Conversion of the Data Structure - General Considerations

The operation of the Adabas Bridge for DL/I normally does not need any change of the original DL/I definitions. It is sufficient to convert these original DL/I definitions with the ADL Control Block Conversion (CBC) utilities as described in the section *ADL Conversion Utilities for DBDs and PSBs* in this documentation. The resulting Adabas file layout can be used without any further modification by DL/I or Natural applications through the ADL CALLDLI or ADL Consistency Interface respectively.

Nevertheless, it may be advantageous to depart from this "straightforward" method by introducing some modifications before, during and after the conversion. The section *Managing ADL Files* in the *ADL Interfaces* documentation describes the steps necessary to perform this modification.

The following section offers you details of the default Adabas file layout, as well as some hints on modifying the DL/I structures or the Adabas file layout.

This chapter covers the following topics:

- Changes to DL/I Data Structures Prior to the Conversion
- Validating Segment Layouts and Data Types
- Optimization with Respect to Adabas Compression
- Adabas File Layout

Changes to DL/I Data Structures Prior to the Conversion

Before starting the conversion process, you should check whether any modifications to the DL/I definitions are desired. When the conversion is completed, some further change of the DL/I structures requires a time-consuming unload and reload of the data.

Validating Segment Layouts and Data Types

For DL/I applications, the view of a segment is often not the same as defined in the original DL/I DBD. This is possible, because DL/I corresponds with the application program on a segment level, that means, it offers and accepts a whole segment, regardless of the field definitions. Therefore, a DL/I field can contain non-numeric data, even if it is defined as numeric. Since Adabas operates on a field level, problems can occur while loading such incorrect data.

On the other hand, DL/I fields can exist which are defined as alphanumeric but which contain only numeric data. The CBC utility converts such fields to alphanumeric Adabas fields and Adabas will not validate them for numeric contents. The same effect occurs for parts of a segment which correspond to no DL/I field definition, but which are subsequently used for numeric data.

Therefore, you are recommended to compare the segment layout of the DL/I definitions with the redefinitions (like copy-codes, copy-books) in the application programs. Remove all inconsistencies and include field definitions from the copy- codes into the DL/I DBD definition. You can add all field definitions or combine subsequent alphanumeric fields to form one large field. This is advantageous for

the Adabas compression and for the developing of Natural applications as described later.

Note that additional field definitions in the DL/I DBD will not influence your DL/I applications, if the overall segment length is unchanged and you do not define a field as numeric which is supplied with non-numeric values. This is because ADL corresponds with the application program on a segment level like DL/I. But the data is stored with Adabas, and thus you will receive a non-zero response code if an inconsistency in the data type is detected.

Optimization with Respect to Adabas Compression

Once the data is converted, it will be compressed by Adabas as described in the *Adabas Utilities* documentation. Because ADL defines all fields with the "NU" option for the Adabas compression utility ADACMP, the "null value suppression" will also be active.

If the converted data structures do not reflect the structures which are used by the application, the Adabas compression may not operate in an optimal way. Assume for example, there is a segment in a DL/I DBD with no field definitions specified. The application program may redefine this segment by splitting it up into five different parts. If the last part is filled with data, no compression can take place, even if the other four parts are empty.

Another failure of the Adabas compression can be an incorrectly defined field type. If a field contains numeric data but is defined as alphanumeric, it will not be compressed.

Thus, in order to optimize the Adabas compression, the converted data structures and types should reflect the usage in the application programs. This may be achieved by modifying the DL/I definitions prior to the conversion or by changing the Adabas file layout after the conversion.

Adabas File Layout

The ADL Control Block Conversion offers you an automated way to convert DL/I data structures into an Adabas file layout. The following sections outline the default layout and how you can modify it:

- Default Layout Generated by the ADL Conversion Utilities
- Changing the Default Layout
- Considerations for Natural / Adabas applications

Default Layout Generated by the ADL Conversion Utilities

The ADL Control Block Conversion (CBC) converts every element of a DL/I physical DBD into an element of an Adabas file. Additionally some fields are generated to reflect the hierarchical structure. The information about the other DL/I definitions, like PSBs, logical or secondary index DBDs, are kept in the ADL directory file. These structures do not have any influence on the Adabas file layout. The only exceptions are user data fields in index DBDs.

In the following, the translation of the different elements is described in detail:

- A physical DBD corresponds to an Adabas file (an "ADL file"). You may split up the segments of the DBD into different files as described in the next section.

- For every DL/I segment, the CBC utility generates one Adabas group. The overall length of all Adabas fields of this group is the same as the length of the DL/I segment. The name and the overall length of the group must not be changed after the CBC utility runs. There are special rules for logical child segments, which are explained later in this section.

- Every DL/I field is converted to an Adabas field. Redefined or overlapping fields are split up into the smallest units. The CBC utility generates one Adabas field for every unit. The only exception is the root sequence field, which is never split up. For those parts of the DL/I segments which correspond to no DL/I field, so-called Adabas "filler fields" are generated.

An Adabas field has the same type and length as the corresponding DL/I field. Filler fields are always defined as alphanumeric. If an Adabas field would be longer than 253 bytes, it is split up. This arbitrary splitting up may result in problems for developing Natural applications. See the section *Considerations for Natural/Adabas Applications* later in this documentation on how to avoid these problems.

- For every secondary index, one Adabas field is generated as part of the Adabas file, which contains the source segment data. This field is one of the ADL internal fields and must not be altered by Natural applications.

- If an index DBD contains user data fields, then the corresponding Adabas fields are generated additionally to the secondary index field.

- With every ADL file, the ADL physical pointer fields Z0 - Z8 are generated as needed. These internal fields are used by the ADL CALLDLI Interface in order to locate the hierarchical position of a DL/I segment. The contents of these fields must never be altered by Natural applications. The maintenance of all ADL internal fields for Natural applications is performed by the *ADL Consistency Interface*, as described in the *ADL Interfaces* documentation.

- For every child segment type, the ADL CBC utility generates the logical pointer fields (so-called "partial concatenated key fields", short "PCK fields") of all parent segments. This concerns the physical as well as the logical hierarchical structure, regardless of which file contains the definition of the corresponding segment group. These fields reflect the hierarchy to Natural applications. For DL/I applications they are maintained by the ADL CALLDLI Interface automatically.

- The following fields are allocated as Adabas descriptors in an ADL file:

  - the root sequence field,

  - the physical pointer fields Z0 and Z1,

  - the secondary index fields.

  - Logical relationships between two DL/I DBDs require the definition of logical child segments on both sides. The data may be stored on both sides or only on one side. A logical child segment is called "real" if the data is stored on its side; otherwise it is called "virtual".

  The ADL CBC utility will include the elements which correspond to a logical child segment only in one Adabas file, regardless how it was defined in DL/I. ADL will choose the side for which the corresponding logical child segment fulfils one of the following criteria (in order of precedence):

  - The logical child segment has dependent segment types (so-called "variable intersection data").

- The logical child segment is real, while the paired one is virtual.
- Both logical child segments are real, and the one in question is to be converted first.

If a logical child segment contains no intersection data (in other words, no data apart from the DL/I concatenated key), no Adabas group for this segment will be generated at all. In fact, all information about this segment is kept in the ADL PCK fields.

Changing the Default Layout

The layout of an ADL file is primarily defined by the DL/I DBD definition. Thus, every modification of the DBD will alter the Adabas file layout.

During the CBC utility run, you can specify the `"FNR"` parameter with the `GENSEG` function. Then the segments will be distributed on different Adabas files. You may also specify the `"ADANAME"` parameter, in order to define the name of the Adabas group, which correspond to a DL/I segment type.

Once the DL/I data structures have been converted to Adabas, some changes are still possible:

- You may define additional fields "inside" or "outside" of segments (i.e. the corresponding groups), provided that the overall length of these groups remains unchanged.
- You may define descriptors, superdescriptors, etc.
- You may alter the Adabas DBID and file numbers of the ADL files by copying the file and reconvert the DBD definitions.

For more information concerning the Adabas layout of an ADL file, see the section *Managing ADL Files* in the *ADL Interfaces* documentation.

Considerations for Natural / Adabas applications

Once the DL/I data structures are converted to Adabas, DDMs can be generated with SYSDDM or with Predict which are used by Natural applications. Because the incorporate functions of these tools use the Adabas layout, you should consider the DL/I structures before the conversion because this defines how the Adabas fields looks like.

As described earlier, a field which would be longer than 253 bytes is split up. This might cause problems for Natural applications, if for example this splitting is in the middle of a numeric value. Thus, you are recommended to force a splitting manually at a better place. This can be performed either by including additional DL/I field definitions into the DBD or by a reorganization of the Adabas fields inside the affected group.

Another reason for avoiding such long Adabas fields is the inflexibility of these fields. As described in the section *Managing ADL Files* in the *ADL Interfaces* documentation, it is easy to increase the size of a field as long as it does not exceed the maximum size of 253 bytes.

You can distribute the segments of a DL/I DBD into different Adabas files. This can be helpful if Natural programs have to identify segment data and segments with no sequence fields are involved. Splitting up the data into different files might also affect the Adabas performance, as described in the section *Performance Considerations* in the *ADL Interfaces* documentation.

If you are using Adabas FASTPATH and some of the segments in a DBD contain constant data, like tables, etc., you should also consider choosing another file number for these segments. Then Adabas FASTPATH can work in a more efficient way.

In order to access dependent segments with Natural applications, you may include descriptors and superdescriptors in the Adabas file. You can do that by adding an entry to the ADACMP cards which are produced by the ADL CBC utility. Once the data has been converted into the ADL file, you can use the Adabas ADAINV utility in order to generate descriptors, superdescriptors etc. Such a superdescriptor might consist of the PCKs of all parent segments and the sequence field of the segment to be accessed. If this segment does not have a sequence field, the superdescriptor would also return data of other segment types under the same parent. In order to avoid this, you can choose another file number during the CBC utility run for the affected segment.