マルチメディア通信と分散処理 43 － 2
（１９８９．９．２８）

OSI RDAプロトコルの実装

西山 智† 小花 貞夫† 杉山 敬三† 堀内 浩規‡ 鈴木 健二†
†国際電信電話株式会社
‡(株)オーエスアイ・プラス

筆者らはOSIに基づくデータベースアクセスプロトコルである、OSI RDA(遠隔データベース・アクセス)の実装を行った。本実装では、対象とするデータモデルとしてリレーショナル型(RDB)を想定し、RDBの標準データ操作言語であるSQLによる遠隔データベース・アクセスを実現することとした。このために、RDAで要求する機能をRDBの機能に対応付けた。また、応用エンティティの実現では、筆者らが提案するALS(応用層構造)の概念に従った汎用的なプロセス構成とした。また、RDAの応用として分散したデータベースに対してデータの分散を意識しないで検索を可能とする分散データベースアクセスのアプリケーションを実現した。本稿ではこのOSI RDAソフトウェアの構成および機能分担について報告するとともに、実現したRDA機能について考察検討する。

Implementation of OSI RDA Protocol

Satoshi NISHIYAMA†, Sadao OBANA†, Keizo SUGIYAMA†,
Hiroki HORIUCHI‡ and Kenji SUZUKI†
†KDD Kamifukuoka R & D Labs. 2-1-15, Ohara, Kamifukuoka, Saitama, 356
‡OSI Plus Corp. 2-1-23, Nakameguro, Meguko-ku, Tokyo, 153

In this paper, an implementation methodology for the OSI RDA (Remote Database Access) and its application are discussed. In this methodology, in order to implement RDA SQL specialization, a effective mapping between the concepts used in RDA and those in RDB (Relational Database Management System) is proposed and each function needed for a RDA sever is realized by a combination of RDA sever software and functions provided by RDB. Furthermore, to realize an AEI (Application Entity Invokation) effectively, the software of RDA ASE is implemented according to the ALS (Application Layer Structure) concept. A software which supports database accesses for distributed databases is also developed as the application of OSI RDA.

1. はじめに

近年、データベース機能は情報処理システムに必須の機能となった。この結果、他のシステムに遠隔ログインしたり、ファイル転送を行うのと同様に、他のシステムに存在するデータベースをアクセスしたいという要求が増加している。そこでOSI(開放型システム間相互接続)においても、OSI RDA(遠隔データベース･アクセス)の標準化を進めている。データベースの大規模化に伴い、今後、ネットワーク上に分散した分散データベースの構築技術が重要になると考えられるが、OSI RDAはOSI環境下で分散データベースを構築するための重要な要素となる。このため、筆者らはOSI RDAソフトウェアの実装を行った。

本実装では、RDAで実際に転送するデータベース操作言語として、RDA専化の標準化の進んでいるSQLを使用した。またAEIの実現の際には、ALS(応用層構造)に基づいて筆者らが提唱している汎用的な応用層のプロセス構成[7]を使用した。さらに、RDAを用いて、分散データベース･アクセスのアプリケーションを実現した。本稿では、このOSI RDA実装のソフトウェア構成および機能分担について報告するとともに、実現した機能について検討･考察を行う。

2. RDA

RDAはOSI環境下において遠隔からデータベースをアクセスするためのサービスとプロトコルを規定する。論理的にはRDAは図1に示すようにRDAサービスを要求するクライアントとそのサービスを提供するサーバからなる。具体的には、図2に示すようにクライアントはそのプロトコルを実現するOSI応用層のAEI(応用エンティティ･インボケーション)と、そのサービスを利用するクライアント･プロセスからなる。またサーバはAEIと、データベース管理システム(DBMS)と接続するサーバ･プロセスからなる。

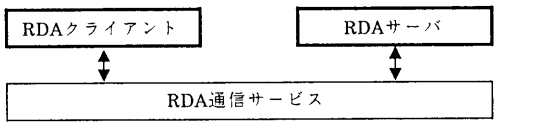

図1 RDAクライアント･サーバ･モデル

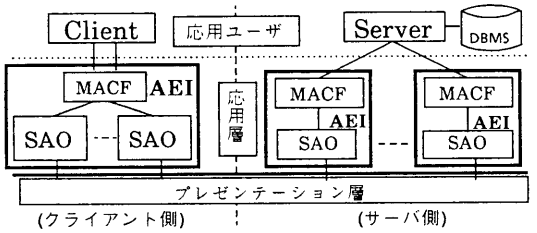

図2 RDAの応用層のモデル

クライアントがサーバにアクセスする際には、ダイアログと呼ばれる接続関係がサーバとクライアントの間に確立される。ダイアログが確立している間はクライアントはサーバのデータベースにアクセスすることができる。

また、データベースにアクセスする際必要となるトランザクション機能を提供するために、RDAは①基本コンテクスト、②CCRコンテクスト、③TP(トランザクション処理)コンテクストの3種類の応用コンテクストを規定している。①は応用層の下位サービスとしてACSE、ROSEのみを使用し、トランザクション機能を提供しない。また、②はACSE、ROSEの他にCCRASEを使用してトランザクション機能を提供する。また③はCCRに加えてTPを使用してトランザクション機能を提供する。これらの選択は、応用アソシエーション確立時に、応用コンテクストの折衝により決定される。RDA自身は特定のデータベース操作言語に依存しないが、実際の通信に際しては特定のデータベース操作言語に専化したものが用いられる。現在はリレーショナル型(RDB)のデータベース操作言語であるSQLが専化として標準化が進められている。

3. 基本設計

3.1 実装の基本方針

RDAの実装は以下の方針で行うこととした。

1) RDA標準草案(DP9579)[1]に準拠する。
2) サーバのデータベースとしてRDBを使用し、RDAで転送するデータベース操作言語として、SQLを使用する。SQLの詳細な文法については、RDAのSQL専化に関する標準草案[2]及びISOのSQL標準[3]に従う。
3) AEIの構成は、応用層構造(ALS)に基づき筆者らが定めた応用層の実装方針[7]に従う。
4) CCRASEを使用し、CCRサービスによるトランザクション機能を提供する[10]。
5) サーバは複数のクライアントに対してサービスを提供可能とする。
6) 今回、RDAの応用として分散データベース･アクセスのアプリケーションを作成する[5][6]。
7) 利用者インタフェースとして、端末利用者に対するSQL言語インタフェースを提供する。
8) VAX8700(OS:VMS)上に実装し、既存のOSIプレゼンテーション、ACSE、ROSEなどのソフトウェアを利用する。また、実際のDBMSとしてORACLEを使用する。

3.2 RDAとRDBとの対応

RDAの実装では、実際のDBMS(ここではRDB)との間の対応関係を明確にする必要がある。

1) RDAサーバにおける複数利用者のサポート

RDAサーバでは、複数の利用者をサポートする必要が生じるが、一般にRDBでは1つのプロセスから複数の操作を非同期に実行させることができない。このため、複数利用者をサポートするには、プロセスを複数生成する必要があるが、単にRDAサーバを複数生成するとそれらが占有するプロセス空間が大きくなる。このためRDBにインタフェースする部分のみ別プロセスとして複数生成

することとした。このプロセスは、あらかじめ一定数を生成しておき、RDAサーバが管理･割当てを行う方法と、利用者のRDAサーバへのダイアログ生成に対応して動的に生成する方法が考えられる。後者の方がダイアログの生成･消滅と明確に対応し管理が容易なため、後者の方法を採用した。

2) 資源と利用者の扱い

RDAでは資源管理の単位として、データベースとして意味をなすデータの集合を想定しているのに対し、RDBでは各テーブルや行が資源管理の単位である。RDBでRDAの想定している資源に近い概念は各利用者の所有しているテーブルや行の集合である。従って、各利用者の所有するテーブル群を利用者単位でRDAの資源とみなすこととした。すなわち、資源名はRDB利用者に対応する。

利用者については、RDAの利用者とRDBの利用者が1対1に対応すると仮定し、そのマッピングはRDAのサーバ側で行う。

3) データベース機能および操作言語

RDAのSQL専化では、転送されるデータベース操作言語としてISO標準のSQLを使用する。実際の個々のRDBのSQLは、詳細においてISOのSQLと仕様が若干異なる場合がある。このため、これらの間の変換が必要となる。この変換をRDAサーバで実施すると他のRDBを使用する場合に移植性が減少するため、これらの変換はRDBとインタフェースするプロセスで実施することとした。

3.3 RDAアプリケーションとしての分散データベースアクセスの機能[5][6]

今回、RDA実装のアプリケーションとして分散した複数のデータベースに対して、利用者がデータの分散を意識しないで検索を行えるメカニズムを実現することとした(データの分散透過性の提供)。このため各分散処理ノードは他のノードの持つ資源に関する知識を持っており、分散処理が必要になった場合は利用者からの操作を受けたノードが操作分割･結果合成を行う、図3に示すようなモデルを作成した。各ノードは対等の機能を持つと仮定する場合、各ノードはRDAの論理モデル上クライアントとサーバの両方の機能を持ち、さらに利用者に対してデータの透過性を提供するために、操作の分割および結果の合成の機能が必要となる。

また、分散処理のために各ノードは他のノードのデータベースに関する知識を持っていることが必要になる。このため、今回のアプリケーションでは各データベースのスキーマは変更されないことを仮定している。

4. RDAのソフトウェア構成

3節の議論をもとに実装したソフトウェアの構成を図4に示す。ソフトウェアは大きく①RDAサーバ/クライアント、②AEI、③利用者インタフェース、④プレゼンテーション層以下の下位プロトコルの4つの部分に分けられる。

4.1 RDAサーバ/クライアント

RDAサーバ/クライアント部は、RDAサーバ/クライアント機能を提供するRDAプロセスと複

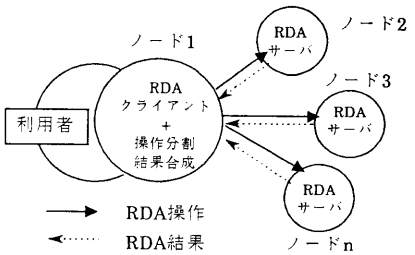

図3 RDAを用いた分散データベース･アクセスのモデル

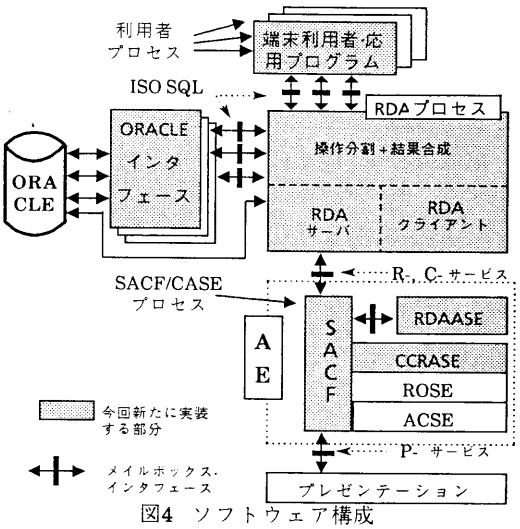

図4 ソフトウェア構成

数のRDBにインタフェースするプロセス(ここでは、ORACLEを使用するのでORACLEインタフェースプロセスと呼ぶ)からなる。

4.2 AEI

AEIはSACF/CASEプロセスとRDAASEプロセスからなる。AEIの実現にあたっては、[7]に基づいて、SACFの機能を実現するモジュールを明示的に導入した[8]。また今回、MACFについては明示的にプロセスを導入せず、必要な複数アソシエーション制御機能は応用プロセス(RDAプロセス)で実現することとした。

実装の基本方針4)により、CCRを用いてトランザクション機能を実現するため、AEIはSACFの他に、ACSE、ROSE、CCRASE及びRDAASEのモジュールからなる。これらの内、ACSEとROSEについては、既に作成したモジュール[12][13]を使用した。また今回の実装では、SACFモジュール導入に伴い、プロセス間のインタラクションでのオーバヘッドが増加しないように、比較的共通に使用されるASE(ACSE、ROSEとCCRASE)についてはSACFとまとめて1プロセスとして実現した。

4.3 利用者インタフェース

今回実現した分散データベースアクセスのアプリケーションを使用する利用者として、端末利用者およびプログラムを想定した。今回特に、端末

利用者のためのSQLを提供する端末インタフェースプログラムを作成した。

4.4下位プロトコル

プレゼンテーション層以下の下位プロトコルは筆者らが既に他のOSIプロトコルの実装で作成したソフトウェアを使用した。但し、プロセス間通信に使用しているキューインタフェースであるVMSのメイルボックス･サービスは1メッセージ長に制限がある。OSIディレクトリと同様、RDAについても応用PDUのサイズとして大きなものが想定されるため、セション層以上については、1PDUの受渡しを複数回のメイルボックス･メッセージで実現可能とし、大きなサイズのPDUに対応させた。

5. AEIの実現

5.1 RDAASE

RDAASEは、R-サービスプリミティブとA-及びRO-サービスプリミティブとのマッピングを行い、RDAのプロトコル機械を実現する。SACF/CASEプロセスとは、メールボックスを介してインタフェースする。

5.1.1 例外的PDUの扱い

RDAでは、ROSEからRO-Reject-U又はRO-Reject-Pといった例外的なPDUを受けた場合の処理が記述されていない。そこでこれらのPDUについては全て対応するR-サービスプリミティブを新たに定義し、RDAASEでマッピングを行うこととした。また、あわせてACSEの例外的なプリミティブについてもRDAASEでR-プリミティブに形式上マッピングし、RDAプロセスとSACF間のインタフェースをR-とC-サービスプリミティブに統一した。すなわち、RO-Reject-U/PはR-Reject-U/Pにマッピングされ、またA-P-Abort/A-AbortはRDAASEによって、R-P-Abort及びR-Abortにマッピングされ、RDAプロセスに通知される。RDAプロセスがこれらのプリミティブを受信した場合の処理については、6.3節で述べる。

5.1.2 R-Cancel

既に発行した操作に対する中止操作、R-Cancelを新しくプロトコルに追加した。RDAの実装ではRDAASEがROSEの直接のサービス利用者となる。ROSEのサービスを使用する際に付与する必要のあるインボークIDは、一般には直接のサービス利用者が管理するのが簡便である。しかし本実装では、RDAプロセスがR-Cancelでの中止対象操作の指定およびR-Reject-U/Pの処理の際にインボークIDを知っている必要があるため、RDAASEではなくRDAプロセスがインボークIDの管理を行っている。RDAASEはSACFを経由して、R-サービスプリミティブと共にインボークIDをRDAプロセスから通知される。

以上より、RDAASEで扱うサービスプリミティブとそのマッピングは表1のようになる。

5.2 CCRASE

CCRASEはCCRのサービスプリミティブ(C-プリミティブ)とプレゼンテーションのサービスプリミティブ(P-プリミティブ)のマッピングを行い、CCRのプロトコル機械を実現する。CCRASEの実装の詳細については、[11]を参照のこと。

R-サービスプリミティブ	マッピングされる下位サービスプリミティブ
R-Associate	A-Associate
R-Release	A-Release
R-Abort	A-Abort
R-P-Abort	A-P-Abort
R-Open R-Close R-InvokeDBL R-ExecuteDBL R-DefineDBL R-DropDBL R-Cancel	RO-Invoke
R-Reject-U	RO-Reject-U
R-Reject-P	RO-Reject-P

表1 RDAASEでのマッピング

5.3 SACF

SACFはプレゼンテーション層および応用プロセス(今回はRDAプロセス)から渡されるサービスプリミティブを適切に各ASEに振り分けを行う。また必要に応じて、応用アソシエーションの状態等を各ASEに通知する[7][8]。

6. RDAプロセス

RDAプロセスは、今回実装したアプリケーション実現のためにRDAクライアント/サーバの両方の役割を果たす。利用者プロセスから操作要求を受けた場合、資源に関する知識をもとに要求された操作を解析し、ローカルなデータベースにアクセスあるいは、RDAクライアントとして他のデータベースにアクセスし、それらの結果を合成して、利用者プロセスに返す。また、RDAプロトコルによる操作要求を受けた場合、通常のRDAサーバとして動作する。

6.1 RDAクライアントとしての動作

利用者プロセスからの操作要求を受けた場合、RDAプロセスはデータの位置透過性を提供するための分散処理を行うとともに、RDAクライアントとして他のRDAサーバにアクセスする。図5及び図6にRDAクライアントとしてのRDAプロセスの処理の流れを示す。

6.1.1 利用者の認証

利用者プロセスはRDAプロセスにアクセスを開始する際に、分散データベースアクセスとしての利用者の認証を行う。この利用者名はそのままRDAの利用者名としても使用される。認証が完了すると、図7に示すトランザクション管理テーブル(TMT)が生成される。また、ORACLEインタフェースプロセスを1つ生成し、ローカルDBにその利用者としてログインする。

6.1.2 トランザクションの開始

もしトランザクションを開始する場合、利用者プロセスは必要な資源をトランザクション開始時

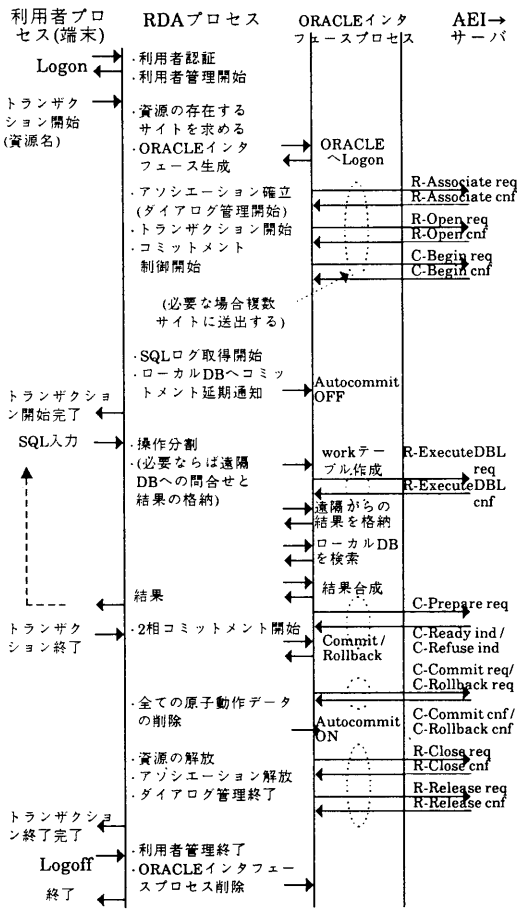

図5　クライアント側の処理(トランザクションあり)

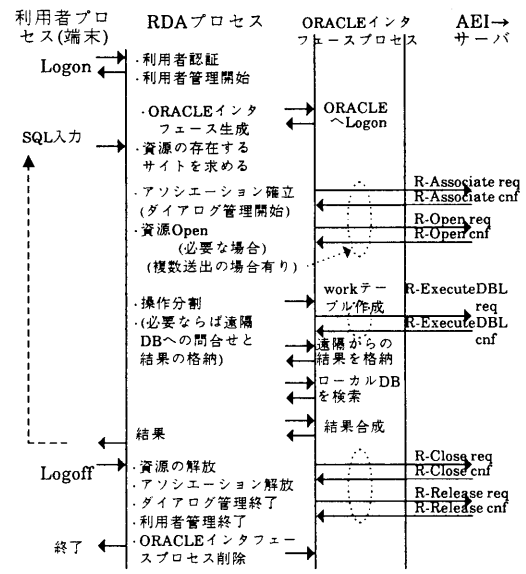

図6　クライアント側の処理(トランザクションなし)

に指定する。RDAプロセスはこの指定をもとに、資源が存在する各ノードに対して、アソシエーション設定、資源のオープンおよびトランザクション開始の指示を行う。各ノードに対するアソシエーションやトランザクションの状態、あるいはRDAサーバとのダイアログ識別子は図7に示すアソシエーション管理テーブル(AMT)に設定される。RDAプロセスはTMT中のアソシエーション総数と確立済みアソシエーション数やCCRプリミティブ受信数を比較することにより全てのアソシエーションの処理が完了したかどうかを判断し、次の処理に移る。

6.1.3　SQLの処理

利用者プロセスからのSQLはSQLをASN.1で符号化したものをプリミティブ形式にしたフォーマットでRDAプロセスに渡される。SQLを受信すると、RDAプロセスはどのノードに資源が存在するかを解析する。トランザクション中の場合、オープンする資源をトランザクション開始時に明示しているため、SQL操作実行時には全てのアソシエーション設定と資源のオープンは完了しているが、トランザクション中でない場合、必要に応じてアソシエーション確立や資源のオープンを行うこととしたため、必要なノードに対してアソシエーション確立や資源のオープンがされていない場合がある。この場合各AMTのアソシエーション状態から判断し、これらの処理を行う。

各資源のオープン終了後、各ノードに対してR-ExecuteDBLを送出してその結果をORACLEインタフェースプロセスを介してローカルなDBに格納する。また、ローカルなDBに対する操作を実行する。個々のノードに対する全ての結果が得られた後ローカルDBの機能を用いて最終的な結果の合成を行い、結果を利用者プロセスに返す。

なお、結果を利用者プロセスに返送する際には、結果に加えて、カラム名などの結果に対するスキーマ情報を分散資源に対する知識から得て、あわせて返すこととした。利用者プロセスではこれらの情報を知らないため、結果を表示するためにこれらの情報が必要となるためである。

6.1.4　トランザクションの終了と資源の解放

トランザクションを行っている場合、利用者プロセスからのトランザクション終了指示により、CCR2相コミットメントの2相目を開始する。各ノードに対してC-Prepareを送出し、結果をもとにC-CommitあるいはC-Rollbackを送出しトランザクションを終了させる。また、資源の解放とアソシエーションの解放(ダイアログの消滅)を行う。ダイアログが消滅した場合、対応するAMTも解放される。但し、トランザクションでない場合、資源とダイアログの解放は利用者プロセスのログオフ時に行われる。

6.1.5　利用者プロセスのログオフ

利用者プロセスのログオフによりTMTを解放するとともに、ローカルDBからログオフしORACLEインタフェースプロセスを削除する。

6.2 RDAサーバとしての動作

AEIからR-Associate指示を受けた場合、RDAプロセスは通常のRDAサーバとして動作する。サーバ側動作の場合の処理の流れを図8に示す。

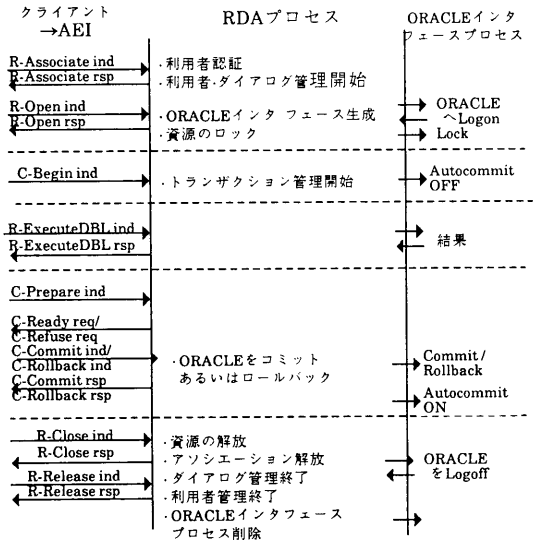

図8 サーバ側の処理

6.2.1 ダイアログ生成および資源の取得

R-Associate指示を受信するとRDAサーバは利用者の認証を行った後、図9に示すAMTとTMTを生成する。クライアント動作の時と異なり1トランザクションに対するアソシエーションの数が1つに限られるため、クライアント動作時のテーブルを単純化したものを使用している。また、ORACLEインタフェースプロセスの生成を行う。

その後R-Open指示を受信すると、R-Openで指定された資源名を用いてORACLEインタフェースプロセス経由でORACLEにログインする。R-Openで指定されている資源の排他制御については、ORACLEの排他制御がRDAと異なるため、ORACLEの機能にまかせることができない。しかしORACLEにローカルな利用者がいることを考慮し、①RDAサーバとしての排他制御はRDAプロセスで行い、②ORACLEでも排他制御を行いローカルな利用者への対処を行うこととした。①については、RDA標準案に準拠した制御を行っている。また②については、ISOのSQLでは排他制御のためのSQL文が定義されておらず、RDAプロセスとORACLEインタフェースプロセス間のインタフェースにORACLEに準拠したlock文を追加し、この文を使用することにより排他制御を行っている。この際、RDAの排他制御の状態とORACLEのそれとが一致しないため、表2のようなマッピングを行っている。ローカル利用者が既に資源を専有しており、ORACLEからlock失敗が通知された場合、RDAプロセスは通常の排他制御の場合と同様にクライアントにエラーを通知する。

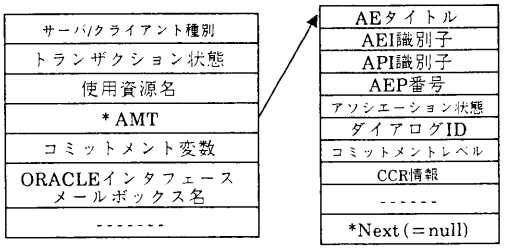

図9 RDAサーバ動作時のAMTとTMT

RDAの状態	ORACLEの状態
shared retrieval	(対応するものはない)
shared update	(対応するものはない)
protected retrival	SHARE
protected update	EXCLUSIVE
exclusive	EXCLUSIVE(*)

*：ORACLEのEXCLUSIVEは他の利用者が検索を行えるためローカル利用者を完全には排除できない。

表2 ローカル利用者に対する資源の排他制御

6.2.2 トランザクション管理

CCRユーザとして要求されるコミットメント機能及び回復制御機能はデータベースのコミットメント/ロールバック機能及び障害回復機能を用いて実現する。すなわち、C-Prepareによりトランザクションの開始が指示された場合、ORACLEの自動コミットメント機能を停止することでトランザクション状態とする。また、C-Commit及びC-Rollbackはそれぞれ SQL のcommit work 文及び

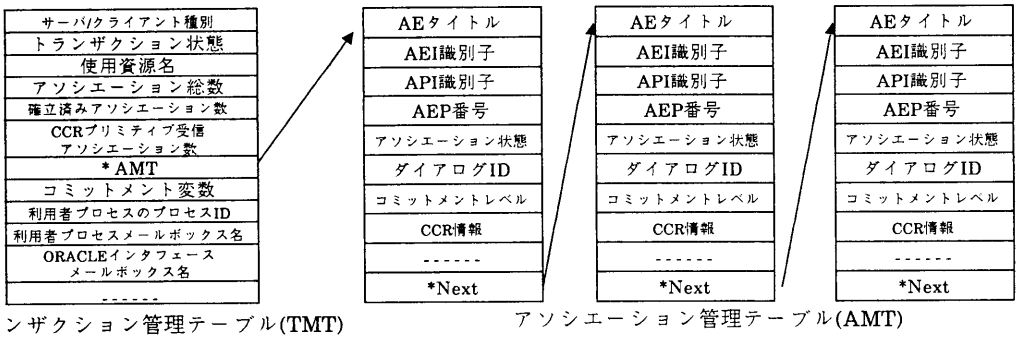

図7 RDAクライアント動作時のアソシエーション管理テーブルとトランザクション管理テーブル

rollback work文にマッピングされる。さらに、ORACLEの自動コミット機能の回復が行われる。

また、障害時からの回復についてはトランザクション開始後は、実行したSQL文等回復に必要な情報を通常の外部ファイルにログとして記録することとした。障害後、ローカルなDBMSの回復機能によりトランザクション初期の状態に戻した後、このログを使用して必要な状態に復帰させる。

6.2.3 データ操作実行

データ操作要求(R-ExecuteDBL)は、トランザクション状態や資源状態からその操作の正当性を検証し、SQLに変換してORACLEインタフェースプロセスに送り、実行させる。また、操作定義要求(R-DefineDBL)については、SQLに操作定義を行う機能がないのでRDAプロセス内にDBL定義として記憶し、R-InvokeDBL要求時にORACLEインタフェースプロセスに転送し、実行させる。データ操作定義の際には、実際に対応するテーブルが存在するかどうかといった操作の正当性についてはチェックを行っていない。

6.2.4 ダイアログ終了

ダイアログ消滅時に対応するORACLEインタフェースプロセスをORACLEからlogoutさせた後に消滅させる。

6.3通信障害に対する対処

アソシエーション切断等の通信が発生した場合、RDAプロセスは以下の処理を行う。

1)R-Reject-U/P受信

RDAプロセスがサーバ動作の場合、R-Reject-U/Pを受信しすると、一度クライアントにR-Abortを送信し、クライアント側からのアソシエーションの再確立を待つ。RDAプロセスがクライアント動作の場合、①プリミティブ中のパラメータとして示されているROSのインボークIDが不明な場合、利用者プロセスに対してエラーを通知するとともに、R-Abortをサーバに送信する。トランザクション中の場合は、その後CCRサービスにより障害回復処理が行われる。②インボークIDが確認できる場合には対応する操作のエラーとして利用者プロセスに通知する。

2) R-P-Abort/R-Abort受信

RDAプロセスがサーバ動作の場合、R-P-Abort/R-Abortを受信すると、そのアソシエーションによってサポートされていたダイアログにたいしてタイマをかける。一定時間以内にそのダイアログを支援するアソシエーションが設定されない場合、ダイアログを消去、データベースに対してロールバックする。また、アソシエーションが回復した場合は障害回復処理を行う。RDAプロセスがクライアント動作の場合、①トランザクション中でない場合、利用者プロセスにエラーを通知するとともに、新たなSQL操作で必要となるまでアソシエーション回復動作をとらない。一方②トランザクション中の場合は、アソシエーション切断後ただちにR-Associate要求を送出してアソシエーションの再確立を試みる。もし、確立できない場合、一定時間おきにアソシエーション確立を試みる。もし、ダイアログの停止可能時間を超えた場合、利用者プロセスにエラーを通知し、ダイアログを消去する。

7. ORACLEインタフェースプロセス

ORACLEインタフェースプロセスは、利用者プロセスがRDAプロセスにloginする際に生成される。また、RDAプロセスがサーバとして新しいダイアログを開始した時に生成され、利用者プロセスのlogoutあるいはダイアログの終了により消滅する。

ORACLEインタフェースプロセスはRDAプロセスと、RDAのSQL専化に従ってASN.1符号化したISO SQLによりインタフェースする。ORACLEインタフェースプロセスはこのISO SQLを変換してORACLEのSQLを生成し、ORACLEにアクセスする。ISOのSQLとORACLEのSQLは概ねそのまま対応し、また表3に示すように一部は単純な予約語等の置き換えにより変換が実現できるが、一部のISOのSQL構文については、対応する機能がORACLEのSQLには提供されておらず、表3のような解釈を行った。

ISO SQL	ORACLE SQL	対処
SOME	ANY	
COMMIT WORK	COMMIT	
ROLLBACK WORK	ROLLBACK	
CHARACTER型	CHAR型	
REAL型	FLOAT型	
LIKE...ESCAPE	(なし)	サポートしない
UNION ALL	(なし)	UNIONにマッピング
DECIMAL型	(なし)	NUMERIC型にマッピング
APPROX型	(なし)	サポートしない

表3 ISOとORACLEのSQLの変換例

また、RDAプロセスとのインタフェースには、ORACLEにlogon/logoffするための文、ORACLEでローカル利用者と排他制御するためのlock文およびデータベース操作を中断させるcancel文が追加されている。RDAプロセスはこれらの文を用いて、ORACLEインタフェースプロセスの動作の制御を行う。

8. 考察

8.1実装したRDAの機能

RDAの標準草案に殆どの点において準拠することができた。RDAのプロトコル上で不明確であったRO-Reject等の扱いについてもRDAプロセスでの処理を明確化し、実装上の問題とはならなかった。また、CCRASEを用いたトランザクション機能の実現についても、実際に使用したデータベースのトランザクション機能を利用することで容易に実現できた。また、RDAのSQL専化として規定されているISOのSQLと実際に使用したORACLE

のSQLについては、若干の構文の違いがあるものの、殆どそのまま対応付けることができた。また、筆者らの提唱する応用層の実装方法[7]に従うことで、他のOSIの実装で作成した応用サービス要素をソフトウェア部品として使用でき、応用エンティティを効率的に実現できた。

8.2 最新のRDAプロトコル[14][15]への対応

今回の実装は[1]に準拠しており、以下の点で現在の標準草案[14]とは異なっている。

a) ダイアログ関係

実装では、ダイアログはR-Associateにより開始(再開)され、R-Releaseによりダイアログの終了(中断)が行われる。これらはROSのBINDおよびUNBIND(すなわちA-AssociateおよびA-Release)にマッピングされる。一方、現在はアソシエーション確立後に、R-BeginDialogue等の操作によりダイアログが制御され、これらはROSの操作により転送される。これらについては、プロトコルを変更することで容易に対処できる。

b) R-Cancel、R-Status

既に発行した操作の状態を調べるR-Status操作と、中止するためのR-Cancel操作が追加された。また、これらの操作で対象となる操作を識別するために、各操作にオペレーションIDパラメータが追加された。R-Cancelについては既に若干異なるフォーマットで実装しており、その変更は容易である。各操作に対して追加するオペレーションIDは、今回RDAプロセスがROSのインボークIDを管理しているためその値がそのまま使用できると考えられる。R-Statusについても容易に実現できると考えられる。

8.3 RDAの分散データベースへの適用性

今回作成したアプリケーションは、利用者プロセスから要求を受けたノードが分散処理を全て行う。しかし、処理効率を考慮すると他のノードで処理した方が効率的である場合もある。RDAでは、このような処理を依頼するような操作は提供していないため、他のプロトコルと組合るなどの方法で、効率の良い分散アクセスのメカニズムを今後検討する必要がある。

8.4 RDAでの大量データの扱い

RDAはROSを使用して遠隔操作を実現しているが、その性格からディレクトリシステムと同様1つの操作に対する結果が非常に大きなものになる可能性がある。OSIでは各プロトコルに対して扱うプリミティブのサイズを制限していないが、実装上扱えるサイズには限界がある。ROSのプリミティブをRTSを用いて分割転送することも考えられるが、RTSはCCRとは現在同時に使用できないためRDAの場合適用することはできない。また、この場合も応用プロセス、RDAASE及びROSEの間は大きなプリミティブを扱う必要がある。従って、これらの応用では、応用プロセスで適当なサイズに分割して送出するようなメカニズムを今後検討していく必要がある。

9. おわりに

本稿では、RDAプロトコルの実装について報告した。本実装では、転送するデータベース操作言語として、ISOでRDA専化として標準化の進んでいるSQLを使用し、サーバ側ではRDBとしてORACLEを用いてデータベース機能を実現した。また、RDAプロトコルを用いて、分散したデータベースに対してデータ位置の透過性を提供するアプリケーションを作成した。今後は、RDA単体としての速度の評価および分散処理を行った場合の処理系全体の処理効率についての評価を行うとともに、RDAを用いた分散データベース実現のための手法についても検討を進めていく予定である。

最後に日頃御指導頂くKDD上福岡研究所 小野所長、浦野次長に感謝します。

参考文献

[1]: ISO/JTC1/SC21/WG3 Editors Working Draft for Generic RDA 2nd DP, Sept., 1988

[2]: ISO/JTC1/SC21/WG3 N675 Remote Database Access: SQL Specialization, Sept., 1988

[3]: ISO/DIS 9075 Database Language SQL, May., 1986

[4]: 西山, 杉山, 小花, "OSI RDA実装の基本設計", 第37回情処全大, 1988.

[5]: 小花, 西山, 杉山, "OSI RDAを用いた分散データベース・アクセスの実現(1)-全体構成-",第38回情処全大, 1989.

[6]: 西山,小花, 杉山, "OSI RDAを用いた分散データベース・アクセスの実現(2)-分散処理メカニズム-",第38回情処全大, 1989.

[7]: 小花,西山, 杉山,鈴木 "OSIにおける応用層構造(ALS)の実現方法の提案",第37回情処全大, 1988.

[8]: 杉山,小花,西山, "OSIの応用層構造(ALS)におけるSACFの実装",第39回情処全大, 1989発表予定

[9]: 杉山,西山,小花, "OSI CCR実装の基本設計",第37回情処全大, 1988.

[10]: 杉山,西山,小花, "OSI CCRを用いたRDAトランザクション機能の実現方法の提案",第38回情処全大, 1989.

[11]: 小花,杉山,西山, "OSI CCRの実装",情処マルチメディア通信と分散処理研究会 43-3,1989

[12]: 小花,加藤,鈴木, "OSIプレゼンテーション, ACSE, FTAMプロトコルの実装と評価",情処論文誌 Vol.30, No.7, Jury, 1989

[13]: 小花,西山,鈴木, "OSIディレクトリシステムの実装と評価",情処マルチメディア通信と分散処理研究会 42-11,1989

[14]: ISO/JTC 1/SC21 N3606 2nd DP Generic RDA, June, 1989.

[15]: ISO/JTC1/SC21/WG3 N844 Remote Database Access: SQL Specialization, June., 1989.