4C−3

オブジェクト指向データベースにおける設計トランザクションの実装方式

安村 義孝
NEC C&C 研究所

1 はじめに

分散環境上で行われるCADやCASE、グループウェアなどの協調作業を想定した場合、複数のユーザが様々なツールを用いてデータベース(DB)を共有して利用する。このような設計環境を支援するために、オブジェクト指向データベース管理システム(OODBMS)では設計トランザクションを提供することが不可欠となっている。設計トランザクションは従来の定型業務用のトランザクションとは異なった特徴を持つために、より高度な機能を持つトランザクションモデルが必要である[1]。

そこで、協調するトランザクションを階層構造で管理してこの階層構造を動的に再構築することができ、トランザクションの永続性を持つトランザクションモデルを提案する[5]。本稿では、実行効率や柔軟性等を考慮して、OODBMS Odin[4]にこのモデルに基づく設計トランザクションを導入するために、クライアント-サーバ方式のアーキテクチャに基づいた実装方式について述べる。

2 Odinのアーキテクチャ

Odinを利用するアプリケーションはOdin/C++を用いてコンパイルし、DB操作を行うためのOdinカーネルを構成するライブラリ群とリンクして作成される。Odinのアーキテクチャはクライアント-サーバ方式に基づいており、任意のアプリケーションがネットワーク環境上に分散された複数のDBへアクセスすることができる。リモートのDBが位置するサイト毎に各クライアントに対応するサーバを実行時に起動させる。また、複数のトランザクションによるDBアクセスの並行制御を行うために、デーモンプロセスであるCCM(Concurrency Control Manager)が各サイトに1つずつ存在する構成になっている(図1)。

サーバとクライアント間のデータの受渡しやDBに対するアクセスはページ単位で行われる。トランザクション内でオブジェクトへのアクセスが発生したら、そのオブジェクトが存在するDBとページをカーネル内で特定し、リモートDBの場合はサーバへ要求する。実際にDBにアクセスする前にCCMへページ要求を出す必要がある。CCMではサイト内の全DBの一貫性を保証するためにカーネルから出されたページ要求をスケジュールする。CCMから許可を取得後、DBからページをキャッシュへ格納する。アプリケーションはキャッシュ上に載っているページ内のオブジェクトを永続オブジェクトと意識しないで利用することができる。

3 設計トランザクション

協調作業を支援する設計トランザクションに要求される事項としては、中間状態のデータの受渡し、トランザクションのグループ化、長時間に渡る作業の支援、データの更新通知などが考えられる。これらの事項を満たし、個々のアプリケーションに柔軟に対応することができるだけではなく、実行効率も考慮する必要がある。

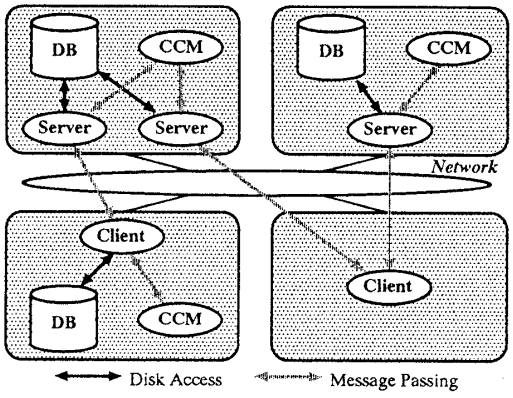

図1: Odinのアーキテクチャ

3.1 モデルの概要

大規模な設計プロジェクトはいくつかのチームに分けられ、設計者は個別の作業空間を持って各自に与えられた作業を行う。また、チーム内ではオブジェクトを共有して利用することも考えられる。そこで、協調作業を行うトランザクション(協調トランザクション)を階層構造で管理する。階層の根にはDB自身を表す仮想的なトランザクションが存在する。個々の協調トランザクションは利用中のオブジェクトの集合体であるオブジェクトイメージを持ち、子の協調トランザクションは親の協調トランザクションのオブジェクトイメージを経由してDB内のオブジェクトにアクセスする。つまり、階層構造の上位に位置するオブジェクトイメージが共有空間を提供するのである。これにより、協調トランザクションに個別のオブジェクトビューを持たせ、一種の作業空間を提供することができる。兄弟の協調トランザクションは親の階層で指定された正当性(並行制御プロトコル)に基づいて制御される。階層毎に異なる正当性を提供することで、アプリケーションに柔軟に対応することができる[2]。

この階層構造は、実行時に協調トランザクションに対して以下のような操作を施すことによって動的に再構築することができる。

1. **従属** (depend) … 親子関係の成立
2. **分離** (isolate) … 親子関係の断絶
3. **分割** (split) … 兄弟の生成
4. **併合** (merge) … 兄弟の結合
5. **共用** (share) … グループ化

この操作を用いることで、予めトランザクションの階層構造を構築しておく必要がなくなる。また、共用操作によりトップダ

A Design Transaction Implementation Method
for Object-Oriented Databases
Yoshitaka Yasumura
C&C Research Laboratories, NEC Corporation

ウンだけではなく、ボトムアップにも階層を構成することができる。

アプリケーションのプロセスに跨るトランザクション(長時間トランザクション)を実現するために、協調トランザクションに永続性を持たせる。協調トランザクションの実行を中断(suspend)することによって、そのオブジェクトイメージを凍結させる。アプリケーションを再起動した時に永続トランザクションを再開(resume)することによって、凍結されたオブジェクトイメージでオブジェクトにアクセスすることができるようになる。

信頼性の向上やDB操作のモジュール化のために、プロセス内の短時間トランザクションはそれ自身入れ子構造に構成することができる。協調トランザクションを利用するには、任意の短時間トランザクションをそれに束縛(bind)させる。束縛された短時間トランザクションは協調トランザクションの要素トランザクションとなる。そして、要素トランザクションの子の短時間トランザクションがコミットした時に、束縛されている協調トランザクションのオブジェクトイメージにオブジェクトの更新が反映されることになる。このように、協調トランザクションの階層と入れ子トランザクションの階層を融合して利用しているように見える。

3.2 実装方式

設計トランザクションの実装方式として、共有のための共用DBと個人用の私用DBを用意し、共用DBから私用DBにオブジェクトをチェックアウトして更新を行い、作業が終了したら共用DBにチェックインするという方式が提案されている[3]。しかし、この方式は柔軟性に欠け、頻繁にチェックアウト/チェックインする時には実行効率もよくない。このような欠点を克服するために、個々の協調トランザクションをそれを利用しているアプリケーションのサーバにマッピングする形で実現する。つまり、サーバ内のキャッシュが協調トランザクションのオブジェクトイメージと見なされるのである。協調トランザクションの階層構造の管理とそれに対する制御はCCMによって行われ、サーバとCCMが交信することによってアクセス制御を行う(図2)。

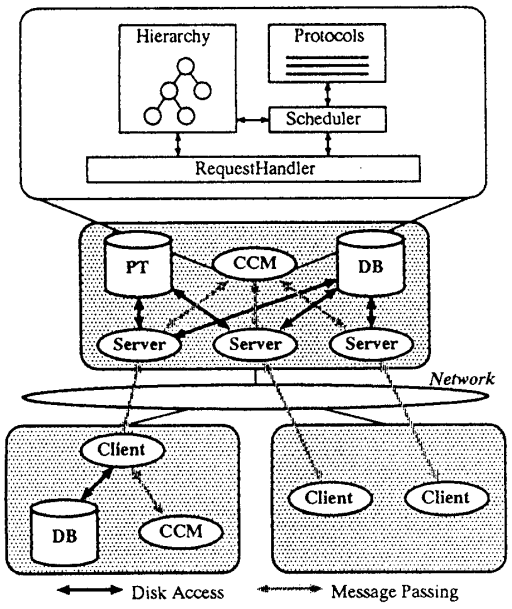

図2: 設計トランザクションの実装方式

協調トランザクションを永続化する場合は、サーバのキャッシュ内に存在するオブジェクトを永続トランザクション用のDB領域(PT)に格納する。永続トランザクションの下位に位置する協調トランザクションはCCMの指示によりこのPT内に存在するオブジェクトにアクセスすることもある。また、メッセージ通信のオーバヘッドを低減するために、ローカルサイトに予め用意しておいた個人専用のDB(私用DB)にオブジェクトを明示的にチェックアウトして格納することもできる。この私用DBはローカルサイトに存在するCCMによって制御される。

3.3 トランザクション管理

トランザクションの管理はCCMで一括して行う。CCMは次のようなモジュールで構成される。RequestHandlerはトランザクション操作、ページ要求、オブジェクトのチェックイン/チェックアウトなどのカーネルからのメッセージを受け付ける。Hierarchyはトランザクションの階層構造を管理しており、ページのロックやバージョン情報、待ち行列等を個々のノードで保持する。Schedulerは階層毎に指定された並行制御プロトコルに従ったトランザクションのアクセス制御を行う。この並行制御プロトコルはProtocols内で管理されている。

RequestHandlerがページ要求を受け付けた時にはSchedulerに制御を渡す。Schedulerはそのトランザクションの上位の階層に指定された並行制御プロトコル(Protocols内にある)に対する妥当性を検査する。これによりページ要求が受理される場合は、その協調トランザクションからHierarchy内の階層を上位の方向にたどり、適切なサーバ(またはPT)を選択して要求元に返答を返す。ページ要求を受理された協調トランザクションはそのサーバ(PT)にアクセスしてページを取得する。このようにして、協調トランザクションのオブジェクトビューを実現する。

4 おわりに

本稿では、OODBMS Odinに協調作業を支援するための設計トランザクションを実装する方式について述べた。従来のような共用DBと私用DBを用意する方式とは違い、実行効率や柔軟性を考慮してクライアント-サーバ方式に組み合わせている。このため、設計トランザクションを利用しない通常のアプリケーションと同等の実行効率が得られる、という特徴がある。今後はOODBMSの並行制御と絡めた協調作業のための正当性についてさらに検討を重ね、実装を進めていく予定である。

謝辞

本研究を含めOdinの研究開発に関して、日頃から議論や助言をして頂いているNEC C&C研究所 鶴岡邦敏課長および木村裕主任に感謝致します。

参考文献

[1] Barghouti, N. S. and Kaiser, G. E.: Concurrency Control in Advanced Database Applications, *ACM Comput. Surv.*, Vol. 23, No. 3, pp. 269–317, 1991.

[2] Heiler, S., Haradhvala, S., Zdonik, S., Blaustein, B. and Rosenthal, A.: A Flexible Framework for Transaction Management in Engineering Environments, in *Database Transaction Models for Advanced Applications*, Elmagarmid, A. K. eds., Morgan Kaufmann, pp. 87–121, 1991.

[3] Kim, W., Ballou, N., Garza, J. F. and Woelk, D.: A Distributed Object-Oriented Database System Supporting Shared and Private Databases, *ACM Trans. Inf. Syst.*, Vol. 9, No. 1, pp. 31–51, 1991.

[4] 木村裕, 鶴岡邦敏, 波内みさ: Odinデータベースシステムのアーキテクチャと言語, アドバンスト・データベースシステム・シンポジウム'92講演会資料, pp. 63–72, 1992.

[5] 安村義孝: 協調作業のための柔軟なトランザクション処理モデル, 日本ソフトウエア科学会第10回大会論文集, pp. 65–68, 1993.