データベースシステム　100−11
（１９９４．１０．１４）

知的分散データベースシステムの高信頼化方式

佐藤 英昭，原嶋 秀次
{sato,haras}@ssel.toshiba.co.jp

（株）東芝　研究開発センター
システム・ソフトウェア生産技術研究所
〒210 川崎市幸区柳町 70

我々はオブジェクト指向分散システムである知的分散システム上のデータベースとして知的分散データベースの研究・開発を行なっており，これまでにリレーションを管理するオブジェクトの集合によりRDBMSを実現する方式を提案し，その実装を行なった．

今回，本システムの高信頼化の１つとして，シャドウ・ページングを用いたデータ更新の無効化機能を持つファイルシステムを実装し，これによるリレーションデータの管理機構を実現したので報告する．本稿では，本方式の概要と本方式によりデータ更新内容やデータの完全なコピーが必要なく高速な処理が可能であることを示す．

A Reliable Mechanism of IDPS-DB

Hideaki Sato, Shuuji Harashima
{sato,haras}@ssel.toshiba.co.jp

System & Software Engineering Laboratory
Research & Development Center
TOSHIBA Corp.

We are developping a database management system on IDPS which is object-oriented distributed system. We've implemented facility of RDBMS as set of objects which manage there own relations.

We implemented a reliable mechanism for this system which uses file system with notion of shadow pages. This paper discribes out-line of the mechanism and shows that by using this mechanism we can abort update fast without recording of previous operation or copying whole relation which includes target tuples.

1 はじめに

東芝では，分散オブジェクト指向システムである知的分散システム (IDPS) [3]-[5] を基盤とした知的分散データベース (IDPS-DB) の研究・開発を行なっている [6]-[9]．現在のところ，データモデルとしてはリレーショナルモデルを採用し，リレーショナルデータベースとしての基本機能である関係演算や B^+-tree によるインデックス機能などを作成してきた．

本報告では，IDPS-DB の高信頼化を実現するために作成した，データ更新の無効化機能を持つファイルシステムの設計とそのデータベースへの組み込みについて報告する．

以下，第 2 章では知的分散システムの概要について述べ，第 3 章では知的分散データベースの構成について述べる．第 4 章では今回作成したファイルシステムの構成と動作を述べ，第 5 章でデータベースへの組み込み方法について説明する．最後に第 6 章でまとめと今後の課題について述べる．

2 知的分散システムの概要

知的分散システムは，オブジェクト指向分散システムのアーキテクチャであり [3]-[5]，データと手続きからなるオブジェクトが互いに協力・協調することによって柔軟性，拡張性の高いシステムの実現を可能とする．

知的分散システムは現在 LAN によって接続された計算機上に構築されており，分散環境化でのオブジェクト間の通信機能を提供する．また，各オブジェクトを管理する集中管理機構を持たないので，システム全体の状況の保持，管理を行なうような実体は存在しない．したがって，オブジェクト間の同期や排他制御，負荷分散や機能配分などシステム全体に渡る管理は各オブジェクトの協力・協調によって行なわれる．

このように，オブジェクトが自身の機能の実現に必要な機構だけでなく，他オブジェクトと協力・協調する機構を持つので，オブジェクトの追加や変更，削除，計算機間の移動等が局所的な操作で実現でき，柔軟性，拡張性の高いシステムの実現が可能となる．

ここで，協力とは与えられたジョブの処理をオブジェクトが互いに処理結果をやりとりすることによって実行することであり，協調とはオブジェクト間で処理の競合を調整し，システム資源の有効かつ適切な利用（排他管理，負荷および機能の配分管理）をはかることを意味する．

図 2.1 に知的分散システムにおけるオブジェクトの構成を示す．オブジェクトは，自律的かつデータ独立である．オブジェクトが自律的であるとは，自身の行動規範を自身が有し，かつその行動方法（すなわち手続き）を自身に有していることを，データ独立であるとは，その行動規範が外部から隠蔽されており，決められたインターフェースを通してしか外部からアクセスできないことを意味する．

オブジェクトの構成要素であるデータと手続きは図 2.1 に示すように，共通知識と固有知識に分けて考えることができる．共通知識は排他制御，故障管理，負荷分散などのシステム全体の管理に必要なデータと手続き群からなり，固有知識はオブジェクトに固有な性質を表すデータと手続き群からなる．前述したオブジェクト間の協力・協調は，各オブジェクトの持つ共通知識によって行なわれる．

システムに与えられたジョブは，前述のようにオブジェクト間の協力・協調によって処理される．この場合，協力・協調に必要となる他のオブジェクトが，どこにどれだけいるかといった情報が必要になるが，知的分散システムでは放送通信を使ってメッセージ交換を行なうことにより，必要になった時点でこれらの情報を得ることができる．このため，アプリケーションはオブジェクトの位置や多重度を直接意識する必要がなく，柔軟で拡張性の高いシステムの構築が可能である．

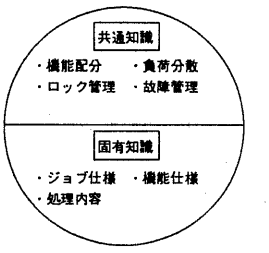

図 2.1: 知的分散システムのオブジェクト構成

3 知的分散データベースの概要

3.1 知的分散データベースの背景

知的分散データベースは，前述の知的分散システム上に実現されており，従来のようなデータベース管理システム (DBMS) は存在せず，ある機能を持ったオブジェクトが放送通信によりメッセージを送受信しながら協調動作することによってデータベースの機能を実現している．これにより，

1. 計算機構成が変化しやすい，
2. データの物理的な位置が変化する，
3. 利用可能なデータが動的に変化する，

など，アプリケーションがどのデータにアクセスすれば必要な処理が実行されるのか判断できないようなシステムにおけるデータベースとして有効である．すなわち，アプリケーションが実行したい処理の内容だけを伝えれば，本データベースを構成するオブジェクト群の協力・協調動作によって必要な処理を行なうことが可能である．これは，データベースシステムとしての柔軟性や拡張性を念頭においたものであり，複雑多様化するシステムに対応できるデータベースシステムの実現を目指している．

また，現在のところデータモデルとしてはリレーショナルモデルを採用しているが，今後オブジェクト指向データモデルもサポートしていく予定である．

3.2 知的分散データベースの構成

図 3.1 に知的分散データベースのシステム構成を示す．UNIX 上に実現されたミドルウェアである知的分散システムをベースとし，以下に示す3種類のオブジェクトによりデータベースを構成している．

- **サイトマネージャオブジェクト (SMGR)**
  システムを構成する各マシン毎に存在し，そのマシンに存在するリレーション名やオブジェクト数といったマシンに固有なデータベースのディクショナリ情報の一部を管理する．

- **質問処理オブジェクト (QPO)**
  アプリケーションからの問い合わせ処理を受け持つ．すなわち，アプリケーションからの処理要求を受けつけて，その時点でのシステムの状況を確認して処理戦略をたて，本データベースを構成する各オブジェクトへ必要な処理を依頼する．また，各オブジェクトからの結果を受け，アプリケーションへ返す処理を行なう．したがって，アプリケーションは QPO とのやりとりだけを行なえば良く，アプリケーションとデータベース間のインターフェースの役割を果たす．

- **データオブジェクト (DTO)**
  データベースで使用されるデータを管理するオブジェクトであり，リレーションのスキーマや実データおよびインデックス用のデータ等を管理する．また，1つの DTO で複数のリレーションを持つことができ，意味的に関連のあるリレーションをまとめて1つのオブジェクトとして管理することが可能である (図 3.2 参照)．このようなリレーション管理方式を採用することにより，次のような効果が期待できる [9].

1. 意味的にまとまりのあるリレーションの集合を効率良く扱える．
2. 意味的にまとまりのあるリレーション間で，操作の制約などアプリケーションに応じた処理を容易に実現できる．
3. システムを効率良く動作できる．

この他に，データベースを使う側のアプリケーションオブジェクトが存在し，マンマシンインターフェースを備えたオブジェクト等を必要に応じて作成することになる．

4 ファイルシステム

4.1 目的

知的分散データベースでは，これまでデータの更新時にエラーが起こった場合，データファイルの内容は保証されていなかった．このため，エラー

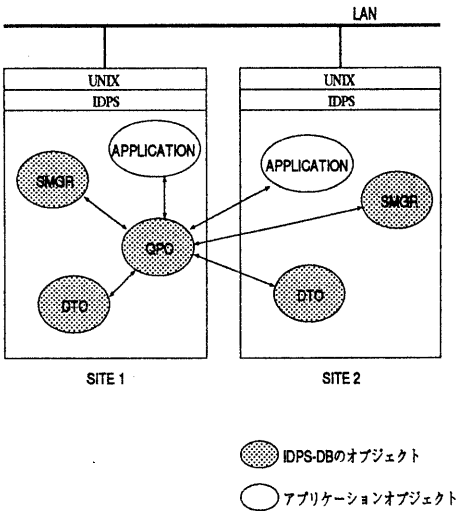

図 3.1: 知的分散データベースのシステム構成

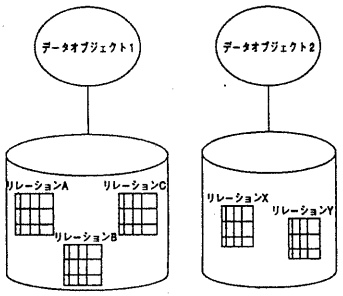

図 3.2: データオブジェクトの構成

発生時にはデータの整合性が保たれず，利用者が作成したバックアップデータまで遡って回復する以外方法がなかった．

そこで本報告では，更新処理を実行する以前にデータベースシステム内部でバックアップデータを保持し，データ更新時にエラーが発生しても更新前の状態へ復帰できるような機構を実現し，データベースの信頼性を高めることを目的とする．なお，データのバックアップを効率良く行なうため，文献 [10] で提案されている Shadow Paging の手法を基本としてファイルシステムの設計を行なった．

4.2 ファイルシステムの構成

図 4.1 に今回作成したファイルシステムの構成を示す．対象とする 1 つのデータファイルに対し 4 つの管理用ファイル (PAGE TABLE FILE 1,2,3 および MODIFIED PAGES BITMAP) を設け，これらを用いてデータのアクセスおよび更新前データへの復帰動作を可能にする．

PAGE TABLE FILE は，DATA FILE の各ページに対応するページ番号を保持し，これを順番に読み出すことによりデータファイルをシーケンシャルにアクセスできる．例えば図 4.2 に示すように，ページテーブルファイルの先頭からページ番号を 1，2，0 と読み出すことにより，それに対応してデータファイルを A，B，C の順に読むことになる．今回の設計では 3 つの PAGE TABLE FILE を用意し，それぞれ更新前，現在，およびテンポラリーのデータを保持するために使用する．

また，データファイルの空き領域を管理するために，FREE PAGE BITMAP という領域を PAGE TABLE FILE に設ける．これは，ページの使用・未使用をビット列で表したものである．

この他に，データ更新中に更新されたページを管理するための MODIFIED PAGES BITMAP をファイルとして持つ．ただし，このファイルはテンポラリーとして用いられ，データ更新が確定された後ではファイルの内容を参照することはない．

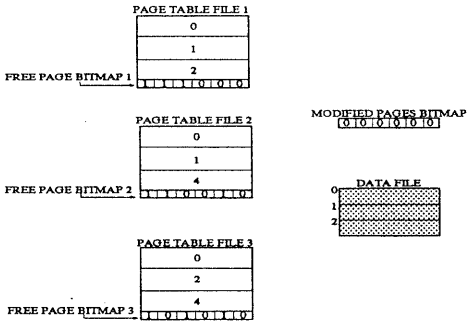

図 4.1: ファイル構成

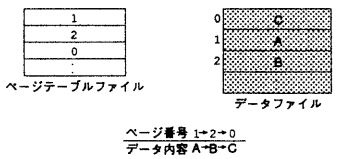

図 4.2: ページテーブルとデータファイルの対応

4.3 データの更新処理

今回作成したファイルシステムによるデータの更新手順を以下に示す.

更新を行う前に図 4.3 のような状態であるとき, DATA FILE の 2 番目のページの内容を変更する場合の処理を図 4.4 に示す. 具体的には次のような流れとなる.

Step 1. DATA FILE オープン時に PAGE TABLE FILE CURRENT をメモリ上に読み込む.

Step 2. メモリ上のページテーブルを参照して, データファイルから該当するページをメモリへ読み込む.

Step 3. データを更新する.

Step 4. 更新した内容をデータファイルへ書き込む. このとき, 2 番目のページを直接変更せずに別の空いている領域へ書き込む.

Step 5. メモリ上のページテーブルの値を, 書き込まれたページ番号の値に変更する. このとき, 変更されたページ番号に対応するメモリ上の MODIFIED PAGES BITMAP と, FREE PAGE BITMAP CURRENT の値を変更する.

Step 6. メモリ上の PAGE TABLE と BITMAP をファイルへ書き込みクローズする. このとき, PAGE TABLE と FREE PAGE BITMAP は PAGE TABLE FILE temp ファイルへ書き込む.

Step 7. 以下他のファイルに対する処理があれば同様に実行する. 無い場合は更新の確定処理 (コミット) を行い, PAGE TABLE FILE current を prev temp を current に変更する.

以上の手順に従い, ページテーブルを用いてデータファイルを扱うと, ファイルへの正しいアクセスが可能となる.

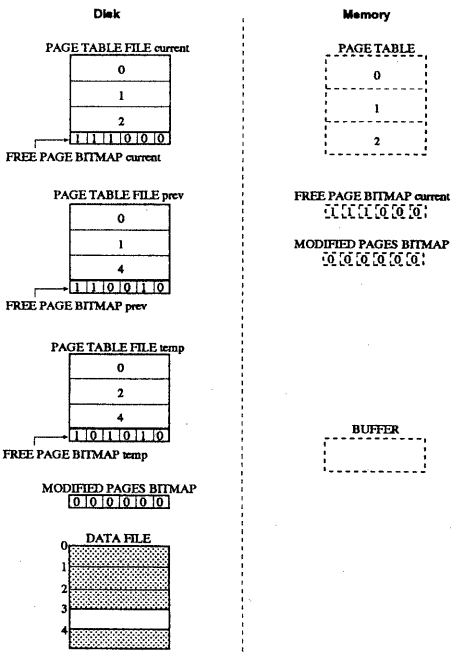

図 4.3: 更新前の状態

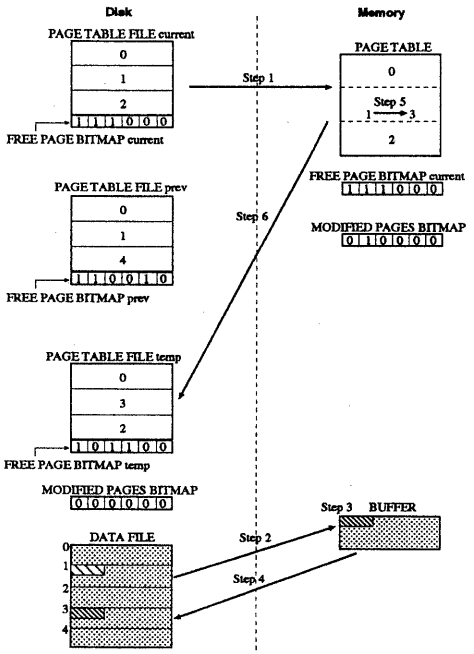

図 4.4: データファイルの更新処理

次に，対象とするファイルの空き領域を管理するために使用する FREE PAGE BITMAP について，実際に空き領域を管理する具体的な手順を以下に示す（図 4.5 参照）.

Step 1. ファイルオープン時に PAGE TABLE FILE current と prev の FREE PAGE BITMAP をメモリに読み込み，それらの OR をとったもの (OR BITMAP) をメモリ上に保持する.

Step 2. OR BITMAP からファイルの空き領域を判断し，メモリ上の PAGE TABLE を変更するときに用いる.

Step 3. メモリ上の PAGE TABLE を変更したら，それに対応する FREE PAGE BITMAP current を変更する.

以上の方法により，対象ファイルの空き領域を管理する.

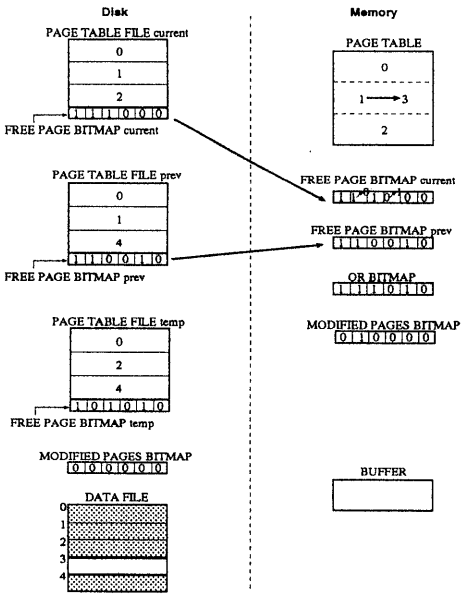

図 4.5: ビットマップファイルの処理

4.4 ファイルシステム操作関数

今回作成したファイルシステムを操作するための関数の一覧を表 4.1 に示す．通常 UNIX のシステムコールで提供されるものと同様な処理を行う関数以外に，ファイル操作を一つのトランザクションとしてとらえ，更新処理を有効にするか無効にするかの処理を行う関数 TrnCommit(), TrnAbort() を作成した．これにより，障害発生時のリカバリー処理がファイル単位で可能となった.

表 4.1: ファイルシステム操作関数

関数名	処理内容
FileCreat	対象ファイルの作成
FileOpen	対象ファイルのオープン
FileRead	対象ファイルからの読み込み
FileWrite	対象ファイルへの書き込み
FileClose	対象ファイルのクローズ
FileUnlink	対象ファイルの削除
TrnCommit	更新内容の確定
TrnAbort	更新内容の破棄

5 ファイルシステムのデータベースへの組み込み

前章で説明したファイルシステムを知的分散データベースへ適用し，障害からの回復を行うための機構について説明する．

5.1 従来のデータ更新処理

アプリケーションがデータベース中のデータを更新する場合，従来は質問処理オブジェクト (QPO) の qpmd メソッド内の fins(),edit(),dlt() といった処理を呼び出すことにより，それぞれ，ファイルからのデータの挿入，データの編集，データの削除といった処理を行っていた．また，これら3つの関数は，データオブジェクト (DTO) の dtmd メソッド内の inst(),mdfy(),dlt() といった処理を起動することにより，データファイルへのアクセスを行っていた（図 5.1 参照）．

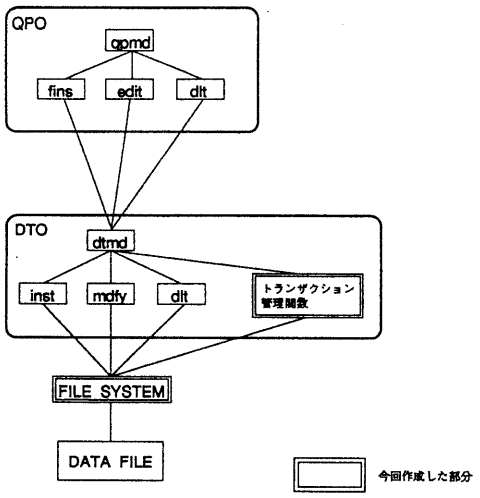

図 5.1: データベース内部の構成

5.2 ファイルシステムを用いたデータ更新処理

今回作成したファイルシステムを適用してデータベースの信頼性を確保するために，図 5.1 に示すようなトランザクション管理関数を DTO へ作成した．これは，データファイルへのアクセスをトランザクションとして管理し，更新操作の開始およびコミット，アボートといった処理を行うものである．このとき，QPO の fins(),edit(),dlt() といったデータ更新関数をトランザクションの単位とし，それぞれの関数にトランザクションの開始，コミット，アボートといった処理を追加した．これにより，QPO の関数単位で更新データのコミット，アボートが可能となった．

6 まとめと今後の課題

知的分散データベースのデータに対する高信頼化を実現するために，データ更新の無効化機能を有するファイルシステムを作成しそれをデータベースへ組み込んだ．組み込みに関しては，データの更新処理をトランザクションとして管理する関数を設け，それを用いることによりデータの整合性を維持できるようになった．

以上のように，以前に比べデータベースシステムの信頼性が高くなったのは確かであるが，このファイルシステムを組み込むことによるパフォーマンスの低下や，このファイルシステムを用いてもデータの整合性がとれなくなるような状況について，具体的な評価をまだ充分に行なっていないため定量的な効果を示すまでには至っていない．

したがって，これからの課題としては，このファイルシステムを用いたデータベースのテストを充分に行なって定量的な解析をするとともに，分散環境で複数トランザクションを扱うための並行処理制御などの機構について設計，実装していくことが必要である．

参考文献

[1] "特集 自律分散システム" 計測と制御 Vol.29, No.10, pp.877-952, 1990

[2] "特集 自律分散システムの新たなる展開" 計測と制御 Vol.32, No.10, pp.789-861, 1993

[3] 田村 信介 他 "知的分散システムのアーキテクチャ" 電気学会論文誌 Vol.108-C, No.6, 1988.

[4] 関 俊文 他 "知的分散 OS" 情報処理学会 マルチメディア研究会資料, 1988.

[5] 関 俊文 他 "オブジェクト指向分散システムにおける放送待機冗長処理方式" 電気学会論文誌 Vol.114-D, No.3, 1994.

[6] 原嶋 秀次 他 "知的分散データベースシステム" 情報処理学会 情報学基礎研究会資料, 1988.

[7] A.Howells,*et.al.* *"Partial Queries in a Decentralised Distributed Relational Database"* Proc. of Future Database '92, pp.97-105, 1992.

[8] K.Nagase,*et.al.* *"IDPS Database System"* Proc. of JWCC-6, pp.99-106, 1991.

[9] 永瀬 恵子 他 "知的分散データベースにおけるデータ管理方式" 情報処理学会 データベースシステム研究会資料, 1993.

[10] R.Elmasri and S.B.Navathe. *"Fundamentals of Database Systems"* The Benjamin/Cummings Publishing Company, Inc. 1989.

[11] 滝沢 誠 "データベースシステム入門技術解説" ソフトリサーチセンター, 1991.

[12] 宇田川 佳久 "オブジェクト指向データベース入門" ソフトリサーチセンター, 1992.