情報処理学会論文誌　データベース　Vol. 4　No. 1　50–63 (Mar. 2011)

オブジェクト・リレーショナルデータベース管理システムにおけるプラグイン機構の開発

土田正士[†1]　河村信男[†1]　中野幸生[†1]
原憲宏[†1]　石川博[†2]

ネットワークコンピューティング時代には，画像，音声，動画などマルチメディアデータを取り込んだ情報システムの構築が不可欠となりつつある．マルチメディアデータは，データ量が大容量なだけではなく，メディア操作および格納形式が多様化かつ順次拡張されることに特徴がある．従来までの RDB（リレーショナルデータベース）およびメディアごとに別サーバを有する情報システムでは，システム運用およびアプリケーション開発に多くの課題があった．これらの課題に対して，提案する ORDBMS（オブジェクト・リレーショナルデータベース管理システム）では，それぞれのマルチメディア情報に対して，検索および格納などの操作に対応するライブラリ群を，自由に追加，登録できるプラグイン機構を開発した．プラグインとしては，SGML 文書，文字列，XML への構造検索をはじめとして，類似画像検索，空間検索を実装した．

Development of Plug-in Architecture on Object-relational Database Management System

Masashi Tsuchida,[†1] Nobuo Kawamura,[†1]
Yukio Nakano,[†1] Norihiro Hara[†1]
and Hiroshi Ishikawa[†2]

In the age of network computing, composite multimedia data such as images, voices and movies are indispensable to information systems. The amount of data is huge in size, and operations of media and structures of store format, are also flexible and expandable. The information systems having the hybrid architecture of the conventional RDB (relational database) and independent media servers for each media have many issues in the administration of information systems and the developments of applications. The proposed ORDBMS (object-relational database management system) introduces new plug-in architecture in which various multimedia method libraries for storing and retrieving can be installed easily into the kernel of the DBMS. The repertoire of plug-in is not only full-text search specifying its structure to SGML (standard generalized markup language) documents, character string, and XML, but it is also similarity retrieval from images and geo-spatial information retrieval.

1. はじめに

インターネットに代表されるネットワークコンピューティングにおいては，利用者に親しみを与える情報提供形態が重要である．そのため，画像や音声，動画といったマルチメディアデータを取り込んだ情報システムの構築が必要不可欠となりつつある．

マルチメディア情報は，データ量が大容量なだけではなく，メディア操作や格納形式が多様化かつ順次拡張していくことに特徴がある．これに対応するマルチメディアデータ処理系として，オブジェクト・リレーショナルデータベース（ORDB）が期待されている[1)]．ORDB は，RDB モデルをベースとし，オブジェクト指向モデルを取り入れた DB である．

一方，国際標準化機構 ISO では，データベース言語 SQL の標準仕様を策定している[2)]．現時点では，SQL2008 が最新であり，利用者定義型（以下では，UDT（User-Defined Type）と呼ぶ），型継承などのオブジェクト指向モデルを取り入れた拡張が規定されている．これにより，複雑な構造を持つマルチメディアデータをデータベースに容易に取り込めるようにする枠組みとして活用が可能である．

ORDB では，データに対する操作機能を利用者定義のルーチンによって提供する．ここではプラグイン（plug-in）と呼ばれる機構によって，ルーチンを実装するモジュール（以下では，プラグインモジュールと呼ぶ）を ORDBMS に組み込む．これにより，DBMS 自体を変更することなく，各種のメディアデータを扱う先進的な機能を実装するプラグインモジュールを，容易に DB システムに取り込むことができる．また，このプラグインモジュールは DBMS と独立して開発することができる．すなわち，この方式によれば，たとえば構造化文書の構造指定検索などの高度な機能を持ち，また n-gram 方式による高速な全文検索機能をプラグインとして開発し，ORDB に組み込むことで，拡張可能なアーキテクチャが実現できる．さらに，文書に限らず画像や地図データなどを管理する機能を持つプラグイン

†1 株式会社日立製作所ソフトウェア事業部
Software Division, Hitachi Ltd.
†2 静岡大学情報学部
Faculty of Informatics, Shizuoka University

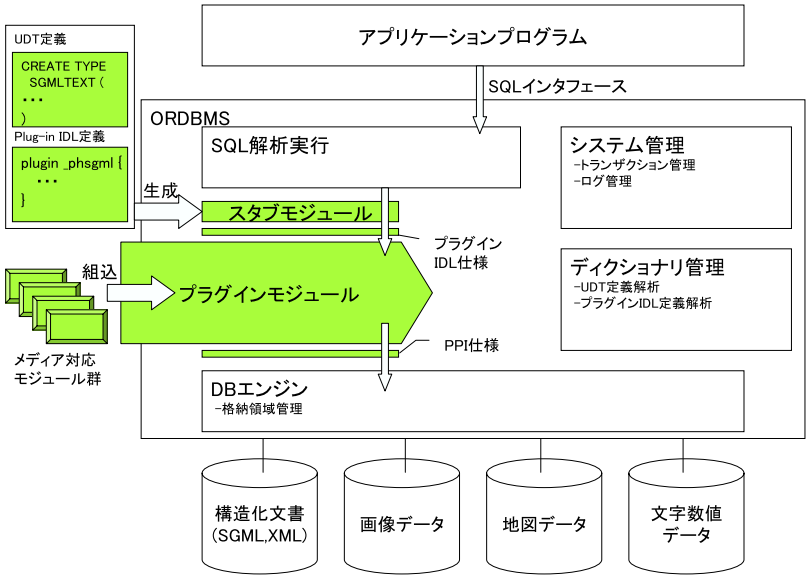

図 1　ORDBMS の構成
Fig. 1　Configuration of ORDBMS.

モジュール群を組み込めば，多様にシステムを拡張することもできるようになる．

そのために，ORDB にプラグイン機構を開発し，また Shared Nothing 方式の並列 DB に適用することで，マルチメディア情報管理システムの基盤となる ORDBMS を開発した[3)–7)]．開発した ORDBMS を適用したシステムの例を**図 1** に示す．本論文の研究範囲は，プラグイン機構である．具体的には，プラグイン機構に関して，プラグイン IDL[⋆1]（Interface Definition Language）と PPI[⋆2]（Plug-in Programming Interface）を開発した[9),10)]．

プラグイン機構について，マルチメディア情報システムへの要件分析と課題の検討を基にして，ORDBMS の開発およびその実装を通して適用性を確認した．以下では，2 章で課題について述べ，3 章でプラグイン機構のアプローチについて解説し，4 章で具体的に実装したプラグイン事例による適用評価を示し，5 章で関連研究を述べ，6 章でまとめる．

⋆1 3.3.1 項で後述するが，データベースへの操作，トランザクション制御およびデータベース回復処理に対応した順序，タイミングの処理契機を規定する，プラグインの外部仕様を記述するための宣言的言語である．
⋆2 3.4 節で後述するが，データ操作，データ制御，ファイル操作，およびサービスに区分する関数インタフェースからなり，DB リソースへのアクセス手段を提供する．

2. 課　　題

ORDBMS はデータに対する操作機能を利用者定義のルーチンによって提供するが，本章ではプラグイン機構によって，関数の実装モジュールを ORDBMS に組み込む場合の課題について述べる．

まず，ORDBMS の基本機能として，データベースの回復処理を実現することは信頼性を達成するための主要な要件である．プラグイン機構によってデータベースに対する操作群が提供されるが，拡張可能なアーキテクチャを実現するために，ORDBMS と独立に開発するので，この回復処理も組み込まれなくてはならない．ここで回復処理とは，ACID 特性の耐久性（Durability）を保証することを指す．また，回復処理はトランザクションおよびログを基にして実装する必要もある[11)]．ここで，この回復処理の実装に必要な，トランザクション情報，ログ情報，格納領域情報を DB リソースと呼ぶ．回復処理はこれらの DB リソースを利用して実現されるが，トランザクション制御およびデータベース回復処理に対応した順序，タイミングで処理を行わないと，データベースの破壊を招く可能性がある．したがって，このような ORDBMS に組み込まれる実装モジュールの開発は，ORDBMS の実装に関する知識を理解しておく必要もあり，プログラミングの難易度が高いと考えられ，信頼性を確保するための施策が必要である．

一方で，ORDBMS はトランザクション処理での利用を想定しており，回復処理の性能も重要である．すなわち，トランザクション処理の性能を維持しつつ，回復処理の実現にともなう実装モジュールの起動および実行にともなうオーバヘッドを最小限にとどめ，ORDBMS 全体として性能を確保したい．

SQL[2)] では，データに対する操作を行うルーチンの実装方式として，以下の 2 つを規定している．SQL で規定するルーチンには，関数および手続きが含まれる．

- SQL 起動ルーチン（SQL-invoked routine）
  ルーチンの実装は，SQL 手続き文からなる．表，列，インデクスなど SQL 定義文で規定されるスキーマオブジェクトだけが，操作の対象である．
- 外部起動ルーチン（Externally-invoked routine）
  ルーチンの実装を，SQL 以外の C 言語などのプログラミング言語で記述する．ルーチンの実行では，その記述のコンパイル結果であるオブジェクトコードを実行する．

SQL 起動ルーチンは，SQL 文によりスキーマオブジェクトへのアクセスが可能で，SQL

手続きは計算完備である．しかし，回復処理の実装に必要な DB リソースはスキーマオブジェクトとして規定はされておらず，それら DB リソースへの操作記述，すなわち表 1 に後述するロールバックおよびロールフォワード処理の記述は制約されている．

一方，外部起動ルーチンは，C 言語などのプログラミング言語を利用して，領域格納手段としてファイルへのアクセスは可能であり，トランザクション情報およびログ情報からなる DB リソースへのアクセスおよびデータベースの回復処理の記述が可能である．また，外部起動ルーチンが DBMS とは異なるアドレス空間で実行する方法[17]，および外部起動ルーチンが実行するアドレス空間を DBMS と同じアドレス空間にするのか，あるいは異なるアドレス空間にするのかを選択する方法[1] があるが，トランザクション処理での性能を考慮すれば，外部起動ルーチンが実行するアドレス空間と DBMS が実行するアドレス空間を同じとする必要がある．

以上，本研究の課題をまとめる．

- 耐久性の保証
  DB リソースを利用した回復処理の実現
- 外部起動ルーチンの起動および実行の高速化
  トランザクション処理での性能要求に応えること
- プラグインモジュールの組み込みを簡単化
  ORDBMS に組み込まれる外部起動ルーチンは，ORDBMS の実装に関する知識を理解して開発される必要があり，難易度が高いと考えられるプラグインモジュールの組み込みを簡単にすること

本論文では，これらの課題を解決するために，DB リソースを利用した回復処理を実装するプラグインモジュールを ORDBMS に組み込み，またプラグインモジュールを外部起動ルーチンで呼び出す，プラグイン方式を採用した．このプラグインモジュールは，DB リソースを利用してデータベースへの操作，トランザクション制御およびデータベース回復処理に対応した順序，タイミングの処理契機に応じて処理の記述を行う．

プラグイン方式の特徴についてまとめる．

(1)　DB リソースへのアクセスおよび回復手段
　プラグインモジュールの実装から DB リソースを引数とし，データベースへの操作，トランザクション制御およびデータベース回復処理に対応する関数群を容易に呼び出し，かつデータベースの回復処理を支援する PPI を提供する．

(2)　容易なプラグインモジュールの組み込み
　ORDBMS の基本機能であるデータベースへの操作，トランザクション制御およびデータベース回復処理に対応した順序，タイミングの処理契機を規定する，プラグインの外部仕様に基づくプラグイン IDL を開発する．このプラグイン IDL に従いプラグインの動作を記述する，プラグイン IDL 記述ファイルをプラグイン IDL コンパイラが解析し，ORDBMS からプラグインを呼び出すときに必要な定義データとスタブモジュールを自動生成する．また，プラグインを組み込むツールを提供し，コマンドにより容易にプラグインを組み込めるようにする．

(3)　プラグインモジュールの高速起動
　プラグインモジュール起動のオーバヘッドを少なくし，かつ ORDBMS の並列処理に合わせて並列実行するために，ORDBMS と同じアドレス空間で SQL 文の実行からなる DB 処理を行うスレッドと同一スレッドでプラグインを実行する方式を採用した．

3. アプローチ

3.1 概　　要

プラグイン機構を利用したデータ処理の概要を**図 2** を用いて示す．

この例では，データベースに登録された SGML 文書からプラグインの機能を利用して全文検索を行う．SGML 文書への操作を実装するデータプラグインモジュール（SGML データプラグインモジュールと呼ぶ）と，n-gram 方式のインデクス処理系を実装するインデクスプラグインモジュール（n-gram インデクスプラグインモジュールと呼ぶ）を利用する[⋆1]．

これらを，図 2 中の 1〜8 を用いて説明する．

(1)　アプリケーションから ORDBMS に，全文検索を要求する SQL を発行する．
　この例では，本文の内容に「日立製作所」という文字を含む文書の文書名を取得する SQL を発行する．

(2)　ORDBMS の FES（フロントエンドサーバ）がアプリケーションからの要求を受け付ける．
　FES の SQL 解析部は，ディクショナリ管理と通信し，SQL を意味解析して DB 処理

⋆1 プラグインモジュールには，UDT への操作を実装するデータプラグインモジュールと，インデクス処理系を実装するインデクスプラグインモジュールの 2 種類からなる．詳細は，3.5 節に後述する．

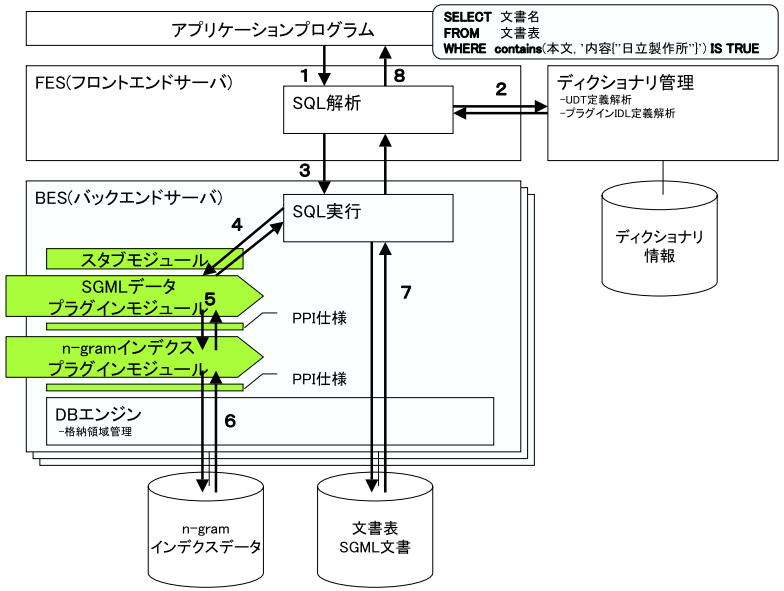

図 2　プラグインを利用した SGML 文書全文検索処理の例
Fig. 2　Example of SGML document full-text search using plug-in.

を行うための中間コードを生成する．なお，ディクショナリには，スキーマオブジェクトおよび DB リソースに関連する定義情報が管理されている．SQL 文の解析処理でプラグインとして実装されているルーチン contains() が指定されていることを識別すると，SGML データプラグインモジュールを呼び出すための中間コードを生成する．また，インデクス定義でルーチン contains() に対応付けられている n-gram インデクスプラグインモジュールを呼び出す中間コードも生成する．

(3)　FES が生成した中間コードを複数の BES（バックエンドサーバ）に送信し，並列に DB 処理を行うよう要求する．
BES の SQL 実行制御部が，中間コードに従って DB 処理を行う．

(4)　BES は，条件判定処理において，中間コードに設定された情報を基に，プラグインモジュールを呼び出す．
BES がプラグインモジュールを動的にロードし，BES の同一スレッドで実行する．
SGML データプラグインモジュールの関数 contains() に対応するプラグインモジュールを起動する．呼び出しはスタブモジュールを介して行う．

(5)　SGML データプラグインモジュールが，PPI を通してインデクス利用要求を行う．
PPI では，中間コードの情報を基に，n-gram インデクスプラグインモジュールを呼び出す．

(6)　n-gram インデクスプラグインモジュールは，PPI を用いて ORDBMS で管理されているファイルにアクセスし，インデクスデータを参照し，全文検索処理を行う．

(7)　全文検索の処理結果に応じて，BES の SQL 実行処理では文書表から文書名を取得してまとめて FES に返す．

(8)　FES は複数の BES から受け取った処理結果をまとめてアプリケーションプログラムに返す．

この例において，本論文で開発した特長的な部分を以下に示す．

- UDT のルーチンをプラグイン方式で実装
  UDT のルーチンをプラグイン方式による外部起動ルーチンとして実装した．プラグインで提供する機能を，UDT のルーチン呼び出しで提供できるので，アプリケーションと親和性の高いインタフェースが実現できる．
- 定義情報を基にしたプラグイン呼び出し
  ディクショナリにプラグインを呼び出すための定義情報を保持する．SQL 解析処理は，この定義情報を基に SQL を解析し，プラグインを呼び出すための中間コードを生成する．この定義情報を変更すれば，呼び出されるプラグインが変更される．これにより，プラグインの追加・変更・削除を柔軟に行うことができる．
- データプラグインモジュールとインデクスプラグインモジュールを分離
  データ操作処理およびインデクス処理に特化したプラグインモジュールを，それぞれデータプラグインモジュール，インデクスプラグインモジュールとする．これにより，n-gram 方式を実装する n-gram インデクスプラグインモジュールを，SGML データプラグインモジュール以外にも適用するなど，再利用を図ることができる．
- BES の処理と同一スレッドでプラグインを動的ロードして実行
  BES の処理スレッド上で動的ロードしてプラグインモジュールを実行する．これにより，モジュール起動のオーバヘッドが小さく，高速にモジュールが実行できる．さらに，複数の BES による並列処理で，プラグインモジュールも並列に実行することができる．

3.2　プラグインモジュールのルーチン定義

プラグインモジュールで提供する機能を SQL インタフェースで利用するために，UDT

```
CREATE TYPE SGMLTEXT (
    PRIVATE sgmltext BLOB,

    PUBLIC FUNCTION SGMLTEXT (sgml_text BLOB)
        RETURNS SGMLTEXT
        LANGUAGE C
        EXTERNAL NAME 'lib_phsgml.sl!_phsgml_constructor'
        PARAMETER STYLE PLUGIN,

    PUBLIC FUNCTION contains (sgml_text SGMLTEXT,
                              condition VARCHAR(32000))
        RETURNS BOOLEAN
        LANGUAGE C
        EXTERNAL NAME 'lib_phsgml.sl!_phsgml_contains'
        PARAMETER STYLE PLUGIN,

    ....
)
```

:ルーチン名
:プラグインモジュールの実装関数名

図 3　UDT のルーチン定義例
Fig. 3　Example of routine definition using UDT.

```
plugin _phsgml id 30001 datatype SGMLTEXT { ...
 _phsgml_contains (
    in       SGMLTEXT       sgml       indicator(sgmli),
    in       VARCHAR(32000) condition  indicator(conditioni),
    returns  BOOLEAN        judgement  indicator(judgementi),
    inout    DBIFB          dbifb,
    pickup   ROWID          r_rowid
    ) as UDT_FUNCTION
      SCAN_TYPE,
      COUNT_SURROGATE (_phsgml_contains_count) ;

 _phsgml_before_insert (
    inout    DBIFB          dbifb,
    inout    NEW_VALUE      sgml       indicator(sgmli),
    in       BLOB           sgmldef    indicator(sgmldefi)
                              setter (_phsgml_load_sgmldef_1),
                                      sgmldef,
                                      EVALUATE_AT_PARAM_BIND)
    ) as BEFORE_INSERT ;

 _phsgml_start_thread (inout DBIFB dbifb)
      as START_THREAD ;

...};
```

:契機指示子
:プラグインモジュールの実装関数名
:DBリソース参照(dbifb)
:標識変数(indicator)
:行ID情報(ROWID)

図 4　プラグイン IDL の記述例
Fig. 4　Example of plug-in IDL.

のルーチンを定義する．**図 3** に，プラグインモジュールのルーチン定義例を示す．この例では SGML 構造化文書データを表現するために SGMLTEXT 型を定義する．

UDT の定義では，CREATE TYPE に続いて，型名を記述する．続いて，属性とルーチンの定義を記述する．

ルーチンの定義は，隠蔽レベル，ルーチン種別，ルーチン名，引数，戻り値の順に記述する．LANGUAGE C は C 言語で実装されたモジュールであることを示す．EXTERNAL NAME '...' は，外部起動ルーチンとしての実装であることを示す．プラグインモジュールは，共用ライブラリ名 ! 実装関数名の形式で示す．PARAMETER STYLE PLUGIN は，プラグイン方式による実装であることを示す．

この定義は，ディクショナリに保持され，SQL 解析時にプラグインを呼び出すための情報として用いる．

3.3　プラグイン・インタフェース定義

3.3.1　プラグイン IDL 定義

プラグイン IDL は，ORDBMS の基本機能であるデータベースへの操作，トランザクション制御およびデータベース回復処理に対応した順序，タイミングの処理契機を規定する，プラグインの外部仕様を記述するための宣言的言語である．

プラグインモジュールは ORDBMS とは独立に開発される．プラグインが ORDBMS に組み込まれたときに，そのプラグインの外部仕様を ORDBMS が認識する必要がある．プラグインの外部仕様を記述するための言語としてプラグイン IDL を開発した．プラグインモジュールの開発者は，プラグインの動作を記述するプラグイン IDL 記述ファイルを作成する．**図 4** に，プラグイン IDL の記述例を示す．

キーワード plugin に続いて，プラグイン名，プラグイン ID，対応する UDT を指定する．続いて，{ } の中にプラグインモジュール内の実装関数の仕様を記述する．以下の記述が可

表 1　プラグイン実装の呼び出し契機
Table 1　Invocation timing of plug-in implementation.

契機種別	処理内容	契機指示子	
ルーチン呼び出し[1]	UDT定義で指定したルーチンの呼び出し時に制御が渡る	UDT_FUNCTION	
		−	
データ操作[1]	行データの挿入，更新，および削除などデータ操作の前後のタイミングで呼び出される	BEFORE_INSERT	AFTER_INSERT
		BEFORE_UPDATE	AFTER_UPDATE
		BEFORE_DELETE	AFTER_DELETE
		−	AFTER_ADD_COLUMN
		BEFORE_DROP_COLUMN	AFTER_DROP_COLUMN
		BEFORE_PURGE_TABLE	AFTER_PURGE_TABLE
システム制御[1),2)]	プロセス，スレッド，およびトランザクションの開始終了時点，コミット処理時点で呼び出される	START_PROCESS	TERMINATE_PROCESS
		START_THREAD	TERMINATE_THREAD
		BEGIN_TRANSACTION	PREPARE_COMMIT
		−	COMMIT
回復処理[1),2)]	ロールバック，ロールフォワードの開始終了時点，ロールバック処理時点，およびロールフォワード処理時点で呼び出される	START_ROLLBACK_PROCESS	TERMINATE_ROLLBACK_PROCESS
		START_ROLLBACK	TERMINATE_ROLLBACK
		ROLLBACK	−
		START_ROLLFORWARD_PROCESS	TERMINATE_ROLLFORWARD_PROCESS
		ROLLFORWARD	−
インデクスメンテナンス[2)]	インデクスの定義，初期化，作成，削除，および遅延更新の各処理時点で呼び出され，またインデクスのエントリ登録，検索，更新前後，削除の各処理時点で呼び出される	DEFINE_INDEX	BUILD_INDEX
		INDEX_SEARCH	INDEX_COUNT
		INDEX_INSERT	−
		INDEX_BEFORE_UPDATE	INDEX_AFTER_UPDATE
		INDEX_DELETE	−
		DROP_INDEX	PURGE_INDEX
		INITIALIZE_INDEX	INDEX_MAINTENANCE_DEFERRED

注記
1）データプラグインモジュールで呼び出される
2）インデクスプラグインモジュールで呼び出される

能である．

(1)　外部起動ルーチン呼び出しに現れない引数の指定
プラグインの仕様定義では，ルーチンの引数には現れない引数の指定も可能である．たとえば，ORDBMS の内部情報を保持した dbifb を指定できる．dbifb は，DB リソースをアクセスするために，PPI を利用するときに受け渡すことができ，PPI では dbifb の内部情報を参照して動作する．このほかに，受け渡す引数がナル値であるかを示す標識変数（indicator と記述），行 ID 情報（ROWID と記述）が指定できる．

(2)　プラグインモジュールを呼び出す契機の指定
as に続いて，プラグインモジュールを呼び出す契機が指定できる．**表 1** にプラグインモジュールを呼び出す契機を示す．これらの中で，UDT 定義で指定されるルーチンのルーチン呼び出し契機以外では，行データの挿入，更新，および削除のタイミングで呼び出されるデータ操作契機，トランザクション処理，コミット処理に関連するシステム制御契機，データベースの回復処理に関連する回復処理契機，およびインデクス処理に関連するインデクスメンテナンス契機からなりプラグインモジュールが呼び出される．これら呼び出す契機で提供する機能は，ORDBMS の実現のために必要とされる機能要件を満たすものである[1),11)]．

(3)　ORDBMS の処理と密接に連携した実行制御の指定
以下の指定によって，プラグインモジュールを ORDBMS の処理と密接に連携して起動することができる．

(a)　スキャンタイプ処理指定（SCAN_TYPE）
通常，インデクスを設定していない表に対しては，ORDBMS は表の行 1 つ 1 つに対して繰り返し処理を行う．このような処理形態をテーブルスキャンという[11)]．テーブルスキャンでは，表の行 1 つ 1 つに対してルーチンを起動することになる．
これに対し，インデクスを利用して検索条件を判定する場合は，ORDBMS はインデクス機能により選択された行に対して繰り返し処理を行う．このような処理形態をインデクススキャンと呼ぶ．
プラグイン IDL で実装関数に SCAN_TYPE を指定すると，ORDBMS はインデクススキャンの処理形態をとり，その関数により選択された行について繰り返し処理を行うようになる．

(b)　プラグインモジュールの実装関数どうしでの値の受け渡し（setter）
プラグインモジュールの実装関数どうしでの値の受け渡しを可能にする．
図 5 (a) の例では，setter を用いて，実装関数 _phsgml_score() が _phsgml_contains_with_score() から値を受け取るように指示している（図中では破線で対応関係を示す）．
_phsgml_contains_with_score() は，n-gram インデクスプラグインモジュールを利用し，全文検索するとともに，その検索でのスコア情報 r_score を得る．_phsgml_score() は，そのスコア情報を引数 score_inf で受け取り，結果としてスコア値を返す．
これを利用する SQL 文を図 5 (b) に示す．ルーチン contains_with_score() の全文検索で絞り込んだ文書について，それぞれの文書名と全文検索でのスコア値を得る SQL である．
ルーチン contains_with_score() 自体は，WHERE 句の条件として指定される

```
plugin _phsgml id 30001 datatype SGMLTEXT { ...
 _phsgml_contains_with_score (
   in       SGMLTEXT       sgml       indicator(sgmli),
   in       VARCHAR(32000) condition  indicator(conditioni),
   returns  BOOLEAN        judgement  indicator(judgementi),
   inout    DBIFB          dbifb,
   pickup   ROWID          r_rowid,
   out      BLOB           r_score    indicator(r_scorei)
   ) as  UDT_FUNCTION
     SCAN_TYPE ;
 _phsgml_score (
   in       SGMLTEXT       sgml       indicator(sgmli),
   returns  INT32          score      indicator(scorei),
   inout    DBIFB          dbifb,
   in       BLOB           score_inf  indicator(score_infi)
                             setter (_phsgml_contains_with_score,
                                     r_score)
   ) as UDT_FUNCTION ;
...};
```

実装関数間での値の受け渡し指示

(a) setterにより実装関数の値の受け渡しを指示したプラグインIDL

```
SELECT 文書名, score(本文)
FROM   文書表
WHERE  contains_with_score(本文, '内容{"日立製作所"}') IS TRUE
```

値の受け渡し関係

(b) 実装関数間での値の受け渡しが行われるSQL
ルーチンscore( )の実装関数は、contains_with_score( )の実装関数の処理結果で得られるスコア値r_scoreを受け取って、その結果を返す。

図 5　プラグイン実装関数どうしでの値の受け渡しの例
Fig. 5　Example of value noification between plug-in functions.

```
plugin _phsgml id 30001 datatype SGMLTEXT { ...
  _phsgml_contains (
    in       SGMLTEXT       sgml       indicator(sgmli),
    in       VARCHAR(32000) condition  indicator(conditioni),
    returns  BOOLEAN        judgement  indicator(judgementi),
    inout    DBIFB          dbifb,
    pickup   ROWID          r_rowid
    )
      as UDT_FUNCTION
      SCAN_TYPE,
      COUNT_SURROGATE (_phsgml_contains_count) ;
 _phsgml_contains_count (
    in       SGMLTEXT       sgml       indicator(sgmli),
    in       VARCHAR(32000) condition  indicator(conditioni),
    returns  BOOLEAN        judgement  indicator(judgementi),
    inout    DBIFB          dbifb,
    out      INT32          r_count    indicator(r_counti)
    ) ;
...};
```

実装関数間での値の受け渡し指示

(a) count_surrogateによりサロゲート関数を指定したプラグインIDL

```
SELECT COUNT(*)
FROM   文書表
WHERE  contains(本文, '内容{"日立製作所"}') IS TRUE
```

値の受け渡し関係

(b) カウントサロゲート関数が起動するSQL

図 6　カウントサロゲート関数の例
Fig. 6　Example of count_surrogate function.

述語であり，真偽の値のみを返すだけで，スコア値を返すことができない．そこで，検索結果でスコア値を返すには，SELECT 句の射影項目にスコア値を返す指示が必要である．setter を用いた実装関数 score() の仕様定義よってスコア値を返すようにすれば，SQL では直観的な記述となる．すなわち，スコア値の受け渡しを SQL で指定する必要がない．

したがって，setter 指定により ORDBMS がプラグイン実装関数間の値の受け渡しを支援することにより，このような SQL 文を効率良く処理することができる．

(c)　カウントサロゲート関数（count_surrogate）

検索条件に合致したレコードの件数のみを返す実装関数を指定できる．件数の算出に特化した関数を代わりに実行することにより，処理の高速化が図れる．**図 6** (a) の例では，実装関数_phsgml_contains() に，カウントサロゲート関数として_phsgml_contains_count() を指定している．これに従い，図 6 (b) のような SQL を実行する場合，ORDBMS はルーチン contains() に対応付けられている実装関数_phsgml_contains() を起動する代わりに，_phsgml_contains_count() を起動する．_phsgml_contains_count() は，カウントサロゲート関数として，件数値を返す引数 r_count を持っている．ORDBMS はこの引数から得た値を件数値とする．

3.3.2　UDT，プラグイン IDL，およびプラグインモジュールの関係

プラグイン IDL 定義では，表 1 のプラグインモジュールを呼び出す契機で示すように，UDT 定義中でプラグインモジュールによる実装であることが指定されている外部起動ルーチンが起動されたときに，どのような引数を必要とするかを，それぞれの実装関数ごとに指定する．また，UDT 定義のルーチンに対応した実装関数を呼び出すルーチン呼び出し契機以外では，行データの挿入，更新，および削除のタイミングで呼び出されるデータ操作契

機，トランザクション処理，コミット処理に関連するシステム制御契機，データベースの回復処理に関連する回復処理契機，およびインデクス処理に関連するインデクスメンテナンス契機の指定が可能である．

UDT で定義された SGMLTEXT 型は，利用者に対して contains(), contains_with_score(), score() というルーチンを提供する．これらのルーチンは，それぞれ，プラグインモジュール内の実装関数では，_phsgml_contains()，_phsgml_contains_with_score()，_phsgml_score() によって実装されることが，SGMLTEXT 型の UDT 定義によって指定される．これらの実装関数が PPI を使用する際に必要な引数は，プラグイン IDL で指定が可能である．図 4 の記述例では，利用者によって呼び出されたルーチン contains(sgml,condition) は，対応する実装関数の呼び出し時には，_phsgml_contains(sgml, sgmli, condition, conditioni, judgement, judgementi, dbifb, r_rowid) として，6 個の引数が付け加えられる．これらは，実装関数の戻り値（judgement），引数の標識変数（sgmli，conditioni，および judgementi），DB リソースへのアクセスを行うための引数（dbifb），検索結果として返される行識別子（r_rowid）からなる．また，プラグインモジュールの初期化，終了処理やトランザクションとの同期処理をプラグインモジュールが行う必要がある場合に備えて，それらの契機でプラグインモジュールの関数を呼び出すような指定も可能になる．

さらに，データプラグインモジュールは，実装関数群からなるプラグインモジュール，それに対応する UDT 定義，およびプラグイン IDL から生成されるプラグインモジュール制御情報の 3 つから構成される．プラグインが ORDBMS に導入される場合，UDT 定義とプラグインモジュール制御情報がディクショナリに読み込まれ，一方でプラグインモジュールが UDT 定義の EXTRENAL NAME 句で指定される共用ライブラリの位置に組み込まれる．

3.3.3　プラグイン IDL コンパイラ

プラグイン IDL コンパイラは，プラグイン IDL の記述を解析し，ORDBMS が認識可能な定義情報や，プラグイン開発に必要なファイルなどを自動生成する．プラグイン IDL が生成するファイルを以下に示す．

(1)　プラグイン定義情報ファイル
この定義情報を ORDBMS の意味解析および最適化が判断して，ORDBMS と密接に連携したプラグインの実行制御を行う．プラグイン登録コマンドを用いて，DB システムにプラグインを登録する場合，この定義情報をディクショナリに登録する．

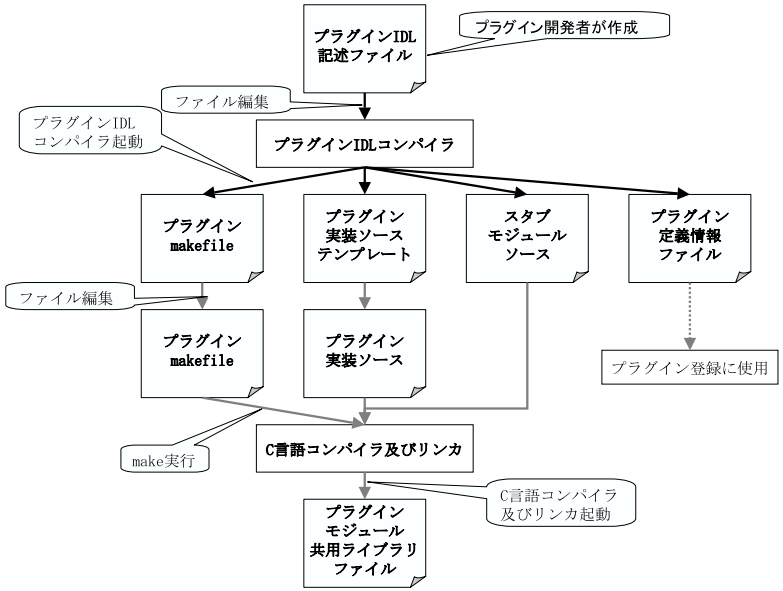

図 7　プラグイン IDL コンパイラと出力ファイルの概要
Fig. 7　Description of plug-in IDL compiler and output files.

(2)　スタブモジュールのソースファイル
ORDBMS からプラグインモジュールを呼び出す際に仲介を行うスタブモジュールのソースコードを自動生成する．このソースをコンパイルしたオブジェクトをプラグインモジュールに加えておく．

(3)　プラグインモジュール実装ソースのテンプレートファイル
プラグイン開発者は，このファイルを基にプラグインを開発することができる．プラグインモジュールを作成する makefile も生成する．

図 7 に，プラグイン IDL と出力するファイルの概要を示す．

3.4　プラグイン・プログラミング・インタフェース

PPI は DB リソースへのアクセス手段を提供する．関数インタフェースで容易に ORDBMS のログに基づく回復処理を呼び出すことができ，ACID 特性の耐久性が実現できる．具体的には，データ操作，データ制御，ファイル操作，およびサービスに区分する関数インタフェースを開発した．**表 2** にその種類と機能を示す．これら PPI で提供する機能は，ORDBMS

表 2　PPI の種類と機能
Table 2　Kind and function of PPI.

機能		処理内容
データ操作	検索処理の制御	インデクスプラグインモジュールによりインデクスが設定されて利用できる状態にあるか，1SQL内で初回の検索要求であるか否かを確認する．
	UDT値の操作[1)]	UDT値の生成，属性値の設定，および取得を行う．
	LOB値の操作[1)]	LOB値の生成，取得，および更新を行う．
	インデクス検索[1)]	インデクスプラグインモジュールによりインデクスが張られていれば，インデクスの検索要求を発行する．
	インデクス更新情報の取得[2)]	表データが更新された場合，インデクスに関する更新情報を取得する．
データ制御	ログデータ取得	プラグインモジュールが回復処理を行うためにログデータを取得する．取得したログデータは，プラグインIDLで指定される回復処理契機で該当するプラグインに渡す．
	排他制御	プラグインモジュールで決められた規則に従い，資源名を用いて排他制御を行う．
ファイル操作	論理ファイル操作	ORDBMS管理のLOB用格納領域にページ単位で読み書き可能な論理的なファイルを作成する．トランザクションとの同期を保証し，バックアップの対象となる．
	DBファイル操作	DBファイルの生成，削除，読み書きを行う．ただし，ファイル内容への変更に関して，トランザクションとの同期は取らず，バックアップの対象にはならない．
	OSファイル操作	OSファイルの生成，削除，読み書きを行う．ただし，ファイル内容への変更に関して，トランザクションとの同期は取らず，バックアップの対象にはならない．
サービス	UDT列情報の取得[1)]	データ操作の対象列に関するディクショナリ情報(表名称，列名称，プラグイン句で指定する任意の文字列)を参照する．
	レジストリ参照サービス	プラグインモジュール固有の制御情報を取得する．
	レジストリ更新サービス	プラグインモジュール固有の制御情報を更新する．
	OSサービスインタフェース	OSシステムコールを直接発行せずに，このOSサービスインタフェースを介して間接的に呼び出す．
	エラー通知	プラグインモジュール内部で発生したエラーをODBMSに通知する．

注記
1）データプラグインモジュールだけで呼び出される
2）インデクスプラグインモジュールだけで呼び出される

の実現のために必要とされる機能要件を満たすものである．

プラグインモジュールは，制御が渡されると DB リソースをアクセスしながら要求された処理を実行することができる．この DB リソースへのアクセス方法は PPI である．具体的には，C 関数呼び出しによるアクセスインタフェースであり，UDT インスタンス，データページ，およびファイルシステムへのアクセス，排他制御，プラグイン独自の回復を行うためのログ取得の操作が提供される．これらの操作を使用するためには，DB リソースへのアクセスを行うための引数が必要になる．この引数は利用者から与えられるものではなく，プラグインモジュールが起動される際に，ORDBMS から引数として取得されなければならない．このために，プラグイン IDL で，プラグインモジュールの処理で使用するために必要な引数を取得できるようにインタフェースを定義する必要がある．

なお，プラグインの種別（データプラグインモジュールおよびインデクスプラグインモジュール）により要求される機能の差があるため，プラグイン種別により使用可能な PPI を分けている．

3.5　データプラグインモジュールとインデクスプラグインモジュールの分離

データ処理機能を提供するデータプラグインモジュールと，インデクスに特化した機能を提供するインデクスプラグインモジュールを分離した．

インデクスプラグインモジュールにより，ORDBMS が標準に提供している B-tree 方式のインデクス機能以外に，新しいインデクス機能を追加することができる．

本論文では，SGML パーサなどデータ操作を処理する部分と，全文検索を高速化するための n-gram 方式のインデクスを実装する部分を独立させ，それぞれプラグインで組み込めるようにする．これにより，n-gram 方式のインデクスプラグインモジュールを SGML 以外のデータにも適用可能とする．

データプラグインモジュールから，PPI を経由してインデクスプラグインモジュールの検索機能を利用することができる．特定のデータプラグインモジュールからではなく，複数のデータプラグインモジュールから共通に利用することもできる．

インデクスプラグインモジュールには，CREATE INDEX 文，DROP INDEX 文などのインデクス実装に固有の呼び出し契機がある．この契機を利用して，インデクスを保守することができる．たとえば，大量の文書の登録・更新時には一時的にインデクスを削除しておくことにより，インデクス保守の効率化を図ることができる．

3.6　プラグインオプションとレジストリ表

表の列に SGML 文書を格納する際，SQL による列単位の操作では，その列に格納する SGML 文書は同一の文書構造を持つことが期待される．この場合，列単位で文書構造情報などを指定することが必要になる．すなわち，列単位でプラグインに必要な情報を保持する．

そこで，列単位にプラグインに関連する情報を指定可能にするプラグインオプションを提供する．また，この関連情報を保持するためのレジストリ表を提供する．

図 8 は，SGML 文書の解析に必要な DTD（Document Type Definition）を列に対応させて定義した例である．

表作成時に，列定義のプラグインオプション（PLUGIN に続く' ' で囲まれた文字列）に DTD 名称を指定する．レジストリ表には，DTD 名称をレジストリキーとし，DTD の内容を値として登録する．プラグインは，PPI を用いて列に定義された DTD 名称を取得し，それをレジストリキーとして対応する DTD 内容を読み出すことができる．

なお，SQL 文で処理対象となるすべての列のプラグインオプションは，SQL 文の処理に先立って FES の SQL 解析処理がディクショナリから読み出し，中間コードに保持する．

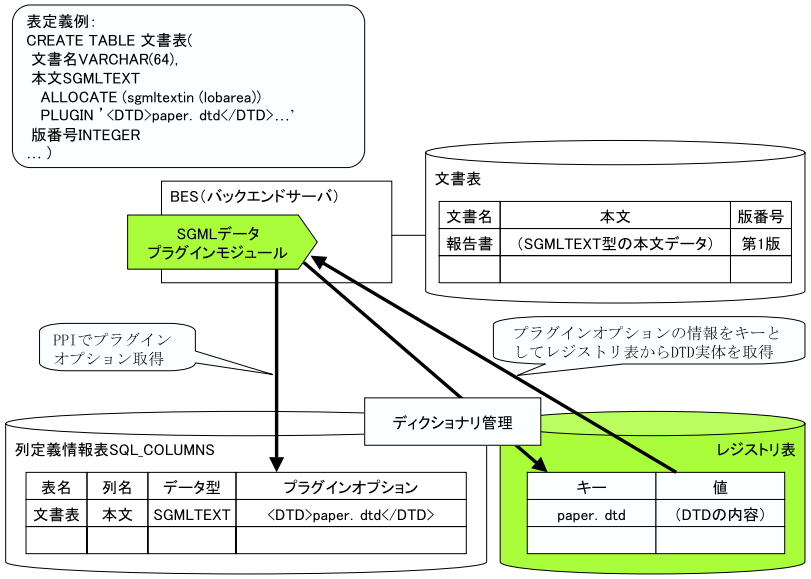

図 8　列定義でのプラグインオプション指定の例
Fig. 8　Example of plug-in option specification in column definition.

よって，プラグインからのプラグインオプションの参照は，メモリ参照であり高速に行える．レジストリ表の値もディクショナリサーバや FES でキャッシングしており，サーバ間の交信を削減している．

4. 適 用 評 価

マルチメディアデータを扱う事例として，全文検索，文字列検索，XML 検索，画像検索，および空間検索のプラグインを開発し，その適用性について評価する．**表 3** は，それらプラグイン群で実装されたデータプラグインモジュール，インデクスプラグインモジュールの組合せ，および提供される代表的な操作群である．

4.1 UDT によるプラグイン拡張について

UDT によるプラグイン拡張によって，マルチメディアデータを扱うことの適用性を確認した．

(1)　マルチメディアデータ管理の一元管理

表 3　プラグイン実装例
Table 3　Example of plug-in implementation.

種別	データプラグインモジュール	インデクスプラグインモジュール	ルーチン一覧		
			入力操作	出力操作	検索操作
全文検索[1]	SGMLTEXT	NGRAM[2]	-SGMLTEXT	-extracts -score	-contains -contains_with_score
文字列検索	FREEWORD	IXFREEWORD[2]	-FREEWORD	-extracts -score	-contains -contains_with_score
XML検索	XML	IXXML[2]	-XML	-extracts -score -XMLQUERY[3]	-contains -contains_with_score -XMLEXISTS[3]
画像検索	IMAGEFEATURE	FEATUREINDEX[4]	-IMAGEFEATURE	-GETDISTANCEIMAGEDATA -GETDISTANCEFEATUREDATA -GETFEATUREDATA	-SEARCHIMAGEDATA -SEARCHFEATUREDATA
空間検索	GEOMETRY	SPATIAL[5]	-GEOMETRY -GeomFromText -GeomFromWKB	-extracts -AsText -AsBinary	-Within -WithinRough -IntersectIn -IntersectInRough

注記
1) SGML文書及びXML文書の構造名を指定した検索も可能
2) n-gram方式のインデクスを実装
3) W3C XPath 1.0及びXQuery 1.0で規定
4) 多次元木方式のインデクスを実装
5) 四分木方式のインデクスを実装

従来のマルチメディア管理システムでは，メディア管理サーバと DB サーバとを独立に同時に管理することが一般的であった[18]．このような形態では，システム運用管理において，メディアデータと DB データとの間で整合性の矛盾がないように管理する必要があり煩雑であった[12]．

本論文では，マルチメディアデータを ORDBMS で一元管理することができるようになった．既存の RDB 資産を活かしたうえで，マルチメディアデータも管理することができる．一元管理することにより整合性を維持する運用が容易になる．さらに，マルチメディアデータを格納する単位は列であり，その列に回復処理を有するプラグインモジュールからなる UDT を対応付けるので，ORDBMS 全体として回復処理が実現できる．

(2)　SQL インタフェースで統一したマルチメディア操作

従来のメディア管理サーバと DB サーバを同時に利用するシステムでは，DB にアクセスする場合は SQL インタフェースで，メディアデータにアクセスする場合はサーバ独自のインタフェースを利用しなければならなかった．このような形態では，アプ

リケーション開発が煩雑であった．

本論文の RDB モデル拡張では，マルチメディアデータのアクセスを SQL インタフェースに統一することにより，アプリケーション開発が容易になる．

また，オブジェクト指向モデルを取り入れることにより，オブジェクト指向のアプリケーションとの親和性が高く，生産性の向上を図ることができる．本論文で示す ORDBMS は，データベース言語 SQL の標準仕様[2] に準拠し，利用者定義型に基づく型継承，多重定義を実現する．

表 3 では，各プラグインで提供されるデータ型で提供されるルーチン群を示す．アプリケーションから同一の操作概念については，総称的に同じルーチン名を付けることでオブジェクト指向のアプリケーションとの親和性に配慮した．たとえば，全文検索，文字列検索，XML 検索については，いずれもテキストを対象にするので，検索条件 contains および contains_with_score，ランク値算出 score は，同じルーチン名としている．また，検索した対象を取り出す出力操作 extracts については，全文検索，文字列検索，XML 検索に加えて，空間検索でも，同じルーチン名としている．

4.2　プラグイン機構について

これら全文検索，文字列検索，XML 検索，画像検索，および空間検索に対応するプラグインモジュールを DB システムに組み込んだ結果から，プラグイン機構の適用性を確認した．

- 本論文で提案するプラグイン IDL を基に，プラグインの外部仕様を規定してプラグインモジュールを開発した．
- ルーチンをプラグインの実装関数として UDT で定義し，ルーチン呼び出しを含む SQL を実行した．ORDBMS と密に連携したプラグインの実行制御を確認することができた．
- 複数の BES による並列 DB 環境で，プラグインを並列に実行することができた．
- PPI が提供する回復機能により，プラグインから DB リソースをアクセスすることでデータベースの回復処理を実現した．
- データプラグインモジュールおよびインデクスプラグインモジュールを分離して開発し，それぞれ独立に組み込んで実行した．また，インデクスプラグインモジュールの共有を図ることができた．

具体的には，プラグインを，メディア操作に対応する機能を ORDBMS に追加するデータプラグインモジュールと，メディアに対する高速検索機能を追加するインデクスプラグインモジュールに分類し，それぞれについて定義を規定した．データプラグインモジュールの外部仕様は，UDT 定義によって規定され，その実装はプラグインモジュールと呼ばれるプログラムによって行われる．一方，インデクスプラグインモジュールは，利用者から直接的な呼び出しインタフェースを持たないが，インデクスの保守のために規定する実装関数を，プラグイン IDL を用いてインタフェースの記述を行う．

また，プラグイン IDL では，UDT 定義中でプラグインでの実装を指定されているルーチンが呼び出された場合，プラグインモジュールのどの実装関数が呼び出されるかを指定できるような仕様とした．さらに，それぞれの実装関数が呼び出される際に ORDBMS の実現に必要な引数をプラグイン IDL で指定する仕様とした．この場合，UDT 定義で指定されたルーチン呼び出しのほかに，更新処理およびシステム制御系の呼び出し（トランザクションの開始終了時，セッションの開始終了時，回復処理の開始終了時，インデクス保守時）の定義を指定する仕様とした．

表 3 では，各プラグインで提供されるデータ型とそのインデクス実装との対応付けを示す．たとえば，全文検索，文字列検索，XML 検索については，いずれもテキストを対象にするが，それらのテキストを対象にする検索を高速化する手法はいずれも n-gram 方式で実装されている．もし，データプラグインモジュールおよびインデクスプラグインモジュールを分離せずに開発すると，同じ n-gram 方式を採用しても個別の実装が必要となる．一方で，データプラグインモジュールおよびインデクスプラグインモジュールを分離することで，実装の独立性が高まり，拡張性も容易となることが期待できる．

一方で，いくつかの考慮点が考えられる．

(1)　プラグインモジュールは，DB 処理と同一のスレッドで実行されるので，スタックサイズの制約，システムコールの発行制限に配慮した難易度の高いスレッドプログラミングが要求される．

(2)　データベースの回復処理に対応したプログラム設計および実装を必要とされる．

(3)　ORDBMS の実行環境中でプラグインモジュールを DB 処理スレッドと同一のスレッドで実行するので，DB 処理のプロセスサイズが増大するため，スループットへの影響を測る必要がある．

5. 関 連 研 究

(1)　ルーチン呼び出し方式の高速化について

ORDB は，従来の RDB をベースとし，オブジェクト指向モデルを取り入れた DB である．主要な RDBMS ベンダは ORDBMS を開発し，製品化している[1),13),17)]．この ORDBMS は，データベース言語 SQL の標準仕様に準拠する形で実装が行われ

ている[14]．SQL データベースの上にオブジェクト指向機能を実現するために利用者定義型が導入されている．利用者定義型では，値の振舞いを規定する操作を関数として定義ができ，これらの操作はルーチンとして指定できる．ルーチンのインタフェースは利用者定義型の中で宣言され，そのルーチンの実装は利用者定義型とは別にルーチン定義によって行うことができる．具体的には，ルーチンの実装は SQL 起動ルーチンあるいは外部起動ルーチンである．

DB2 [⋆1,13] は，外部起動ルーチンが実行するアドレス空間を DBMS と同じアドレス空間にするのか，あるいは異なるアドレス空間にするのかを選択できる．利用者定義関数の定義時に，FENCED あるいは NOT FENCED を指定する．NOT FENCED を指定すると，当該関数は DBMS と同じアドレス空間で実行される．しかし，関数の呼び出しが高速化される利点を有するものの，関数の動作で引き起こされる突発的な不具合からデータベースが破壊される可能性がある．そのため，NOT FENCED を指定する利用者定義関数は，DBA 権限を有する利用者が定義可能とし，権限管理の配下に置く．また，利用者定義関数のテスト段階では，DBMS と異なるアドレス空間で実行する FENCED を指定し，正しく動作することが信頼できれば，NOT FENCED を指定することを推奨している．

提案方式では，性能を維持するために外部起動ルーチンを DBMS と同じアドレス空間で実行するが，一方で回復処理を記述する外部起動ルーチンを実行することにより引き起こされる不具合でデータベースを破壊する可能性がある．信頼性の確保および性能向上は，トレードオフの関係にある．

(2)　機能の組み込みの容易化について

Oracle [⋆2] データカートリッジ[17] の実装において，インデクスと表データとの一貫性の維持，ファイルおよびデータベースとページなど永続化記憶との間の管理（トランザクション，バックアップ，データベース回復，記憶領域割当て）に留意することが示されている．また，静的変数を使用せずにスレッドセーフに関数を記述し，OS システムコールは発行しないなどのコーディング規則が明記されている．

Informix [⋆3] は，DataBlade [⋆4] と呼ぶモジュールを追加する機構を有する[1]．種々のマルチメディアに対応する DataBlade を有する．この DataBlade ではアクセスメソッド実装で，インデクスに対するロック確保，解放によって，DBMS のロック管理と連携が可能である．また，データを回復可能とする，ログ管理との連携も可能である．さらに，アクセスメソッドでは，ディスク上のページを扱うために，DBMS のバッファ管理と連携が行われ，当該バッファへのページ書き込みを制御することも可能である．また，インデクスによるアクセスメソッド実装には，インデクススキャンの開始，スキャン中のレコード取り出し，レコード操作（挿入，削除，更新），インデクススキャンの終了など，12 個の関数が用意される[1]．

MySQL [⋆5] は，Pluggable Storage Engine Architecture でストレージエンジン API を公開しており，InnoDB [⋆6] など組み込まれたエンジンでの稼動実績がある[15),16]．このストレージエンジン API は，RDB で提供する既定義型に対して，最適化で生成する実行プランを実行解釈する処理で各行レベルのデータ操作およびインデクス操作を規定している．ただし，DB エンジン全体を組み込む作業を前提にした API 仕様であり，マルチメディアに対応したデータ型拡張，DB 回復の処理規定，および柔軟にモジュールを追加できる機構とはなっていない．

いずれも，API 仕様，およびマニュアルに記載されるコーディング規則に従って実装するしかないが，提案方式では，DB リソースを利用してデータベースへの操作，トランザクション制御およびデータベース回復処理に対応した順序，タイミングの処理契機に応じる処理の記述を可能とすることで難易度が高いと考えられるプラグインモジュールの組み込みを簡単にしている．

6. おわりに

本論文では，様々なメディアデータの多様化かつ拡張に対応可能なオブジェクト・リレーショナルデータベース管理システムを開発し，オブジェクト指向アプリケーションと親和性の高い SQL インタフェースを提供することで，先進的なマルチメディア管理技術を取り込んで容易に拡張可能な DB システム基盤を提供することができた．具体的には，下記の特長を有するプラグイン機構を実装した．

⋆1 DB2 は，米国およびその他の国における International Business Machines Corporation の商標です．
⋆2 Oracle および Oracle Database 10g は，Oracle Corporation およびその子会社，関連会社の米国およびその他の国における登録商標または商標です．
⋆3 Informix は，米国およびその他の国における International Business Machines Corporation の商標です．
⋆4 DataBlade は，米国およびその他の国における International Business Machines Corporation の商標です．
⋆5 MySQL は，Oracle Corporation およびその子会社，関連会社の米国およびその他の国における登録商標または商標です．
⋆6 InnoDB は Innobase Oy 社の商標です．

- プラグイン実装のルーチン
- プラグイン・インタフェース定義
- プラグイン・プログラミング・インタフェース
- DB と同一スレッドでのプラグイン実行
- データプラグインモジュールとインデクスプラグインモジュールの分離
- プラグインオプションとレジストリ表

また，全文検索，文字列検索，XML 検索，画像検索，および空間検索のプラグインを開発し，これらの技術の適用性を確認した．

(1) マルチメディアデータ管理の一元管理マルチメディアデータを DB サーバで一元管理することができた．既存の RDB 資産を活かしたうえで，マルチメディアデータも管理することができた．一元管理することにより整合性を維持する運用が容易になった．さらに，ORDBMS の回復機能により高信頼なデータ管理が可能になった．

(2) SQL インタフェースで統一したマルチメディア操作マルチメディアデータのアクセスを SQL インタフェースに統一することにより，アプリケーション開発が容易になった．また，オブジェクト指向モデルを取り入れることにより，オブジェクト指向のアプリケーションとの親和性が高く，生産性の向上を図ることができた．今後，マルチメディアデータを順次データベースシステムに取り込むことがますます要望され，このようなマルチメディアデータの追加が容易になることが期待できる．

参 考 文 献

1) Stonebraker, M.: *Object-Relational DBMSs The Next Wave*, Morgan Kaufmann Plublishers (1996).
2) ISO/IEC 9075-2 Database Language – SQL Part 2: Foundation (SQL/Foundation) (2008).
3) 鳥居俊一，根岸和義：高拡張性を目指した並列 RDB サーバ，電子情報通信学会技術研究報告，DE94-49, pp.41–48 (1994).
4) 日立製作所：共通マニュアル スケーラブルデータベースサーバ HiRDB Version 5.0（Object Option 付）(1998).
5) 鳥居俊一，加藤寛次，正井一夫ほか：マルチメディアデータベース基盤技術の開発，平成 9 年度次世代電子図書館システム研究開発事業論文集，pp.11–16 (1998).
6) 鳥居俊一，加藤寛次，松澤　茂ほか：マルチメディアデータの一元管理を実現する HiRDB Universal Server，日立評論 1998 年 5 月号 (1998).
7) Torii, S., Kato, K. and Masai, K.: Integrated Multimedia Database, *Hitachi Review*, Vol.47, No.6, pp.296–299 (1998).
8) 岩田守弘，金子英司，中野幸生ほか：ORDB における多重定義関数の呼び出し処理の高速化，情報処理学会第 58 回全国大会講演論文集 2T-3 (1998).
9) 小林　挙，土田正士，山本洋一ほか：ORDB におけるプラグイン組込み仕様と実行制御，情報処理学会第 58 回全国大会講演論文集 2T-2 (1998).
10) 原　憲宏，河村信男，土田正士ほか：ORDB におけるメディア対応ライブラリ組込みのためのアーキテクチャ提案，情報処理学会第 58 回全国大会講演論文集 2T-1 (1998).
11) Gray, J. and Reuter, A.: *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publiskers (1993).
12) 土田正士，小寺　孝：SQL2003 ハンドブック，ソフトリサーチセンター (2004).
13) Chamberlin, D.: *Using the New DB2: IBM's Object-Relational Database System*, Morgan Kaufmann Publiskers (1996).
14) 土田正士，小寺　孝，芝野耕司：SQL の 20 年と現状および今後の展開（後編），情報処理，Vol.45, No.6, pp.624–630 (2004).
15) MySQL 5.1 Reference Manual: Chapter 13. Storage Engines, available from ⟨http://dev.mysql.com/doc/refman/5.1/en/index.html⟩ (accessed 2010-09-20).
16) MySQL 5.0's Pluggable Storage Engine Architecture (White Paper, October 16, 2005), available from ⟨http://dev.mysql.com/why-mysql/white-papers/⟩ (accessed 2010-09-20).
17) Oracle Database データカートリッジ開発者ガイド 10g リリース 2(10.2) B19251-02, available from ⟨http://otndnld.oracle.co.jp/document/products/oracle10g/102/⟩ (accessed 2010-09-20).
18) Melton, J., Michels, J.E., Josifovski, V., et al.: SQL and Management of External Data, *SIGMOD Record*, Vol.30, No.1, pp.70–77 (Mar. 2001).

(平成 22 年 9 月 20 日受付)
(平成 22 年 12 月 13 日採録)

(担当編集委員　市川 哲彦)

土田　正士（正会員）

株式会社日立製作所ソフトウェア事業部先端情報システム研究開発部担当部長．1983 年筑波大学大学院理工学研究科修了後，同年株式会社日立製作所システム開発研究所を経て，2003 年より現職．著書に『SQL スーパーテキスト』（技術評論社），『SQL2003 ハンドブック』（SRC 社）．訳書に『SQL：1999 リレーショナル言語詳解』（J. メルトン著，共訳，ピアソン・エデュケーション），『標準講座 XQuery』（J. メルトン，S. バクストン著，共訳，翔泳社）．日本データベース学会理事，ISO/IEC JTC 1/SC 32 WG3 日本代表．情報処理学会情報規格調査会 SC32 専門委員会幹事および SC32/WG3(SQL) 小委員会委員，2005 年同標準化貢献賞受賞．日本データベース学会，ACM 各会員．

河村　信男（正会員）

株式会社日立製作所ソフトウェア事業部先端開発プロジェクト室主管技師．1981 年愛媛県立松山工業高等学校情報技術科卒業後，同年株式会社日立製作所システム開発研究所を経て，2002 年より現職．

中野　幸生（正会員）

株式会社日立製作所ソフトウェア事業部先端情報システム研究開発部主任技師．1982 年香川県立多度津工業高等学校電子科卒業後，同年株式会社日立製作所システム開発研究所を経て，2003 年より現職．

原　　憲宏

株式会社日立製作所ソフトウェア事業部 DB 設計部主管技師．1992 年東京工業大学理学部情報科学科卒業後，同年株式会社日立製作所システム開発研究所を経て，2004 年より現職．

石川　　博（フェロー）

静岡大学情報学部情報科学科教授．東京大学理学部情報科学科卒業．東京都立大学を経て 2006 年より現職．東京大学博士（理学）．著書に『次世代データベースとデータマイニング』（CQ 出版社），『JavaScript によるアルゴリズムデザイン』（培風館），『データベース』（森北出版）等．国際的論文誌 ACM TODS，IEEE TKDE，国際学会 VLDB，IEEE ICDE 等を含め学術論文多数．1994 年情報処理学会坂井記念特別賞，1997 年科学技術庁長官賞（研究功績者）受賞．情報処理学会データベースシステム研究会主査，情報処理学会論文誌（データベース）共同編集委員長，International Journal Very Large Data Bases Editorial Board，日本データベース学会理事歴任．電子情報通信学会フェロー．ACM，IEEE 各会員．