Recovery/Restart Design

This chapter discusses the design aspects of database recovery and restart. Proper recovery and restart planning is an important part of the design of the system, particularly in a database environment. Although Adabas provides facilities to perform both restart and recovery, the functions must be considered separately.

This chapter covers the following topics:

- Adabas Recovery
- Planning and Incorporating Recoverability
- Matching Requirements and Facilities
- Transaction Recovery
- End Transaction (ET) Command
- Close (CL) Command
- Reading ET Data
- System or Transaction Failure
- Limitations of Adabas Transaction Recovery
- Adabas Checkpoint Commands
- Exclusive File Control
- User Restart Data

Adabas Recovery

Recovery of database integrity has the highest priority; if a database transaction fails or must be cancelled, the effects of the transaction must be removed and the database must be restored to its exact condition before the transaction began.

The standard Adabas system provides transaction logic (called *ET logic*), extensive checkpoint/logging facilities, and transaction-reversing backout processing to ensure database integrity.

Restarting the database following a system failure means reconstructing the task sequence from a saved level before the failure, up to and including the step at which the failure occurred-including, if possible, successfully completing the interrupted operation and then continuing normal database operation. Adabas provides a recovery aid that reconstructs a recovery job stream to recover the database.

Recoverability is often an implied objective. Everyone assumes that whatever happens, the system can be systematically recovered and restarted. There are, however, specific facts to be determined about the level of recovery needed by the various users of the system. Recoverability is an area where the DBA needs to take the initiative and establish necessary facts. Initially, each potential user of the system should be

questioned concerning his recovery/restart requirements. The most important considerations are

- how long the user can manage without the system;
- how long each phase can be delayed;
- what manual procedures, if any, the user has for checking input/output and how long these take;
- what special procedures, if any, need to be performed to ensure that data integrity has been maintained in a recovery/restart situation.

Planning and Incorporating Recoverability

Once the recovery/restart requirements have been established, the DBA can proceed to plan the measures necessary to meet these requirements. The methodology provided in this section may be used as a basic guideline.

1. A determination should be made as to the level and degree to which data is shared by the various users of the system.
2. The recovery parameters for the system should be established. This includes a predicted/actual breakdown rate, an average delay and items affected, and items subject to security and audit.
3. A determination should be made as to what, if any, auditing procedures are to be included in the system.
4. An outline containing recovery design points should be prepared. Information in this outline should include
    - validation planning. Validation on data should be performed as close as possible to its point of input to the system. Intermediate updates to data sharing the record with the input will make recovery more difficult and costly;
    - dumps (back-up copies) of the database or selected files;
    - user and Adabas checkpoints;
    - use of ET logic, exclusive file control, ET data;
    - audit procedures.
5. Operations personnel should be consulted to determine if all resources required for recovery/restart can be made available if and when they are needed.
6. The final recovery design should be documented and reviewed with users, operations personnel, and any others involved with the system.

Matching Requirements and Facilities

Once the general recovery requirements have been designed, the next step is to select the relevant Adabas and non-Adabas facilities to be used to implement recovery/restart. The following sections describe the Adabas facilities related to recovery/restart.

Transaction Recovery

Almost all online update systems and many batch update programs process streams of input transactions which have the following characteristics:

- The transaction requires the program to retrieve and add, update, and/or delete only a few records. For example, an order entry program may retrieve the customer and product records for each order, add the order and order item data to the database, and perhaps update the quantity-on-order field of the product record.
- The program needs exclusive control of the records it uses from the start of the transaction to the end, but can release them for other users to update or delete once the transaction is complete.
- A transaction must never be left incomplete; that is, if it requires two records to be updated, either both or neither must be changed.

End Transaction (ET) Command

The use of the Adabas ET command

- ensures that all the adds, updates, and/or deletes performed by a completed transaction are applied to the database;
- ensures that all the effects of a transaction which is interrupted by a total or partial system failure are removed from the database;
- allows the program to store up to 2000 bytes of user-defined restart data (ET data) in an Adabas system file. This data may be retrieved on restart with the Adabas OP or RE commands. The restart data can be examined by the program or TP terminal user to decide where to resume operation;
- releases all records placed in hold status while processing the transaction.

Close (CL) Command

The Adabas CL command can be used to update the user’s current ET data (for example, to set a user-defined *job completed* flag). Refer to the section *User Restart Data* for more information.

Reading ET Data

After a user restart or at the start of a new user or Adabas session, ET data can be retrieved with the OP command. The OP command requires a user ID, which Adabas uses to locate the ET data, and a command option to read ET data.

The RE command can also be used to read ET data for the current or a specified user; for example, when supervising an online update operation.

System or Transaction Failure

The autobackout routine is automatically invoked at the beginning of every Adabas session. If a session terminates abnormally, the autobackout routine removes the effects of all interrupted transactions from the database up to the most recent ET. If an individual transaction is interrupted, Adabas automatically removes any changes the transaction has made to the database. Each application program can explicitly back out its current transaction by issuing the Adabas BT command.

Limitations of Adabas Transaction Recovery

The transaction recovery facility recovers only the contents of the database. It does not recover TP message sequences, reposition non-Adabas files, or save the status of the user program.

It is not possible to back out the effects of a specific user’s transactions because other users may have performed subsequent transactions using the records added or updated by the first user.

Adabas Checkpoint Commands

Some programs cannot conveniently use ET commands because

- the program would have to hold large numbers of records for the duration of each transaction. This would increase the possibility of a deadlock situation (Adabas automatically resolves such situations by backing out the transaction of one of the two users after a user-defined time has elapsed, but a significant amount of transaction reprocessing could still result), and a very large Adabas hold queue would have to be established and maintained;
- the program may process long lists of records found by complex searches; restarting part of the way through such a list may be difficult.

Such programs can use the Adabas checkpoint command (C1) to establish a point to which the file or files the program is updating can be restored if necessary.

Exclusive File Control

A user can request exclusive update control of one or more Adabas files. Exclusive control is requested with the OP command and will be given only if the file is not currently being updated by another user. Once exclusive control is assigned to a user, other users may read but not update the file. Programs that read and/or update long sequences of records, either in logical sequence or as a result of searches, may use exclusive control to prevent other users from updating the records used. This avoids the need for placing each record in hold status.

Checkpointing Exclusively Controlled Files

Exclusive control users may or may not use ET commands. If ET commands are not used, checkpoints can be taken by issuing a C1 command.

System or Program Failure

In the event of a system or program failure, the file or files being updated under exclusive control may be restored using the BACKOUT function of the ADARES utility. This utility is not automatically invoked and requires the Adabas data protection log as input. This procedure is not necessary if the user uses ET commands (see the section *Transaction Recovery*).

Limitations of Exclusive File Control

The following limitations apply to exclusive file control:

- Recovery to the last checkpoint is not automatic, and the data protection log in use when the failure occurred is required for the recovery process. This does not apply if the user issues ET commands.
- In a restart situation following a system failure, Adabas does not check nor prevent other users from updating files which were being updated under exclusive control at the time of the system interruption.

User Restart Data

The Adabas ET and CL commands provide an option of storing up to 2000 bytes of user data in an Adabas system file. One record of user data is maintained for each user. This record is overwritten each time new user data is provided by the user. The data is maintained from session to session only if the user provides a user identification (user ID) with the OP command.

The primary purpose of user data is to enable programs to be self-restarting and to check that recovery procedures have been properly carried out. The type of information which may be useful as user data includes the following:

- The *date and time* of the original program run and the time of last update. This will permit the program to send a suitable message to a terminal user, console operator, or printer to allow the user and/or operator to check that recovery and restart procedures have operated correctly. In particular, it will allow terminal users to see if any work has to be rerun after a serious overnight failure of which they were not aware.
- The *date of collection* of the input data.
- *Batch numbers*. This will enable supervisory staff to identify and allocate any work that has to be reentered from terminals.
- *Identifying data*. This data can be a way for the program to determine where to restart. For example, a program driven by a logical sequential scan needs to know the key value at which to resume.
- *Transaction number/input record position* . This may allow an interactive user or batch program to locate the starting point with the minimum of effort. Although Adabas returns a transaction sequence number for each transaction, the user also may want to maintain a sequence number because
    - after a restart, the Adabas sequence number is reset;
    - if transactions vary greatly in complexity, there may not be a simple relationship between the Adabas transaction sequence number and the position of the next input record or document;

    - if a transaction is backed out by the program because of an input error, Adabas does not know whether the transaction will be reentered immediately (it may have been a simple keying error) or rejected for later correction (if there was a basic error in the input document or record);

- *Other descriptive or intermediate data*; for example, totals to be carried forward, page numbers and headings of reports, run statistics.

- *Job/batch completed flag*. The system may fail after all processing has been completed but before the operator or user has been notified. In this case, the operator should restart the program which will be able to check this flag without having to run through to the end of the input. The same considerations apply to batches of documents entered from terminals.

- *Last job/program name*. If several programs must update the database in a fixed sequence, they may share the same user ID and use user data to check that the sequence is maintained.

A user's own data can be read with either the OP or RE command. User data for another user can be read by using the RE command and specifying the other user's ID. User data for all users can be read in logical sequential order using the RE command with a command option; in this case, user IDs are not specified.