HASSO–PLATTNER–INSTITUT
für Softwaresystemtechnik GmbH an der Universität Potsdam

Dr. Heiko Schuldt

Concepts and Technologies for ERP Systems

Exercise 4 **SS 2003**

Transactions and Logical Units of Work in SAP R/3: ABAP Dialog Programming

Introduction

This practical exercise completes our activities in implementing banking support inside SAP R/3. You will develop dialog transactions that allow to create entries to the accounts of the IPH bank. Since some of these dialogs encompass several Dynpros that perform changes to the underlying database, you have to create appropriate lock objects in order to correctly implement LUWs.

Preparation (only needed when Exercise 2 has not been completed)

All those who do not have finished exercise 2 yet have first to create view `ZHPIxxCUST` and tables `ZHPIxxACC` and `ZHPIxxENTR`. The detailed explanation how this can be done is given in the preface of Exercise 3.

Question 1: Implement a First Dialog Transaction

After the implementation of reports in the last exercise, we will design and implement dialog transactions that support the creation of new entries in our banking scenario.

This dialog transaction is supposed to consist of a single Dynpro (screen) in which entry data can be specified. In addition to the graphical user interface and the data fields contained in

this screen, also control flow has to be defined (PBO and PAI modules) in a module pool (which is a special ABAP program)

Some nice colleague of yours has already started implementing such a dialog transaction. Unfortunately, he did not manage to complete this task. The name of the module pool he has prepared is `ZHPI00SingleDynpro`. Copy this program to your name space (i.e., to `ZHPIxxSingleDynpro`) and save it in your development class. For copying, you can go to the overview on program objects in the Object Navigator (Development → ABAP Workbench → Overview → Object Navigator). There, you can choose Program and `ZHPI00SingleDynpro`; display this program (using the icon with glasses). With right mouse → copy ... on the name `ZHPI00SingleDynpro` in the list of object names, you can copy the module pool and all associated objects. Check all objects that are listed (and not only the module pool itself), i.e., also screens and user interfaces.

Display your copy `ZHPIxxSingleDynpro` in the Object Navigator. Your dialog transaction consists of a single screen (Dynpro) with number `0100`. In the list of program objects, you can navigate to this screen. Essentially, this screen consists of the definition of control flow (Flow Logic) and a specification of the data elements (Element List) used in a dialog screen. Go to the flow logic specification first. Here, only module calls are possible (but no ABAP commands!). The screen painter (the tool where the graphical layout of the screen is defined) can be reached by using Layout. Fields that have the same name as variables in the PBO and PAI module, respectively, are automatically copied to/from this dialog mask (when PBO or PAI is activated). Important hint: In here, xx has of course to be replaced by your login number!

You find the ABAP sources of the PBO module and the PAI module in module pool `ZHPIxxSingleDynpro`. Complete all these program skeletons. The dialog transaction is supposed to collect all required information on an entry (customer number, account number, and amount). The entry ID shall be determined automatically (by incrementing the maximum entry ID of this account). This information shall be written to the database in the PAI module (create a new account entry record), but only if the account balance remains positive. In addition, the account balance shall be consistently updated.

Hint: Do not forget to explicitly activate all Dynpros. Please bear also in mind that variables are only copied to/from the GUI when they have the *same* name than variables in the associated modules.

After having implemented the Dynpro, you can define a dialog transaction. Use transaction ID `ZSDxx` for this purpose (via Screen → Other object ...). In the dialog box, check transaction and specify the transaction ID. You then have to specify a program name and the number of the first screen to be displayed in this transaction. After having defined a transaction, you can invoke your Dynpro via the transaction code /nZSDxx in the command field (for the ease of usage, you can add this transaction ID to your personal favorites). Without the definition of a transaction ID, you can only simulate this transaction, but you cannot execute it.

Further Hint: If you cannot find the command field, it can be displayed by clicking on the small triangle in the upper left corner of the SAPGui window which will then make the command field appear.

Question 2: Define Lock Objects

The first Dynpro was implemented with only one Dynpro. When implementing more complex sequences of dialog masks with several Dynpros, we have to apply the SAP lock management to guarantee the consistency of data even across several Dynpros.

To this end, you have to define a lock object first. The lock objects shall combine semantically related records of the account and of the entries table and shall restrict the access to these records. Lock objects are dictionary objects and are therefore managed by the ABAP dictionary. Create a lock object and name it `EZHPIxxACC`, where again xx has to be replaced by your login number. Note the exception from the naming scheme here: user-defined lock objects have to start with the prefix `EZ`. The lock object shall be defined over the primary table `ZHPIxxACC` and should have `ZHPIxxENTR` as secondary table. Choose a lock mode that allows to consistently modify records and/or to insert new entries. Which lock parameters are automatically generated for the lock object?

Question 3: Dialog Transaction with Locks

A more sophisticated dialog transaction shall better support the insertion of account entries. This dialog transaction is rudimentary defined in module pool `ZHPI00DialogTransaction`. Copy this module pool to your name space. It contains four screens: `0100` takes as input a customer number and an account number. In the PAI module, the associated account information should be retrieved from the database, an entry ID should be generated, and the lock object that has been defined in the previous exercise should be called. The following screen `0200` shall output account data and should allow the the input of entry data. In the PAI module, a test is required that checks whether or not the entry amount is allowed (the account balance must not become negative; the strategy of our bank is very restrictive for the time being) and whether no constraint is violated when inserting an entry. For the latter, the return code of the `INSERT` statement can be used. In case of success, the entry should be inserted into the database in the PAI module, together with an update of the account balance, and a change to screen `0300` has to be defined. Screen `0300` outputs the updated account information together with the entry data. In case of failures (i.e., entry amount would lead to negative account balance, or due to a lock conflict), a rollback is necessary. In order to output an error message, you have to change to screen `0400`. You do not need to worry about the release of the lock object; this will be done automatically with either rollback or commit.

Complete the individual modules of the module pool. Take also a look at the field lists of all four screens in the Screen Painter. Make also sure that the proper modules are called at PBO and PAI. Make your program available by defining a transaction `ZDIxx` starting with Dynpro `0100`.

Try to access the same account in two parallel transactions and verify whether the synchronization of the enqueue process using the lock object is actually working. A list of locks currently being held is available at transaction SM12 that can be started by /oSM12 in a separate window.

Question 4: Dialog Transaction with Update Task

The final dialog transaction you will have to implement requires the usage of a separate update task to handle the interactions with the database. First, copy the module pool of the previous question with all associated objects to the program `ZHPIxxUpdateTask`.

In order to apply the decoupled update paradigm, you first have to define a function group `ZFGxx` (choose Function Group in the Object Navigator and enter `ZFGxx`). A function group encompasses several individual function modules. After the definition of the function group, you can define the update function as a function module there. As a name for this update function module use `Z_ZHPIxx_UPDATETASK`[1]. Function modules can be defined via Create → Function Module by right clicking to the name of the function group. As attributes, you have to choose Update module as processing type and Start immediately (with the latter specification, a distinction between V1 and V2 functions can be made; start immediate corresponds to a V1 function). The interface definition of the function can be found via (Import/Export). The function needs two input parameters (both correspond to records of SAP tables): account with reference type `ZHPIxxACC` (type spec. LIKE) and entry with reference type `ZHPIxxENTR`. Check Pass Value for both input parameters. In the source code of the function, you first have to declare both tables that will be updated. The contents of the two input parameters can be copied via `MOVE input_parameter TO table.` Then, inserts and updates of database tables can be handled similar to the previous exercise.

Now, you have to replace the direct interaction with the database by the call of the update function. This update function shall be executed in a dedicated update process (`CALL FUNCTION ... IN UPDATE TASK`). When invoking the update task, the corresponding parameter values have to be specified. In order to start the update task not before the end of the dialog transaction, `Commit Work` or `Rollback Work` must not appear before the final Dynpro (`0300` in case of success and `0400` in case of failures). Define transaction ID `ZUTxx` for this dialog transaction.

Verify whether the update is really effected at the end of the transaction (i.e., by taking a look at the entry data using the Data Browser when the dialog transaction is still in `0300`). The Data Browser can be found at Utilities → Table Contents → Display when you have started the ABAP Dictionary to display the corresponding table.

Submission of results: Submission of results is not mandatory. However, you are invited to submit your results by email to `heiko.schuldt@hpi.uni-potsdam.de` indicating: "ERP: Exercise 4" in the subject line. This email should contain your observations from question 2 and your SAP R/3 user ID (the programs can then be accessed via your development class).

[1] Similar to lock objects, this is also an exception of the SAP naming convention (prefix `Z_`).