Robert Cremanns

IBS Financial Services

Bank Analyzer

Einführung in das Parallelverarbeitungstool

16. Mai 2002

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument gibt eine Einführung in das Parallelverarbeitungstool (PV-Tool) von Thomas Bollmeier. Weitere Details sind in der Entwicklerdokumentation beschrieben. Diese befindet sich im IBU FS InfoCenter unter IBU FS – Projects – NAD – Tools & Technology – Job Control – Entwicklerdokumentation.

1.1 Einschränkungen

Dieses Dokument beschreibt nicht alle Möglichkeiten des PV-Tools. Insbesondere wird nur der Fall beschrieben, dass die Funktionsbausteine zum Starten von Parallelisierungsläufen synchron durchlaufen werden, d.h. die Funktionsbausteine werden erst nach Ende aller parallelen Prozesse verlassen.

Die Verarbeitung eines Massenlaufs erfolgt in einem oder mehreren Schritten. Dieses Dokument beschreibt nur den Fall, dass ein Massenlauf aus einem Schritt besteht.

2 Einführung in das PV-Tool

2.1 Grundkonzept

Das PV-Tool bietet eine Programmierschnittstelle für Anwendungen zur Implementierung einer Parallelverarbeitung. Hierzu werden durch die Anwendung voneinander unabhängige Einheiten gebildet, die parallel verarbeitet werden können. Diese Einheiten heißen Pakete. Die Anwendung bestimmt für jedes Paket, welche Daten in dem Paket bearbeitet werden sollen und wie die Verarbeitung erfolgen soll. Das PV-Tool übernimmt dann die Abarbeitung der Pakete innerhalb einer parallel laufenden Menge von Hintergrundprozessen. Diese Prozesse werden vom PV-Tool automatisch generiert. Das PV-Tool sorgt außerdem für die Synchronisation der parallelen Prozesse.

Diagramm 1 zeigt das Zusammenspiel zwischen Anwendung und PV-Tool. Die Anwendung startet das PV-Tool durch Aufruf des Funktionsbausteins BANK_MAP_PP_START. Das PV-Tool startet dann die parallelen Prozesse und ruft zu bestimmten Zeitpunkten bestimmte Callback-Funktionsbausteine der Anwendung auf. In diesen Callback-Funktionsbausteinen definiert die Anwendung die Aufteilung in die Pakete und die Verarbeitung in den einzelnen Paketen.

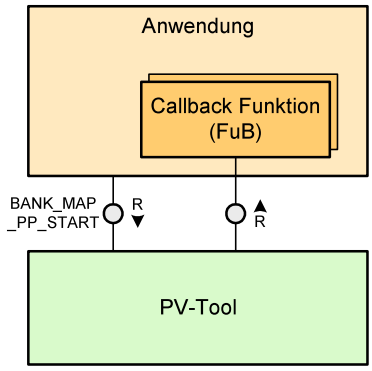

Diagramm 1 Zusammenspiel zwischen Anwendung und PV-Tool

Es gibt eine feste Menge von Zeitpunkten, zu denen das PV-Tool Callback-Funktionsbausteine der Anwendung aufruft. Diese Callback-Funktionsbausteine heißen auch Methoden und werden durch Nummern gekennzeichnet. Die wichtigsten Methoden sind:

- 0205: Paketvorlagen erzeugen
- 1000: Initialisierung eines Arbeitspaketes
- 1300: Objekte bearbeiten

Diese Methoden werden in den folgenden Abschnitten erläutert.

Für jede zu parallelisierende Anwendung muss im Customizing des PV-Tools eine sogenannte Anwendungsart definiert werden. Weiterhin muss im Customizing des PV-Tools für jede Anwendungsart und für jede Methode der Name des entsprechenden Callback-Funktionsbausteins bekannt gemacht werden.

2.2 Einfachste Verwendung des PV-Tools

Das PV-Tool bietet vielfältige Möglichkeiten. Dieser Abschnitt beschreibt die einfachste Verwendungsmöglichkeit.

Das PV-Tool wird mit dem Funktionsbaustein BANK_MAP_PP_START aufgerufen. Dieser Funktionsbaustein hat u.a. folgende Importparameter:

Funktionsbaustein BANK_MAP_PP_START

Name	Typ	Bedeutung	Import(I) / Export(E)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_STR_APPL_PARAM	(von der Anwendung zu definieren)	globale Anwendungsparameter	I

Alle anderen Importparameter sind optional und werden hier nicht verwendet.

In dem Parameter I_APPLCATG muss die Anwendungsart übergeben werden. In dem Parameter I_STR_APPL_PARAM können globale Anwendungsparameter übergeben werden. Global bedeutet hier, dass die Anwendungsparameter für den gesamten Lauf des PV-Tools gelten und nicht Paket-spezifisch sind. Der Typ dieses Parameters ist eine Struktur, die von der Anwendung im ABAP Dictionary definiert werden muss. Jedes Feld dieser Struktur muss in ein Character-Feld der Länge 45 passen. Diese Einschränkung kann allerdings auch umgangen werden.

Diagramm 2 zeigt den Ablauf. Die Anwendung startet das PV-Tool durch Aufruf des Funktionsbausteins BANK_MAP_PP_START. Das PV-Tool ruft dann den Callback-Funktionsbaustein der Anwendung für die Methode 0205 (Paketvorlagen erzeugen) auf. Dies bedeutet im einfachsten Fall, dass die Anwendung dem PV-Tool nur die Anzahl der Pakete mitteilt. Im folgenden wird dann jedes Paket durch eine Nummer identifiziert. Die Nummern der Pakete sind 1, 2 usw. bis Anzahl der Pakete. Anschließend startet das PV-Tool die parallelen Prozesse. Im Diagramm ist nur einer dieser parallelen Prozesse gezeigt. In jedem der parallelen Prozesse wird in einer Schleife solange ein weiteres Paket bearbeitet, solange es noch unbearbeitete Pakete gibt. In dem im Diagramm gezeigten parallelen Prozess ist nur die Bearbeitung von zwei Paketen gezeigt. Für jedes Paket, das bearbeitet wird, ruft das PV-Tool zwei Methoden auf, und zwar Methode 1000 (Initialisierung eines Arbeitspaketes) und Methode 1300 (Objekte bearbeiten). Mit der Methode 1000 übergibt das PV-Tool der Anwendung alle Parameter, insbesondere die globalen Anwendungsparameter und die Paketnummer. Typischerweise sind alle Callback-Funktionsbausteine einer Anwendung in einer Funktionsgruppe zusammengefasst, so dass die Anwendung die Parameter in den globalen Variablen der Funktionsgruppe speichern kann und somit zur Verfügung stehen, wenn das PV-Tool die weiteren Funktionsbausteine aufruft. Mit der Methode 1300 fordert das PV-Tool die Anwendung auf, die Verarbeitung des Pakets durchzuführen. Die Methode 1300 hat keine Import-Parameter.

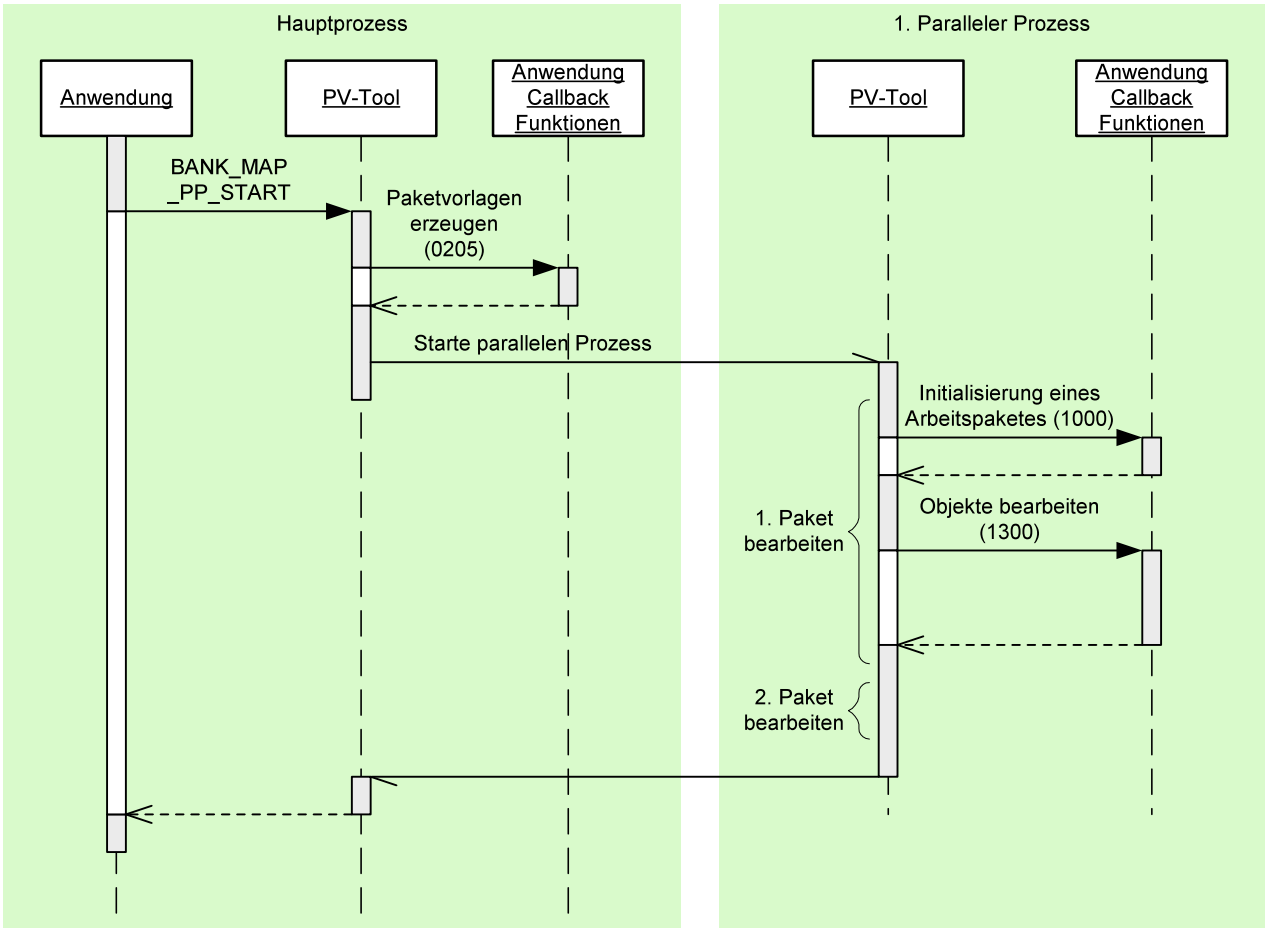

Diagramm 2 Beispielablauf

Es werden hier nur folgende Parameter der Methode 0205 verwendet:

Methode 0205: Paketvorlagen erzeugen

Parameter	Typ	Bedeutung	Import(I) / Export(E)
I_STR_PARAM	(von der Anwendung zu definieren)	globale Anwendungsparameter	I
E_CNT_PACKAGES	BANK_DTE_PP_CNT_PACKAGES	Anzahl der Pakete	E

Der Wert des Importparameters für die globalen Anwendungsparameter ist der, der dem PV-Tool mit Aufruf des Funktionsbausteins BANK_MAP_PP_START übergeben wurde. In dem Exportparameter E_CNT_PACKAGES übergibt die Anwendung die Anzahl der Pakete.

Die Methode 1000 hat u.a. folgende Parameter:

Methode 1000: Initialisierung eines Arbeitspaketes

Name	Typ	Bedeutung	Import(I) / Export(E)
I_STR_APPLPARAM	(von der Anwendung zu definieren)	globale Anwendungsparameter	I
I_STR_PACKAGE_KEY	BANK_STR_PP_PACKAGEKEY	Schlüssel der Paketvorlage	I

Der Wert des Importparameters für die globalen Anwendungsparameter ist der, der dem PV-Tool mit Aufruf des Funktionsbausteins BANK_MAP_PP_START übergeben wurde.

In dem Importparameter I_STR_PACKAGE_KEY teilt das PV-Tool der Anwendung mit, welches Paket bearbeitet werden soll. Der Typ BANK_STR_PP_PACKAGEKEY hat zwei Komponenten. Die zweite Komponente ist die Paketnummer.

Es werden keine Parameter der Methode 1300 verwendet.

Zusammenfassend müssen also folgende Schritte durchgeführt werden, um das PV-Tool in der oben beschriebenen Weise zu nutzen:

- Im ABAP Dictionary muß eine Struktur für die globalen Anwendungsparameter definiert werden.
- Die Funktionsbausteine für die Methoden 0205, 1000 und 1300 müssen implementiert werden.
- Im Customizing des PV-Tools (Transaktion bank_cus_pp) muß eine Anwendungsart definiert werden und für diese Anwendungsart müssen die Namen der Funktionsbausteine für die Methoden 0205, 1000 und 1300 angeben werden. Alle anderen Felder für die Anwendungsart bleiben leer.

Nun kann das PV-Tool durch Aufruf des Funktionsbausteins BANK_MAP_PP_START mit folgendem Muster gestartet werden:

```
CALL FUNCTION 'BANK_MAP_PP_START'
  EXPORTING
    I_APPLCATG      =
    I_STR_APPL_PARAM =
```

In dem Parameter I_APPLCATG wird die Anwendungsart und in dem Parameter I_STR_APPL_PARAM werden die globalen Anwendungsparameter übergeben.

2.3 Zentrale Begriffe

Das folgende Diagramm fasst die wichtigsten Begriffe des PV-Tools zusammen und zeigt deren Zusammenhänge.

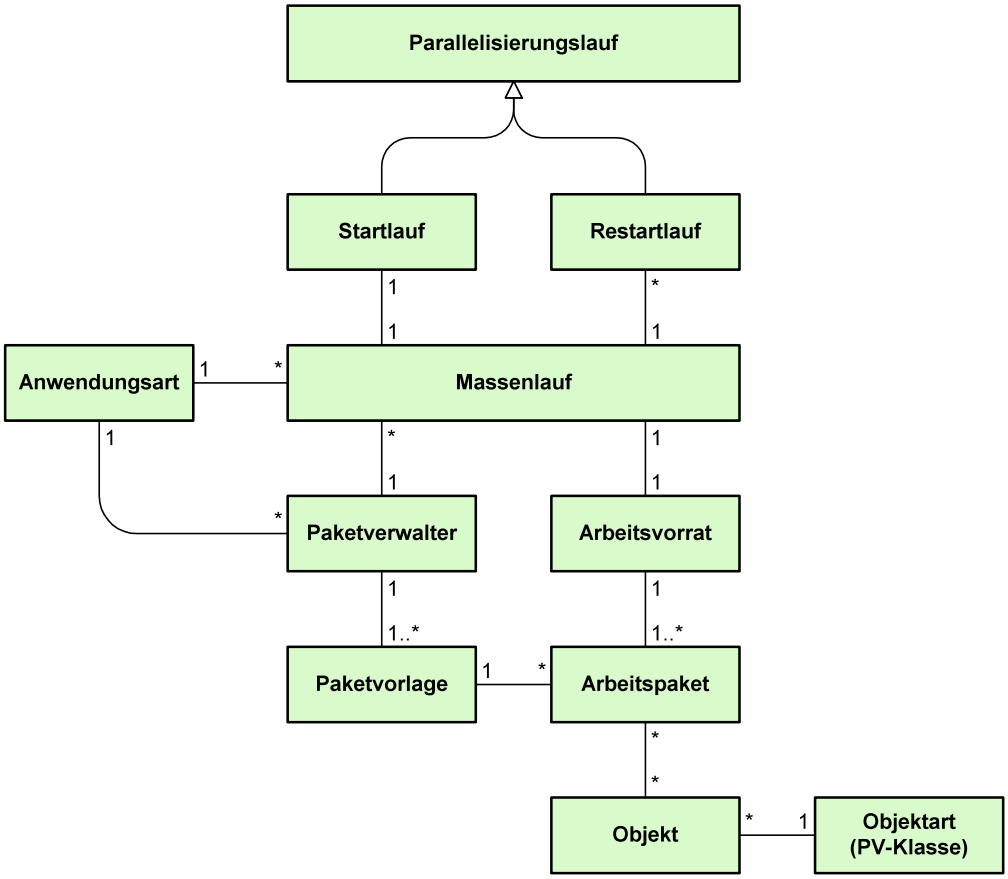

Diagramm 3 Zentrale Begriffe

Eine wesentliche Eigenschaft des PV-Tools wurde bisher noch nicht erwähnt: Das PV-Tool bietet die Möglichkeit, einen Arbeitsvorrat zu verwalten. Ein Arbeitsvorrat besteht aus der Menge der offenen, d.h. noch nicht abschließend bearbeiteten Objekte. Damit verbunden ist eine Wiederaufsetzfähigkeit der Parallelverarbeitung, d.h. beim Wiederaufsetzen der Parallelverarbeitung werden nur noch die offenen Objekte erneut bearbeitet. Der Arbeitsvorrat wird unten näher erläutert.

Das PV-Tool bietet also die Möglichkeit, Objekte im Arbeitsvorrat zu verwalten. Es macht keine Vorgaben über die Art der zu verarbeitenden Objekte. Objekte können z.B. Konten oder Aufträge sein. Um die verschiedenen Arten unterscheiden zu können, werden im Customizing des PV-Tools (Transaktion bank_cus_pp) **Objektarten** definiert. Anstatt Objektart wird manchmal auch der Begriff PV-Klasse verwendet. Die einzelnen **Objekte** werden vom PV-Tool über einen **Objektschlüssel** (Typ BANK_STR_OBJKEY) bestehend aus Objektart (Typ BKK_OBJCATG = CHAR 4) und **Objektnummer** (Typ BKK_OBJNO = CHAR 40) angesprochen.

Die Parallelverarbeitung wird mit dem Funktionsbaustein BANK_MAP_PP_START gestartet. Wiederaufsetzen erfolgt mit dem Funktionsbaustein BANK_MAP_PP_RESTART. Die Ausführung des Funktionsbausteins BANK_MAP_PP_START wird im folgenden mit **Startlauf** bezeichnet und die Ausführung des Funktionsbausteins BANK_MAP_PP_RESTART mit **Restartlauf**. Ein **Parallelisierungslauf** ist ein Startlauf oder ein Restartlauf.

Jeder Startlauf erzeugt einen neuen **Massenlauf**. Es gibt eine 1-zu-1-Beziehung zwischen den Startläufen und den Massenläufen. Ein Massenlauf wird durch einen Schlüssel (Typ BANK_STR_RUNKEY) eindeutig identifiziert. Ein Massenlauf wird vom PV-Tool in der Datenbank gespeichert, zusammen mit den Parametern mit denen der entsprechende Startlauf gestartet wurde und mit dem Schlüssel des entsprechenden Arbeitsvorrats. Jeder Massenlauf hat seinen eigenen Arbeitsvorrat. Der Arbeitsvorrat wird während des Startlaufs angelegt und vom PV-Tool auf der Datenbank gespeichert wird. Der Massenlaufschlüssel wird normalerweise vom PV-Tool vergeben und ist Exportparameter des Funktionsbausteins BANK_MAP_PP_START und Importparameter des Funktionsbausteins BANK_MAP_PP_RESTART. Ein

Restartlauf bezieht sich also immer auf einen schon existierenden Massenlauf mit dem zugehörigen Arbeitsvorrat.

Jeder Massenlauf gehört zu genau einer **Anwendungsart**. Die Anwendungsart ist Importparameter des Funktionsbausteins BANK_MAP_PP_START.

Ein **Paketverwalter** definiert die Paketeinteilung für Parallelisierungsläufe. Ein Paketverwalter besteht aus einer Menge von **Paketvorlagen**. Eine Paketvorlage besteht mindestens aus einer Paketnummer. Eine Paketvorlage kann auch weitere Daten enthalten, und zwar:

- Intervallgrenzen
- Objektliste
- Anwendungsparameter zum Paket

Diese Möglichkeiten werden in Abschnitt 2.4 genauer beschrieben. Im Abschnitt 2.2 bestand jede Paketvorlage nur aus der Paketnummer, hatte also keine weiteren Daten. Die Paketvorlagen eines Paketverwalters sind durchnummeriert von 1 bis Anzahl der Paketvorlagen. Ein Paketverwalter wird durch einen Schlüssel (Typ BANK_PACKMANID) eindeutig identifiziert, den das PV-Tool vergibt. Der Schlüssel einer Paketvorlage (Typ BANK_STR_PP_PACKAGEKEY) besteht aus zwei Komponenten, dem Schlüssel für den Paketverwalter und der Paketnummer (Typ BKK_INTERVNO).

Paketverwalter werden von der Anwendung in der Methode 0205 definiert und vom PV-Tool in der Datenbank gespeichert. Ein Paketverwalter wird nach der Definition nicht mehr geändert. Jeder Massenlauf hat genau einen Paketverwalter, der für alle Parallelisierungsläufe dieses Massenlaufs gilt. Im allgemeinen wird für jeden Massenlauf ein neuer Paketverwalter erzeugt, und zwar am Anfang des entsprechenden Startlaufs durch ein oder mehrere Aufrufe der Methode 0205. Übergibt die Anwendung im ersten Aufruf der Methode 0205 nur die Anzahl der Pakete, dann erfolgt kein weiterer Aufruf und jede Paketvorlage besteht nur aus der Paketnummer. Die Anwendung kann beim ersten Aufruf weitere Daten zur ersten Paketvorlage übergeben. Dann erfolgen weitere Aufrufe, und zwar ein Aufruf für jede Paketvorlage.

Es gibt auch die Möglichkeit, Paketverwalter mit dem Funktionsbaustein BANK_MAP_PACKMAN_CONSTRUCTOR zunächst unabhängig von einem Massenlauf zu definieren. Solche Paketverwalter heißen Standard-Paketverwalter und können dann für mehrere Massenläufe wiederverwendet werden. Sie werden dann beim Aufruf des Funktionsbausteins BANK_MAP_PP_START als Importparameter übergeben.

Jeder Massenlauf hat einen eigenen Arbeitsvorrat. Der **Arbeitsvorrat** besteht aus einer Menge von Arbeitspaketen. Jedes **Arbeitspaket** besteht aus einer Liste von Objektschlüsseln (bestehend aus Objektart und Objektnummer) mit einem Status zu jedem Objektschlüssel. Der Arbeitsvorrat wird während des Startlaufs angelegt. Die Objekte und der Status der Objekte teilt die Anwendung dem PV-Tool über Callback-Funktionsbausteine mit. Der Arbeitsvorrat wird vom PV-Tool persistent auf der Datenbank gespeichert. In jedem weiteren Parallelisierungslauf desselben Massenlaufs teilt die Anwendung dem PV-Tool über Callback-Funktionsbausteine für jedes Objekt den veränderten Status mit.

Ein Arbeitsvorrat gehört zu genau einem Massenlauf, der wiederum genau einen Paketverwalter hat. Somit gehört zu einem Arbeitsvorrat genau ein Paketverwalter. Dieser Paketverwalter legt auch die Einteilung des Arbeitsvorrats in Arbeitspakete fest. Konkret gibt es eine 1-zu-1-Beziehung zwischen den Arbeitspaketen eines Arbeitsvorrats und den Paketvorlagen des zugehörigen Paketverwalters.

2.4 Paketvorlagen erzeugen

Ein Paketverwalter besteht aus einer Menge von Paketvorlagen. Eine Paketvorlage besteht mindestens aus einer Paketnummer und kann auch weitere Daten enthalten. Der Schlüssel einer Paketvorlage (Typ BANK_STR_PP_PACKAGEKEY) besteht aus zwei Komponenten, dem Schlüssel für den Paketverwalter (Typ BANK_PACKMANID) und der Paketnummer (Typ BKK_INTERVNO). Die Paketvorlagen eines Paketverwalters sind durchnummeriert von 1 bis Anzahl der Paketvorlagen.

Paketverwalter werden immer mit der Methode 0205 definiert. Das PV-Tool ruft für einen Paketverwalter ein- oder mehrmals hintereinander die Methode 0205 auf, um die Daten zu diesem Paketverwalter von der Anwendung zu erfragen und speichert anschließend den Paketverwalter in der Datenbank.

Die Methode 0205 hat folgende Schnittstelle:

Methode 0205: Paketvorlagen definieren

Parameter	Typ	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_STR_PARAM		globale Anwendungsparameter	I
I_STR_PACKAGE_KEY	BANK_STR_PP_PACKAGEKEY	Schlüssel der Paketvorlage	I
E_LIMIT_HIGH	BKK_OBJNO	Obergrenze Paket	E
E_LIMIT_LOW	BKK_OBJNO	Untergrenze Paket	E
E_TAB_OBJKEY	BANK_TAB_OBJKEY	Objektliste	E
E_CNT_PACKAGES	BANK_DTE_PP_CNT_PACKAGES	Anzahl der Pakete	E
E_STR_PACKPARAM		Anwendungsparameter zum Paket	E
NOT_FOUND			A

Es gibt folgende Möglichkeiten, Paketverwalter zu definieren:

- Anzahl der Pakete
- Intervallgrenzen
- Objektliste
- Anwendungsparameter zum Paket

Für einen Paketverwalter muss eine dieser Methoden festgelegt werden. Alle Paketvorlagen eines Paketverwalters werden also nach derselben Methode definiert.

In allen Fällen außer dem ersten wird die Methode 0205 mehrmals aufgerufen, und zwar für jedes Paket genau einmal. Bei jedem Aufruf werden also die Daten zu einem Paket übergeben. Gibt es kein weiteres Paket, übergibt die Anwendung die Ausnahme NOT_FOUND. Die Anwendung muss in diesen Fällen nicht explizit die Anzahl der Pakete übergeben.

2.4.1 Paketvorlagen definieren: Anzahl der Pakete

Die Methode 0205 wird nur einmal aufgerufen und dabei wird nur die Anzahl der Pakete übergeben. Diese Möglichkeit ist im Abschnitt 2.2 beschrieben. Jede Paketvorlage besteht nur aus einer Paketnummer und enthält sonst keine weiteren Daten.

2.4.2 Paketvorlagen definieren: Intervallgrenzen

Jedes Paketvorlage hat eine Untergrenze und eine Obergrenze. Diese werden in den Parametern E_LIMIT_LOW und E_LIMIT_HIGH mit der Methode 0205 übergeben.

2.4.3 Paketvorlagen definieren: Objektliste

Mit der Methode 0205 wird zu jedem Paket in dem Parameter E_TAB_OBJKEY eine Objektliste übergeben. Der Typ dieses Parameters (BANK_TAB_OBJKEY) ist eine interne Tabelle mit Zeilentyp BANK_STR_OBJKEY. Diese Struktur besteht aus der Objektart und der Objektnummer.

2.4.4 Paketvorlagen definieren: Anwendungsparameter zum Paket

Die oben beschriebenen Möglichkeiten, Paketvorlagen zu definieren, werden vom PV-Tool mit eigenen Datenbanktabellen unterstützt, d.h. die Intervallgrenzen und die Objektlisten werden vom PV-Tool in eigenen Datenbanktabellen gespeichert. Sollen die Paketvorlagen nach anderen Kriterien gebildet werden, muss die Anwendung hierzu eigene Datenbanktabellen verwenden.

Mit der Methode 0205 können zu jedem Paket in dem Parameter E_STR_PACKPARAM beliebige Anwendungsparameter zum Paket übergeben werden. Der Typ des Parameters E_STR_PACKPARAM ist eine Struktur. Diese Struktur wird von der Anwendung definiert und muß ein Feld MANDT haben und das Include BANK_STR_PP_PACKAGEKEY für den Schlüssel einer Paketvorlage beinhalten. Die weiteren Felder können von der Anwendung beliebig definiert werden, insbesondere kann die Struktur also auch eine geschachtelte oder tiefe Struktur sein.

Weiterhin muss ein Tabellentyp im ABAP Dictionary angelegt werden, der als Zeilentyp die oben beschriebene Struktur hat. Im Customizing des PV-Tools muss bei der Anwendungsart dieser Tabellentyp als Typ für die Paketparameter angegeben werden.

Außerdem muss die Anwendung eine eigene Datenbanktabelle für die Anwendungsparameter zu den Paketen anlegen. Diese Tabelle muss ein Feld MANDT als Schlüsselfeld haben und das Include BANK_STR_PP_PACKAGEKEY für den Schlüssel einer Paketvorlage beinhalten. Die Felder dieses Include müssen ebenfalls Schlüsselfelder sein. Die weiteren Felder können von der Anwendung beliebig definiert werden. Als Typ für den Parameter E_STR_PACKPARAM der Methode 0205 kann zum Beispiel auch diese Tabelle gewählt werden.

Schließlich müssen die Methoden 0206 (Parameter zu Paketen sichern) und 0207 (Parameter zu Paketen löschen) implementiert werden. Diese müssen nur implementiert werden, wenn Anwendungsparameter zum Paket verwendet werden. In diesem Fall ruft das PV-Tool immer dann, wenn es einen Paketverwalter in den eigenen Datenbanktabellen speichert oder löscht, zusätzlich noch die Methode 0206 bzw. 0207 auf. Die genauen Schnittstellenbeschreibungen dieser Methoden befinden sich in der der Entwicklerdokumentation.

Der Ablauf ist wie folgt: Bei den Aufrufen der Methode 0205 übergibt die Anwendung dem PV-Tool die Anwendungsparameter zu den einzelnen Paketen jeweils in Form einer Struktur. Diese Daten werden vom PV-Tool in einer internen Tabelle gespeichert. Der Typ dieser Tabelle ist der von der Anwendung im Customizing hinterlegte Tabellentyp. Mit Aufruf der Methode 0206 übergibt das PV-Tool die interne Tabelle an die Anwendung und fordert die Anwendung auf, den Inhalt der internen Tabelle in der speziell dafür angelegten Datenbanktabelle zu speichern. Dabei ist zu beachten, dass das PV-Tool möglicherweise die Methode 0206 mehrfach aufruft, also die Daten zu einem Paketverwalter nicht in einem Aufruf an die Anwendung übergibt, sondern auf mehrere Aufrufe verteilt.

Wird diese Möglichkeit, Paketvorlagen zu definieren, gewählt, dann sind also die obigen Schritte durchzuführen. Werden keine Anwendungsparameter zu Paketen verwendet, dann entfallen die obigen Schritte, insbesondere wird der Parameter E_STR_PACKPARAM der Methode 0205 nicht verwendet, der Customizing-Eintrag für den Typ für die Paketparameter entfällt, es muss keine Datenbanktabelle für die Anwendungsparameter zu den Paketen angelegt werden, und die Methoden 0206 und 0207 werden nicht implementiert.

Wie oben beschrieben, werden die Anwendungsparameter zu den Paketen von der Anwendung an das PV-Tool übergeben und anschließend zurück bevor sie dann von der Anwendung in der Datenbank gespeichert werden. Dieses möglicherweise umständliche Verfahren kann dadurch umgangen werden, indem die Anwendung die eigentlichen Anwendungsparameter zu den Paketen nicht in der Methode 0205 an das PV-Tool übergibt, sondern in einer eigenen internen Tabelle speichert. Bei den Aufrufen der Methode 0206 müssen dann die Daten zu den entsprechenden Paketvorlagen aus der eigenen internen Tabelle gelesen werden um dann auf der Datenbank zu speichern.

2.4.5 Vergleich

Die erste Möglichkeit, Paketvorlagen zu bilden, bei der nur die Anzahl der Pakete angegeben wird, ist vermutlich nur in wenigen Fällen ausreichend. In den meisten Fällen müssen die Paketvorlagen außer der Paketnummer weitere Daten enthalten.

Häufig erfolgt die Einteilung in Pakete dadurch, dass die Anwendung festlegen möchte, welche Objekte in welchem Paket bearbeitet werden sollen. Das ist möglich durch die explizite Angabe von Objektlisten. Die Objektlisten sollten dabei so bestimmt werden, dass die verschiedenen Pakete möglichst gleich viele

Objekte enthalten. Der Nachteil dieses Verfahrens ist, dass der Zeitaufwand zur Bestimmung der Objektlisten sehr groß sein kann. Dabei ist zu beachten, dass die Bestimmung der Paketvorlagen im Hauptprozess stattfindet, also noch nicht parallelisiert.

Eine Alternative ist, die zu bearbeitenden Objekte pro Paket implizit durch Angabe von Intervallgrenzen festzulegen. Die Intervallgrenzen sind dabei so zu bestimmen, dass jedes zu bearbeitende Objekt innerhalb genau einem Intervall liegt. Der Vorteil dieser Methode ist, dass die eigentliche Selektion der Objekte innerhalb der Intervalle erst in den parallelen Prozessen erfolgt. Der Nachteil dieses Verfahrens ist, dass es schwierig ist, die Intervalle so festzulegen, dass jedes Paket möglichst gleich viele Objekte enthält. Den gesamten Bereich von Objektnummern in gleich große Intervalle aufzuteilen ist möglicherweise nicht ausreichend, wenn die Objekte in diesem Bereich nicht gleichverteilt sind.

Sind die Daten, die die Anwendung pro Paketvorlage speichern will, nicht in Form von Intervallgrenzen oder Objektlisten darstellbar, so ist die vierte Möglichkeit zu wählen.

2.5 Ablauf

Das folgende Diagramm zeigt den prinzipiellen Ablauf eines Parallelisierungslaufs (Startlauf und Restartlauf), insbesondere zeigt es die wichtigsten Methoden mit den Zeitpunkten, wann diese Methoden vom PV-Tool aufgerufen werden. Nicht alle Methoden sind hier gezeigt.

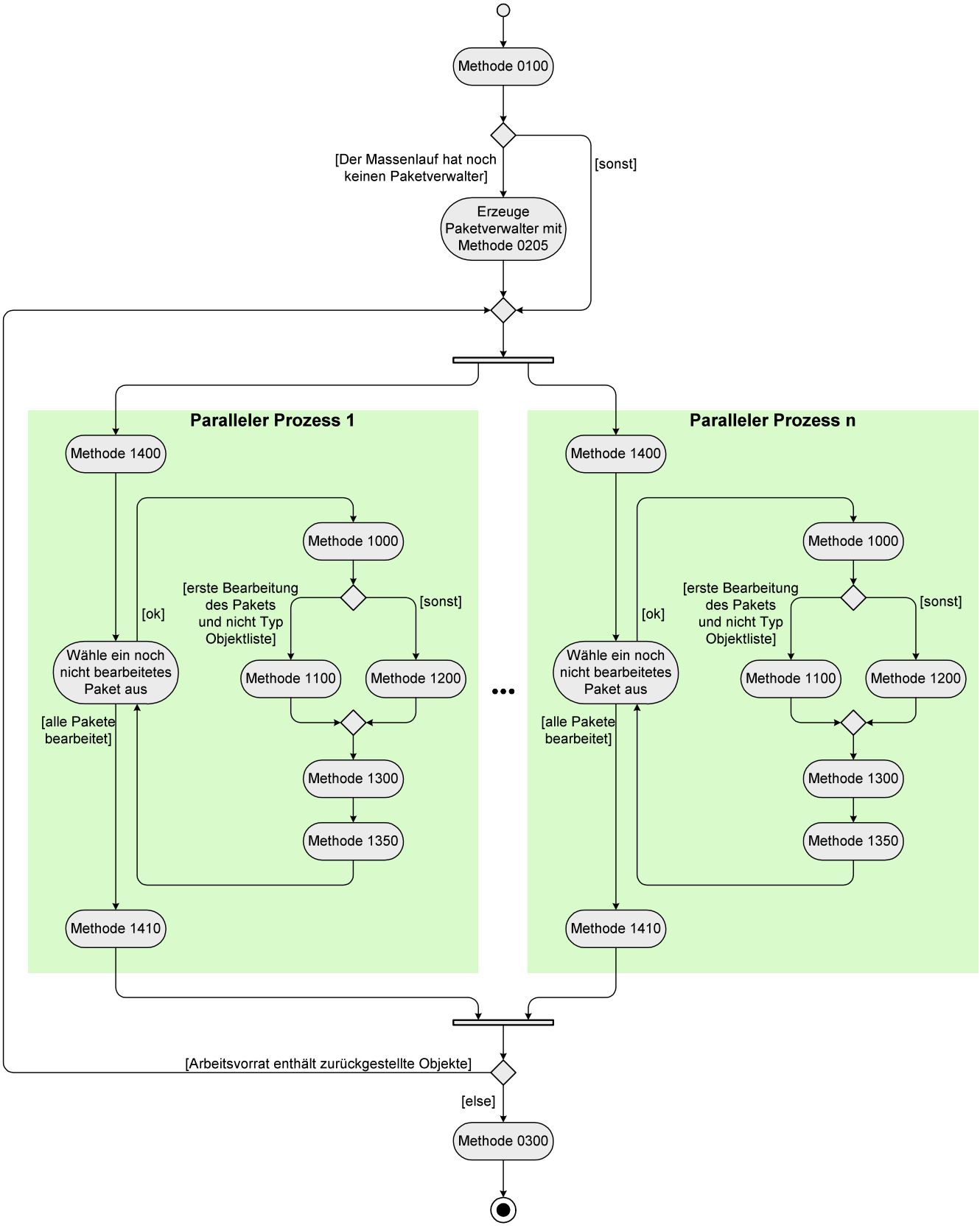

Diagramm 4 Ablauf

Nachdem das PV-Tool die Methode 0100 aufgerufen hat, ruft es, falls dem Massenlauf noch kein Paketverwalter zugeordnet ist, die Methode 0205 ein- oder mehrmals auf zur Definition eines Paketverwalters. Dies ist also nur bei einem Startlauf der Fall und nur wenn kein Standard-Paketverwalter beim Aufruf des Funktionsbausteins BANK_MAP_PP_START übergeben wurde.

Anschließend erzeugt das PV-Tool die parallelen Prozesse. Innerhalb von jedem parallelen Prozess wird eine Schleife durchlaufen, in der pro Durchlauf ein noch unbearbeitetes Paket angefordert und danach durch die Anwendung bearbeitet wird. Zwischen Erzeugen der parallelen Prozesse und anschließender Synchronisation wird jedes Paket genau einmal bearbeitet. Die Zuordnung eines Pakets zu einem der parallelen Prozesse und die Reihenfolge der Pakete, die in einem parallelen Prozess bearbeitet werden, ist nicht festgelegt.

Nach Synchronisation der parallelen Prozesse ist also jedes Paket genau einmal bearbeitet worden, und es gibt einen Arbeitsvorrat, also zu jedem Paket eine Liste von Objekten mit Status. Wie schon erwähnt, gibt es die Möglichkeit des Wiederaufsetzens, also im Restartlauf werden nur noch die offenen Objekte des Arbeitsvorrats noch einmal bearbeitet. Das Diagramm zeigt noch eine weitere Möglichkeit. Innerhalb desselben Parallelisierungslauf gibt es die Möglichkeit, dass die Bearbeitung wiederholt wird. Das ist abhängig vom Status der Objekte im Arbeitsvorrat. Die wichtigsten Ausprägungen, die der Status eines Objekts haben kann, sind:

- Initial: Das Objekt wurde noch nicht bearbeitet
- Finished: Das Objekt wurde erfolgreich bearbeitet
- Retry: Das Objekt ist zurückgestellt
- Restart: Das Objekt ist zum Wiederaufsetzen vorgemerkt

Der Status Retry bedeutet, dass das Objekt im selben Parallelisierungslauf noch einmal bearbeitet wird. Der Status Restart bedeutet, dass das Objekt erst im nächsten Restartlauf wieder bearbeitet wird. Ein erfolgreich bearbeitetes Objekt wird nicht noch einmal bearbeitet.

Die gezeigte Schleife wird also solange wiederholt, bis es keine zurückgestellten Objekte (Status retry) mehr im Arbeitsvorrat gibt. Eine Endlosschleife kann hier durch Customizing-Einstellungen verhindert werden (siehe Abschnitte 3.3.2.2.3 und 3.5.2). Bei jedem Durchlauf der Schleife, die mit Erzeugen der parallelen Prozesse beginnt und mit der Synchronisation dieser Prozesse endet, wird jedes Paket wieder genau einmal bearbeitet.

Der Ablauf bei der Bearbeitung eines einzelnen Pakets innerhalb eines parallelen Prozesses ist wie folgt: Zuerst wird immer die Methode 1000 aufgerufen. Mit dieser Methode übergibt das PV-Tool der Anwendung alle Parameter, die für die Bearbeitung des Pakets notwendig sind. Diese Daten werden typischerweise von der Anwendung in globalen Variablen der Funktionsgruppe gespeichert und stehen somit zur Verfügung, wenn das PV-Tool die nachfolgenden Methoden aufruft.

Die Methode 1000 hat nur Import- und keine Exportparameter. Die wichtigsten Importparameter sind:

Methode 1000: Initialisierung eines Arbeitspaketes

Name	Typ	Bedeutung	Import(I) / Export(E)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart	I
I_STR_APPLPARAM		globale Anwendungsparameter	I
I_STR_PACKAGE_KEY	BANK_STR_PP_PACKAGEKEY	Schlüssel der Paketvorlage	I
I_LIMIT_LOW	BKK_OBJNO	Untergrenze	I
I_LIMIT_HIGH	BKK_OBJNO	Obergrenze	I
I_XRESTART	XFELD	Kennzeichen: Wiederaufsetzen	I

Anschließend wird entweder die Methode 1100 oder 1200 aufgerufen. Diese Methoden betreffen den Arbeitsvorrat.

Bei der ersten Bearbeitung eines Pakets in einem Massenlauf (also insbesondere nicht im Restartlauf) muss das entsprechende Arbeitspaket gebildet werden. Das Arbeitspaket besteht aus einer Liste von Objekten mit Status. Falls der Paketverwalter nicht vom Typ Objektliste ist, ermittelt die Anwendung in Methode 1100 diese Objektliste und übergibt sie an das PV-Tool. Das PV-Tool bildet dann das Arbeitspaket mit dieser Objektliste. Alle Objekte erhalten den Status initial.

Die Schnittstelle der Methode 1100 ist:

Methode 1100: Selektion von Anwendungsdaten

Name	Type	Bedeutung	Import(I) / Export(E) / Ausnahme (A)
E_TAB_OBJKEY	BANK_TAB_OBJKEY	selektierte Objekte	E
NOT_FOUND		keine Objekte gefunden	A

Falls der Paketverwalter vom Typ Objektliste ist, wird nicht die Methode 1100 aufgerufen, sondern das PV-Tool übernimmt die Objektliste zum Paket aus dem Paketverwalter in das entsprechende Arbeitspaket, wobei auch jedes Objekt den Status initial erhält. Zusätzlich übergibt das PV-Tool die Objektliste der Anwendung mit der Methode 1200. Diese Methode hat folgenden Importparameter:

Methode 1200: Nachselektion von Anwendungsdaten

Name	Typ	Bedeutung	Import(I) / Export(E)
I_TAB_OBJKEY	BANK_TAB_OBJKEY	wiederaufzusetzende Objekte	I

Zusätzlich hat die Anwendung mit dieser Methode die Möglichkeit, über die Exportparameter (hier nicht aufgeführt) Änderungen an dieser Objektliste vorzunehmen.

Bei der wiederholten Bearbeitung eines Pakets wird immer die Methode 1200 aufgerufen, weil das entsprechende Arbeitspaket mit der Liste der zu bearbeitenden Objekte bereits existiert.

Nach Aufruf der Methode 1100 bzw. 1200 kennt nun die Anwendung das Arbeitspaket mit den zu bearbeitenden Objekten und wird mit Aufruf der Methode 1300 vom PV-Tool aufgefordert, diese Objekte zu verarbeiten. Die Methode 1300 hat folgende Schnittstelle:

Methode 1300: Objekte bearbeiten

Name	Typ	Bedeutung	Import(I) / Export(E)
E_TAB_STATUS_CHANGE	BANK_TAB_PP_STATUS_CHANGE	Status der Objekte	E

In dem Exportparameter übergibt die Anwendung an das PV-Tool für jedes Objekt einen Status, und zwar finished, retry oder restart. Falls der Status eines Objekts retry oder restart ist, speichert das PV-Tool diesen Status im Arbeitsvorrat. Objekte mit Status finished werden aus dem Arbeitsvorrat gelöscht.

Bei der ersten Bearbeitung eines Pakets bei einem Restartlauf setzt das PV-Tool den Status aller Objekte im Arbeitsvorrat auf initial.

Die Methoden 0100, 1350, 1400, 1410 spielen eine untergeordnete Rolle und werden hier nicht näher erläutert. Wie im Diagramm zu sehen, werden diese Methoden zu folgenden Zeitpunkten aufgerufen:

- 0100: am Anfang von jedem Parallelisierungslauf

- 1350: am Ende von jeder Paketbearbeitung
- 1400: am Anfang von jedem parallelen Prozess
- 1410: am Ende von jedem parallelen Prozess

Die Methode 0300 wird einmal am Ende von jedem Parallelisierungslauf aufgerufen. Diese Methode hat u.a. folgende Parameter:

Methode 0300: Am Ende des Parallelisierungslaufs

Name	Typ	Bedeutung	Import(I) / Export(E)
I_RUNSTATUS	BANK_DTE_PP_RUNSTATUS	Status des Massenlaufs	I
E_APPL_RETURNCODE	SY-SUBRC	Rückgabecode der Anwendung	E

Die Funktionsbausteine BANK_MAP_PP_START und BANK_MAP_PP_RESTART haben als Exportparameter den Status des Massenlaufs und den Rückgabecode der Anwendung. Der Status des Massenlaufs wird vom PV-Tool bestimmt und kann folgende Werte annehmen:

- Abgebrochen (3): Dieser Status tritt auf, wenn ein technischer Fehler aufgetreten ist oder wenn es noch Objekte mit Status initial gibt (was ein Fehler der Anwendung wäre).
- Nachbereitung erforderlich (5): Dieser Status tritt auf, wenn es noch offene Objekte im Arbeitsvorrat gibt.
- Erfolgreich beendet (6)

Der Rückgabecode der Anwendung wird von der Anwendung in der Methode 0300 bestimmt.

Falls es am Ende eines Parallelisierungslaufs keine offenen Objekte mehr im Arbeitsvorrat gibt, dann löscht das PV-Tool alle Daten zum Massenlauf von der Datenbank, insbesondere den Arbeitsvorrat und den Paketverwalter (es sei denn, es handelt sich um einen Standard-Paketverwalter).

Es wird empfohlen, dass die Anwendung keine COMMIT-WORK-Befehle absetzt. Das PV-Tool setzt eigene COMMIT-WORK-Befehle ab, und zwar im Hauptprozess vor dem Erzeugen der parallelen Prozesse und in den parallelen Prozessen am Anfang und Ende der Bearbeitung von jedem Paket. Setzt die Anwendung keine COMMIT-WORK-Befehle ab, so ist die Konsistenz zwischen Zustand der Anwendungsdaten und Objektstatus gewährleistet. Ist für eine Anwendung das Absetzen von COMMIT-WORK-Befehlen unumgänglich, muß zuvor für die geänderten Objekte der MAPI-Funktionsbaustein BANK_MAP_PP_CONFIRM_OBJECTS aufgerufen werden (siehe Entwicklerdokumentation).

3 Anbindung einer Anwendung an die Parallelverarbeitung

Dieses Kapitel dient als „Kochbuch“, mit dessen Hilfe ein Anwendungsentwickler in die Lage versetzt werden soll, eine Parallelverarbeitung seiner Anwendung zu implementieren.

3.1 Implementierung der Methoden

Die Anwendung muß zunächst die in Abschnitt 2 beschriebenen Methoden durch entsprechende Funktionsbausteine implementieren, mindestens die Methoden 0205, 1000 und 1300. Alle anderen Methoden sind optional. Um innerhalb der Funktionsbausteine auf gepufferte Daten gemeinsam zugreifen zu können, empfiehlt es sich, die Funktionsbausteine in einer eigenen Funktionsgruppe zusammenzufassen.

Es gibt eine Demo-Anwendung (Anwendungsart DEMO). Die zugehörigen Methoden können als Vorlage für die eigenen Methoden verwendet werden.

3.2 Persistente Speicherung der Anwendungsparameter

Wie schon beschrieben, können beim Aufruf des Funktionsbausteins BANK_MAP_PP_START in dem Parameter I_STR_APPL_PARAM globale Anwendungsparameter übergeben werden. Der Typ dieses Parameters ist eine Struktur, die von der Anwendung im ABAP Dictionary definiert werden muss. Die globalen Anwendungsparameter werden nach dem Start vom PV-Tool in der Datenbank gespeichert, damit sie in den parallelen Prozessen verfügbar sind. Dazu wird entweder der Persistenzdienst des PV-Tools genutzt oder die Anwendung legt eigene Datenbanktabellen zur Speicherung der globalen Anwendungsparameter an.

Nutzt die Anwendung den Persistenzdienst des PV-Tools, dann ist nichts weiter zu tun. Das ist allerdings nur möglich, wenn jedes Feld der Struktur für die globalen Anwendungsparameter in ein Character-Feld der Länge 45 passt. Denn das PV-Tool speichert die Parameter in eigenen DB-Tabellen in Character-Feldern der Länge 45. Ein möglicher Nachteil ist auch die relativ speicherplatzintensive Form der Speicherung.

Um eine eigene Datenhaltung der Anwendungsparameter eines Massenlaufs zu realisieren, muß die Anwendung eine oder ggf. mehrere Datenbanktabellen anlegen. Der Schlüssel der Datenbanktabellen muß den Schlüssel des Massenlaufs umfassen. Außerdem muß die Anwendung zusätzlich noch die Methoden 0110, 0120 und 0130 implementieren.

3.3 Systemcustomizing

Die Transaktion für das Systemcustomizing heißt bank_cus_pp.

3.3.1 Objektart

Hier können die Objektarten gepflegt werden. Jede Objektart hat einen Schlüssel (CHAR 4) und eine Bezeichnung. Die restliche Felder für eine Objektart können leer bleiben.

3.3.2 Anwendungsart

Hier können die Anwendungsarten gepflegt werden. Jede Anwendungsart hat einen Schlüssel (CHAR 4) und eine Bezeichnung.

Falls bei der Paketvorlagenbildung Anwendungsparameter zum Paket verwendet werden, so muss unter „Parametertyp“ ein entsprechender Eintrag gemacht werden (siehe Abschnitt 2.4.4).

Soll das PV-Tool automatisch Anwendungsprotokolle erzeugen, so kann unter „Objekt“ das entsprechende Anwendungsprotokollobjekt angegeben werden.

3.3.2.1 Implementierte Methoden

Pro Anwendungsart werden hier den Methoden die Namen der Anwendungsfunktionsbausteine zugeordnet, die diese Methoden implementieren. Die Einträge für Methoden, die nicht verwendet werden, bleiben leer.

3.3.2.2 Technische Einstellungen

Pro Anwendungsart können folgende technische Einstellungen vorgenommen werden.

3.3.2.2.1 Kennzeichen: alle Objekte persistent

Ist das Kennzeichen gesetzt, dann speichert das PV-Tool bei der ersten Bearbeitung eines Pakets in einem Parallelisierungslauf das entsprechende Arbeitspaket zweimal auf der Datenbank, einmal vor Ausführung der Methode 1300 (alle Objekte haben den Status initial) und einmal nach Ausführung der Methode 1300. Das bedeutet also, dass Objekte, die in der Methode 1300 erfolgreich bearbeitet werden (Status finished) und genauso wie die nicht erfolgreich bearbeiteten Objekte schon im Arbeitspaket auf der Datenbank gespeichert sind, nach Ausführung der Methode 1300 wieder aus dem Arbeitspaket auf der Datenbank gelöscht werden. Es gibt also häufige Einfüge- und Löschoperationen auf der Datenbank.

Ist das Kennzeichen nicht gesetzt (Voreinstellung), dann speichert das PV-Tool bei der ersten Bearbeitung eines Pakets in einem Parallelisierungslauf das entsprechende Arbeitspaket nur einmal auf der Datenbank, und zwar erst nach Ausführung der Methode 1300. Objekte, die bei der ersten Bearbeitung erfolgreich bearbeitet werden (Status finished), werden also überhaupt nicht im Arbeitsvorrat auf der Datenbank

gespeichert. Vorteil ist die höhere Performance, aber im Abbruchfall stehen die selektierten aber noch nicht bearbeiteten Objekte nicht mehr zur Verfügung.

3.3.2.2.2 Prozentsatz zurückgestellter Objekte

Mit diesem Prozentsatz wird eine Toleranzschwelle definiert, oberhalb derer das PV-Tool eine automatische Wiederholung der Verarbeitung für die zurückgestellten Objekte (Status retry) unmittelbar nach Synchronisation der parallelen Jobs erneut startet. Die Schwelle ist dabei definiert als prozentuale der Objekte mit Status retry an den insgesamt noch offenen Objekten – d.h. Objekten mit Status initial, retry oder restart.

3.3.2.2.3 Anzahl der wiederholten Durchläufe

Mit dieser Anzahl kann bzw. muss die Zahl der automatischen Wiederholungen für zurückgestellte Objekte in einem Parallelisierungslauf begrenzt werden, um nicht in eine potentielle Endlosschleife zu geraten.

3.4 Start von Parallelisierungsläufen

Ein Startlauf wird mit dem Funktionsbaustein BANK_MAP_PP_START gestartet und ein Restartlauf mit dem Funktionsbaustein BANK_MAP_PP_RESTART.

3.4.1 BANK_MAP_PP_START

Der Funktionsbaustein BANK_MAP_PP_START hat u.a. folgende Parameter. Alle anderen Parameter werden im Normalfall nicht verwendet und sind optional.

BANK_MAP_PP_START					
Name	Typ	Bedeutung	Vorschlags-wert	Optio-nal	Import(I) / Export(E)
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart			I
I_XSIMULRUN	BKK_XSIMULRUN	Kennz.: Simulationslauf	SPACE	X	I
I_STR_APPL_PARAM		Globale Anwendungs-parameter		X	I
I_XLOG	XFELD	Kennz.: Anwendungs-protokoll erstellen		X	I
I_X_USE_DIALOG_WP	XFELD	Kennz.: Keine Hintergrund-verarbeitung		X	I
E_STR_RUNKEY	BANK_STR_RUNKEY	Schlüssel des Massenlaufs			E
E_RUNSTATUS	BANK_DTE_PP _RUNSTATUS	Status des Massenlaufs			E
E_RCD_APPL	SY-SUBRC	Rückgabecode der Anwendung			E

Die Parameter haben folgende Bedeutung:

- I_XSIMULRUN: Ist das Kennzeichen gesetzt, so wird am Ende des Startlaufs der Massenlauf zusammen mit dem Arbeitsvorrat wieder aus der Datenbank gelöscht. Es ist kein Wiederaufsetzen möglich. Dieses Kennzeichen hat keinen Einfluß auf die COMMIT-WORK-Befehle, die das PV-Tool absetzt. Soll die Anwendung an sich simuliert werden, also zum Beispiel keine

Datenbankänderungen an Anwendungsdaten vorgenommen werden, so ist ein weiteres Simulationsflag in den globalen Anwendungsparametern notwendig.

- I_XLOG: Ist das Kennzeichen gesetzt, so erzeugt das PV-Tool in jedem parallelen Prozess ein Anwendungsprotokoll. Es gibt jeweils einen Protokolleintrag zu Beginn und Ende des parallelen Prozesses und einen Protokolleintrag für jedes Paket, das bearbeitet wird, mit Paketnummer und Anzahl der Objekte. Das entsprechende Anwendungsprotokollobjekt wird im Customizing eingestellt (siehe 3.3.2).
- I_X_USE_DIALOG_WP: Ist das Kennzeichen gesetzt, werden keine Hintergrundjobs gestartet. Die Verarbeitung läuft stattdessen sequentiell in einem Dialog-Workprozess. Dieser Betriebsmodus sollte möglichst nur zum Debuggen genutzt werden.
- E_STR_RUNKEY: Das ist der Schlüssel, unter dem der neu generierte Massenlauf angesprochen werden kann.
- E_RUNSTATUS: Der Status des Massenlaufs wird vom PV-Tool bestimmt und kann folgende Werte annehmen:
    - Abgebrochen (3): Dieser Status tritt auf, wenn ein technischer Fehler aufgetreten ist oder wenn es noch Objekte mit Status initial gibt (was ein Fehler der Anwendung wäre).
    - Nachbereitung erforderlich (5): Dieser Status tritt auf, wenn es noch offene Objekte im Arbeitsvorrat gibt.
    - Erfolgreich beendet (6)
- E_RCD_APPL: Der Rückgabecode der Anwendung wird von der Anwendung in der Methode 0300 bestimmt.

3.4.2 BANK_MAP_PP_RESTART

Der Funktionsbaustein BANK_MAP_PP_RESTART hat u.a. folgende Parameter. Alle anderen Parameter werden im Normalfall nicht verwendet und sind optional.

BANK_MAP_PP_RESTART					
Name	Typ	Bedeutung	Vorschlags-wert	Optio-nal	Import(I) / Export(E)
I_PROGN	BKK_PROGN	Massenlauf-schlüssel, 1.Feld		X	I
I_PROGDATE	BKK_PRGDAT	Massenlauf-schlüssel, 2.Feld		X	I
I_PROGNO	BKK_PRGNO	Massenlauf-schlüssel, 3.Feld		X	I
I_APPLCATG	BKK_PAAPPLCATG	Anwendungsart		X	I
I_XSIMULRUN	BKK_XSIMULRUN	Kennz.: Simulationslauf	SPACE	X	I
I_STR_APPL_PARAM		Globale Anwendungs-parameter		X	I
I_XLOG	XFELD	Kennz.: Anwen-dungsprotokoll erstellen		X	I
E_RUNSTATUS	BANK_DTE_PP _RUNSTATUS	Status des Massenlaufs			E
E_RCD_APPL	SY-SUBRC	Rückgabecode der Anwendung			E

Die Parameter haben folgende Bedeutung:

- I_PROGN, I_PROGDATE, I_PROGNO: Das ist der Schlüssel des Massenlaufs, der aus drei Komponenten besteht.
- I_XSIMULRUN: Ist das Kennzeichen gesetzt, dann wird der Arbeitsvorrat nicht verändert, d.h. am Ende des Restartlaufs hat der Arbeitsvorrat denselben Zustand wie vor dem Restartlauf. Das Kennzeichen I_XSIMULRUN hat keinen Einfluß auf die COMMIT-WORK-Befehle, die das PV-Tool absetzt. Soll die Anwendung an sich simuliert werden, also zum Beispiel keine Datenbankänderungen an Anwendungsdaten vorgenommen werden, so ist ein weiteres Simulationsflag in den globalen Anwendungsparametern notwendig.

Die Bedeutung der anderen Parameter ist analog zum Funktionsbaustein BANK_MAP_PP_START.

Wird beim Aufruf des Funktionsbausteins nicht der Schlüssel des wiederaufzusetzenden Massenlaufs in den Feldern I_PROGN, I_PROGDATE und I_PROGNO übergeben, darf die Anwendungsart nicht initial sein. Es muß in diesem Fall außerdem die Methode 0150 (Massenlauf für Restart auswählen) implementiert sein (siehe Entwicklerdokumentation).

3.5 Kundencustomizing

3.5.1 Verteilung der Prozesse auf Server oder Servergruppen

Im View V_TBANK_PP_DISTR kann pro Anwendungsart die Anzahl der parallelen Prozesse, die in einem Parallelisierungslauf jeweils erzeugt werden, und deren Verteilung auf Server oder Servergruppen eingestellt werden. Gibt es dort keinen Eintrag für eine Anwendungsart, so wird jeweils nur ein paralleler Prozess erzeugt.

3.5.2 Technische Einstellungen

Über die Transaktion sm30 können in der Tabelle TBANK_PP_APPCUST technische Einstellungen pro Anwendungsart vorgenommen werden, und zwar dieselben wie im Systemcustomizing (Kennzeichen: alle Objekte persistent, Prozentsatz zurückgestellter Objekte, Anzahl der wiederholten Durchläufe). Das Systemcustomizing überschreibt das Kundencustomizing.