BPEL und Transaktionen

Arne Platzen / Guido Neander
MT AG
Ratingen

Schlüsselworte:

SOA, BPEL, Transaktionen

Einleitung

Transaktionen als Folge von Operationen, welche als eine Einheit betrachtet werden, kommen häufig zum Einsatz und sind gerade in der Datenbankwelt immer ein wichtiges Thema. Mit der steigenden Verbreitung von Service-orientierten Architekturen stellt sich nun die Frage, wie Transaktionskonzepte hier integriert werden können, an welchen Stellen sie notwendig sind, welche Alternativen man bei ihrer Realisierung und welche Folgen ihr Einsatz für den SOA-Architekten haben könnte.

Allgemeines zu Transaktionen

Eine Transaktion besteht aus mehreren Operationen, die eine logische Einheit bilden. Bei der Ausführung von Transaktionen müssen die ACID-Eigenschaften garantiert sein: Eine Transaktion muss vollständig ausgeführt werden (Atomar). Sie muss die Daten in einem konsistenten Zustand hinterlassen (Consistency). Des Weiteren dürfen sich gleichzeitige Transaktionen nicht beeinflussen (Isolation) und das Ergebnis muss manifestiert sein (Durability). Müssen Teile einer Transaktion auf verschiedenen Systemen ausgeführt werden, so handelt es sich um eine verteilte Transaktion.

Notwendigkeit von Transaktionskonzepten in Service-orientierten Architekturen

Um auf die Frage der Notwendigkeit von Transaktionen in Service-orientierten Architekturen einzugehen, sei ein kurzes Beispiel dargestellt. Für eine Urlaubsreise sollen nacheinander ein Flug und ein Mietwagen gebucht werden. Die Buchung erfolgt jeweils über einen externen Service. Der Aufruf des ersten Service, welcher den Flug verbindlich bucht wird erfolgreich ausgeführt. Bei der Ausführung des zweiten Service kommt es jedoch zu einer Ausnahme. Für den entsprechenden Zeitraum ist kein Mietwagen mehr verfügbar. Es stellt sich nun die Frage, was mit dem vorher gebuchten Flug passiert. Soll und kann eine Stornierung erfolgen oder soll die Buchung auch ohne einen Mietwagen beibehalten werden?

Ein Transaktionskonzept ist also immer notwendig, wenn mehrere Services eine logische Einheit darstellen. Insbesondere, wenn es in einem der Services zu einer Fehlersituation kommen kann, muss der konsistente Zustand des Systems gewährleistet werden.

Try-Cancel-Confirm-Mechanismus

Eine mögliche Lösung des oben beschriebenen Problems bietet der Try-Cancel-Confirm-Mechanismus. Hierfür muss es möglich sein, eine Reservierung (TRY) auszuführen, welche den Status der entsprechenden Ressource auf PENDING setzt. Das System muss dafür sorgen, dass die

entsprechende Reservierung eine gewisse Zeit erhalten bleibt. Sobald alle anderen Buchungen (Serviceaufrufe) erfolgreich ausgeführt wurden, wird die Reservierung endgültig bestätigt (CONFIRM). Sollte es in einem der Services zu einem Fehler oder einer Ausnahmesituation kommen, können alle anderen Reservierungen noch storniert werden (CANCEL). So wird verhindert, dass das System in einem inkonsistenter Zustand hinterlassen wird. Neben der Reservierung muss das System also auch Methoden zur Stornierung bzw. Bestätigung anbieten. Da für die Zeit der Reservierung die entsprechende Ressource geblockt wird, ist dieser Mechanismus nicht für alle Situationen und Systeme geeignet.

Logisches UNDO in BPEL

Wenn mehrere Services zu einer logischen Einheit gehören und bei ihrer Ausführung ein Fehler auftritt, muss das System Methoden anbieten, um die bisher erzeugten Änderungen rückgängig zu machen. Häufig reicht es nicht aus, einfach die inverse Operation auszuführen oder ein Rollback auszulösen. Wenn z. B. eine Buchung in einem externen System vorgenommen wurde oder eine Bestätigungs-E-Mail versendet wurde, so müssen diese Operationen fachlich rückgängig gemacht werden. Es muss also für diese Fälle Services geben, welche diese fachlichen Stornomethoden zur Verfügung stellen, um z.B. eine Buchung zu stornieren oder eine weitere Info-E-Mail zu verschicken.

Kompensation in BPEL

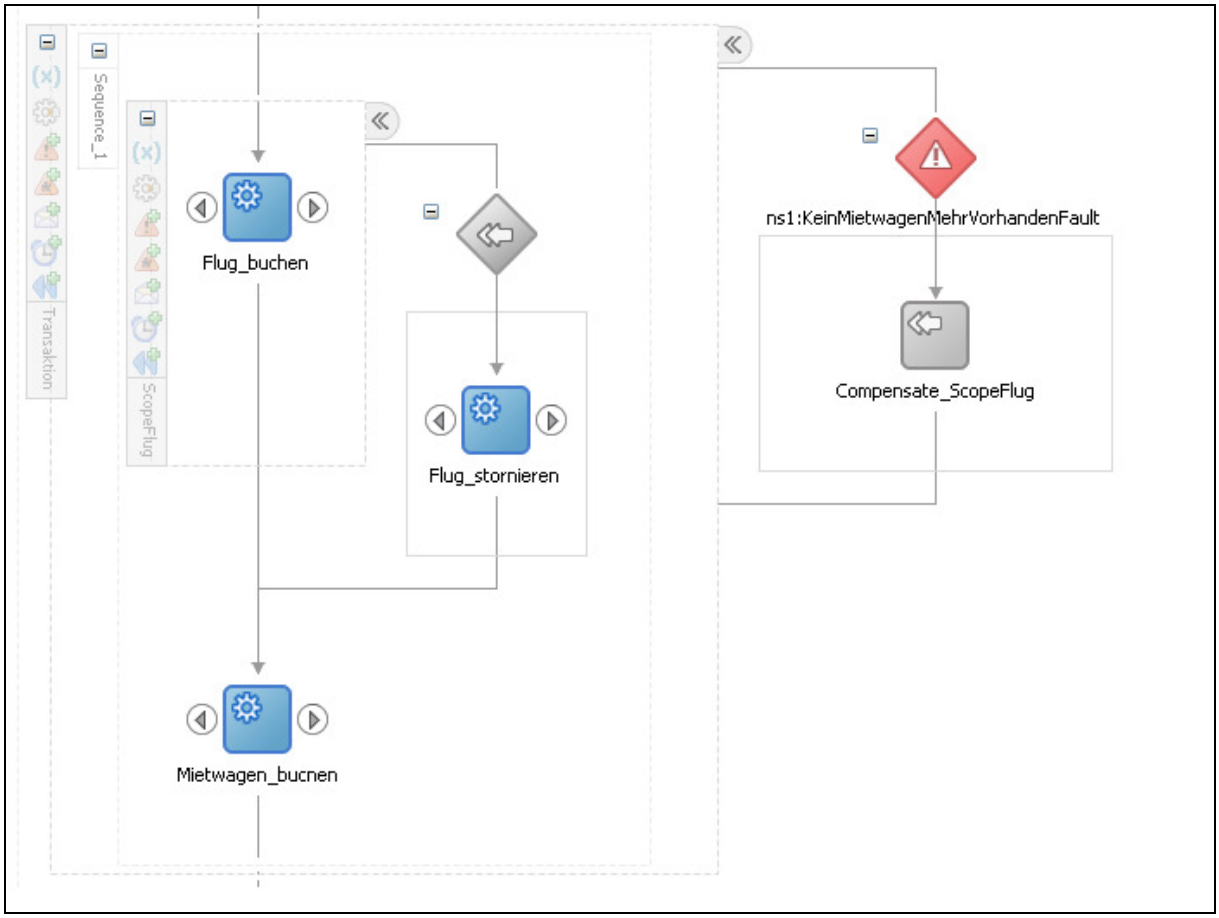

Abb. 1: Das Compensation-Pattern für BPEL

Beim Compensation-Pattern wird eine Stornomethode (fachliches Undo) verwendet, um einen vorhergehenden, erfolgreich abgeschlossenen Teil einer Transaktion wieder rückgängig zu machen. Im Beispiel wird der Service "Flug buchen" erfolgreich ausgeführt. Danach erfolgt der Aufruf des Service "Mietwagen buchen". Bei der Ausführung wird festgestellt, dass für das gewünschte Datum kein Mietwagen mehr vorhanden ist und der entsprechende Fault wird geworfen. Der Fault-Handler des übergeordneten Scopes übernimmt die Verarbeitung. Die Compensate-Aktivität sorgt dafür, dass der Kompensationszweig des Scopes "ScopeFlug" ausgeführt wird. An dieser Stelle erfolgt der Aufruf des Service, welcher die Stornologik enthält: "Flug_stornieren".

Transaktionshandling in BPEL

Anforderungen werden in BPEL in einer JTA (Java Transaction API) – Transaktion ausgeführt. Wenn das Ende des Flows erreicht ist oder wenn eine sogenannte Breakpoint-Aktivität ausgeführt wird, erfolgt die Rückkehr zur übergeordneten Transaktionsinstanz bzw. ein Commit. Bei Breakpoint-Aktivitäten handelt es sich um Receive, onMessage, wait, onAlarm und invoke, falls diese bestimmte Properties enthalten.

Transaktionssteuerung durch BPEL-Properties

Die Transaktionssteuerung in BPEL erfolgt durch die Verwendung von Properties. Bei einem Serviceaufruf kann entweder eine neue Transaktion für den Service erzeugt werden oder es wird die existierende Transaktion des aufrufenden Prozesses verwendet. In Oracle SOA Suite 10g steht dafür die Partner link Property „transaction = participate“ zur Verfügung. In der Version 11g setzt man die Property „bpel.config.transaction=requiresNew/required“ in der BPEL Service Component Section der Datei composite.xml.

Transaktionen im Mediator

Wird die Mediator-Komponente von extern aufgerufen, so dass keine Transaktion zur Verfügung steht, wird eine neue Transaktion erzeugt. Steht jedoch die Transaktion des aufrufenden Prozesses zur Verfügung wird diese vom Mediator auch verwendet.

Sequentielle (synchrone) Routing-Regeln werden alle in einer Transaktion ausgeführt, so dass im Fehlerfall ein Rollback aller Routings durchgeführt wird. Fault-Policies des Mediators werden in diesem Fall nicht beachtet. Bei asynchronen Routing-Regeln wird für jede Regel eine neue Transaktion verwendet. Auch hier erfolgt im Fehlerfall ein Rollback, jedoch können Fault-Policies verwendet werden, um auf Fehler zu reagieren.

Fehlererkennung

Um eine Kompensation auszulösen, muss jeder Service im Fehlerfall einen Fault zurückliefern. Dies ist standardmäßig bei einem synchronen Service der Fall. Bei einem asynchronen Service wird kein Fault zurückgeliefert. Hier muss eine andere Lösung gefunden werden. Der Einsatz eines „Receive Elements“, das in einer Endlosschleife auf ein Ergebnis des Service wartet, ist beim Ausfall des Service keine praktikable Lösung. Ein „Pick Element“ wartet ebenfalls auf ein Ergebnis des Service, jedoch nur bis zu einer zuvor definierten maximalen Wartezeit.

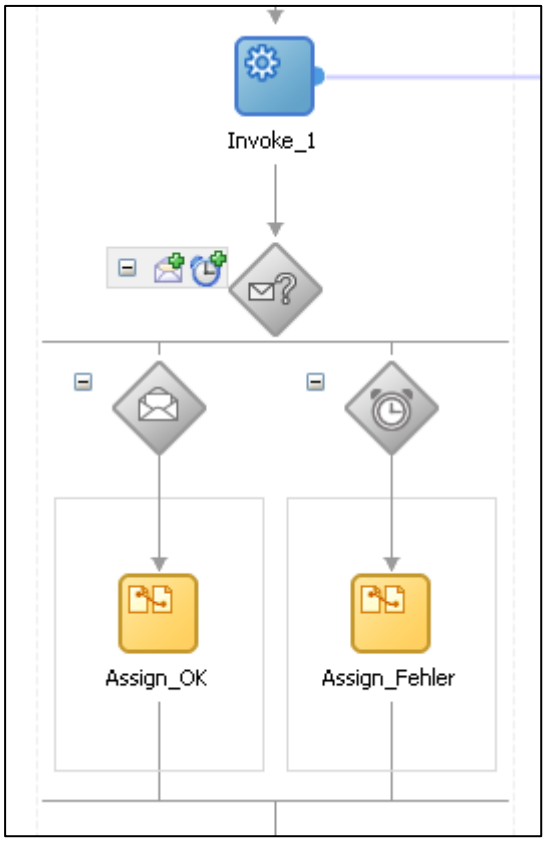

Abb. 2: Das Pick-Element

Das Pick-Element beinhaltet die beiden Fälle „on Message“ und „on Alarm“. Es wartet bis der aufgerufene asynchrone Prozess ein Ergebnis liefert (on Message) oder die maximale Wartezeit* (on Alarm) erreicht wurde und führt dann den entsprechenden Block aus. Hier führt der Ausfall des Service nicht zum „hängen bleiben“ des aufrufenden Service, sondern zu einer definierten Fehlerverarbeitung.

Anforderungen an Transaktionsstandards und -protokolle

Im Gegensatz zu lokalen Transaktionen stellt die Ausführung von verteilten Transaktionen andere Anforderungen an die Transaktionsumgebung, da hier Services in heterogenen Systemen ausgeführt und verwaltet werden müssen.

Eine Transaktion wird durch den Transaktionskontext eindeutig identifiziert. Dieser muss für die Dauer der Transaktion verwaltet und allen beteiligten Systemen kommuniziert werden. Bei verteilten Transaktionen erfolgt dies über Transaktionsstandards und -protokolle. Diese sollen gewährleisten, dass Fehlersituationen nicht zu systemübergreifenden Inkonsistenzen führen.

Bisherige Transaktionsstandards basieren meist auf dem Koordinator-Teilnehmer-Modell. Beispiele hierfür sind WS-C/AT/BA, BTP oder WS-CAF. Häufig lassen sich Transaktionsstandards bezüglich der Länge der von ihnen verwalteten Transaktionen differenzieren. Einige Standards eignen sich eher für kurzlebige Transaktionen, weil Ressourcen während der Verarbeitung geblockt werden. Andere finden bei langlebigen Transaktionen Anwendung.

Anhand der Spezifikation „Web Services Atomic Transaction“ (WS-AT) wird die Umsetzung von Transaktionskonzepten für kurzlebige Transaktionen in verteilten Umgebungen gezeigt.

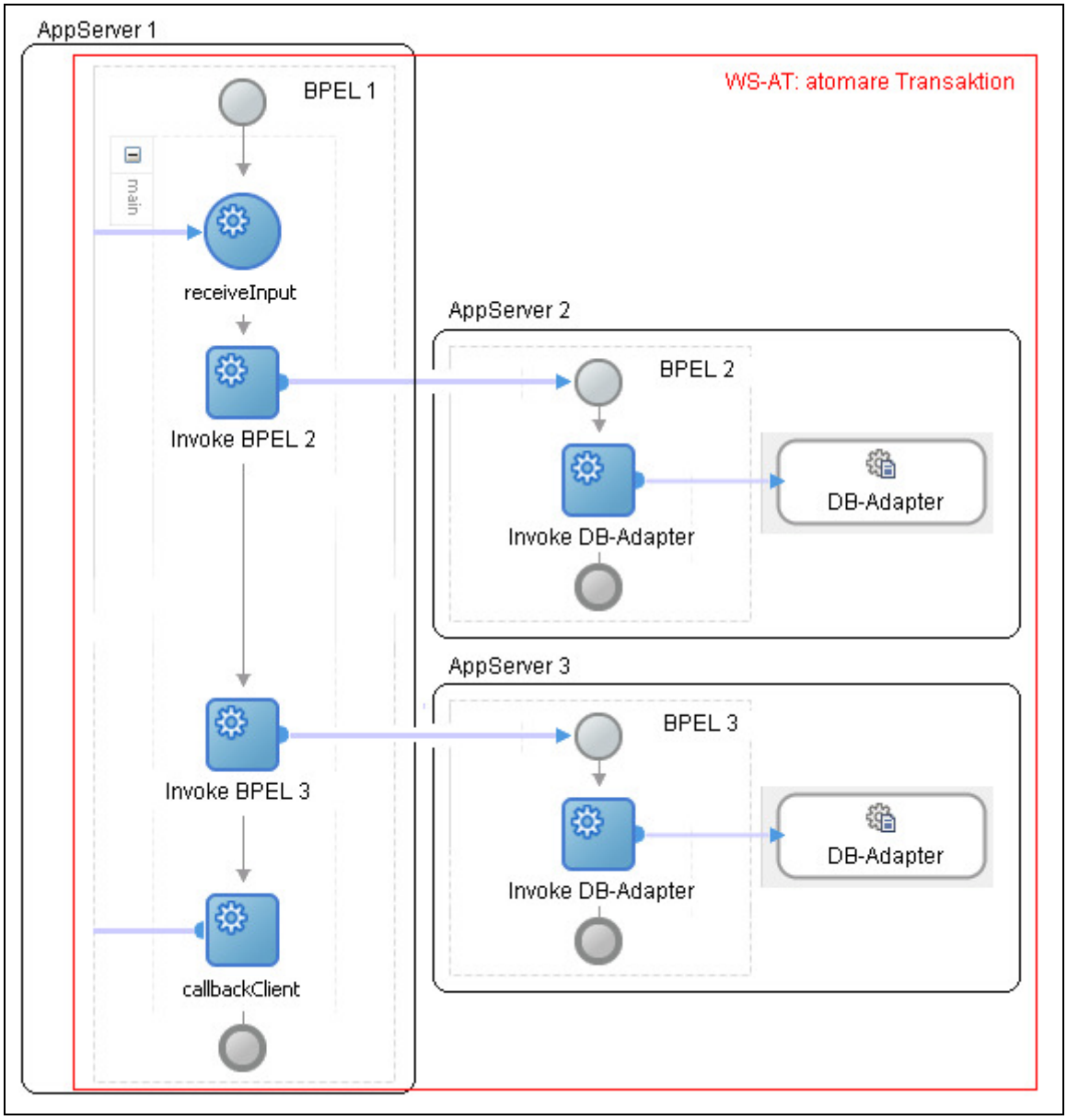

Abb. 3: WS-AT

Der BPEL-Fluss BPEL 1 startet auf dem AppServer1 eine verteilte Transaktion. Beim Aufruf der referenzierten Services BPEL 2 und BPEL 3 auf den Applikationsservern 2 und 3 wird der Transaktionskontext über WS-AT mitgeteilt. Bis zum Abschluss der verteilten Transaktion werden alle beteiligten Ressourcen geblockt. Tritt in BPEL 3 ein Fehler auf, wird die gesamte Transaktion inklusive der Datenbankänderungen in BPEL 2 zurückgerollt. Werden alle Services fehlerfrei durchlaufen, erfolgt für die gesamte Transaktion ein Commit.

Bewertung des Transaktionsstandards: Web Service Atomic Transaktion (WS-AT)

Mit WS-AT können die ACID-Eigenschaften prinzipiell über mehrere verteilte Services auf verschiedenen Applikationsservern erfüllt werden. Da die Ressourcen während der Ausführung geblockt werden, ist WS-AT nur für kurzlaufende Prozesse geeignet. Problematisch ist der Ausfall des Transaktionskoordinators während der Commit-Bearbeitung. Die Implementierungen der verwendeten Middleware-Hersteller müssen miteinander kompatibel sein.

Der Weblogic Server 11gR1 (10.3.3) und die Oracle SOA Suite 11g R1 PS2 (11.1.1.3) unterstützt den Standard WS-AT in der Version 1.2. Um WS-AT auf einem Weblogic Server nutzen zu können, muss die Domain für WS-AT konfiguriert sein. In den BPEL Prozessen ist über Properties ein Transaktionsstart zu setzen und bei den Referenzen ist WS-AT zu aktivieren.

Zusammenfassung Transaktionsstandards

Vorrangig sollten immer lokale Transaktionen verwendet werden, sofern dies möglich ist. Verteilte Transaktionen sollten nur dann angewendet werden, wenn man sie unbedingt benötigt. Sie sollten so einfach wie möglich gestaltet werden. Eine über mehrere, heterogene Systeme und Produkte verteilte Transaktion ist häufig komplex und fehleranfällig.

Fazit

Es ist frühzeitig in einem Projekt zu entscheiden, ob eine Transaktionsverarbeitung notwendig ist, um die Transaktionsumsetzung bereits in einem frühen Stadium mit in die Planung einbeziehen zu können. Eine einfache Transaktionsrealisierung in BPEL mit Kompensation und fachlichem Storno lässt sich häufig sehr leicht integrieren. Kompliziertere Szenarien, die Reservierungsmechanismen oder Transaktionsstandards beinhalten, haben meist einen höheren Realisierungsaufwand zur Folge.

Kontaktadresse:

Arne Platzen / Guido Neander
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon: +49 (0) 2102-309 61-0
Fax: +49 (0) 2102-309 61-10
E-Mail arne.platzen@mt-ag.com / guido.neander@mt-ag.com
Internet: www.mt-ag.com