ソフトウェア工学 101−10
（1994. 11. 18）

アクティブデータベースの動向と応用へのインパクトについて

小島 功

電総研情報ベース研究室

kojima@etl.go.jp

要旨

本稿では、新しいデータベース応用に対するアクティブデータベースの有効性について議論する。アクティブデータベースでは、イベントや更新の波及、システムの動的な挙動とその対応が扱えるなど、データベースシステムの機能を大きく拡張している。一方で、様々な計算機応用では扱うデータが巨大化、複雑化しており、統合的なデータベース管理の必要性と、かつ、その上での高度な処理が要求されつつある。

ここでは、そのようなアクティブデータベースの機能／動向をまとめると共に、その応用への可能性について例を用いて議論する。ビューや一貫性制御など、データベースシステム自身の持つ機能が統合的に支援できるということと、ワークフローや金融支援といった、時間制約や動的な対応の要求される新しい応用にも用いられることなどを示す。

キーワード：アクティブデータベース,ECA アーキテクチャ

A Survery on the Trends and Applications of Active Database Systems

Isao KOJIMA

Information Base Section, Electrotechincal Laboratory

Abstract

This paper discusses the trends and applications of active databases. Active databases based on the ECA architecture provide flexible functionality including event handlings and transaction processings. Using ECA rules, it is possible to support several new database applications also with integrating the existing database fucntions like views.

Keywords: Active databases, ECA Architecture

1 まえがき

アクティブデータベース (Active Database Systems)[2] は、1980年代の終りごろから提案され始め、90年代に入って盛んに研究、開発の行なわれている研究分野である。この背景としては、データベースシステムの応用が広がるにつれ、そこで行なわれるデータ処理が複雑化、高度化し、処理を自動化する要求が高まっていることが考えられる。
データベースの状態を定常的にモニタし、その変化に応じて必要な処理を自動的に行なうような機能は重要で、データベース分野に限らず様々な計算機応用で何らかの形で支援されている。エキスパートシステムなどの例に限らず、既存のデータベースシステムにおいても、様々な形で条件のモニタリングや処理の自動化を実現するような機能は導入されている。
しかし、応用処理の複雑化やオブジェクト指向データベースに代表されるようなシステム自身の機能の高度化に伴ない、比較的ボトムアップ的、後づけ的に実現されてきたこれらの機能は、統合的な枠組みの中で支援する必要性が生じていると考えられる。このような立場を基礎として問題を扱うことが、アクティブデータベースの研究の基本的な特徴といえる。

2 アクティブデータベースの概要

2.1 ECA アーキテクチャ

アクティブデータベース [3] とは、データベースの内容変化など、システムの扱う様々な状況の変化事象（イベント）を検知し、それに対応する処理を自動的に起動（トリガ）するような機能を持つデータベースシステムである。
従って、データベースの状況を常時監視（モニタ）する機能と、プログラムなど処理の呼び出し機構を持っている。また、処理の結果によってデータベースの状態がさらに更新されても引き続き事象が発生、処理が継続できるようフィードバックできるようになっている。
検知すべき状況の変化の記述や、対応する操作の記述は一種の規則（ルール）として書かれ、各規則はトランザクションとして並行に処理される。この例では、機械のテストと対応する処理を行なっている記述を示しているが、一般にはデータベースを使った処理を自動化する目的から、ルールの条件や動作の記述については、問い合わせ言語などが良く使われる。簡単な例を 図 1 に示す。
このような規則の記述は、次のように大きく 3 つの要素から構成されることが多い。この要素を何らかの形式で支援するシステム構成を、広く **ECA(Event-Condition-Action)** アーキテクチャと呼んでいる。

1. あるルールの評価を行なうタイミングを決定する**事象 (Event)**

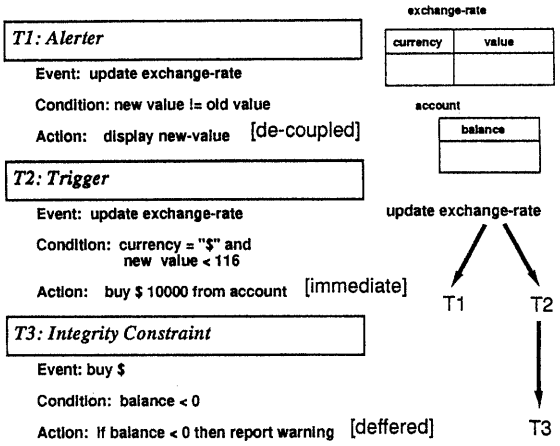

図 1: アクティブデータベースにおける、処理の記述の例

システムは次のような事象を検知し、ルールの評価を始める。

- 検索, 更新など一般のデータベースの操作。
- メソッドプログラムの起動や、コミット / アボートといった、トランザクション制御文など、実行中のシステムの状態変化。
- タイマ、クロックなどを使った時間（相対、絶対、期間）の指定。
- 応用プログラムなど外部から通知される事象。
- 以上の組合せ（and/or, 順序）によるもの。

2. 起動されたルールのうち、検査すべき部分である条件 **(Condition)** 条件は、論理式やデータベース言語などによる宣言的な記述や、検査プログラムなどを用いることが多い。起動されたルールはこの条件を評価し、満たした結果を次の動作に処理させる。条件の記述の方法によって、例えば入力データのチェックなど、データ一つ一つで評価されるような記述と、平均にかかる制約検査など、データ集合全体で評価されるような記述が可能で、モデルによって異なる。

3. 条件を満した場合に実行する動作 **(Action)** 次のような種類の動作が考えられる。

- 更新、検索といったデータベース操作
- ルール処理、イベントシグナルの生成といったシステムの制御操作

- 利用者の記述した応用プログラム

一般のルール処理では、エキスパートシステムなどのように Condition-Action ルールあるいは、if-then ルールとして、条件と操作の組として扱われることが多いが、ECA アーキテクチャでは評価を行なうタイミングとその対象を事象という要素として別に扱う。事象を起こすデータの対象や操作を明示的に記述して与えておけば、システムが監視すべき操作の範囲を制限でき、負荷を軽減できる。また、利用者にとっても、表現が自然でルールの起動の状態を分かりやすく記述、理解できると考えられる。

2.2 結合モード (coupling mode: カップリング・モード)

アクティブデータベースでは、登録されたルールの条件の成立やイベントの伝達により、更新操作を含む複数のルールが並列に起動、処理される。また、ルールの処理結果によって新たなイベントが発生する場合、さらに次のルール集合がネスト的に起動される。トランザクション処理の機能は不可欠である。
ここで、条件の評価と続く動作の実行との関係を、2つのトランザクションの親子関係と考え、この組み合わせ方にいくつかの選択枝を与えて利用者が指定できるよう考える。こうすれば、評価と実行を同じトランザクションとして扱うことも、別のトランザクションのように扱うこともでき、先のような要求に適する。このような提案の例として HiPAC で言う結合モード (coupling mode)[6] があり、次のような種類が提案されている。

1. immediate, **即時モード** 呼び出すトランザクションをサブトランザクションとし、親トランザクションを中断して先に実行する。
2. deffered、**遅延モード** 親トランザクションのコミットの寸前にサブトランザクションとして実行する。
3. de-coupled、**独立モード** 呼び出すトランザクションを全く別のトランザクションとして実行する。

図 2 にその順序関係を示す。これを使えば、例えば更新に伴う値の変更関係とルール処理の評価順序などを自由に指定することが可能である。
図 1 の 3 の例 deffered では、親となる買い動作のコミット前に対応する子トランザクションとしての Action が評価されるので、親ルールの結果を子ルールが反映している。と同時に、親のコミット前に実行されるので、警告を発生させるようなプログラミングが可能である。

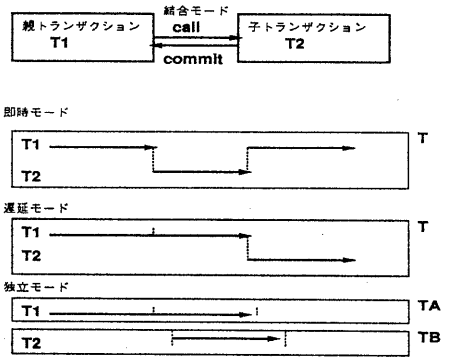

図 2: カップリング・モードの実行順序関係

2.3 特徴と動向

アクティブデータベースには、次のような特徴がある。

- イベントとそれに基づく処理の実行によって、様々なデータベース処理を自動的に行なうことができる。
- 結合モードの導入などによって、トランザクション処理のプログラミングが可能
- ECA ルール自身のオブジェクト化によってルール管理やプログラム管理といった機能を持たせることができる。

また、他のソフトウェアシステムと比較すると、次のような相違がある。

- プログラミングシステムに比較して、扱うデータが大規模な応用に向く。
- ルールシステムやエキスパートシステムに比較すると、波及的な更新操作や複雑なトランザクション処理が可能になっている。

現在研究されているシステムの多くは実験システムであり、言語／モデルなど様々なアプローチと、それに伴う問題 [3, 1] がある。

- **記述言語の問題**：関係データベースを元にした場合は宣言的なルール処理との親和性が良いが、手続き処理や応用とのインターフェイスに問題がある。また、オブジェクト指向などプログラミング言語ベースのものは、例えば複雑なイベントの記述方法などを支援する必要がある。
- **システム実現上の問題** ：データベース上でのルール処理は集合操作ベースなので効率的でな

い。また、条件に含まれる時間制約を保存するた目のトランザクション処理アルゴリズムや、効率的なイベント検出などの手法が必要である。

- **検証、デバッグの問題**：能力の高い記述言語を提供するので、誤ったプログラムでデータベースの内容が破壊されないよう、効率的な検証の手法を提供する必要がある。処理の停止性あるいは結果の一意性を保証するための支援ツールや、更新などを扱う理論的な枠組み [4] などの提供が必要になる。

3 アクティブデータベースの応用

アクティブデータベースは様々なレベルでのデータベース処理の自動化を実現するのに有効である。この応用 [5] としては、

- トリガなど既存のシステムでも実現されていた機能を、ルール記述に よって統合的に実現する。
- その高機能を使って、従来あまり支援されてなかった新しい応用に用いる。

の２つの側面がある。それぞれについて述べる。

3.1 データベースシステムの支援すべき機能の提供

例として**導出データやビューの維持** を取り上げる。更新の波及するシステムは一貫性の保持に使えるので、例えばビューのスナップショットの保持や実体化ビューの管理などが容易にプログラムできる。現データを B, ビューを V とし、V は給料の高い従業員の集合 (B.sal > value) とする。

- **実体化ビューの実現** 実体化ビューは実データを持つ別のデータとして保持され、更新時にその内容が保持される。挿入の例を示す。削除、追加の処理も同様に記述できる。

```
Event: inserted into B
Condition: if new.sal > value
Action: insert new into V
```

- **仮想ビューの処理** 仮想ビューの場合は、実体がなく、検索時に動的に作成される。

```
Event: retrieve from V
Condition:
Action: retrieve from B where sal > value
```

- **ビューの更新** ビューに対する更新は、基底データに対する更新に変換される。

```
Event: updated V.attribute
Condition:
Action: update B
        set attribute = new attribute
        where id      = new id
```

この場合、ビュー更新は一般にはあいまいなので、あいまいでない更新可能な属性かどうかの判定あるいは利用者への問い合わせを条件部につけることが考えられる。

このように、従来のデータベースビューの導出方法をアクティブデータベースを使えば一般的に扱えるので、より機能の高いビューを実現できる。

3.2 その他の応用

同様に、次のような機能がアクティブデータベースによって統合的に扱うことができると考えられる。

- **トリガ／アラータの提供** トリガもアラータも、共にデータベースの更新その他の状態変化に対して、あらかじめ定められたデータベース操作を起動するものである。アラータとトリガは、起動される操作の種類が異なる（アラータは警告、トリガは一般の操作）が、ECA ルールによる扱いは、これらを統合して記述することを可能にする。
- **意味制約の保持** アクティブデータベースの元々の動機にある通り、データベース上で成立する意味制約の保持はアクティブデータベースの実現によって柔軟に実現できる。汎用に近いルール記述により、キー制約や参照制約といった特定の制約保持要素を用いずに済み、制約の表現とその扱いを高機能化できる。
- **リアルタイム／時間応用の実現** 時間による処理／イベントによる処理により、定期的な処理や日常的な処理の自動化にはアクティブデータベースの機構は有益である。一方、条件よる管理を使って、日常業務から外れた例外的な場合のプログラム対応も可能と考えられる。
- **利用権制御** 利用者のアクセス権限などの制御を行なう場合、例えば読み出しに対しても制御が必要である。高度なルール処理はアクセス制御に有用な概念であり、このようなイベントにも対応できる。一方、利用者がルールを登録することでデータが洩れて導出されないよう、ルール自身にも権限や利用者制御の概念が必要になる。
- **データ交換、スキーマ統合** 複数のデータベースを使った巨大な意志決定システムなどの環境では、その異種性のためにデータの相互の交換方法や統合管理の方法を固定的にしておくことが

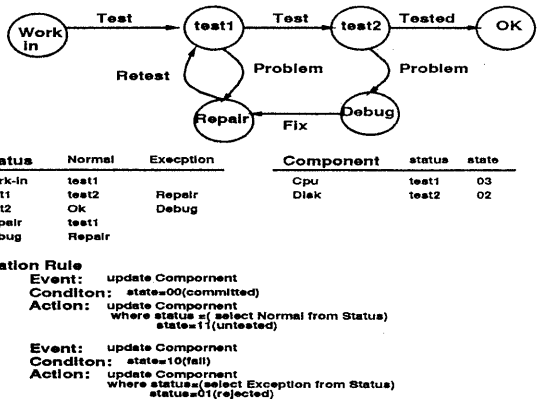

図 3: 処理手順のデータベース化

難しい。アクティブデータベースは例えば相互の自動的なデータ交換をルール処理としてプログラム可能にでき、より柔軟な環境を実現できる。

ここで示した機能は従来のシステムでは後づけ的に実現されている。一方、これらは目的は異なるものの、必ずしも機能的には大きく異なるものでないと考えることができる。つまり、アクティブデータベースを使うことで、より一般的、統合的なアプローチが可能になる。

4 アクティブデータベースの応用分野

応用側からの要求として比較的注目をあび、実現性が高そうなものとして次のようなものが考えられる。

4.1 ワークフロー管理

グループ作業やプロジェクトの工程管理など、様々な条件／制約下での実行管理などか考えられる。ワークフロー管理は時間的な制約も必要な一方、既存のワークフロー管理プログラムは管理データの巨大化に対応する必要があり、アクティブデータベースの期待される応用の一つである。
基本的には、図3のように、扱うデータをデータベース化して共有環境に置き、その間のフローや制約条件などをECAルールによって書くことになる。例を示す。

- バージョニング データベースRの更新に伴うバージョンVの管理を行なう場合、同じサイズのコピーを保存するのは効率的でない。例えば、差分ログによるバージョンの管理を行なうとして、

```
Event: retrieve from V
Condition:
Action: retrieve from V+ union R
        where not in V-
```

により、最新の版を得ることができる。中間の版を検索したい場合には、日付またはバージョン番号をパラメータとして、

```
Event: retrieve from V
Condition: backupdate < DATE < currentdate
Action: retrieve from
(retrieve from V+ where date < DATE)
 union R where not in
(retrieve from V- where date < DATE)
```

のような形式で行なう。また、Vの実体はログであるので、

```
Event: insert into V
Condition:
Action: insert into V+

Event: delete from  V
Condition:
Action: delete from V+
insert into V-
```

のように、削除データをログV－に挿入することで情報を保存する。操作履歴の全体はV＋とV－をマージし、日付順でソートすることによって得られる。

- フロー管理フローの記述もECAによって記述することができる。例えば、文書データベースの処理において、時間管理は

```
Event: time deadline
Condition: if (retrieve from Document
           where status="incompleted")
Action: send warning message to boss
```

のようになる。同様にイベントの順序関係を記述する用語(precedes,follows)によってフローが記述可能である。また、結合モードを使うことで例えば、

```
Event: update document
Condition:
Action: if review(boss)=OK then commit
                              [deffered]
```

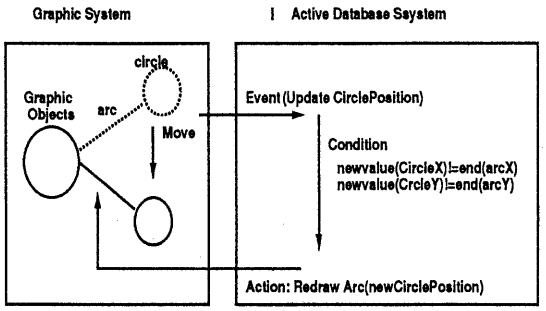

図 4: グラフィックオブジェクトのデータベース化

は、ドキュメントの編集作業であるが、一連の処理の中で最終的に review を通ればよいというフローを示す。一方、

```
Event: update document
Condition:
Action: if review(boss)=OK then commit
                                [immediate]
```

は、即座にレビューを要求し、その結果を待つことになる。

4.2 マルチメディア、空間情報の管理

例えば地理情報のような空間データは、道路の接続関係などそのトポロジを制約条件の形で記述するのが自然である。このように地理情報オブジェクト間の関連を ECA ルールにより記述してデータベースの一貫性を保つことが考えられる。例えば図 4のように、画像の拡大操作に伴う接続関係の維持は、ルールによって記述、保持することができる。
また、利用者インターフェイスなどグラフィック操作環境などの応用でも、ウィンドウシステムで見られるようなイベント管理と対応する動作という枠組みは用いられている。マルチメディアのように、複数のデータ形式の間に例えば音声と画像の出力がある条件の下で同期して出力するなどという記述にも有益である。一般にはイベント処理などはウィンドウシステムなどインターフェイス側の処理になるが、アクティブデータベースを使うことで、インターフェイスとデータベースとの接続を密にして、インタラクティブ性を向上させることができる点が特徴である。

4.3 その他の応用

その他自動化を含む高度処理、意志決定支援、あるいは共同作業管理などに関連して、いくつかの応用が考えられる。

- 金融応用 最初の例で示したように、証券や為替のデータベースを用いて価格変動を分析し、必要な売買などを行なうアナリスト的な支援や、データベースによる在庫管理と発注の自動化といった利用が考えられている。また、小切手の電子的な処理や抵当不動産の管理などへの利用なども含め、比較的大規模な応用が考えられている。

- ＦＡ パワープラントの管理などでは、監視している値をデータベース化しておき、それを通してプロセスの監視や制御を行なうことができる。この場合は、現実のプロセスの制御が可能にするため、処理の実時間性を保証する問題が生じる。

- データ分析／発見 様々な解析のためのルールの登録により、データの分布や相関の分析、あるいは条件による自動的な構造化、分類といった、データベースから法則的な情報を得るのに使うことができる。

- エンジニアリング応用 ソフトウェアエンジニアリングやＣＡＤのような共同作業環境において、バージョン管理やデータのローカルな変更に伴う通知、伝搬、一貫性の保持などに用いることができる。

その他にもネットワーク管理や移動体の制御に ECA の実時間制約を用いるという応用などがある。

5 あとがき

アクティブデータベースの概要とその応用について述べた。多くのシステムが研究開発の途中であるものの、ここで示した機能のうちには SQL 3 などで検討されているものもあり、導入されて実現される可能性があるものが多い。

参考文献

[1] ,大蒔他: "ブルーバブル情報ベースシステムに関する調査報告", 電総研調査報告第２２３号、 1993.03.

[2] U.Dayal: "Active Database Systems," 3rd International Conference on Data and Knowledge Bases, pp.150-169, 1988.

[3] 小島："アクティブデータベースシステム" データエンジニアリングフォーラム 1994.10

[4] 小島他：" プロセス代数による、アクティブデータベースの支援について" 情報処理学会研究報告 1993.05

[5] RIDE94', Research Isshues in Data Engineering , 1994.02

[6] IEEE Bulletin of Data Engineering 1992 Vol15,No1-4