a time. A user can put the record in hold status (that is, the ISN of the record is put in the hold queue) using the commands S4 (find with hold), L4/L5/L6 (read with hold), A1/E1 (update/delete) with the hold option specified, N1/N2 (add record with hold), or HI (hold record).

  If a record requested for hold status is already being held by another user or utility, the user issuing the record hold command is put in wait status until the record becomes available, at which time Adabas reactivates the command. You may request that a response code be returned if you do not want to be put in wait status.

  Records in hold status can be accessed (found and read) by users who do not seek to hold the record.

  Records in hold status are released by issuing the ET or CL commands. Options are available with ET and BT commands to release records selectively. The CL command releases all records in hold status for the issuing user.

- **Avoiding Resource Deadlock:** Resource deadlock occurs when two users are each waiting for a record held by the other user. Adabas protects against such a user deadlock situation by detecting the potential deadlock and returning a response code to the second user after putting the first user in wait status. This occurs if the two users are serviced by the same nucleus (a non-cluster nucleus or the same cluster nucleus). But even in a single nucleus, a deadlock might not be detected if an ISN involved in the deadlock is being held as a shared resource by more than one user. For more information, read about shared hold status.

- **Shared Hold Status:** Shared hold status can be used to lock data records in shared mode, rather than in exclusive mode. This allows your database users to read the same record in parallel transactions, but ensures that no one can update the record concurrently.

  Using shared hold status, your users can protect large object values from concurrent updates without locking out other users who may need to read the same LOB value or other LOB values in the same record. It also allows your users to protect the records they read against concurrent updates for specific periods of time:

  - For the duration of the read command;
  - When the next record in a sequence is read;
  - When the user’s transaction ends;
  - Indefinitely.
  - Option C puts the record in shared hold status for the duration of the read operation. It ensures that the version of the record being read has been comitted by the last updater. This option is available for the L4, L5, L6 and S4 commands. For the S4 command, the shared hold remains in place for the duration of the read operation.
  - Option Q puts the record in shared hold status until the next record in the read sequence is read or the read sequence or transaction is terminated, whichever happens first. It ensures that the record being read cannot be updated concurrently until the next record in the sequence is read (or the transaction is terminated). This option is available for the L4 (when command option 2 is set to "N"), L5, L6 and S4 commands.

- Option S puts the record in shared hold status until the end of the transaction. It ensures that the record being read cannot be updated concurrently until the transaction is terminated. This option is available for the HI, L4, L5, L6, RI and S4 commands.

- Option H keeps a record in shared hold status indefinitely (until the next ET or BT command). This option is available for the BT and ET commands. Records in shared hold status at the time of the BT or ET command are kept in shared hold status beyond the end of the transaction until another ET or BT command is issued (without this H option or the prefetch or multifetch options). Any records in exclusive control are also changed to shared hold status beyond the end of the transaction.

If the same record is placed in shared hold status more than once (using the C or S options or the Q option for different read sequences), it stays in shared hold status until all of the specified hold lifetimes have expired.

For more information about these command options and their detailed functioning in each command, read about the individual commands in *Commands*.

For complete information about shared hold updating, read *Shared Hold Status*.

- **Exclusive Control Updating:** Users who use logical transaction commands (ET/BT) are called ET logic users.

  Alternatively, a user can request exclusive control of one or more Adabas files for the duration of the user session. If the file or files for which exclusive control is requested are not already opened for update by another user or utility, exclusive control is granted and the user becomes an exclusive update (EXU) user. Adabas treats EXU users as non-ET logic users.

  Adabas does not place an ISN in hold status for EXU users. Adabas disables hold logic processing for files being updated under exclusive file control.

  Instead of using ET commands, EXU users can request checkpoints to act as reference points; for example, updates applied after a checkpoint can be removed.

For complete and detailed information about competitive updating, read *Competitive Updating*

Timeout Controls

Adabas times out:

- transactions that exceed a specified limit; and
- users who are inactive for a specified amount of time.

Transaction Time Limit

Adabas provides a transaction duration time limit for ET logic users. The time limit is set with the ADARUN TT parameter; an override for a specific user can be set using the OP command.

If a transaction exceeds the prescribed limit, Adabas generates a BT (back out transaction) command to remove all the updates performed during the transaction and release all held records. The user can then either repeat the backed out transaction from the beginning or begin another transaction.

Non-Activity Time Limit

All users are subject to a non-activity time limit; different limits can be set for different user types and for specific users within each user type.

If a user exceeds the prescribed limit and the user is

- an ET logic user, Adabas backs out the current transaction, releases all held records and command IDs, and deletes the user's file list.
- an EXU user, Adabas deletes the user's file list and releases all command IDs. The user loses his EXU user status and becomes an access-only user.
- an access-only user, Adabas deletes the user's file list.

Backout, Recovery, and Restart

Backout, recovery, and restart may be required when a user or Adabas session is interrupted due to a timeout (read *Timeout Controls*); a program error when Adabas determines that a transaction cannot be completed successfully; an Adabas, hardware, or operating system failure; or a power failure.

A *user session* is a sequence of Adabas calls optionally starting with an open (OP) command and ending with a close (CL) command. A *user* 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.

An *Adabas session* 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.

User Program Error

A user program that is processing a transaction can detect that the transaction cannot be completed successfully. In this case, a BT (back out transaction) command is used to remove or back out the incomplete transaction.

If a user program error causes logical damage to the database, it may be necessary to recover the affected files using the ADASAV and ADARES utilities.

Adabas, Hardware, or Operating System Failure

After any failure that causes the Adabas nucleus to terminate abnormally, an automatic procedure is executed when Adabas is reactivated to bring the database to a physically and logically valid status. All partially executed update commands are reset; all incomplete transactions are backed out.

The automatic procedure comprises three steps: repair the database, autorestart, and autobackout:

- Database repair modifies the database to the status it would have had if a buffer flush had just been completed at the time of failure. That is, all blocks in the database are at a status that enables the nucleus to perform normally.

- Autorestart backs out updates of single update commands that were partially executed when the system failed. It resolves internal inconsistencies in the database and ensures physical integrity.
- Autobackout, which is performed only for ET logic users, backs out updates of user transactions that were partially executed when the system failed. Adabas performs an internal BT (back out transaction) followed by autorestart, and then informs the user that the last transaction has been backed out.

The autobackout routine is executed at the end of an ET session that was terminated with HALT. It is also executed automatically at the beginning of the next Adabas session to remove any updates performed within transactions that did not complete successfully.

After autobackout execution, the database contains updates only from logically complete transactions.

Note:
ET users can manually back out an incomplete transaction at any time by issuing the BT (back out transaction) command. Read *Maintaining Database Integrity*.

If an Adabas, hardware, or operating system failure results in physical damage to the database, it may be necessary to recreate the database using the ADASAV and ADARES utilities.

Power Failure

Depending on the hardware, a power failure during an I/O operation may damage the Adabas blocks that were being processed. This damage cannot be detected during autorestart and therefore can result in problems later such as unexpected response codes or lost database updates.

If the ADARUN IGNDIB=YES parameter is set, the autorestart routine checks whether a buffer flush was active when the session interruption occurred. If a buffer flush was in process, the autorestart shuts down and Adabas alerts the user to the potential problem and includes a list of the files being updated when the buffer flush was in process. The DBA must then determine whether a power failure occurred.

If the cause of a session interruption

- *is* a power failure, Software AG strongly recommends recovering the affected files using the ADASAV and ADARES utilities.
- is *definitely not* a power failure and the integrity of the information on the output hardware can be guaranteed, the database can be reactivated immediately. Database recovery is not necessary.

Extended Error Handling and Message Buffering

The error handling and message buffering facility helps implement 24X7 operations by analyzing and recovering from certain types of errors automatically with little or no DBA intervention. It also generates additional information so that the error can be diagnosed by the user and by Software AG.

A wrap-around message buffer collects Adabas messages for later review by Adabas Online System in case online access to the console or to DDPRINT messages becomes unavailable. The buffer aids problem analysis and performance tuning.

The error handling functions of the facility can be invoked from the operator console or from Adabas Online System.

User exits and hyperdescriptor exits that are essential to the operation of the Adabas nucleus can be marked as critical (the default) or not:

- a *critical* user exit is not affected by the error handling and message buffering facility: an abnormal termination in it causes the Adabas nucleus to terminate abnormally as well.
- for a *notcritical* user exit, the facility maintains an active Adabas nucleus, optionally refrains from invoking that exit, takes a dump of the nucleus at the point when the exit failed, and issues messages to the system log to inform the DBA of the problem. The DBA can then examine the diagnostic information, use it to resolve the problem, then load and reactivate the corrected exit.

The extensions (plug-in routines or PINs) analyze and, in some cases, determine the cause of an abend while allowing the nucleus to continue processing. Each PIN service routine handles a predefined condition when encountered, allowing the Adabas nucleus to

- remain active when it otherwise would terminate abnormally; or
- print extended error diagnostics as an aid to error recovery.

While the PIN is executing, most Adabas functionality is available to it as the registers at the time of the abnormal event are available. The PIN determines whether it is safe to allow the nucleus to continue processing and prints appropriate messages to notify the DBA.

Based on its execution, a PIN can either transfer control to the Adabas nucleus so that it can resume normal processing-usually with a response code -- or it can return control to the error handling and message buffering facility, allowing the Adabas nucleus to terminate abnormally.

A PIN can also be used to format an intelligent dump in a number of circumstances to help debug a particular response or abend code.

A special PIN routine user exit can be used to obtain additional information about response codes and abends. The user exit allows you to specify particular response codes or response code/subcode combinations to be monitored. Once you have modified the user exit, you can reload it and make your changes effective without bringing the database down.