データベースシステム 119－62
（１９９９．７．２３）

移動体計算環境におけるアクティブデータベースのECAルール実行監視機構の設計と実装

寺田 努† 莫 君† 村瀬 亨‡ 塚本 昌彦† 西尾 章治郎†
†大阪大学大学院工学研究科情報システム工学専攻
‡住友電気工業株式会社システムエレクトロニクス研究開発センター

無線通信や計算機ハードウェア技術の急速な発展に伴って，ユーザは無線通信機能を持つ携帯端末を用いて，場所を固定せずにネットワークを介して情報を利用することが可能になった．筆者らは，このような環境において移動体がもつデータを統合利用するために，アクティブデータベースを拡張し，移動体の接続，切断，データ交換などを処理するAMDS(Active Mobile Database System)を提案・実装してきた．AMDSの動作言語であるECAルールは，記述能力が高く，連鎖的に実行させることで複雑な処理が記述できる一方，予期しない異常動作を起こす可能性がある．そこで，本研究では，AMDSを拡張し，ECAルールの連鎖実行の検出や，連鎖回数のチェックなどを行なうECAルール実行監視機構の実現を目的とする．本機構を用いることで，ECAルールを用いたアプリケーションをより安全に運用できるようになる．

Design and Implementation of an ECA Rule Execution Monitering Mechanism in Active Database in Mobile Computing Environments

Tsutomu TERADA† Jun MO† Toru MURASE‡ Masahiko TSUKAMOTO† Shojiro NISHIO†
†Department of Information Systems Engineering, Graduate School of Engineering, Osaka University
‡Systems and Electronics R & D Center, Sumitomo Electric Industries, Ltd.

As technologies of wireless communications and computer hardware grow rapidly, users can access a variety of information from anywhere using handy terminals with wireless communication equipments. In this environment, to integrate and use the data held by mobile host, we proposed and implemented an AMDS (Active Mobile Database System) as a kernel system for data and mobile host management in mobile computing environments. The behavior definition language of AMDS, ECA rules, has high description ability, thus enabling users to define complicated behavior. However, it is possible that the execution of ECA rules may fall into a chain of unexpected behaviors. In this paper, we realize an ECA rule execution monitoring mechanism to detect a chain of ECA rules and check the chain frequency, and integrate this mechanism into AMDS. Using this mechanism, applications with ECA rules can be used more safely.

1 はじめに

近年，無線通信技術や計算機ハードウェア技術の急速な発展に伴って，無線通信機能をもつ携帯端末を用いることにより場所を固定せずにネットワーク上のさまざまな資源を利用することが可能になった．この新しい計算環境は移動体計算環境と呼ばれる．移動体計算環境のモデルは図1に示すように，固定ネットワークに無線通信可能な移動端末を含んだ形態である．移動体計算環境においては，以下のような新しいサービスを提供できるようになる．

- 会社内などにおいて，各社員は一台ずつ各自のスケジュールが入力された携帯端末を持ち歩き，本社でスケジュールデータを統合して全社員のスケジュールを管理し，効率的に仕事を割り当てる．
- 遊園地などのアミューズメント施設で入場者に一台ずつ携帯端末を持たせ，各アトラクションの待ち時間等の情報を提供する．
- 美術館や博物館において，入館者は一台ずつ携帯端末を持ち，展示物に近づくと自動的にその展示物の詳細情報が表示される．

このようなサービスを実現するためには，移動体から，または移動体上でデータを収集する必要がある．しかし，従来の分散データ管理技術では移動するホストを考慮していないため，移動体のネットワークへの接続・切断やデータの収集を自動的に行なう共通の基盤が望まれている．このような要求に対し，筆者らは，イベント駆動型データベースであるアクティブデータベースを拡張することで，移動体計算環境における各種のイベントを容易に扱うことができるアクティブモー

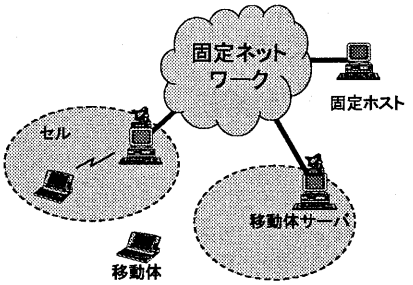

図 1: 移動体計算環境

バイルデータベース (Active Mobile Database System: AMDS) の研究を行なってきた [3][8]．AMDS では，イベント，コンディション，アクションの 3 つを一組として記述する ECA ルールによって動作を記述する．イベントには従来のアクティブデータベースのイベント (更新・挿入・削除など) に加え，移動体の接続・切断などを扱うイベントも提供している．

ECA ルールは，アクションの実行によって新たなイベントを引き起こすことができる．ECA ルールを連鎖的に実行させることで，複雑な動作が実現できる一方，無限ループなど，予期せぬ動作に陥る可能性がある [4]．そのため，ECA ルールの実行を監視してシステムの異常動作を回避する必要がある．そこで，本研究では，移動体計算環境において，ECA ルールの連鎖によるシステムの異常動作を実行時に検出するための ECA ルール実行監視機構を実現した．以下，2 章では AMDS の概要について述べ，3 章では今回実現した ECA ルール監視機構について述べる．4 章で評価及び考察を行ない，最後に 5 章で本研究のまとめと今後の課題について述べる．

2 AMDS

AMDS はアクティブデータベースを拡張して移動体計算環境に適合させたものである．アクティブデータベースは，データベースの内界・外界で起こる事象の発生に対して，規定された処理を行なうデータベースである [2]．その動作は，発生する事象 (イベント)，ルールの発火条件 (コンディション)，実行される操作 (アクション) の 3 つの組みで表わされる ECA ルールで記述される．

```
create rule 接続 on CONNECT
  then do
   SEND( new.from, "Request_" );
```

図 2: ECA ルール例 1

```
create rule 返信 on RECEIVE
  where new.header = 'Request_'
  then do data = QUERY("select s.*
                  from Schedule s");
   SEND( new.from, "result_", data );
```

図 3: ECA ルール例 2

AMDS には，従来のアクティブデータベースで提供されている，データベースに関するイベント (挿入，削除，更新，検索) に加えて，移動体の動作に関するイベント (移動体のセルへの接続・切断)，AMDS 間の通信に関するイベント (データの受信) が用意されている．アクションには AMDS 間のデータ送信関数，データベース対する問い合わせ関数などが提供されている [5]．

AMDS は一般のアクティブデータベースと同様 ECA ルールで動作する．AMDS における ECA ルールの記述例を図 2，3 に示す．例 1 に示した接続ルールは，移動体が接続してきたときに，その移動体に対してデータ要求を行なう移動体サーバ用のルールである，例 2 に示した返信ルールは，移動体サーバからの要求パケットを受信したとき，スケジュールデータを送り返す移動体用ルールである．

移動体サーバでは，移動体が接続してきたときに接続ルールが起動し，接続ルールのアクションにより，移動体の返信ルールが起動する．このように，ECA ルールでは，アクションの実行が新たなイベントを発生させることができるため，1 つのイベントの発生によって複数の ECA ルールを連鎖的に実行させることができる．そのため，ECA ルールを連鎖的に実行させることで，複雑な動作が実現できる．

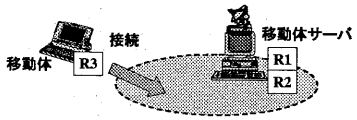

図 4: 異常動作の例

表 1: ルールの内容

ルール	E	C	A
R1	移動体接続	–	データ要求
R2	パケット到着	身元確認	身元要求
R3	パケット到着	–	身元要求

3 ECA ルール実行監視機構

本章では，ECA ルールの連鎖による異常動作の例を示し，アクティブデータベースの異常動作検出手法と，本研究で提案する ECA ルール実行監視機構について説明する．

3.1 ECA ルールの異常動作

ECA ルールは，アクションの実行が新たなイベントを発生させることができるため，複数の ECA ルールを連鎖的に実行させることができる．そのため，ECA ルールを連鎖的に実行させることで複雑な動作を実現できる一方，無限ループなど予期しない連鎖を起こす可能性があり，このような異常動作を検出することが必要となる．

図4，表1に，異常動作の例を示す．R1，R2は移動体サーバに格納されているルール，R3は移動体に格納されているルールである．移動体がセルに接続すると，R1が起動され，移動体サーバは移動体に身元問い合せを行なう．問い合せにより移動体のR3が起動され，移動体は移動体サーバに移動体サーバの身元問い合せを行なう．これによりR2が起動されるが，このルールは再びR3を起動するため，R2とR3の間に無限ループが発生する．

移動体がもつルールと移動体サーバがもつルールを個別に見ても，ループを起こすかは判断しづらいが，移動体の移動や，もっているルールの変化などにより，異常動作を起こしてしまう．

このような異常動作を解析的に検出するのは非常に困難であるが，ECA ルールを用いたアプリケーションを構築する場合，ECA ルールが異常動作を行なわないことを保証する必要がある．したがって，異常動作が起こるかどうかをあらかじめ調べたり，異常が実際に起こったときにそれを検出するような仕組みが必要となる．

3.2 異常動作の検出手法

アクティブデータベースにおける異常動作検出には，次に示す2つの手法が考えられる．

- 事前検出

  データベース (ECA ルール) が動作していない状態で，トリガグラフと呼ばれる有向グラフを用いて論理的に異常動作の有無を調べる．

- 実行時検出

  実際にデータベースが動作している状態で，リアルタイムに異常動作の有無を検出する．

事前検出で用いるトリガグラフ [6] とは，ある ECA ルールから次に引き起こすルールに対して有向のパスをはるという作業をすべてのルールに対して行なったもので，出来上がったグラフ中にループが存在していなければ，そのルール群は安全であるとするものである．トリガグラフは，その信頼性を高めるためのさまざまな拡張がされており [1][7]，アクティブデータベースの無限ループ検出においては非常に信頼性が高い．

しかし，トリガグラフは，本来ループとならないルール群まで無限ループと判断する可能性がある．また，AMDS で想定する移動体計算環境においては，移動体の移動によりネットワーク構成が動的に変化する．さらに，AMDS では，ECA ルールのサイト間でのやりとりも頻繁に起こるため，複数サイト間にまたがる ECA ルールの相互関係を考慮する必要がある．そのため，完全なトリガグラフを作成することは困難である．

このように，移動体計算環境上では移動体の移動により，ネットワーク状態は多様であり，そのすべての場合を考えて静的検出を行なうのは現実的ではない．そこで，本研究では実際にシステムを実行させながら，ECA ルールの連鎖実行を監視することで，システムの異常動作を検出する．

3.3 異常検出のパラメータ

提案する実行監視機構では，ルールカウンタ，タイムスタンプという2つのパラメータを用いて異常動作を検出する．

- ルールカウンタ

  現在のECAルールの連鎖的な実行回数を表わす．

- タイムスタンプ

  ECAルールの連鎖実行が開始された時刻が刻まれている．

ECAルール実行監視機構では，新たな連鎖実行が開始されるたびに，ルールカウンタとタイムスタンプを作成する．ルールが連鎖的に実行されている間，カウンタは増加し続け，値が一定以上になったり，タイムスタンプに刻まれた時刻から一定以上経っても連鎖が終了しなかった場合，異常動作が起こったと判断する．

ルールカウンタや，タイムスタンプから経過した時間が異常動作だと判断する値は，ユーザやアプリケーションが設定するが，その値については4章で考察する．

3.4 移動体の移動

本節では，異常動作検出の信頼性を高めるための，移動体の移動を考慮した拡張について述べる．移動体計算環境では，ネットワーク構成の変化により，それまで記録していたカウンタの内容が無効になる場合があるため，次のような処理を行なうことで異常動作検出の信頼性が高まる．

- 1ループ中に異なる2つの場所で，同じ移動体による連鎖の中継が起こった場合は，1回目の移動体中継以前のカウントは無効にする．

- ループを中継していた移動体が切断したら，切断以前のカウントは無効にする．

また，さまざまなホスト間を移動しながら仕事を行なうタイプのアプリケーションでは，多数のホスト間を移動しながらデータを収集するようなルールが考えられる．そのような場合，カウンタのリミットを大きくすることで対応できるが，あまりカウンタのリミットを大きくしてしまうと，異常動作が検出できない．そこで，このようなアプリケーションにも対応するため，ルールカウンタを次の2段階に分ける．

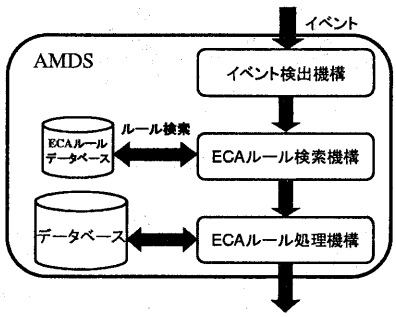

図 5: AMDSのシステム構成

- グローバルなカウンタリミット

  とにかくこの値を超えたら絶対に異常動作とする値．大きな値を設定する．

- ホスト内連鎖のみに設定されるカウンタリミット

  ホストのループが起こっていないときは，ホスト間で連鎖の移動が起こった場合，それまでのカウンタの値は考慮されず，各ホスト内の連鎖がこの値を超えなければ異常動作とはみなさない．ただし，ホストのループが起こっていた場合(同じホストを2回通っていた場合)は，この値を合計カウントと比較する．

このように，カウンタのリミットを2つ設けることによって，ホスト間を移動しながらデータを集めるタイプのアプリケーションに関してはある程度連鎖数が大きくなっても異常動作とならず，かつ，実際にホスト内で異常動作が起こった場合や，ホスト間でループが発生した場合の異常動作検出は行なえることになる．

3.5 実行監視機構の設計

以上に述べたECAルール実行監視機構を加えたAMDSのシステム構成を図5，6に示す．以下，システムの各部について説明する．

- イベント検出機構

  システム内外に起こったイベントを検出する．

- ECAルール検索機構

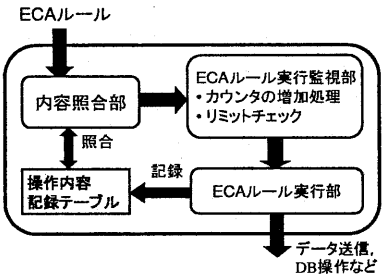

図 6: ECA ルール処理機構の構成

イベント検出機構から受け取ったイベントを受け取り，ルールベースから対応するECAルールを検索する．存在すれば，それらのルールすべてを取り出し，コンディションの評価を行ない，実行されるべきルールをECAルール処理機構に渡す．

- ECA ルール処理機構

  受け取ったECAルールを実行する．

3.5.1 連鎖性の判断

AMDS では，イベント検出機構において検出したイベントが，新たに起こったイベントであるのか，あるいは他のルールのアクションから引き起こされた連鎖中のイベントなのかが判断できない．そこで，ECAルール処理機構を，図6に示すように拡張し，イベントの連鎖性を判断する機構を組み込む．

ECA ルール実行時に，そのアクション内容およびカウンタ，タイムスタンプの値を操作内容記録テーブルに記録しておく．新たにECAルールが実行される場合は，内容照合部において，操作内容記録テーブルに記録されたアクション内容と，現在のイベント内容を比較することで，どのルールのアクションから連鎖したのかを判断する．連鎖でないと判断した場合は，新たにカウンタを設置する．ECAルール実行監視部では，カウンタの増加処理や，カウンタ，タイムスタンプのリミットチェックを行なう．

アクションがデータ送信であった場合は，現在のカウンタとタイムスタンプの値を送信パケットに付加する．受信側は，データ受信イベントを処理する際，カウンタとタイムスタンプを復元して用いる．

また，3.4節で示したように，ループの中継移動体が切断したらそれ以前のカウントを無効にする拡張を行なうためには，操作内容記録テーブルに，ループを中継した移動体情報と，そこまでのカウントの値も記録する必要がある．そして，移動体の切断イベントを処理する際に操作内容記録テーブルをチェックし，該当移動体が含まれるデータを書き換える．また，中継移動体情報を用いれば，移動体の中継がループしているかどうかがわかるため，ホストのループの有無によって処理を切り替える拡張も可能となる．

3.6 システムの実装

以上に説明したECAルール実行監視機構を実装した．実装には，IBM 社の Think Pad770E を用い，Windows95 上で，Visual Basic5.0 および Visual C++5.0 を用いて行ない，無線通信には赤外線通信を用いた．さらに，実装したAMDS 上に，スケジュール管理アプリケーション及び，博物館アプリケーションを実装し，ECAルール実行監視機構の動作を確認した．

4 考察

本章では，実行監視機構で用いるパラメータの値設定及び，実行監視機構がトラフィックや処理時間に与える影響について議論する．

4.1 アプリケーション毎の設定

AMDS で構築するアプリケーションは以下に示すように分類できる．

- リアルタイム性が必要なアプリケーション

  セル内に存在する人を1秒ごとに記録するなど，処理に時間制約をもつアプリケーションでは，タイムスタンプのリミットを短く設定することによって，制限時間内に達成できなかった処理を検出できる．

- 一つの処理に長時間かかるアプリケーション

  全国の社員のデータを収集するなど，処理に非常に時間がかかるアプリケーションでは，タイムスタンプは利用せず，カウンタの値を大きくするか，あるいは3.4節で述べたように，ホスト内とホスト

表 2: トラフィックと処理時間の増加

監視レベル	処理時間 (%)	トラフィック (byte)
なし	100	4098
通常	102	4448
拡張	105	5127

外で別のカウンタリミットを用いることで，信頼性の高い異常検出を行なうことができる．

- その他のアプリケーション

  処理時間が短く，リアルタイム性を考慮しなくてもよいアプリケーションでは，タイムスタンプとルールカウンタを統合利用することで，異常動作の検出を行なえる．

このように，ルールカウンタとタイムスタンプのリミットを適切に設定することで，さまざまなタイプのアプリケーションに対応できる．

4.2 トラフィックと処理時間の増加

ECA ルール実行監視機構を用いることで，AMDS アプリケーションの安全性が増加する．一方，カウンタやタイムスタンプの管理および，他サイトへの連鎖情報の送信などにより，トラフィックや処理時間も増加する．

本研究では，AMDS 上のスケジュール管理アプリケーションにおいて，トラフィックとルール処理時間の増加を調べた．結果を表2に示す．監視レベル'なし'は実行監視を行なわない状態，'通常'は，ルールカウンタとタイムスタンプを用いた場合，'拡張'は，さらに3.4節で述べた拡張を行なった場合である．結果からわかるように，ルール処理においては，コンディションの評価に時間がかかるため，実行監視機構による処理時間への影響はほとんど見られない．トラフィックに関しては，'通常'の場合はそれほど増加していないが，'拡張'の場合は25% 程度も増加している．これは，スケジュール管理アプリケーションが多くのホスト間の連鎖によって処理を実現しているためであると考えられる．したがって，特に拡張した手法が必要である場合以外は，'通常'を用いるべきである．

5 おわりに

本研究では，AMDS における ECA ルール実行監視機構の設計および実装を行なった．提案した機構を用いることで，移動体計算環境においても ECA ルールの異常動作が検出できる．今後は，カウンタの最適なリミット値について考察すると共に，静的検出も含めた AMDS の安全性について更に考察する予定である．

謝辞

本研究は，日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「マルチメディア・コンテンツの高次処理の研究 (Project No. JSPS-RFTF97P00501) および文部省科学研究費補助金重点領域研究 (1)「高度データベース」(課題番号08244103) の研究助成によるものである．ここに記して深謝の意を表す．

参考文献

[1] A.P.Karadimce and S.D.Urban, "Refined Trigger Graphs: A Logic-Based Approach to Termination Analysis in an Active Object-Oriented Database," *ICDE'96*, pp. 384–391 (1996).

[2] 石川博: "アクティブデータベース," 情報処理, vol. 35, no. 2, pp. 120–129 (1994).

[3] 村瀬亨, 塚本昌彦, 西尾章治郎: "アクティブデータベースシステムによる移動体計算環境におけるデータ統合," 電子情報通信学会データ工学研究会, vol. 95, no. 287, pp. 41–48 (1995).

[4] 村瀬 亨, 塚本 昌彦, 西尾 章治郎: "移動体環境におけるアクティブデータベースの安全性について," 情報処理学会研究報告, 96-DBS-106, vol. 96, no. 11, pp. 33–40 (1996).

[5] 成田藤智, 村瀬亨, 塚本昌彦, 西尾章治郎: "移動体環境におけるアクティブデータベースの設計と実装," 電子情報通信学会総合大会講演論文集, pp. 315–316 (1996).

[6] S.Ceri and J.Widom, "Deriving Production Rules for Constraint Maintenance," *Proc 16th VLDB Conf*, pp. 566–577 (1990).

[7] S.Y.Lee, T.W.Ling, "A Path Removing Technique for Detecting Trigger Termination," *EDBT*, pp. 341–355 (1998).

[8] T. Murase, M. Tsukamoto, and S. Nishio: "A system Platform for Mobile Computing base on Active Database," *in Proc. International Symposium on Cooperative Database Systems for Adavanced Applications*, vol. 2, pp. 424–427 (1996).