Getting Started with Large Object (LB) Fields

Adabas 8 introduces large object fields (LB fields). These are fields that may contain much more data than the 253 bytes of normal alphanumeric fields or the 16,381 bytes of LA fields. Fields containing large objects are defined with the new LB option. The theoretical maximum size of the value of an LB field is just short of 2 GB; practical usable sizes are smaller.

Adabas stores 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*.

Note:
LOB files must be managed independently from their associated base files. For example, using ADADBS REFRESH to refresh the base file will not automatically refresh the LOB file; a separate, independent ADADBS REFRESH job for the LOB file must be run independently. Likewise, Adabas Online System (AOS) and Adabas Manager file functions must be applied separately to the LOB file from the base file.

To use LB fields with Adabas 8, complete the following steps:

1. Define an FDT with one or more fields assigned the LB option. You can use the ADACMP COMPRESS function to create LB fields. For example:

   ```
   ADACMP COMPRESS
   ADACMP FNDEF='1,AA,8,A,DE'                        Key field
   ADACMP FNDEF=...
   ADACMP FNDEF='1,AZ,250,A,NU'              Data field
   ADACMP FNDEF='1,L1,0,A,LB,NV,NU,NB'               Binary LOB field
   ```

   This defines field L1 as a large object field. The NV and NB options specify that this field is not subject to character conversion, nor to the compression of trailing blanks. The NU option specifies that this field is null-suppressed (fields with the NB option must also have either the NU or NC option defined). In effect, Adabas will not modify the provided field values in any way. For further information about the LB field option and all field options (including the NV and NB field options), read *Field Options*.

   If you provide input data for ADACMP to compress, the input can contain either short LB field values (up to 253 bytes) or large LB field values (greater than 253 bytes). In the uncompressed input record, at the place that corresponds to the LB field (in the example, L1), you may provide an empty LB value by specifying an inclusive length of X'00000004' that is not followed by LB data (since the LB value is empty). For further information about the structure of the COMPRESS input data, read *Input Data Requirements*

   Or:
   You can also update an existing FDT using the ADADBS NEWFIELD function with an FNDEF specification defining an LB field. (Adabas Online System also provides equivalent functionality.) If you elect to do this, you do not need to define and load a new *base file* and you can skip the next step and proceed with Step 3 of these instructions.

2. Use the ADALOD LOAD function to load the (possibly empty) base file with LB fields into the database.

You will load an empty LOB file separately in the next step, although the LOB and base files can be loaded in either order, or even in parallel. For example:

```
ADALOD LOAD FILE=11,NAME='BASE-FILE'
ADALOD      LOBFILE=12
ADALOD      MAXISN=100000,DSSIZE=5000B
ADALOD      ...
ADALOD      SORTSIZE=100,TEMPSIZE=100
```

The LOBFILE parameter specifies the file number of the LOB file associated with this base file. For complete information about the LOB-related ADALOD parameters, read *LOAD: Load a File*.

Note:
A base file-LOB file pair can also be established by just loading a *LOB file*, if a file with LB fields (*base file*) is already present.

Specify the //DDAUSBA output data set from the ADACMP COMPRESS step as //DDEBAND input data set for the ADALOD LOAD function.

3. Use the ADALOD LOAD function to load an empty *LOB file* and associate it with the *base file* you loaded in the previous step or that you updated in Step 1. (The LOB and base files can be loaded in either order, or even, in parallel.)

```
ADALOD LOAD FILE=12,LOB,NAME='LOB-FILE'
ADALOD      BASEFILE=11
ADALOD      MAXISN=500000,DSSIZE=500000B,NISIZE=4000B,UISIZE=40B
ADALOD      ISNREUSE=YES,DSREUSE=YES,INDEXCOMPRESSION=YES
ADALOD      ...
ADALOD      SORTSIZE=100,TEMPSIZE=100
```

The LOB file type indicates that ADALOD should load a LOB file with a predefined FDT. No //DDEBAND input data set need be supplied if you are loading an empty LOB file.

The BASEFILE parameter specifies the file number of the *base file* associated with this LOB file. For complete information about the LOB-related ADALOD parameters, read *LOAD: Load a File*.

Note:
A base file-LOB file pair can also be established by just loading a *LOB file*, if a file with LB fields (*base file*) is already present.

4. Run a database report using ADAREP and see how the base file and LOB file are shown in the report.

5. Reevaluate and adjust the Adabas nucleus (ADARUN) parameters NISNHQ, NH, LP, LDEUQ, LWP, NAB, and LU to make sure the nucleus will have sufficient resources for processing LB fields. For more information about these parameters, read *Adabas Initialization (ADARUN Statement)*. For information on how these parameters might need adjusting, read *Migrating From Previous Adabas Versions*

6. Write or adjust an application program to load data, including LB field values, into 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.

- One way to work with LB fields is using Natural 4.2. The LB field (for example, L1) of the LOB file is mapped to a dynamic variable in Natural. Otherwise, your application program can be written in much the normal way. Natural will automatically issue the Adabas calls in the format necessary to deal with large LB field values.

- Alternatively, your application program can issue direct calls against Adabas. For calls with *large* LB field values (more than 32 KB of data), you must use the ACBX direct call interface of Adabas 8. (Calls using the ACB direct call interface can be used if all specified LB field values fit into a single 32 KB buffer.) For more information, read *Specifying an ACBX Interface Direct Call*. For example, you could specify the following information in the ACBX direct call:

Direct Call Component	**Specification**

ACBX	`ACBXCMD = N1 (insert record)`
	`ACBXFNR = 11 (base file number)`
	`...`

First format/record buffer segment pair	`FB1 = 'AA,8,A,`**`L1L,4,B`**`,...,AZ,250,A.'`
	`RB1 = 'KEY-1    '` **`X'000186A0'`** `... 'Some arbitrary data...'`

Second format/record buffer segment pair	`FB2 = '`**`L1,*`**`.'`
	`RB2 = X'100,000 bytes of binary data'`

The locations and lengths of the two buffer segment pairs must be given in four Adabas buffer descriptions (ABDs), which are not shown here. For more information, read *Adabas Buffer Descriptions (ABDs)*

The length indicator (L) for the LB field in the first format buffer segment (`L1L,4,B`) specifies that the length of the L1 field value proper is stored at the corresponding place in the first record buffer segment, which indicates a LB field value length of 100,000 bytes (in hex, 186A0). For more information about the length indicator, read *Length Indicator (L)*.

The asterisk (*) length notation in the second format buffer segment (`L1,*`) specifies that the LB field value proper (without any further length information) is stored at the corresponding location in the second record buffer segment. For more information about asterisk length notation, read *Asterisk (*) Length Notation* .