Data Transfer Process

1 Data Transfer Process - Overview

The data transfer process (DTP) includes a description of how data within a BW is to be loaded from a source object to a target object using transformations and filters. The data transfer process forms the template for requests. When a request of this type is processed, data is loaded into a target object from a source object.

2 Main Features of the Functionality

2.1 Requirements

To be able to manage complex data flows in a data warehouse, users require powerful tools in the following areas:

1. Modeling
2. Process design
3. Monitoring
4. Management of incorrect data
5. Runtime: the loading process itself

Enterprise data warehouses (EDW) can contain several layers via which data is loaded. There is typically an input buffer (PSA), an ODS layer, which can be made up of one or more DataStore objects stored one after another, and the multidimensional stores for reporting (InfoCubes and aggregates). It is also necessary to use data for applications such as data mining from DataStore objects and master data tables, whose results are played back into a source system in a ‘closed loop’ scenario. In addition, data is to be transferred from BI into other systems. As far as possible, the data should be transferred incrementally (i.e. using delta technology) from the source systems and within the various layers. The consequences of this requirement for each of the subareas are summarized below.

Aside: Request

The term ‘request’, which is of central importance for BI, is explained here. While there are other ways of modeling a data warehouse, in BI the request concept is the backbone with which each further development must comply. The request is made up of the following components, and in BI the term ‘request’ is used as a synonym for one of these components. It is generally clear from the context.

1. The description of a data request to a data source and the further processing of the data (transformation, filter, transfer to a data target)
2. The data that has been transferred to a data target and its load status
3. The data buffer that is updated during the transfer for restarting (PSA)

2.2 Modeling

It must be possible to arrange and maintain the metadata objects in BI (InfoProviders, transformations, InfoSources, filters, and so on) in a network for the data flow.

2.3 Process Design

Some of the requirements for designing complex processes are already covered by the process chain model for BW 3.0. In principle, the process
chain is a template for the 'run' of this chain that is actually executed, and should run periodically, always in the same form. There must also be such a template object for load processes from one persistent object to another, which on the one hand can be inserted into a process chain as a 'variant', and on the other hand, when the process chain is being executed, generates an 'instance', or an executable copy of itself, to load the data. The modeler must be able to arrange and define an arbitrary number of subprocesses in a manually defined process chain.

2.4 Monitoring

Monitoring must enable the BI administrator to quickly find the sources of errors within a request. There must be a clear reference to the process chain from which the error results and it must be easy to see which objects are affected by the error. The monitor must also identify performance-critical areas. The status of a load process must be transparent at all times so that subsequent processes (such as rolling up in aggregates when loading into an InfoCube) and the administrator can react to it.

2.5 Management of Incorrect Data

At single-record level for all load processes, the management of incorrect data should separate out single records that contain errors, enable the user to correct the cause, and eventually update them once more. The administrator should also be able to delete an entire request consistently from the system. A cross-object status manager knows the dependencies between the requests. The status manager shows the user an impact analysis with the affected data targets and the secondary requests that are based on the request that is to be deleted. It should also be possible to retrigger an incorrect request if the error has been corrected in the meantime (for example, if the table space was exceeded, a transformation was incorrect, master data was missing, and so on). The latter implies that the generation of a request and its actual processing must take place asynchronously. The transformations called during the processing of the request and the update to the data target can mark certain data records as incorrect. The further processing of these data records is stopped, and at the same time the records are written to an error stack. It must be possible to subsequently reload the incorrect records in a separate error request. The processing rule (filter, transformation, data target) is the same as for a normal request, but the source is the error stack and not the data source.

2.6 Runtime

During the runtime of a request, the data is read from the source object, fed through the filters and transformation steps predefined by the request, and then written to the data target. It should be possible to instantiate and start all requests as an instance within the run of a process chain as well as separately. If possible, the requests should be processed in parallel in background processes. Where possible, data should be loaded incrementally, that is, by delta update. In doing so, care must be taken that data that is loaded from a source into a certain target is not transferred multiple times due to overlapping filters. At the beginning of the request processing, the source data that is to be loaded must be locked so that it is not deleted or condensed during the request processing. The request must also be published in the data target so that, for example, the read pointer for the query can be adjusted. After the request has been successfully processed, the source data is released for condensing (or archiving, and so on). To ensure consistency between the data in the source object and the target object, the data is not allowed to be deleted. The data transferred in the request is available for subsequent processes and for reporting. If the data transfer terminates after extraction and transformation steps, which can be time-consuming, have been carried out, it should be possible to restart the transfer with the help of a temporary buffer. The user decides when and where temporary buffers are written and how long the data is to be retained in the buffer. Incorrect data should be immediately separated out by data quality management.

3 The Request (DTPR_*)

3.1 Request ID

As a unique identification, each request is given a 25-character unique ID with prefix 'DTPR_'. The ID is immediately created during the instantiation of a new request and is converted into a SID according to InfoObject 0REQUID. The SID then serves as the key of the tables for the request. The prefix serves to immediately differentiate the request from other requests that were not generated by data transfer processes.

3.2 Status transitions

See DTP Request - Status transitions

4 Relevant tables

See DTP - Relevant tables

5 Why DTP?

This section contains some arguments in brief, why the DTP was introduced with BI 7.0 and which have been discussed in the previous chapter.
- Transfer from one source to one target

<table>
<thead>
<tr>
<th>BI 3.X</th>
<th>BI 7.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>One InfoPackage / Request supplies data for several data targets</td>
<td>Data Transfer Process transports data from one data source to one data target; new source and target types can easily be defined</td>
</tr>
<tr>
<td>Fixed sequence of steps (DataSource / transfer rules / update rules / data target)</td>
<td>Arbitrary number of steps (filter objects/transformations) between two persistent objects</td>
</tr>
<tr>
<td>Fixed source type: Combination DataSource/source system</td>
<td>Init selection available</td>
</tr>
<tr>
<td>Fixed target type: Cube or ODS object or master data</td>
<td>Request produces error records stored in generic error stack</td>
</tr>
<tr>
<td>No init selection for extraction from infoprovder</td>
<td>Error handling into ODS Objects possible</td>
</tr>
<tr>
<td>One request can produce several error requests, depending on the data targets</td>
<td>Resume from persistent buffer</td>
</tr>
<tr>
<td>No error handling into ODS Objects</td>
<td></td>
</tr>
</tbody>
</table>

- **Delta management**
  - BI 3.X: The Receiver for DataSources is merely a logical system, which makes it impossible to have a separate delta management for several data targets within one system
  - BI 7.0: The delta management is maintained by the data transfer process which connects arbitrary sources and targets

**Related Content**

- Page: Table RSBERRORLOG - Growing Too Fast - Sap Note and Recommendations
- Page: DTP Performance issues
- Page: Syntax error in start routine
- Page: DTPs cannot be changed in a closed system
- Page: DTP tables
- Page: How to use previous selection value during next load.
- Page: Change Status of DTP Request
- Page: Advanced method to find out the number of uncompressed requests in an InfoCube