# Database Space Management

This chapter provides the DBA with all pertinent information related to database space management. Information is provided about

- Adabas physical and logical extents;
- Adabas relative block number (RABN);
- the role of the Adabas nucleus and utilities in allocating/deallocating space;
- using the database status report to monitor database space usage;
- potential space utilization problems and recommended action.

This information is organized under the following headings:

## Adabas Physical Extents

An Adabas physical extent (database container) is a collection of physical blocks assigned to a given database component (Associator, Data Storage, Work) during the definition of the database (see the ADADEF utility, ASSOSIZE, DATASIZE and WORKSIZE parameters).

The space for a physical extent is allocated using the standard allocation procedures of the operating system in use.

An Adabas physical extent may be allocated within a single operating system extent which consists of a primary extent only, or may be allocated as a primary extent together with one or more secondary extents. The secondary extents need not be contiguous to the primary extent or to each other.

An Adabas physical extent may be contained on a single physical volume or may extend across multiple volumes. The Associator and Data Storage components may each contain up to about 99 Adabas physical extents. However, your actual real maximum could be less because the extent descriptions of all Associator, Data Storage, and Data Storage Space Table (DSST) extents must fit into the general control blocks (GCBs). For example, on a standard 3390 device type, there could be more than 75 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there are less for another).

# Relative Adabas Block Number (RABN)

Adabas information is stored in space allocated in blocks. A block's size depends on:

- the physical device on which the block is located; and
- the Adabas component to which the block is assigned.

For example, the default device type used by Adabas is the IBM 3380 disk. This device is assumed in many utility and operating parameters as the device type unless another is specified.

When 3380 space is allocated for Adabas, it must be designated as Associator (ASSO), Data Storage (DATA), the Work area (WORK), logging area (PLOG, CLOG, RLOG), sort area, or temp area. A 3380 block allocated to ASSO contains 2004 bytes, but a 3380 block allocated to DATA contains 4820 bytes. Block sizes are predefined for each device type and Adabas component. To review a list of block sizes by device type and component, refer to the sections entitled *Device And File Considerations* in the Adabas installation documentation for the appropriate system platform (z/OS, z/VSE, or BS2000).

Adabas block sizes are not fixed by hardware; however, they are referred to as *physical* blocks to coincide with the level of description used for physical block (FBA) devices. Software AG tries to maintain consistent block definitions, by device type, from release to release. However, in some cases the block size for a component type may change to accommodate expanded Adabas facilities. Thus, a specific Adabas component (PLOG, ASSO, etc.) may need to be reformatted before you can run a new Adabas release.

Adabas identifies and addresses each physical block within a database component (Associator, Data Storage, Work) by its *relative Adabas block number* (RABN), which indicates the block's position relative to the beginning of the component. RABNs are assigned in ascending sequence within each database component, starting with "1". If multiple physical extents are used, the RABN assignment continues across the physical extents.

The first track of the first physical extent of the Associator, Data Storage and Work components is not used. The first track of the second and each subsequent physical extent as well as all extents of TEMP, SORT, CLOG, and PLOG are used.

The number of RABNs that can be assigned to ASSO and DATA depends on the RABNSIZE parameter, which is specified when the database is defined. RABNSIZE specifies the length of relative Adabas block numbers in the database (not the length of the block itself).

- If RABNSIZE=3 (block number is 24 bits or three bytes), the maximum number of RABNs is 16,777,215.
- If RABNSIZE=4 (block number is 31 bits or four bytes), the maximum number of RABNs is 2,147,483,646.

The number of Adabas blocks that can be stored on a given physical unit (track/cylinder/volume) of external storage is different for each database component and for each device type.

For example, using the z/OS information provided in *Supported Device Types*, the number of blocks that can be stored on a given z/OS volume may be calculated as shown in the following examples. In addition, the RABN ranges stored on each VOLSER can easily be determined using the Adabas Online System report function.

**Example 1: Associator database component, model 3380 (880 cylinders are assumed to be available on the volume)**

```
number of ASSO blocks = blocks/track * tracks/cylinder * number of cylinders
= 19 * 15 * 880 = 250,800 Associator blocks
```

Nineteen (19) blocks must be subtracted for the first track of the first Associator physical extent; therefore, the first Associator volume can contain a maximum of 250,781 blocks.

**Example 2: Data Storage database component, model 3370 (748 cylinders are assumed to be available on the volume)**

```
number of DATA blocks = blocks/track * tracks/cylinder * number of cylinders
= 10 * 12 * 748 = 89,760 Data Storage blocks
```

Ten (10) blocks must be subtracted for the first track of the first Data Storage physical extent; therefore, the first Data Storage volume can contain a maximum of 89,750 blocks.

## Adabas Logical Extents

An Adabas logical file extent is a group of consecutive RABNs allocated by the Adabas nucleus or an Adabas utility. For each file loaded into the database, a minimum of one of each of the following types of Adabas logical extents is allocated to the file:

| Logical extent | Allocated from the physical extent . . . |
|---|---|
| Data Storage | Data Storage |
| address converter | Associator |
| normal index | Associator |
| upper index | Associator |

Additional logical extents are allocated by the Adabas nucleus or an Adabas utility when additional space is needed as a result of file maintenance.

A maximum of two address converters may be allocated for any file, one for the file itself and one for any spanned records in the file. If spanned records are not used, only one address converter is allocated.

The total number of logical file extents that can be defined is limited only in that the extent information of all address converters, Data Storage, normal index, and upper index extents for the file must fit into the file control block (FCB). (The extent information is stored in a variable section of the FCB.) For example, on a standard 3390 device type, a file could have more than 40 extents of each type (or there could be more of one type if there are less for another).

When the Adabas nucleus starts up, the FCBs are checked. If there is insufficient FCB space for four further extents, the Adabas nucleus prints messages suggesting that the files should be reordered. In addition, if the last file extent of each type has only five or less free ISNs or blocks left, the nucleus locks the file for utility use only. Normal users trying to access the file will then get response code 17 (ADARSP017) or 48 (ADARSP048).

The "Contents of the Database" section of an ADAREP utility report flags files that may have insufficient FCB space for ten further extents. In the "File Information" section of the report, ADAREP prints a conservative estimate for how many further file extents can possibly be built for each file.

## Adabas Space Allocation and Deallocation

This section provides an overview of Adabas space allocation and deallocation procedures. A full understanding of these procedures will help ensure correct and efficient database space management.

- Free Space Table
- Adabas Nucleus Space Allocation Rules
- Space Allocation with the ADADBS Utility
- Space Allocation with the ADAINV Utility
- Space Allocation with the ADALOD Utility
- Space Allocation by the ADAORD Utility
- Space Allocation by ADASAV (RESTORE FILES Function)

### Free Space Table

All space available for allocation is stored in the free space table (FST). This table contains all RABN extents that are currently available for an allocation to any file.

### Adabas Nucleus Space Allocation Rules

When processing an add or update record command, the Adabas nucleus may need to allocate an additional extent to any of the following file components:

- address converter
- normal index
- upper index
- Data Storage

This section describes the rules used for the allocation.

#### Address Converter (AC)

The size of the address converter is initially defined by the MAXISN parameter in the ADALOD utility. The actual highest expected ISN is slightly higher because the address converter is stored in entire blocks. For example:

- If RABNSIZE=3, MAXISN=5000 on a model 3380 with 668 entries per block (2004/3) results in 8 blocks. The highest ISN expected (before further expansion) is therefore 5343 (668 * 8 - 1).

- If RABNSIZE=4, MAXISN=5000 on a model 3380 with 501 entries per block (2004/4) results in 10 blocks. The highest ISN expected is therefore 5009 (501 * 10 - 1).

If the Adabas nucleus requires an additional extent for a file when executing N1 commands, the allocation routine attempts to locate a new extent of 25% of the current size:

- If an unused extent between 25% and 28% can be found using the free space table (FST), that space is taken immediately;
- If only longer extents are available in the FST, a new extent of exactly 25% is taken;
- If only smaller extents are available in the FST, the longest available extent is taken;
- If an additional AC extent is required, and the maximum has already been assigned, Adabas will return an appropriate response code to the calling program;
- If a file has the attribute "one AC extent only" (e.g., if the file is an expanded file), an attempt to allocate a second AC extent will cause a response code.

## Normal Index (NI), Upper Index (UI), Data Storage (DS)

For the purpose of allocating a new extent, the following formulas are used:

$$Z1 = MIN\left(2 * B, (E-U) * B/U\right)$$

$$Z = MIN\left(MAX(Z1, B/8 + 10), 1000000\right)$$

where

| | |
|---|---|
| B | number of blocks currently allocated. |
| E | highest ISN expected. |
| U | highest ISN currently allocated. |

If an extent found in the FST is contiguous with the end of a previous extent, it is allocated for a maximum of Z blocks.

If *no* such extent can be found in the FST

- but an extent between Z and 9 * Z/8 is found, it is allocated.
- but an extent with more than 9 * Z/8 blocks is found, then a new extent is allocated with exactly Z blocks.
- the longest extent in the FST is allocated as the new extent.

Additionally, if the MAXNI, MAXUI, or MAXDS parameter is specified for the current file, the nucleus allocates no more than the specified maximum number of blocks for the NI, UI, or DS, respectively.

## Space Allocation with the ADADBS Utility

### ADD/INCREASE Associator, Data Storage

If the physical extent for the Associator or Data Storage has been exhausted, the ADD or INCREASE function (using Adabas Online System or the ADADBS utility) may be used to provide additional physical space.

The ADD function requires the allocation of an *additional* data set to the Associator or Data Storage. The new data set may be located on the same or a different device type than those currently in use. Both the Associator and Data Storage may consist of no more than 99 data sets each. However, your actual real maximum will be less because the maximum number of physical extents that you can define is derived from the block size of the first Associator data set (DDASSOR1). For example, on a standard 3390 device type, there could be more than 75 Associator, Data Storage, and DSST extents each (or there could be more of one extent type if there are less for another).

The INCREASE function results in the physical extension of an *existing* data set. The new space must, however, first be formatted using the ADAFRM utility. There is no restriction on the number of times the INCREASE function may be used.

Following an ADD function, the new data set must be formatted using the ADAFRM utility before it can be used, and the appropriate changes must be made to all Adabas job control as described in the Adabas Operations documentation.

After increasing Data Storage, it may be necessary to run the REORASSO function of the ADAORD utility to reorder the Data Storage space table (DSST) to a single extent to allow additional increases in Data Storage.

To permit formatting or reordering, the nucleus session terminates automatically following an ADADBS ADD or INCREASE operation.

### ALLOCATE Function

The ALLOCATE function (Adabas Online System or ADADBS utility) may be used to allocate an extent of a specific size for any of the following file components:

- Data Storage
- address converter
- normal index
- upper index

It is also possible to specify where the extent is to be allocated by specifying a starting RABN.

Using this function, the DBA may, based on knowledge of the projected size of a file, allocate extents of a specific size, rather than having Adabas perform the assignment. This may avoid having Adabas allocate an extent which is too small or too large (see ADALOD utility). MAXNI/MAXUI and MAXDS values in effect for the accessed file are not checked.

### DEALLOCATE Function

The DEALLOCATE function (Adabas Online System or ADADBS utility) may be used to deallocate an extent allocated for any of the following file components:

- Data Storage
- address converter
- normal index
- upper index

It is also possible to specify where deallocation is to begin by specifying a starting RABN. The deallocated space is returned to the free space table (FST).

### DELETE Function

The DELETE function (Adabas Online System or ADADBS utility) causes an existing file to be deleted from the database. All space which was assigned to the file is returned to the free space table and is available for reuse. DELETE can delete complete expanded files only.

### RECOVER Function

The RECOVER function (Adabas Online System or ADADBS utility) may be used to recover space which was allocated during an execution of the ADAINV or ADALOD utility which terminated abnormally. The recovered space is returned to the free space table and is available for reuse.

### REFRESH Function

The REFRESH function (Adabas Online System or ADADBS utility) results in the setting of a file status to 0 records loaded and 1 extent allocated to each file component. Any additional extents other than the first extent are returned to the free space table.

### RELEASE and UNCOUPLE Functions

The RELEASE and UNCOUPLE functions (Adabas Online System or ADADBS utility) results in the deletion of an inverted list or physical coupling lists. The space used for the list can be recovered only by using ADAORD. When releasing a descriptor for an expanded file, each component file must be released individually. ADADBS displays a message whenever a descriptor of an expanded file is being released.

## Space Allocation with the ADAINV Utility

### COUPLE/INVERT Functions

The COUPLE and INVERT functions (Adabas Online System or ADAINV utility) may result in the assignment of additional blocks for the NISIZE file component (but not DSSIZE or MAXISN). This occurs if the available space becomes full during processing of the input data.

In such a case, if there are any index blocks freed during deletion by the nucleus, these blocks are reused. Then, if available, a range of blocks in the free space table whose size is within the range M1 through M2 will be taken.

M1 and M2 are computed as follows:

```
M2 = M1 + M1/8
M1 = MAX (A2, NIB/4 + KZ)
```

where

| | |
|---|---|
| KZ | zap-able value (default = 10) |
| NIB | number of NI blocks in use |

and

```
A2 = MIN (A1, NIB * 2)
A1 = IUN * NIB/IUS
```

where

| | |
|---|---|
| IUN | number of unused ISNs |
| IUS | number of used ISNs |

When inverting a descriptor for an expanded file, each component file must be individually inverted. ADAINV displays a message whenever a descriptor of an expanded file is being inverted.

## Space Allocation with the ADALOD Utility

### LOAD Function

The LOAD function of the ADALOD utility is used to load a file into a database.

### DSSIZE Parameter

The number of blocks or cylinders specified with the DSSIZE parameter is allocated and assigned to the first DS extent at the beginning of ADALOD execution.

The DSRABN or DSDEV parameters may be used to force the allocation to a specific RABN or device.

If during processing of the input data, this first allocated extent becomes full, a search is made for a range of free blocks in the free space table whose size is within the range M1 through M2.

M1 and M2 are computed as follows:

```
M2 = M1 + M1/8
M1 = MAX (A2, DSB/4 + KZ)
```

where

|  |  |
|---|---|
| KZ | zap-able value (default = 10) |
| DSB | number of DS blocks in use |

and

```
A2 = MIN (A1, DSB * 2)
A1 = IUN * DSB/IUS
```

where

|  |  |
|---|---|
| IUN | number of unused ISNs |
| IUS | number of used ISNs |

If enough space is found in the free space table and that space follows immediately an already allocated extent, this space is added to the end of the extent. In this case no new extent is allocated.

If a new extent is needed, the free space table is scanned and the number of blocks needed to satisfy the size of M1 through M2 is taken for the new extent. Up to 99 extents are possible. However, your actual real maximum will be less because the maximum number of physical extents that you can define is derived from the block size of the first Associator data set (DDASSOR1). If space is not available, ADALOD ends with an error message.

The maximum number of logical file extents that you can now define is derived from the block size of the first Associator data set (DDASSOR1). The extent information is stored in a variable section of the FCB. New extents can be added now until the used FCB size reaches the block size of the Associator data set.

## MAXISN Parameter

The MAXISN value is converted into a number of blocks and rounded up to a full block boundary. This range of blocks is allocated at the beginning of ADALOD execution and is assigned to the first address converter extent for the file.

The ACRABN parameter may be used to force the allocation to begin at a specific location.

If during processing of the input data, this first allocated extent becomes full, ADALOD tries to allocate another AC extent whose size is 25% of the sum of all currently existing AC extent sizes.

- If an unused range of blocks is available in the free space table in the range of 25% through 28% of the size currently in use, this range is immediately allocated as a new AC extent for the file;
- If only longer free ranges are available, a new AC extent of 25% is taken from the smallest free range of blocks;
- If only smaller free ranges are available, the largest available is taken.

### NISIZE/UISIZE Parameters

At the beginning of its execution, ADALOD allocates and assigns the blocks or cylinders specified by the NISIZE and UISIZE parameters to the first NI and UI extents, respectively.

The NIRABN and UIRABN parameters can be used to force extent allocation to begin at a specific RABN.

If you omit the NISIZE or UISIZE parameters, ADALOD does not initially allocate NI and UI space. Instead, ADALOD waits until all incoming descriptor values have been written to the Temp data set, and then estimates NISIZE and UISIZE values as follows:

- If no input records were processed:

  ```
  NISIZE = Number of descriptors +1
  UISIZE = 2 blocks
  ```

- If input records were processed:

  For each descriptor in the file, up to 16 temp data set blocks are selected and read. The contents of these blocks are sorted and estimated to the total amount of temp blocks used for this descriptor.

  The chosen algorithm returns the NISIZE and UISIZE values for each descriptor, which ADALOD adds together and then multiplies by the factor K, which is

  ```
  K = (MAXISN - MINISN + 1) / number of records loaded
  ```

  If, during operation, ADALOD determines that the resulting value is not enough, ADALOD allocates subsequent extents during its run. The sizes of these extents are computed in the same way as for additional DSSIZE extents, as described above.

### UPDATE Function

The UPDATE function of the ADALOD utility performs a mass add/delete of records to/from an existing file, reorganizes and (if necessary) expands the Associator and/or Data Storage space.

The ADALOD UPDATE functions allocates additional AC space if the MAXISN parameter specifies a new, higher maximum ISN value-even if the restructuring of the AC, NI and UI performed by UPDATE results in more unused current space. ADALOD UPDATE adds Data Storage space if the current Data Storage space cannot hold the new records.

### MAXISN Parameter

If a MAXISN value is specified for the UPDATE operation that is greater than the current value for the file, the difference between the old and the new MAXISN setting is computed. The number of AC blocks to satisfy this amount is then allocated from the free space table as an additional extent. The ACRABN parameter may be used to force the allocation to begin at a specific location.

If during processing of the input data the current AC and/or Data Storage extent becomes full, ADALOD tries to allocate another AC and/or Data Storage extent whose size is 25% of the sum of all currently existing AC and/or Data Storage extent sizes.

- If an unused range of blocks is available in the free space table in the range of 25% through 28% of the size currently in use, this range is immediately allocated as a new AC extent for the file;
- If only longer free ranges are available a new AC extent of 25% is taken from the smallest free range of blocks;
- If only smaller free ranges are available, the largest available is taken.

## Space Allocation by the ADAORD Utility

ADAORD reorders the respective Adabas Associator component (AC, NI/UI, DSST) and Data Storage to reclaim unusable space for reuse. Although ADAORD functions affect only the selected component files of an Adabas expanded file, there is no change to the logical consistency of an expanded file; therefore, ADAORD does not have to be performed on each component file of an expanded file, unless desired.

| **Function** | **Accessed Table Types** |
|---|---|
| REORFASSO | AC, NI, UI |
| REORASSO | AC, NI, UI, DSST |
| REORFDATA, REORDATA | DS |
| REORFILE, REORDB | AC, NI, UI, DS |

For each accessed file and for each accessed table type (depending on the function), the following action is taken:

- All used space is returned to the free space table.
- All tables with a specific location (ACRABN, DSRABN, NIRABN, UIRABN) are allocated and assigned as a first extent. The sizes used are either supplied (MAXISN, DSSIZE, NISIZE, UISIZE) or taken from the original file.
- All tables without a specific location are allocated and assigned as a first extent.

If one of the extents become full, the same action is taken as described for the ADALOD (UPDATE) utility (see the previous section).

## Space Allocation by ADASAV (RESTORE FILES Function)

If a file to be restored is already present in the database (OVERWRITE parameter must be specified) the space used by all these files is returned to the free space table. If a component file of an expanded file is specified, then all related component files must also be specified.

- RESTORE FILE=...

For each file to be restored, the original RABNs must be available. ADASAV tries to allocate the required extents at their original position with their original size. If one of these allocations fails, ADASAV will terminate with ERROR-060.

- RESTORE FMOVE=...

  For each file to be restored, at least the amount of original space used will be allocated. The allocation of the first extent for each file table can be forced to a specific location by using one of the optional parameters ACRABN, DSRABN, NIRABN, UIRABN. The sizes of these tables may be increased using MAXISN, DSSIZE, NISIZE, UISIZE.

If space is available, multiple input extents may be compressed in a new single extent. If there is not enough contiguous free space available, ADASAV will split the tables over several new extents (up to 99 for each table). If such space is not available, ADASAV will terminate with ERROR-060.

## Using the Database Status Report to Control Space Use

Database space utilization information can be obtained directly from the database status report produced by executing the ADAREP utility or using Adabas Online System.

In addition to a file allocation map and a block allocation map, this report lists the number of blocks

- used (and unused) for the Associator physical extent (or extents);
- used (and unused) for the Data Storage physical extent (or extents);
- allocated for the Work physical extent;
- used (and unused) for each file for the address converter, normal index, upper index, and Data Storage logical extent (or extents).

See the ADAREP chapter in the Adabas Utilities documentation for a detailed explanation of the information provided on this report.

The DBA should frequently review this report to identify potential space utilization problems.

The next section contains guidelines on problems which may be detected using the status report, and recommendations as to what action should be taken to prevent and/or resolve each problem.

## Potential Space Use Problems and Recommended Actions

This section provides a summary of the problems most often encountered concerning database space utilization, and the recommended corrective action to be taken to prevent and/or correct problems.

### Full Physical Extents

1. The Associator physical extent is nearly or completely full.

   - The physical extent may be increased (see ADADBS utility, INCREASE function);

- A new physical extent may be added (see ADADBS utility, ADD function);
- The Associator may be reordered using the ADAORD utility. This will be of benefit only if a large amount of Associator space fragmentation exists;
- Unused file extents can be release using the ADADBS DEALLOCATE function;
- Any Adabas files no longer required may be deleted (see the ADADBS utility; DELETE function);
- Any file coupling lists no longer needed may be deleted (see the ADADBS utility, UNCOUPLE function);

2. The Data Storage physical extent is nearly or completely full.
    - The physical extent may be extended (see the ADADBS utility, INCREASE function);
    - A new physical extent may be added (see the ADADBS utility, ADD function); this is recommended only when the new extent is on a new device type.
    - Data Storage may be reordered (see the ADAORD utility, REORDATA function). This will be of benefit only if a large amount of Data Storage space fragmentation exists, or the Data Storage padding factor is decreased;
    - A given file may be reordered (see the ADAORD utility, REORFILE function);
    - Any Adabas files no longer required may be deleted (see the ADADBS utility, DELETE function).

## Maximum Physical Extents Reached

1. The maximum number of Associator physical extents (about 99) has been reached.
    - The last extent can be increased using the ADADBS INCREASE function;
    - The Associator can be reordered by executing the ADAORD REORASSO function;
    - All files can be unloaded using the ADAORD RESTRUCTURE function and then reloaded into a larger database using ADAORD STORE.
2. The maximum number of Data Storage physical extents (about 99) has been reached.
    - The last extent can be increased using the ADADBS INCREASE function;
    - Data Storage can be reordered (see the ADAORD utility, REORDATA function). This will result in the elimination of Data Storage space fragmentation;
    - All files can be unloaded using the ADAULD utility and then reloaded into a larger database.

## Maximum Logical Extents Reached

1. The maximum logical extents for the address converter, normal index, or upper index for a file has been reached.

    - The REORFILE or REORFASSO function of the ADAORD utility can be executed to reorder all Associator entries for the file.
    - ISN reusage can be invoked using the ADADBS utility.
    - The file can be unloaded, deleted, and reloaded.

2. The maximum logical extents limit for either Data Storage or the Data Storage space table for a file has been reached.

    - The file (and all other files) can be reordered using the REORFDATA or REORFILE function of the ADAORD utility. This condenses multiple Data Storage extents into fewer extents.
    - The file can be unloaded, deleted, and reloaded.