定時トランザクションパターン

Ver.1.0 1999年 7月10日
Ver.2.0 2000年12月29日

矢崎博英

共通のコンテクスト

時の到来によって発生するトランザクションがある。

例　給与
締め日による顧客への請求
決まった日時、地点での観測（正月三が日の神社仏閣の人出など）
公示地価
アメダスの雨量観測（多分）
実地棚卸し　などなど

上記のようなトランザクションや観測を「定時トランザクション」と呼ぼう。定時トランザクションは、毎日3時、毎月２５日、毎月１０日と２０日、隔週の月曜日、偶数月の第3土曜日、４半期末、年間の指定された日など、それぞれに固有のタイミングで生成される。それに対して、ランダムに発生するトランザクションというものがある。これは、その発生が人間の認知と、人間の手による入力に直接依存するものである。ランダムに発生するトランザクションは、そのトランザクションをシステムにエントリするためのインタフェース（GUI など）を用意し、ユーザがそれを使ってデータを入力する。定時トランザクションは、通常は自動的に、多くはバッチ処理などでされる類のものである。ある種の定時トランザクションは、生成される時点でデータベースなどに記録されている情報を使って、人間による入力で捕捉されることなくすべての属性値が自動的に計算される。例えば、給与は社員マスタの基本給や、計算期間の労働時間などから計算されるものである。また、締め日による顧客への請求は､同じく計算期間の注文の合計金額に税や割り引き等を考慮して計算される。また、人間の入力である属性値を捕捉しなければならない別の種類の定時トランザクションもある。実地棚卸しは、人間の手で在庫品を数え上げて、その結果を入力しなければならない｡しかしこの場合でもトランザクションのインスタンスの生成については自動化され得る。生成直後はある属性値が確定していないので、いわば仕掛中のトランザクションであるが、それでもインスタンスが自動的に生成されれば入力の手間を省くことができ、あるいは入力漏れ、重複入力といった誤入力を防ぐことができる。

このように定時トランザクションにおいては、インスタンスの生成をある程度自動化することができる。以下はこの自動化という点に着目し、定時トランザクションに関してのパターンを記述したものである。

補足：定時トランザクションを考えていくと、次のような困難にも出会う。まず、発生時点の精度という問題。これは定時トランザクションが、何月何日のある時刻に必ず発生しなければならないものなのか、何月何日のその日であればいつでもいいのか、またある日を基準にだいたい前後何日程度の範囲内に発生すればいいといったレベルのものなのかというものである。次に、ある条件下でトランザクションの発生が中止される、あるいは他のトランザクションに代替されるという問題。例えばその月に取引きのなかった得意先に対しては請求書の発行が行われない。最後に、ある条件下でトランザクションの発生時点が移動するというものである。トランザクション発生予定日が休日にあたる場合に、発生時点をどのように移動するのかとか、ある手続きが遅れたために、ある得意先に対する請求書発行が遅れるなどということがある。残念ながら今回は、こうした点まで踏み込んで検討することができなかった。これからの課題としたい。

ルールにより発火するトランザクション

定時トランザクションの問題に取り組む前に、ルールにより発火するトランザクションという問題を少し検討してみたい。定時トランザクションは事前に決められている時の到来により発火するトランザクションである。これは一般的にはルールにより発火するトランザクションの一種である。定時トランザクションの場合は、ある時が到来すると、このトランザクションが生成されなければならない、というルールにしたがってトランザクションが生成されるわけである。しかし、ルールは時の到来以外にも考えられる。例えば、あるトランザクションがエントリされたことが、別のトランザクションを自動生成するトリガとなるケースがある。簿記などでの対照勘定の仕訳などがこれに該当する。M.Fowler はこれを転記ルールとしてパターンとして発表した。また、リソースの状態変化がトランザクションを自動生成する例として、在庫量がある数以上を下回った時に、自動発注するしくみを考えることができる。この場合発注トランザクションが自動生成されるのである。

このように、一般的にルールにより発火するトランザクションという問題を考えることができる。ここでルールとは、システム内にある状態変化が起きたときに、それに伴いあるトランザクションが生成されるという、事前に決められている決め事である。ここには、システム内の状態変化というトリガと、結果的に生成されるトランザクション、そしてトリガと結果トランザクションの因果関係を示すルールがある。

問題

システム内のある状態変化がトリガとなって、あるトランザクションのインスタンスが生成されることをルール化したい。

フォース

- トリガとなる状態変化には、いろいろな種類のものがある。
- トランザクションにも、いろいろな種類のものがある。
- トリガとなる状態変化と、その結果としてのトランザクションの結びつきは、あるビジネス・ルールのもとで関連付けられることが多い。ビジネス・ルールが変われば、その関連付けも変わる場合がある。
- トリガは、トランザクションのもつ属性や、その作られ方とは独立して変更し得る。逆にトランザクションのもつ属性や作られ方も、トリガとは独立に変更し得る。

解決

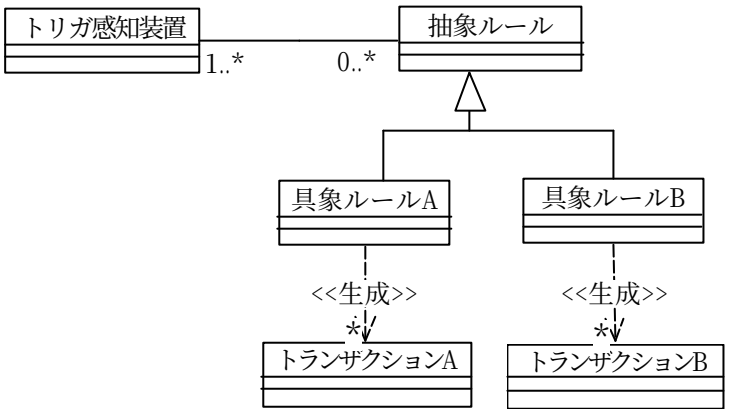
トリガを感知するオブジェクト（トリガ感知装置と名づける）と、トランザクションを作成するルール（略してルールと呼ぶ）を別のオブジェクトとして分離する。トリガ感知装置

オブジェクトはトリガを感知することと、そのトリガによって発火されなければならないトランザクションの作成に責務を持つルール・オブジェクトが何かを知っている。トリガとなる事象が発生すると、クライアントはトリガ感知装置オブジェクトに、対象となるルールオブジェクトを問い合わせる。そして、そのトリガに反応してトランザクションを生成しなければならないルールに対して、トランザクションの生成を依頼する。

トリガ感知装置とルールは責務を明確に切り分けなければならない。つまり、トリガ感知装置は、トランザクションの生成のロジックについては関与しないし、ルールは、トリガがどういうものかについて関与しないのである。

トリガ感知装置オブジェクトは、１つないしは複数のルール・オブジェクトと事前に関連づけられる。また、この関連は変更可能である。また、トランザクションにはいろいろな種類を考えなければならないので、抽象的なルール型と具象的なルール型を用意する必要がある。

構造



関連するパターン

Fowlerの以下のパターンは、ルールにより発火するトランザクションの一種とみなせる。

- 連想関数
- 転記ルール

時間に関するルールへの対処方法

問題

ルールにより発火するトランザクションの一種として、事前に決められた時の到来で発火するトランザクションを取り扱いたい。

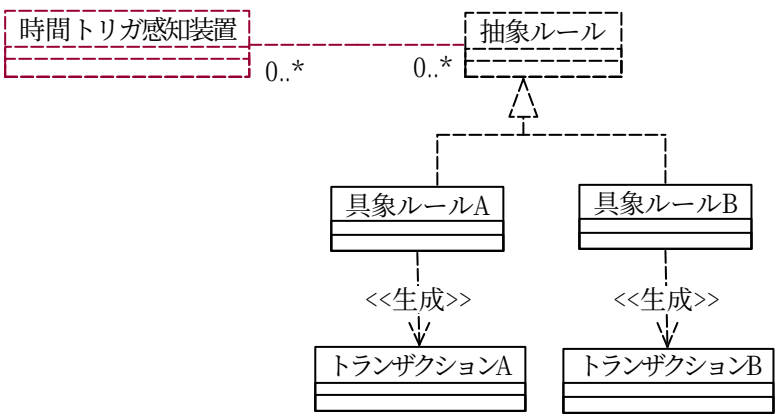
フォース

- トリガとなる時間の刻みの長さは多用である。数年に１度のものもあれば、秒、あるいはさらに細かい単位でトランザクションが発火される場合もある。
- トリガの認識方法として、トリガ感知装置が能動的に、時の到来を監視していなければならない場合と、時をパラメタとして受け取るなどして、その受け取った時が既に訪れたものかどうかを bollean で返せばいいような受動的に感知できればいい場合がある。トランザクションの発生タイミングに誤差が許されないようなアプリケーションの場合、能動的なトリガ監視装置が必要であり、その精度も高いものが要求される。逆に発生誤差の許容範囲が広いものは、高精度のトリガ監視装置はかえってオーバ機能である。

解決

時の到来を感知するオブジェクトをトリガ認識装置とする（時間トリガ感知装置）。フォースで述べたように、アプリケーションの特徴により、トリガ認識装置の認識の仕方やその精度は多様である。また、同じトランザクションでもビジネスルールの変更や競争の激化などで、認識の方法、精度を変えなければならない場合がある。よって、どのようなトリガ認識装置を使っても、あるいは変更があっても、トランザクションの生成ロジックに影響を与えないようにしなければならない。

構造



単一の定時トランザクション

コンテクスト

通常、住宅や事務諸などの家賃の支払は月に一度行われる。また、水道光熱費の支払や電話代の支払などもそうである。月給の受け取りなどもそうである。

こうした収入、支出は、トランザクションであり、仕訳などの方法でデータとして記録されるものである。家賃などの支払トランザクションは、支払額と支払日の両方が事前に決まっている場合が多く、その場合には、家賃支払日の到来により、トランザクションのインスタンスが生成される。人間の入力は必要ない。水道高熱費の類はメータなどの測定により自動化できる（しかし、請求書ベースなので、確定値の再入力が筆ようかもしれない）。月給などは、時間外給の不可などで、月により変動するケースがあり、その場合はインスタンスの生成は自動化できるが、給与額については入力しなければならない。

問題

ある事象がいつ起こるかを予定として管理し、その時が到来したときに、予定されていたトランザクションを自動生成する。

フォース

- それぞれのトランザクションの種類において、ある時間のタイミングでは１つのインスタンスしか発生しない。明細を持っていてもよいが、明細行の変動がない。

解決

「時間に関するルールへの対処方法」で示した解決策を具体化する。

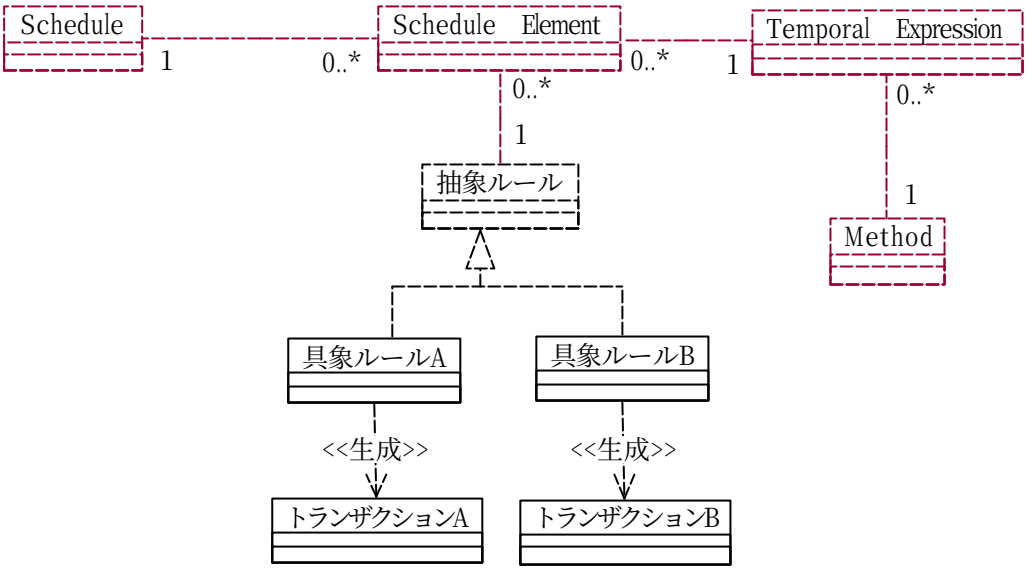
例

Martin Fowler の「Reccurring Events for Calendars」を時間トリガ感知オブジェクトとして利用する例を考えてみよう。

なお、「Reccurring Events for Calendars」の詳細については、Martin Fowler の WEB サイトを当たってほしい。

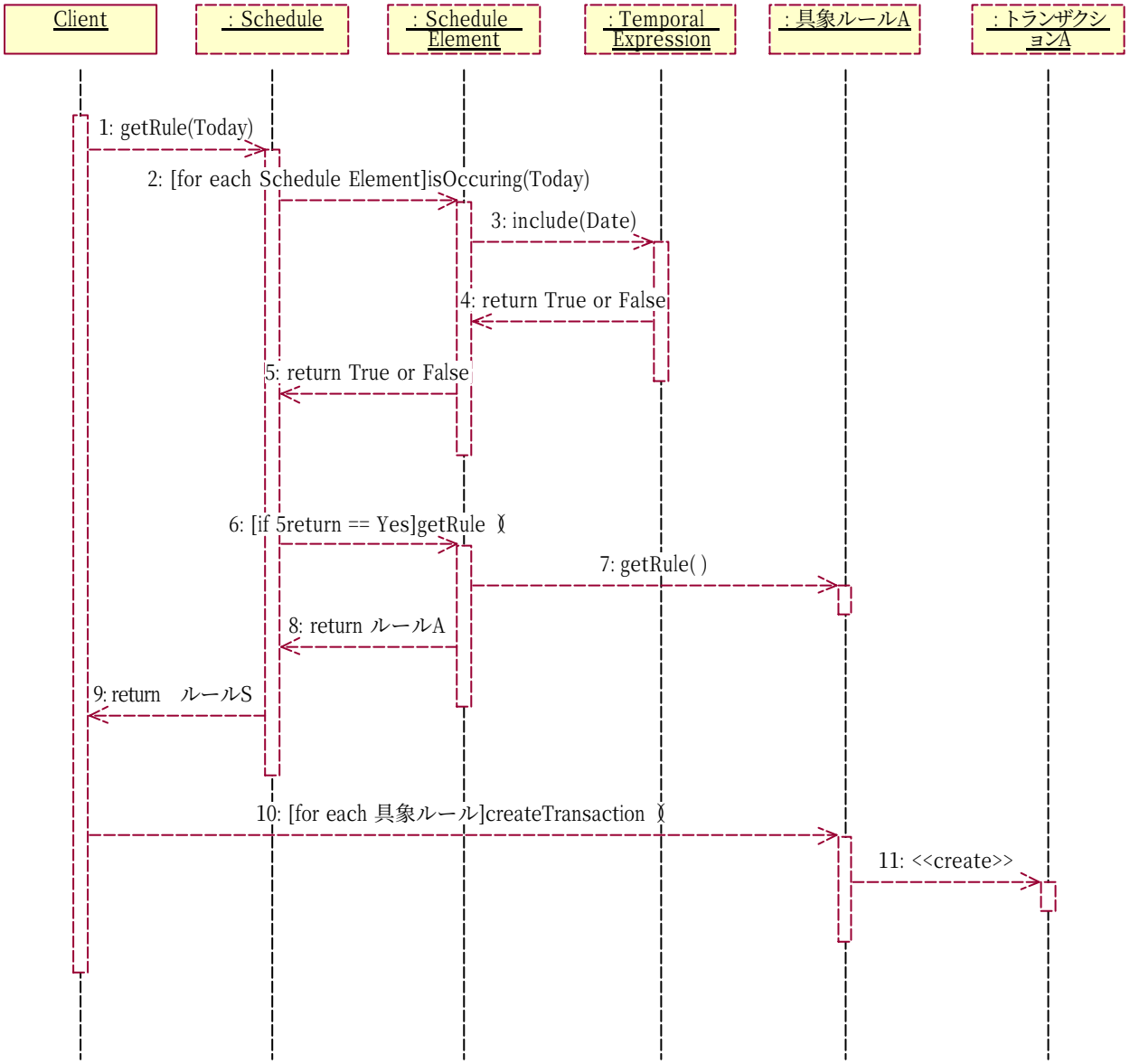
（http://www2.awl.com/cseng/titles/0-201-89542-0/apsupp/events2-1.html）。

Reccurring Events for Calendars を時間トリガ感知オブジェクトとみなし、単一の定時トランザクションパターンの構造を表すと以下のようになる。



上図はオリジナルの「Reccurring Events for Calendars」では、イベントとなっているところを、「抽象ルール」に置き換えたモデルになっている。「Reccurring Events for Calendars」で想定されているユースケースは、スケジュール管理に焦点をあてていて、イベントはその存在だけを示せればよかった。「定時トランザクション」では、「Reccurring Events for Calendars」の持つ予定を管理する機能に加えて、それからトランザクションを生成する機能が必要になる。よって、受身だったイベントに変えて、その位置に能動的なルールを必要とする。

それでは、このモデルで実際に定時トランザクションが生成される流れをシーケンス図で表現してみよう。そのためには、やはりオリジナルには想定されていなかった getRules(Date)というメソッドを Schedule クラスのインスタンス・メソッドとして用意する。このメソッドは、その日付に予定されているルールのインスタンスのコレクションを返すものである。



１）朝１０時になると、業務担当者が「定時トランザクション生成クライアント」を起動する。「定時トランザクション生成クライアント」は「Schedule」を生成する。「Schedule」オブジェクトは、その企業の全ての定時トランザクションのスケジュール（「Shedule Element」）を管理している。

２）クライアントは、マシン日付から今日の日付を取得し、それを引数として、「Schedule」オブジェクトに、その日に発生しなければならないトランザクションのルールオブジェクトのコレクションを返すよう依頼する（Rule[　] getRule

(TODAY))。

３）「Schedule」**は、全ての**「Shedule Element」**インスタンスに対して、それらが関連している**「Temporal Expression」**に、「今日の時点」が該当するか調べさせる**（boolean IsOccuring(Date))。

４）「Schedule Element」**の各インスタンスは、**「Temporal Expression」**オブジェクトに、**「Schedule」**から引数で渡された日付け（すなわち今日）が該当するかチェックを依頼する**（boolenan Includes(Date))。**その結果を**「Schedule」**に返す**（return（TemporalExpression.Includes(Date))）。

５）「Schedule」**は、**SheculeElement.IsOccuring(Date)**のリターン値が** TRUE **の場合、**「SheduleElement」**に対して、**「ルール」**のインスタンスを渡してもらえうよう依頼する**（GetRule()）。

６）取得した「ルール」**のインスタンスに対して、トランザクションの生成を依頼する**（CreateTransaction（　））。

リソース単位の定時トランザクション

コンテクスト

定時トランザクションの中には、同一のタイミングで複数のインスタンスが同時に生成されるようなものがある。例えば給与。これは月のある決められた日に、社員数分だけ発生する定時トランザクションである。あるいは、顧客への請求は、締め日などといった特定の日に顧客毎に発生する。プロジェクトの進捗確認はプロジェクトの数だけ発生する。

こうしたものの多くは、あるリソースに関連したトランザクションとして発生する。つまり発生日に、それぞれのリソース・インスタンス１つ１つに対して１つずつのトランザクションとして発生する類のものである。

問題

リソースのインスタンス毎に別々の定時トランザクションを生成したい。あるいはリソースのインスタンス毎に、定時トランザクションの明細を生成したい。

フォース

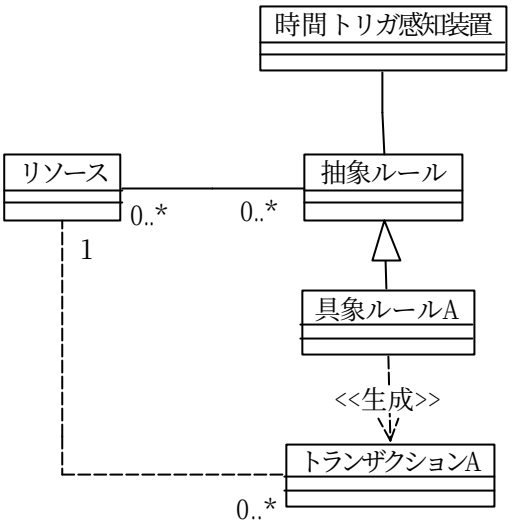
- 生成の対象となるリソース・インスタンスは常に同じものではない。先月まで在職していた社員が退社した場合、その社員の今月の給与トランザクションは発生しない。また、今月に入社した社員については、先月まではその社員の給与トランザクションは発生していないが、今月からは発生することになる。

解決

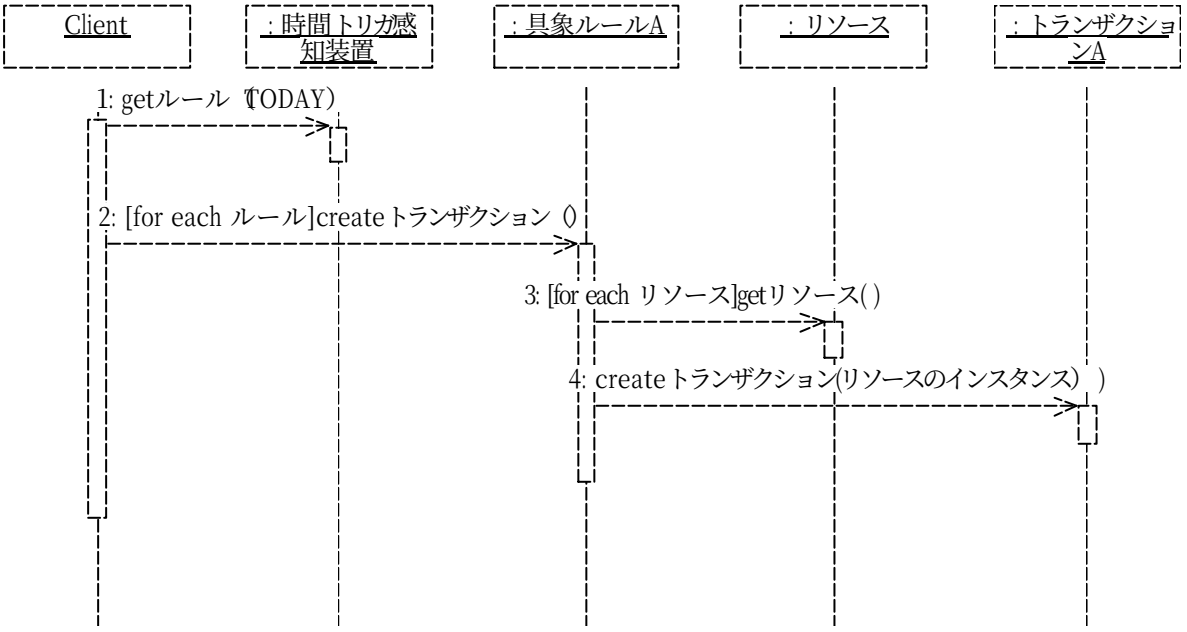
抽象ルールに、そのルールで生成されるトランザクションに関与するリソースの集合を関連づける。抽象ルールは、トランザクションの生成を依頼されたら、自分に関連づけられているリソース１件１件に対して、トランザクションを生成する。

抽象ルールには、関連するリソースをハンドリングするメソッド（追加、削除、参照）を設ける。

構造



インタラクション



例

ルールのインスタンス「給与ルール」には、一人一人の社員インスタンスが対応づけられている。ルールのインスタンス「顧客への請求」には「顧客 A」、「B」、「C」…が対応づけられている。

複数のリソースの組み合わせによる定時トランザクション

コンテクスト

「リソース単位の定時トランザクション・パターン」では、トランザクションの生成に関与するリソースは１トランザクションにつき１つであった。定時トランザクションの中には、複数のリソースの組み合わせがトランザクションの生成に関与する場合がある。例えば、社員の定期健康診断。年に２回、９月と３月に社員の健康診断が行われるとする。そのトランザクションは社員と診断項目の組み合わせで発生する。あるいは、４半期予算。単純な場合でいえば、部署と勘定科目の組み合わせで予算が立案される。このケースは「リソース単位の定時トランザクション・パターン」では対応できない。

問題

複数のリソースのインスタンスの組み合わせ毎に別々の定時トランザクションを生成したい。あるいは複数のリソースのインスタンスの組み合わせ毎に、定時トランザクションと明細の組を生成したい。

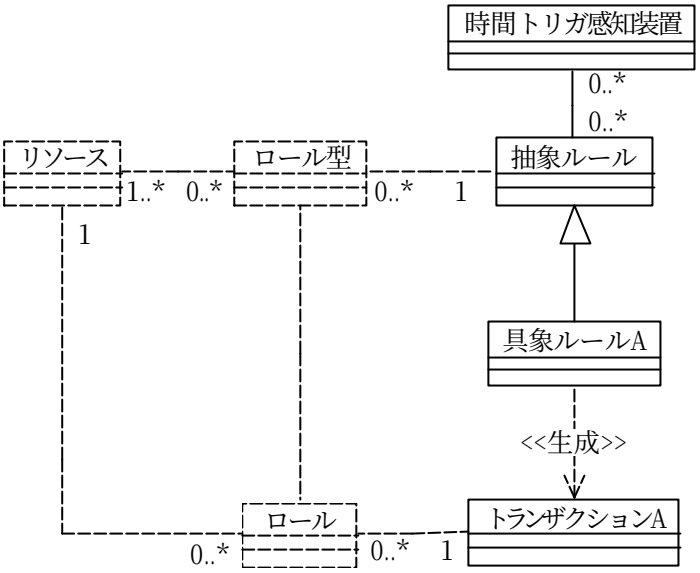
フォース

- 複数のリソースが関与する場合、それぞれの関与の仕方を表すロール型を考えなければならない。
- ロール型毎に参加するリソースの集合は異なる。
- トランザクションの生成は、各ロール型に参加者として待機しているリソースの直積として生成される。
- 複数のリソースが関与する場合、トランザクションのデータの構造として主となるリソースと副となるリソースを考慮してトランザクションを生成しなければならないケースがある。主となるリソース単位にトランザクションが作られ、副となるリソースはその明細となるようなケースである。

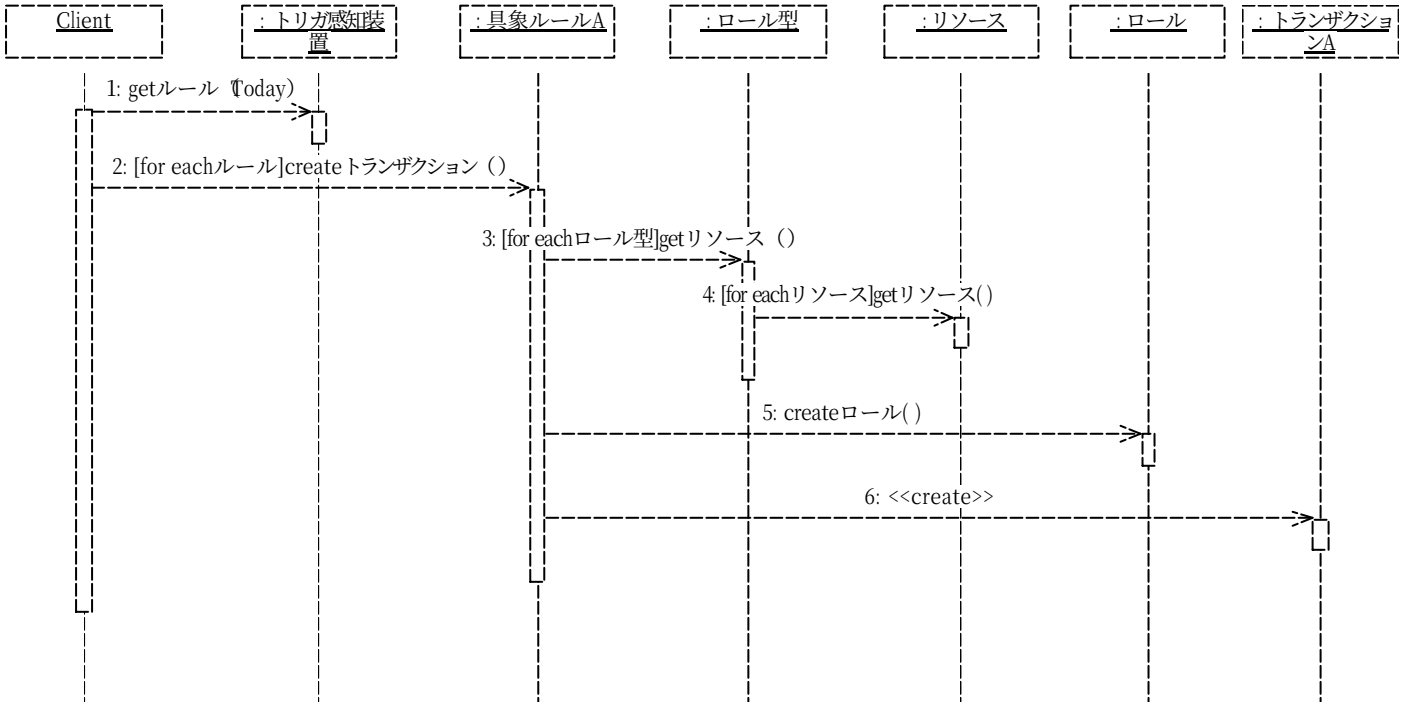
解決

トランザクションにリソースが関与する立場を表すロール型オブジェクトを設ける。各ルールはそれぞれのトランザクションの特性において、複数のロール型オブジェクトと関連を持つ。また、ロール型オブジェクトは、そのトランザクションに対して、そのロール型への参加者としてのリソースの集合と関連を持つ。

構造



インタラクション



例

健康診断の場合は、「ロール型」のインスタンス「被診断者」と「診断項目」の２つのインスタンスが、定時トランザクション型のインスタンス「定期検診」に対応づけられる。

「被診断者」インスタンスにはリソース「社員 A」、「社員 B」、「社員 C」…インスタンス

が対応づけられる。「診断項目」インスタンスには、リソース「身長」、「体重」、「血糖値」…インスタンスが対応づけられる。

　トランザクションの構造は具象サブタイプのロジックで自由に決めることができる。定期検診の場合、例えば、各社員毎に１つずつのトランザクションが生成されて、その明細として、診断項目別の明細行を持つような構造を想定することができるが、この構造を決めるのは具象ルールオブジェクトである。、また、定期検診の場合、自動生成が可能なのは、各診断結果の値は空のトランザクションの生成までである。実際の値は診断結果を受けて人間の手で入力されなければならない（あるいは検診機器からのデータ転送ができるかもしれない)。

リソースの固定化した組み合わせによる定時トランザクション

コンテクスト

「複数のリソースの組み合わせによる定時トランザクション」では、各ロール型に対応している「リソース」インスタンスの全ての組み合わせから定時トランザクションが発生することになる。しかしそうならないケースもある。ロール型別に、それぞれ複数のリソース・インスタンスが待機しているのだけれども、それらの全ての組み合わせではなく、ある決まった組み合わせからしか定時トランザクションが発生しない場合である。

例えば製造部門と間接部門では、実際に使われる勘定科目は同一というわけではない。この場合、予算をたてるときに使用する勘定科目は、部門により異なるのが普通であろう。こうした場合に、リソースの集合にある法則を見つけてリソースのサブタイプ化を行い、「複数のリソースの組み合わせによる定時トランザクション」を適用できる場合もある。しかし、リソースの集合にサブタイプ化できるだけの法則を見出せない場合は、この方法はとれない。つまりはリソースの組み合わせが個別に決定されるような場合である。

問題

個別に決定されるリソースの組み合わせ毎に、定時トランザクションが発生する。

フォース

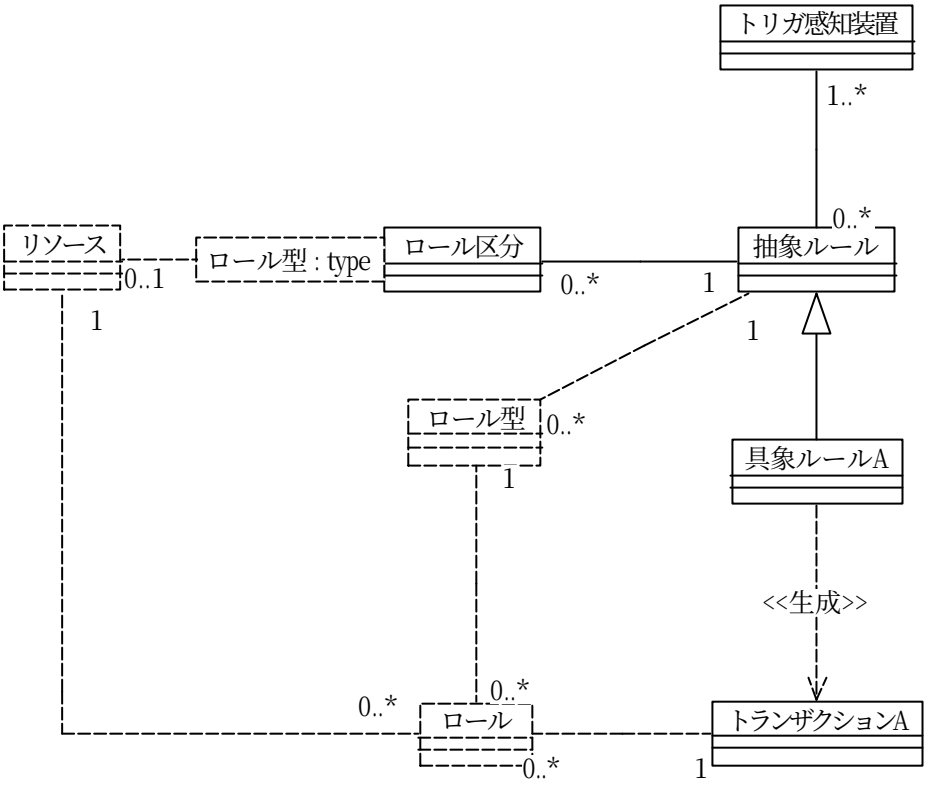
- リソースの組み合わせに法則が見出せない。戦略や歴史的な慣習などで、個別的に決定されたリソースの組み合わせ毎に定時トランザクションが発生する。

解決

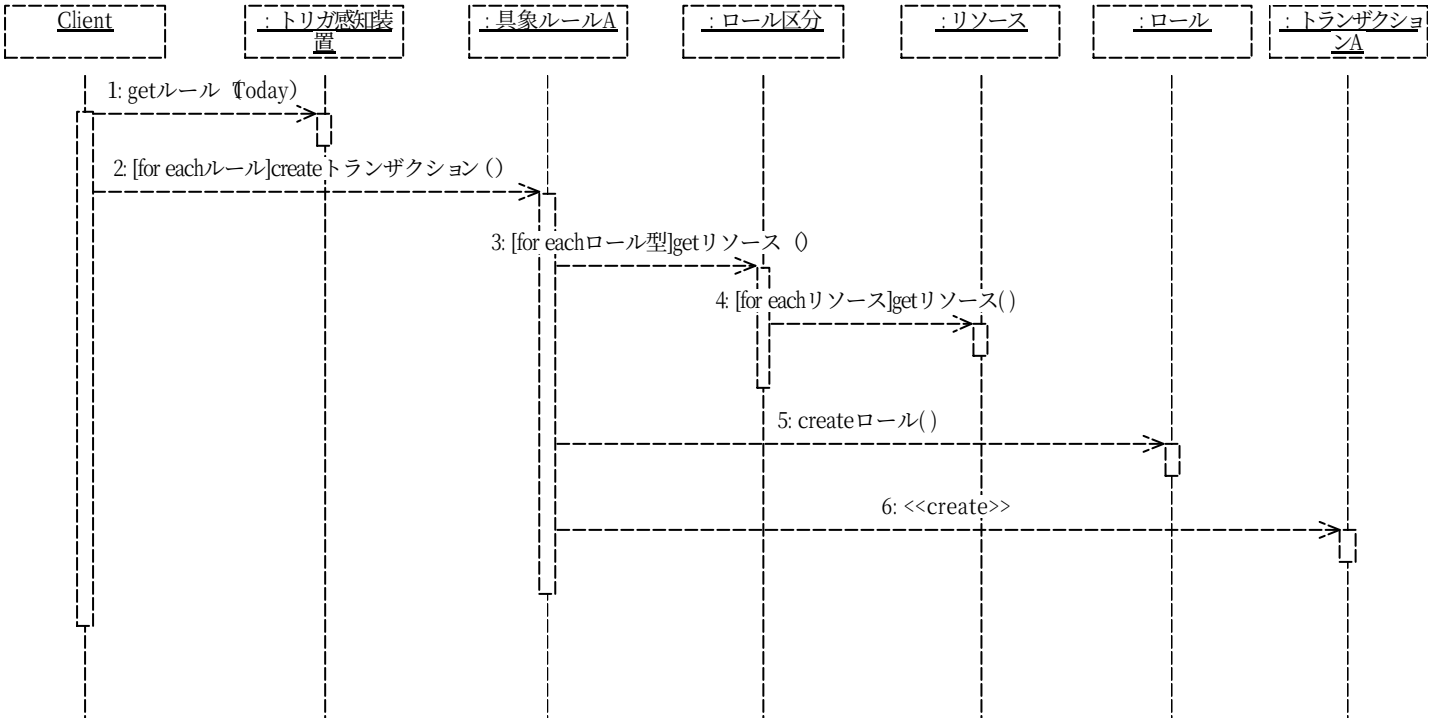
ロール区分は、リソースとの間にキー付き対応づけをもっている。キーはロール型である（「被診断者」、「診断項目」等）。ロール区分のインスタンスは［被診断者／社員 A、診断項目／聴覚］のように、定時トランザクションの発生単位となるリソースの組み合わせである。定時トランザクション型単位での、ロール区分のインスタンスの数と、１回あたりの定時トランザクションのインスタンスの発生数は同じである。

定時トランザクション型に対応づけられている「ロール型」は、「ロール区分」の必ずリソースをつきとめることのできるキーを定義している。

構造


トリガ感知装置
1..*
0..*
リソース
0..1
ロール型 : type
ロール区分
0..*
1
抽象ルール
1
1
ロール型
0..*
1
具象ルールA
<<生成>>
0..*
0..*
ロール
0..*
トランザクションA
1

インタラクション


Client
: トリガ感知装置
: 具象ルールA
: ロール区分
: リソース
: ロール
: トランザクションA
1: getルール（today）
2: [for eachルール]createトランザクション（）
3: [for eachロール型]getリソース（）
4: [for eachリソース]getリソース（）
5: createロール（）
6: <<create>>

謝辞

Ver1.0 を作成する仮定で、藤野さん、友野さんから、有意義なアドバイスをいただきました。また、Ver1.0 を JPLoP の研究会に取り上げていただき、そこで参加者の皆様からたくさんのアドバイスをいただきました。

Ver2.0 につきましては、友野さんがシェファード役を引き受けてくださりました。また、佃さんから、Ver1.0 の内容に関してメールをいただきました。

参考

Fowler の以下のパターン

1） Schedule Element（Reccurring Events for Calendars より）

2） 企業内区分

3） 連想関数

4） 転記ルール

GoF

参考文献

[Fowler96]

Fowler, M. Analysis Patterns -Reusable Object Models. Reading, MA: Addison-Wesley, 1996.

[GoF]