CORBA 環境下における OODBMS の利用

3 X − 7

深津 博樹　田幡 勝　有次 正義　金森 吉成
群馬大学工学部情報工学科

1 はじめに

ネットワーク上に分散された複数のオブジェクト指向データベースを遠隔から利用することは難しい問題である．DB を利用するユーザは，各々の DB の位置や，その実装方法，および DB を管理している DBMS の仕様等を知っていなければならない．

そのような問題を解決するために CORBA[1] の技術を用いる．CORBA では IDL(Interface Definition Language) で記述されたインタフェースを介して，クライアントがサーバプロセス中の CORBA オブジェクトへ処理を依頼するリモートオペレーション呼び出しを行う．このとき，クライアントは依頼をする CORBA オブジェクトの位置や内部の実装方法を知る必要はない．DB 中に存在するオブジェクトは，CORBA オブジェクトではないため，そのオブジェクトに対して，直接リモートオペレーション呼び出しをすることはできない．そこで，CORBA で仕様が定まっている TIE アプローチを用いて，IDL インタフェースと DB に実装されているオブジェクトを連結 (*tie*) する CORBA オブジェクトを生成する．

既存の DB ごとに実装クラスが異なる場合には，それぞれのクラス定義に対応するインタフェースを記述しなければならない．したがって，クライアントは実装クラスごとに異なるインタフェースの異なるオペレーション呼び出しを行うことになる．本論文では，DB が複数，かつ分散されて存在しているときにその内部の実装に関係なく，ユーザは単一のインタフェースを用いるだけで DB を利用できる機構を提案する．具体的環境として CORBA 準拠な Orbix[2] と，オブジェクト指向データベースである ObjectStore[3] を統合することによって，これを実現する．

2 分散環境での永続オブジェクトの利用

2.1 永続オブジェクトへの IDL インタフェースの適用

IDL インタフェースを用いて既存のデータベース中のオブジェクトを利用するための構成を図 1に示す．なお，例としてデータベース中のオブジェクトは，属性に画素値データを持ち，メソッドとしてその画素値データを返すような画像オブジェクト[5] を用いる．

- インタフェース ImageData
  クライアントはこのインタフェースに記述されているオペレーションのみを使用することによって，サーバ側の CORBA オブジェクトにメッセージを送ることができる．
- クラス A_ImageData, B_ImageData
  実装クラスとインタフェースを連結する CORBA オブジェクトのクラス．
- クラス a_ImageData, b_ImageData
  データベース中に存在する画像オブジェクトの実装クラス．

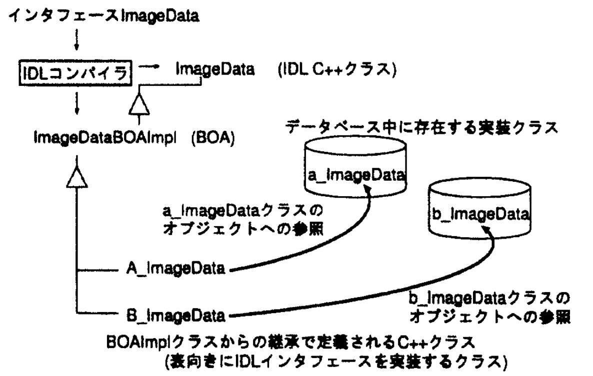

図 1: 永続オブジェクトへの IDL インタフェースの適用

実装クラス a_ImageData, b_ImageData が異なるデータベース中にすでに存在しており，このクラスが IDL インタフェースの各機能を実現する．実装クラスと IDL インタフェースを連結する A_ImageData, B_ImageData クラスを BOA(basic object adapter) から継承するクラスとして定義することにより TIE オブジェクトの役割を果たすようにする．クライアントにとっては A_ImageData, B_ImageData クラスが IDL インタフェースを実装しているクラスとして見える．A_ImageData, B_ImageData クラスのメンバとして，それぞれ a_ImageData, b_ImageData クラスのオブジェクトへの参照を持たせることにより，リモートオペレーション呼び出しに対応した処理を行う．

クライアント側では IDL インタフェースに記述されているオペレーションのみを使用して，サーバ側の CORBA オブジェクトへメッセージを送り，その結果を得ることが可能である．したがって，サーバ側の実装がどうなっているのかを知る必要がない．故に，サーバ側で複数の実装が施されていても，クライアント側ではその違いを意識すること無くプログラミングすることができる．

Utilizing OODBMSs under a CORBA Environment
Hiroki Fukatsu, Masaru Tabata, Masayoshi Aritsugi, Yoshinari Kanamori
Department of Computer Science, Gunma University

2.2　永続オブジェクトの活性化の手順

クライアントからリモートオペレーション呼び出しが行われたとき，呼び出しのターゲットとなるCORBAオブジェクトに対して，呼び出されたオペレーションが実行される．しかし，CORBAオブジェクトがサーバプロセス中に存在しない場合には，データベース中の画像オブジェクトへの参照をメンバとして持つようなCORBAオブジェクトを生成しなければならない．これを実現するために，Orbixで提供されているloaderオブジェクトを用いる[4]．クライアントから要求のあったCORBAオブジェクトが，プロセス中に存在しないときには，オブジェクトフォルトがloaderオブジェクトに通知される．これにより，loaderオブジェクトは，データベース中の画像オブジェクトを活性化し，そのオブジェクトをメンバとして持つようなCORBAオブジェクトを生成する．図2に活性化の手順を示す．

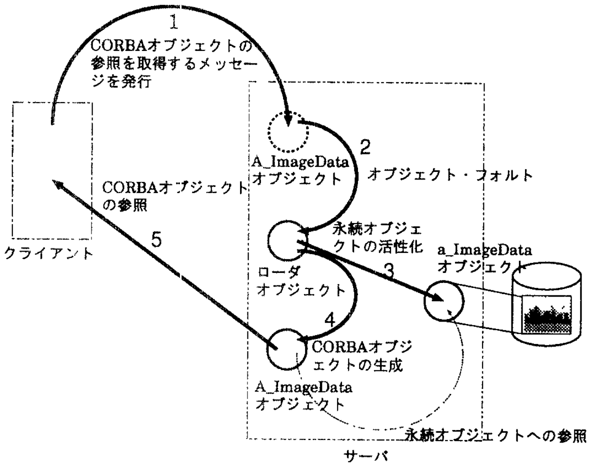

図 2: 永続オブジェクトの活性化の手順

i) クライアントは，リモートオペレーション呼び出しのターゲットとなるCORBAオブジェクトへの参照を取得するメッセージを発行する．
ii) クライアントから要求のあったCORBAオブジェクトが，サーバプロセス中に存在しないため，loaderオブジェクトに対してオブジェクトフォルトが通知される．
iii) loaderオブジェクトは，要求のあった画像オブジェクトをデータベースから活性化する．
iv) 活性化された画像オブジェクトへの参照をメンバとして持つようなCORBAオブジェクトを生成する．
v) サーバはクライアントに対して，CORBAオブジェクトへの参照を返す．この時，クライアントはii, iii，ivの処理がサーバ側で行われていることを意識する必要はない．

3　トランザクション管理

サーバ側の永続オブジェクトに対するオペレーションは全てデータベーストランザクションの中で行われなければならない．なぜならOODBMSのみが，トランザクション中で，永続領域にアクセス可能であるからである．したがって，クライアント側からのすべてのオペレーション呼び出しに対して，トランザクションの開始と終了を示す必要があるが，これをオペレーションの開始前，及び終了後に示さなければならない．それには，Orbixで提供されているfilterオブジェクトを用いる[4]．プロセス単位フィルタで定義されている関数の内，サーバ側にリクエストが着信する前に実行する操作であるinRequestPreMarshal()中でトランザクションを開始する．また，サーバ側から応答が発信する前に実行する操作であるoutReplyPreMarshal()中でトランザクションを終了する．

上記の方法でトランザクションを管理した場合，1つのトランザクション中に1つのオペレーションのみ実行が可能となる．ところが，現実には1つのトランザクション中で複数のオペレーションを実行することが望ましい場合もある．また，連続的に行いたいオペレーション間で，他のクライアントによって別のオペレーションが挿入される恐れもある．それには，サーバ側のプロセスをクライアントごとに1つずつ割り当て，DBMSのロック機構を用いることで対処できる．

4　おわりに

分散環境で構築されている複数のオブジェクト指向データベースを，CORBAの技術を用いて遠隔から利用する機構を実現した．ユーザは，データベースの実装方法およびデータベースがネットワーク上のどこに存在するのかを意識する必要なく，単一のインタフェースを用いるだけで使用することができることを示した．

謝辞

本研究の一部は，文部省科学研究費補助金重点領域研究(課題番号08244101)による．

参考文献

[1] *“The Common Object Request Broker : Architecture and Specification Revised Edition”*. Object Management Group, Inc., July 1995.
[2] IONA Technologies Ltd. *“Orbix 2 Programming Guide”*, October 1996.
[3] Object Design. *“ObjectStore C++ API User Guide Relese 4.0.2”*, July 1996.
[4] F. Reverbel. “Persistence in distributed object systems: ORB/ODBMS integration”. *A Sunrise/TeleMed Progress Report.*
[5] 田幡勝，深津博樹，有次正義，金森吉成．“分散環境における画像オブジェクトの版管理の設計”．第8回データ工学ワークショップ(DEWS’97), pp. 179–184, March 1997.