オブジェクト指向データベースにおけるイベント管理について

2R−4

石川博　久保田和己
株式会社富士通研究所

1．はじめに

我々は，エンジニアリング業務支援等のデータベースの高度応用のためにオブジェクト指向データベースシステムJasmine〔ISHI91〕を研究開発してきた。こうした高度応用では，完全性制約の維持，値の変更伝播のためのイベント／トリガ機能の他に値充足のための制約ルール等，従来よりアクティブな機能を要求する。これらの機能を持つデータベースを総じてアクティブデータベースと呼ぶ。制約管理という側面からのアクティブデータベースの機能についてはすでに報告した〔ISHI92〕。しかし，そこではイベントの組み合わせにより応用を記述する機能については説明しなかった。例えば，設計業務は複数のサブタスクに分解され，それらの柔軟な組み合わせで業務全体が遂行されるが，そのような応用では複合イベントの管理が重要である。ここでは単純なイベントを組み合わせて複合イベントを定義する枠組みを中心に，イベント管理という側面からアクティブデータベースJasmine/Aについて説明する。

2．イベント管理機能

2.1単純イベント

ここではJasmine/Aの提供する完全性制約とトリガ機構を実現する単純イベントについて説明する。それらはすべてデモン〔ISHI91〕としてユーザに提供される。ここでは制約をイベント／コンディション／アクションとしてモデル化する。システム定義のデモンとしてmandatory，multipleデモンがある。mandatoryはインスタンス生成時（即ちinstantiateイベント発生時）に，もしそれが指定された属性に値が挿入されると（即ちinsertイベントが発生すれば）正常終了する。さもなければエラーを引き起こす。またその属性の値が削除されると（即ちdeleteイベントが発生すると），エラーが起きる。multipleが指定された属性（即ち多値属性）に対してのみinsertイベントが発生すれば，値が追加される。一方constraintデモンはinsert／replaceイベントが発生した時にユーザが指定したコンディションが成り立てばコミットし，さもなければアボートする。

さらに属性には4つのデモンが指定でき，if-neededデモンは属性値が参照されると（即ちreferイベントが発生したら），起動される。if-added，if-deleted，if-replacedデモンは，それぞれinsert，delete，replaceイベント発生時に起動される。各デモンはコンディションとアクションをモジュール化する。

2.2　複合イベント

ここでは単純なイベントの組み合わせからなる複合イベントについて述べる。一般にイベントは単純イベントと複合イベントに分類される。単純イベントは，属性の参照や更新，それ以外のシステム定義メソッド（これにはトランザクションの開始，終了などが含まれる）及びユーザ定義メソッドが対応する。複合イベントはひとつ以上の単純イベントの組み合わせである。単純イベントを組み合わせる演算子としては論理演算子（and／or／not）及び順序付け演算子（;）を提供する。例えばE1，E2，E3を単純イベント，F1，F2を複合イベントとすると以下のような複合イベントの例が考えられる。

（例1）F1=E1 and E2.　E1とE2が共に起きたことを示す。

（例2）F1=(E1 or E2) and E3.　E1とE2のどちらかが起きていてかつE3が起きていることを示す。

（例3）F1=E1 and not E3.　E1は起きているがE3は起きていないことを示す。

（例4）F1=E1; E2.　E1のあとにE2が起きたことを示す。

さらにこれらのイベントの組み合わせには，時間指定（at／before／after）をすることができる。例えばT1，T2を時間とすると以下のような例が考えられる。

（例5）F2=E1 at T1.　E1がT1に起きていることを示す。

（例6）F2=E2 and E3 after T1 before T2.　E2とE2がT1以後T2以前に起きることを示す。

次に複合イベントの評価のタイミングについて説明する。イベント／コンディション／アクションのパラダイムにおいて，イベント／コンディションの結合のモードと，コンディション／アクションの結合モードをユーザは指定できる。それぞれ即時（immediate），遅延（deferred），分割（decoupled）のどれかを選択して指定する。2つの結合モードの組み合わせとしては以下のものを許す：（immediate，immediate）（immediate，deferred）（immediate，decoupled）（deferred，deferred）（deferred，decoupled）（decoupled，decoupled）。

immediateはイベントを起こしたトランザクション内で即時に評価されることを示す。deferredはイベントを起こしたトランザクション内で即時ではなくトランザクションのコミット直前に評価されることを示す。decoupledはイベントを起こしたトランザクションとは別トランザクションで評価されることを示す。

複合イベントに関するトランザクションのリカバリについて注意点を述べる。複合したイベントの場合それを構成する単純イベントに対応するトランザクションを，それらの起きた順序と逆順にアボートすればすむわけではない。そのかわりにユーザが，それらのイベントの効果を全体として補償するアクションを定義できる必要がある。そこで複合イベントには補償アクションを属性として記述できるようにする。

以下に単純イベントと複合イベントのオブジェクトを示す。

```
PRIMITIVE-EVENT（単純イベント）
  Name　EventName
  TimeOccurred TIME
  CompositeEvent COMPOSITE-EVENT
```

Name属性は単純イベント（primitive event）の名前を表す。TimeOccurred属性はこのイベントが発生した時刻を時系列で格納する。CompositeEvent属性はこのイベントが構成要素になっている複合イベントの列を表わす。例えばF1 F2となる。

On Event Management in OODB
Hiroshi ISHIKAWA, Kazumi KUBOTA
Fujitsu Laboratories, Ltd.

```
COMPOSITE-EVENT（複合イベント）
  Name  EventName
  TimeOccurred TIME
  PrimitiveEvent PRIMITIVE-EVENT
  EventExp EventExpression
  ConditionExp ConditionExpression
  ConditionValue BOOLEAN
  ActionExp ActionExpression
  CompensatingActionExp ActionExpression
  EC-mode CouplingMode
  CA-mode CouplingMode
```

ここにName属性は複合イベントの名前を表わす。TimeOccurred属性はこのイベントが発生した時刻を時系列で格納する。PrimitiveEvent属性は複合イベントの構成要素である単純イベント（primitive event）の列を表す。例えばE1 E2 E3となる。EventExp属性は前述したイベントの組み合わせ式（event expression）を表す。例えばE1 or E2 and notE3 after T1 before T2となる。ConditionExp属性はこの複合イベントのコンディションを指定する。ConditionValueはコンディションの評価値（trueかfalse）を格納する。ActionExp属性は複合イベントのアクションを指定する。CompensatingActionExp属性はこの複合イベントの効果を補償するアクションを指定する。EC-mode属性には複合イベントが生起したら即コンディションを評価するか，トランザクションの終了前に評価するか，別トランザクションで評価するかを指定する。それぞれ即時モード（immediate），遅延モード（deferred），分割（decoupled）を指定する。CA-mode属性には複合イベントが起きてコンディションが成立したら即アクションを実行するか，トランザクションの終了前に実行するか，別トランザクションで実行するかを指定する。それぞれ即時モード（immediate），遅延モード（deferred），分割（decoupled）を指定する。

具体的イベントは予めインスタンス化され，複合イベントを構成する単純イベントは以下のようにデモンの中で生起される。

```
PropertyName if-added
      { if condition
        then PRIMITIVE-EVENT.put("TimeOccurred", time)
             where PRIMITIVE-EVENT.Name = EventName}
```

あるいは一般にメソッドやプログラムの中で生起される。

```
...
PRIMITIVE-EVENT.put("TimeOccurred", time)
where PRIMITIVE-EVENT.Name = EventName
...
```

2.3　制約ルール

ここでは制約ルールの機能を説明する。前述したデモンが値の変更の伝播を主たる目的としていたのに対して，この制約ルールは値の生成を主たる目的とする。制約ルールは第一級のオブジェクトである。制約ルールにはこのルールが決定するパラメータ（属性）名やこのルールが依存するパラメータを指定する。さらに制約条件を満足する候補値を生成する手段を指定する。それには値の生成，計算，データベース検索，ユーザ入力などの手段を持つ。複数の条件節が指定できる。条件節は制約条件とそれが満たされなかった際に起動されるアクションが複数指定できる。アクションには自分自身もしくは他の制約ルールの起動が指定できる。先頭からはじめてアクションが成功すれば，それ以後のアクションは実行されない。アクションはルールのバックトラックの指定に対応する。条件節が複数指定されている場合はすべてのコンディションが満たされたときにこのルールは成功する。さもなければアボートされる。ルールの起動の順序はルールの依存関係に基づき，ルールのスケジューラが決定する。イベントはルール起動そのもの（generate）であり，条件節にコンディションとアクションが対応する。

3．実現

ここでは複合イベントの論理的な実現方式について説明する。即ち複合イベントと単純イベントに対してそれぞれクラスを定義し，そのメソッドとデモンの組み合わせとして，ブートストラップ的にイベント管理機能を実現する。

```
PRIMITIVE-EVENT（単純イベント）
 property     TIME TimeOccurred  multiple if-added
イベント／コンディション結合：即時モードチェック
{ self.CompositeEvent.check-immediate()
  where self.ComositeEvent.EC-mode = immediate}

COMPOSITE-EVENT（複合イベント）
 method      check-immediate()  即時モードチェック
 { if self.event-eval() & self.condition-eval() then
    if self.CA-mode == immediate  then self.action-eval()
    else self.ConditionValue = true  }
          check-deferred()   遅延モードチェック
 { if self.event-eval() & (self.ConditionValue == true ¦
     self.condition-eval() ) then
    if self.CA-mode = deferred then self.action-eval()
    else self.ConditionValue = true }
          event-eval()     {イベント式の評価 }
          condition-eval() { コンディションの評価}
          action-eval()    {アクションの評価 }
```

システムはトランザクション終了前に遅延モードチェックとして以下の操作を実行する。

```
COMPOSITE-EVENT.check-deferred()
where COMPOSITE-EVENT.EC-mode = deferred ¦
      COMPOSITE-EVENT.CA-mode = deferred
```

4．おわりに

アクティブデータベースについて，オブジェクト指向データベースのコンテクストで複合イベントを中心に，機能と実現を述べてきた。今後は複合イベントを複数の既存データベースから構成できるようにするために，オブジェクト指向データベースを中心とした異種データベースの研究も重要になるであろう。

参考文献

〔ISHI91〕 Ishikawa, H. et al. : The Model, Language, and Implementation of An Object-Oriented Multimedia Knowledge Base Manegement System, accepted for publication in ACM TODS(1991).

〔ISHI92〕石川他：オブジェクト指向データベースにおける制約管理について，情報処理ＤＢシステム研89-10(1992).