分散形データベース管理システムDEIMS-3†

鈴木健司†† 田中豪†† 村田達彦†††
岸本義一†† 伊藤健治†††

本論文では，分散形データベース管理システム DEIMS-3 (DEndenkosha Information Management System version 3) の実現方式および構成について述べる．DEIMS-3 は，大容量高トラヒックな大規模実時間データベースシステムへの適用をめざし，集中形データベース管理機能に対し，以下の分散形データベース機能を拡充している．(1)ネットワークアーキテクチャとして，DCNA のデータベースアクセスプロトコルを採用し，異種データベースとの接続への拡張性のあるスキーマ構成の実現，(2)アプリケーションに対するユーザインタフェースとして，CODASYL 仕様のデータベース操作言語の高水準化をはかり，通信効率のよい分散問合せ処理の実現，(3)ノード間のデータベースアクセスの競合対策として，時間監視による実行効率のよい排他制御の実現，(4)ノード間のデータベース更新に対する障害時の無矛盾性保証として，ノード間の更新同期および整合処理の実現．これら分散形データベース管理機能の実現により目的とする大規模データベースシステムの所要性能を達成することができている．また，これら機能の実現にあたっては，集中形データベース管理機能を実現するソフトウェアからの独立性の高いソフトウェアとして実現している．

1. まえがき

データベースシステムは計算機化対象業務の拡大，データ量の増大につれて大規模化の傾向にある[1]．このような大規模なデータベースシステムを構成する方法として，データベースを地理的に分散配置し，処理することができる分散形データベースは有力な手段の一つである．

分散形データベースシステムの研究としては，これまでに SDD-1[2]，R*[3] など種々報告されている[4]．これらの研究は，主に関係（リレーショナル）モデルに基づくデータベースをもとに，問合せの最適化等が検討されているが，現実に要請がある大規模データベースシステムの構築の観点からの適用性の検討および実現をはかったものはない．

筆者らは，これまで大容量高トラヒックな実時間システムに向けたデータベース管理システム（以下，DBMS）として，CODASYL 仕様[5],[6] に準拠した集中形 DBMS (DEIMS-2) を開発してきた[7]~[9]．この CODASYL 仕様に基づく DBMS は現状技術で高性能なデータベースを実現できるという特長を有し，ほかにも多数商用化されている．本稿では，この CODASYL データベースの特長を活かし，大規模なデータベースシステムに適用するために，集中形 DBMS を分散形 DBMS に拡張した DEIMS-3 の実現方式および構成について述べる．

2. 設計目標

DEIMS-3 が最初に適用対象とするシステムは，データベースの容量が数百 GB 程度，オンライン実時間での処理トランザクションが秒当り約 100 件という定形的な業務処理を数個の分散処理ノードで構成するシステムである．

このようなシステムを実現するために設定した設計目標は以下のとおりである．

（1） 分散形データベース設計の柔軟性

当面の適用対象システムは，全体設計による分散形データベース構成（トップダウン設計）で充足できるが，既存のデータベースを接続し統合化する分散形データベース（ボトムアップ設計）へも拡張性がある構成を実現する．

（2） 通信形態の柔軟性

グローバルネットワークによる地理的な分散と，センタ内のホストおよびバックエンドプロセッサによる機能分散のそれぞれ異なる通信形態に対し，同時に両立しうる統一的な構成を実現する．

（3） アプリケーションの分散不可視化

アプリケーションインタフェースに対し，分散配置を意識させずにデータベースのアクセスを可能とする分散不可視化を実現する．

（4） 高性能化

トランザクション処理における端末の応答時間は，分散処理を含めて通常 3 秒以内を可能とする通信効率

† Distributed Database Management System DEIMS-3 by KENJI SUZUKI, TAKESHI TANAKA, TATSUHIKO MURATA, GIICHI KISHIMOTO (NTT Information Processing and Communication Laboratory) and KENJI ITOH (NTT Data Communication Bureau).

†† NTT 情報通信処理研究所
††† NTT データ通信本部

を実現する．

（5） 高信頼性

分散処理ノードのデータベース障害の波及範囲は極力ローカライズし，全処理ノードの停止とならないような高信頼な分散形データベースシステムを実現する．

（6） ソフトウェア構成の独立性

上記の分散形データベース管理機能の実現にあたっては，集中形 DBMS のソフトウェアからの独立性の高いソフトウェアとして構成し実現する．

3. 実現課題と実現方式

集中形 DBMS を拡張し，2章で述べた設計目標を満たす分散形 DBMS を実現するためには，大別して，データベースの記述情報，すなわちスキーマ構成の面と，データベースのアクセス制御の面から拡張をはかる必要がある．以下，これらの実現課題と実現方式について述べる．

3.1 分散形データベースのスキーマ構成

分散型データベースのスキーマ構成は，ネットワークにおよぶシステム全体の論理的なデータ構造の記述とデータ配置情報の記述を必要とするグローバルなレベルと，各ノードにあるデータベースの記述のローカルなレベルを必要とする[10),11)]．

DEIMS-3 では，トップダウン設計に基づくデータベースを同種の DEIMS-3 間で分散処理する形態を最初の実現範囲とするが，ボトムアップ設計および異種 DBMS との結合を可能とする拡張性を考慮し，以下に示すスキーマ階層とそのデータモデルを設定している[12)]．これらのスキーマ構成を**図1**に示す．

（1） ローカル内部スキーマ

各ノードに存在するデータベースの記述であり，集中形 DBMS である DEIMS-2 により実現されている論理スキーマ，格納スキーマ，物理スキーマを対応させる[8)]．ここで論理スキーマは CODASYL 仕様に基づくネットワークモデルで記述される．格納スキーマはデータベースの物理的な構成要素である配置法，インデックス付け法，ポインタ種別の設定法などの性能に関する要素を記述する．これにより論理スキーマの物理的データ独立性を高めるとともに大容量高トラヒックシステムに向けたデータベースの物理的な記述が可能である．物理スキーマはオペレーティングシステム (OS) の管理する記憶空間へのマッピング方法(ファイル)を記述し，格納スキーマの OS からの独立性を高める．アプリケーションに対する仮想スキーマとして，後述するように分散処理時の変換効率および通信効率を考慮し，高水準スキーマを新設している．これは集中形環境下でも使用できる．

（2） ローカルスキーマ

自ノードのローカル内部スキーマをネットワークに対する見せ方（データビュー）として記述するものである．ローカルスキーマの目的は，各ノードの DBMS の違い，あるいはデータモデルの違いに

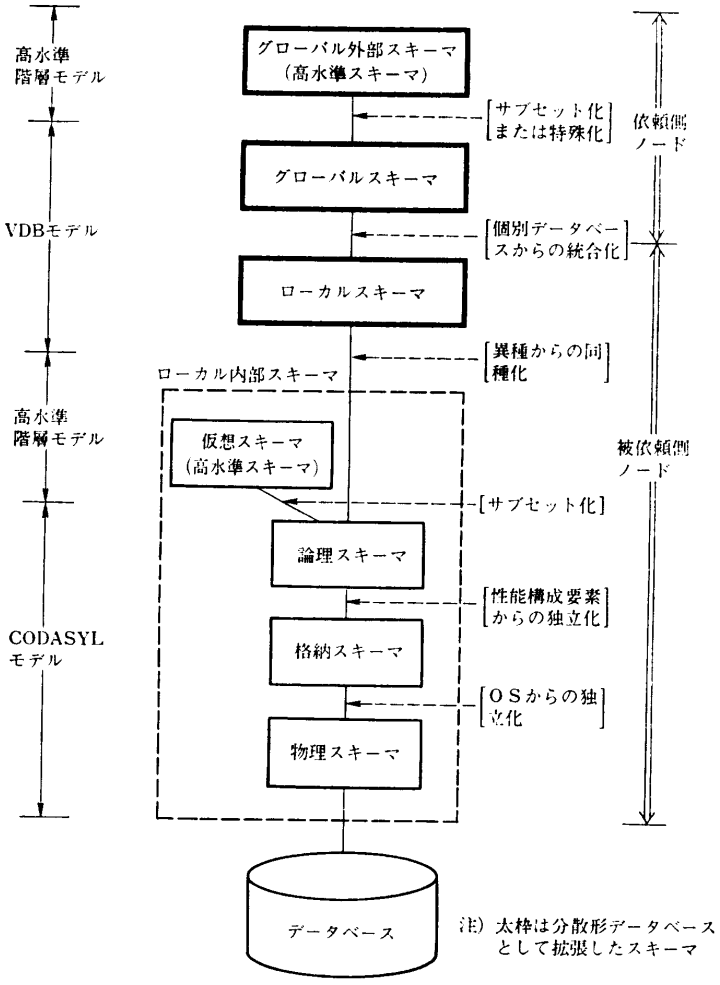


図 1 DEIMS-3 のスキーマ構成
Fig. 1 DEIMS-3 schema architecture.

よるデータベースの異種性を排除し，均質化することである．さらに，ローカル内部スキーマのデータビューの部分的な切り出しを可能とすることである．このローカルスキーマに対するデータモデルとしてDCNA のデータモデル[13]を設定している．

DCNA のデータベースアクセスアーキテクチャは，データ通信網内に存在する種々のデータベースの共用を可能とするために，仮想化されたネットワーク上に各ノードの異種データベースシステム間で共通に認識できる仮想的なデータベース (VDB) を設定し，それに対する標準的な操作法をデータベースアクセスプロトコル (DBAP) として規定している．この VDB のデータモデルは階層形，ネットワーク形，関係形の各モデルの表現要素を包含した統合的なモデルである．またそのデータベース操作は，1回の操作で複数のレコード処理を可能とする高水準な操作コマンドを規定している．

このローカルスキーマの最初の実現範囲は，(4)で述べるグローバル外部スキーマとの変換効率を考慮し，階層形に限定している．そして，DEIMS-3 間による同種の分散形データベースを構成する場合には，ローカル内部スキーマの高水準スキーマにより代替えできるため，ローカルスキーマを不要としている．

(3) グローバルスキーマ

複数のローカルスキーマの統合，あるいは分割を可能とするためのシステム全体のビューの記述である．さらに，データがどのノードに配置されているかを示す分散配置情報を記述する．このグローバルスキーマのデータモデルとして，ローカルスキーマとの変換効率を考慮し VDB モデルとしている．

(4) グローバル外部スキーマ

グローバルスキーマからアプリケーションのデータビューに適合するように，またアプリケーションに必要な部分的なデータビューの記述である．グローバル外部スキーマのデータモデルは，集中形 DBMS からの連続性を考慮すると CODASYL ネットワークモデルとなるが，そのデータベース操作言語 (DML) は，分散環境では以下のような変換効率と通信効率の問題がある．一つはグローバルスキーマに基づく DCNA DBAP への低水準から高水準への変換と，ローカルスキーマからローカル内部スキーマへの高水準から低水準への変換を必要とすること，さらに，通信回数がアプリケーションの DML 発行回数に依存し，1操作1レコードを基本とする DML では通信回数が多くなることなどの問題がある．しかし関係モデルを選定する場合には，DCNA DBAP との整合性はよいが，ローカル内部のスキーマのネットワークモデルへの変換を必要とし，変換効率の問題が残る．そこで多くの定形的な業務処理では，処理効率のよい階層構造で充足できることから，CODASYL ネットワークモデルを階層構造と等価的な木構造に制約し，DCNA の DBAP との整合性のよい高水準な DML を3.3節で述べるように設定している．このデータモデルを高水準階層モデル，そのスキーマを高水準スキーマと呼んでいる．

3.2 分散配置制御

大容量高トラヒックな分散形データベースシステムを実現するためには，トランザクションに対する処理要求が，極力そのノードで完結するようなデータの分散単位を設定できることが必要である．

DEIMS-3 では，アプリケーションのデータビューとして階層構造を設定したことにより，この階層構造に着目し，各ノードの処理要求が該当ノードで完結できるような階層関係とオカレンスの配置を考慮している．すなわち，階層構造を形成するセットタイプの集り (VDB モデルでは，これをセットグループ (SG) タイプという) を論理的な分散単位とした．この SG タイプのオカレンスはデータベースの格納単位であるエリアに格納される．一つの SG タイプは複数のエリアに対応づけられるが，1SG オカレンスは，唯一つのエリアに閉じて格納される．したがって，分散単位は，論理的に重複配置される SG タイプと，物理的に非重複配置される SG オカレンスとエリアである．この分散単位の概念を図2に示す．図2の例では，顧客元帳セットグループタイプは階層構造を形成する普通預金セットタイプ，定期預金セットタイプ等々の一連のセットから構成され，これが論理的な分散単位となる．

分散配置情報の記述はグローバルスキーマにおいて，ノード情報とそれに対するエリア名，あるいは SG オカレンスを一意に識別するレコード中のエントリキーを分散キーとして記述する．これにより，アプリケーションはエリア名あるいはエントリキーを指定し，SG に基づく処理を行うだけでよく，分散配置のノードを意識することなくデータベースをアクセスすることができる．すなわち，アプリケーションの分散不可視化を実現できる．

3.3 ユーザインタフェースと分散問合せ処理

グローバル外部スキーマにおけるアプリケーション

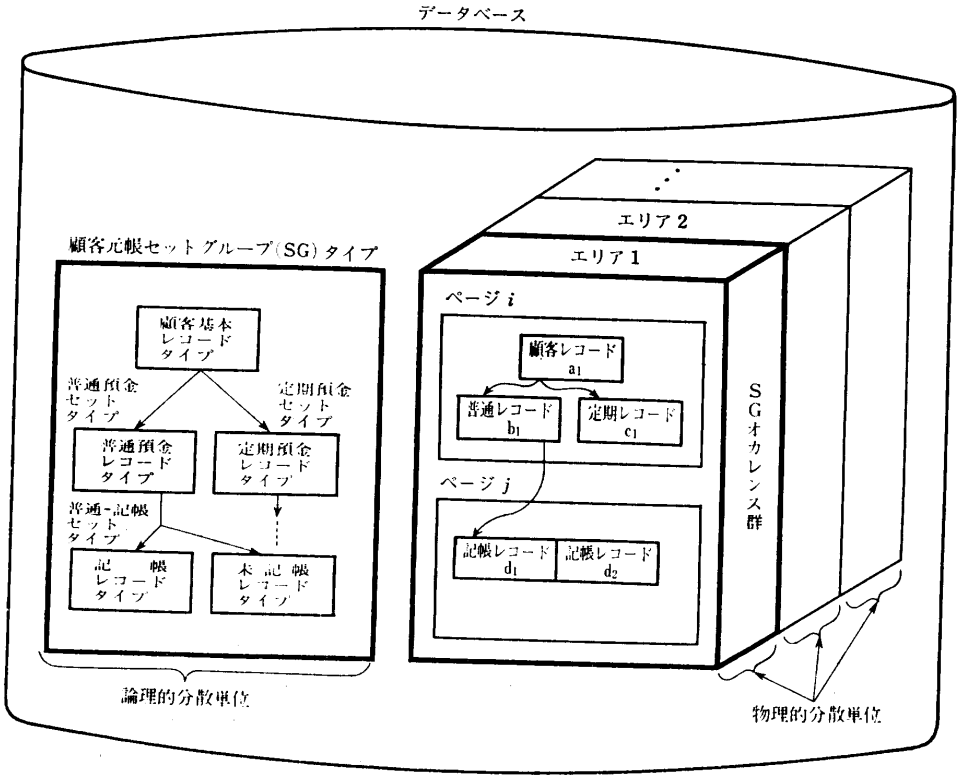


図 2 分散単位
Fig. 2 Distribution unit.

インタフェースとして，3.1 節(4)で述べた分散処理の通信効率の問題を解決するために，CODASYL 仕様の DML を次のように高水準化している[14),15)]．この言語を HDML と呼ぶ．

(1) レコード選択式の高水準化

HDML では，一つの SG オカレンス全体を操作の基本とし，一度の操作で SG オカレンスの選択と，その構成要素である複数のレコードを一括して扱えるように，DML のレコード選択式を高水準化している．この高水準なレコード選択式は，検索命令 (FETCH (Find & Get))，削除命令 (ERASE)，格納命令 (STORE)，更新命令 (MODIFY) で使用でき，セット構造をたどる操作を不要とし，値によるレコードの結合，比較等の操作を可能とする．その結果，AP と DBMS 間でレコードの授受を行うためのユーザ作業域 (UWA) を，1レコード当り任意数のレコードオカレンスを授受できるように拡張している．

さらに，分散処理の通信効率を高めるために，以下の機能を実現している．

(2) HDML の複合化

高水準化された HDML は，複数の命令を連続一括して実行できる場合が多く，分散処理を考慮したとき，これを一括して転送した方が通信効率がよくなる．しかし，AP に対し通信を意識した記述は，分散不可視の観点からは問題がある．そこで，一括転送という概念ではなく，AP からみて，一括してデータベース操作を実行し，結果も最終結果をまとめて知ることでよいという HDML の複合化した実行単位の概念を設けている．この複合化された HDML は，DCNA の対応する DBAP コマンドに変換され，さらに DCNA の重畳コマンド (SUPERPOSE) を用いて，これらのコマンド群は (SUPERPOSE コマンドに対するサブコマンド群として) 一括転送される．

(3) 分散処理の最適化

HDML の操作対象が一つの SG タイプの場合でも，SG オカレンスは複数ノードに配置されるため，レコード選択式の条件によっては操作の対象ノードが複数になることがある．このとき各ノードでの処理は互いに独立して実行可能であるから，各ノードへ並行して処理依頼を行い通信時間の短縮をはかっている．

また，異なる SG タイプ間の結合条件を指定されたとき，その SG オカレンスが異なるノード間に配置さ

れているときには，ノード間にわたる結合条件処理が必要になり，通信効率が課題になる．これに対するDEIMS-3の実現方式は以下の通りである．最初にデータベースアクセスを依頼されたノード（被依頼側ノード）では，依頼された結合処理を実行し，結合できなかったものまで含めて結果をアクセスを依頼したノード（依頼側ノード）へ返却する．そして依頼側ノードでは，結合が未完のものについて再度選択式を作成し，別の被依頼側ノードへ結合処理を依頼する．本方式では通信回数は2回であり，選択結果のデータが多いと転送量が増大する可能性があるが，データ配置に局所性が考慮されている定形処理の実時間システムでは，通常の結合処理は1ノード内で完結する可能性が高いことから実用性は十分である．

3.4 排他制御

集中形DBMSのDEIMS-2では，トランザクションでのデータベース資源の競合制御をロック方式により実現している．このロック対象となる資源は，アプリケーションで宣言されるエリアの使用モードにより決まり，エリアで明示的にロックするか，あるいは暗に対象となるレコードが存在するページをロックするかの方法がある．本ロック方式は，ある資源のロックが要求されたとき，すでに他のトランザクションによりロックされていれば，そのままウェイトさせてもデッドロックにならないかをチェックし，ならなければウェイトし，なればエラーリターンしロールバックさせる方法である．

このロック方式を非重複の分散形データベースに拡張する実現方式としては，他ノードでのロック情報を収集し，グローバルなデッドロックのチェックを行う方法[16]があるが，ロック情報の転送量，チェック量の増大による処理時間の問題があり，大容量高トラヒックの実時間システムでは限界がある．

そこで，DEIMS-3では，各ノードが単独で処理でき，しかもロジックが単純でオーバヘッドの小さい時間監視方式を実現している．これはウェイトする場合に時間監視を設定してウェイトさせ，一定時間を経過しても応答がない場合には，グローバルなデッドロックが発生したものとみなし後処理する方式である．

3.5 更新同期制御

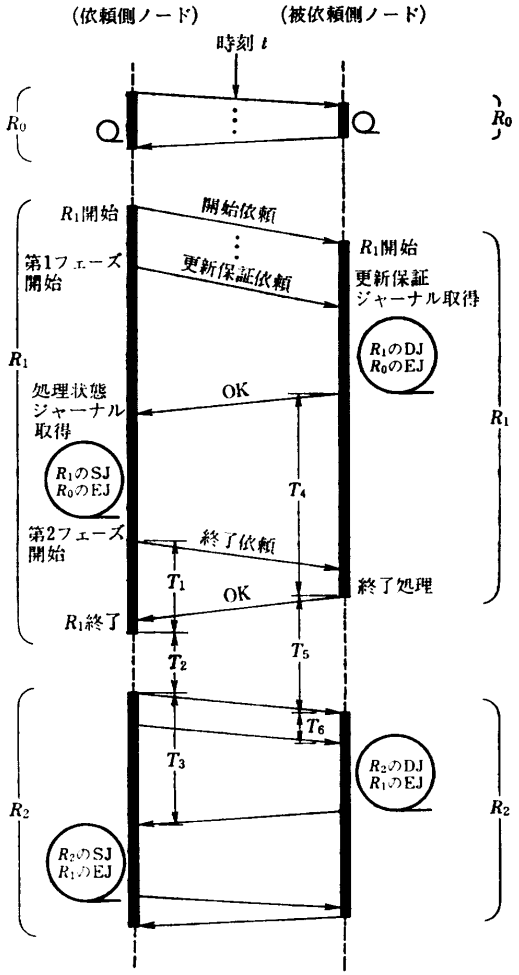
トランザクション処理におけるデータベース更新を保証するために，DEIMS-2では，更新を保証する論理的な処理単位をアプリケーションで宣言できるようにしている．これを救済単位といい，開始命令(RESCUESTART)と終了命令(RESCUEEND)で囲んだ1個以上の命令からなる．この救済単位をもとに，アプリケーションプログラムの論理矛盾，デッドロックによるエラー処理などが生じたとき，直前の救済単位までのデータベースの復旧を可能にしている．また，前述したページ単位の排他制御もこの救済単位と同期して行われる．

救済単位におけるデータベース更新を保証するための処理を更新保証処理といい，これは救済単位内と終了時がある．救済単位内では，メモリ上でデータベースの仮更新を行い，データベースの内容は救済単位開始状態を保証する．救済単位終了時では，次の2フェーズで実更新保証処理を行う．第1フェーズでは，データベースの更新保証として更新後情報のジャーナル取得を行う．第2フェーズでは，ジャーナル取得がすべて正常に完了していればデータベースの実更新処理を行うが，異常に終了していればデータベースの実更新を行わずエラーリターンし，アプリケーションの指示でロールバック等を行う．そして，救済単位の正常／異常終了情報のジャーナル取得を行う．

分散形データベースではノード間に更新が及ぶため，ノード間にわたる更新同期制御が必要になる．前述の救済単位における更新保証処理で，ノード間の同期が必要になるのは救済単位終了時の2フェーズ処理時であり，これは分散処理では2フェーズコミットといわれている．DEIMS-3では，この2フェーズ処理をノード間に拡張している．すなわち，第1フェーズでは，更新を行ったすべての被依頼側ノードに更新保証処理を要求し，すべてのノードの終了同期をとって第2フェーズの処理を依頼する．このことにより基本的なノード間の更新同期保証を実現している．

DEIMS-3では各ノードは従属関係を持たず対等なことから，更新保証処理に必要なジャーナル取得は実更新を行うノードでそれぞれ行われる．しかし，ある被依頼側ノードがプロセッサ障害や通信系障害により他ノードと切断された場合，2フェーズコミットに基づく自ノードのデータベース更新ジャーナル情報だけでは，復旧処理の判断が困難な場合がある．例えば，ある被依頼側ノードでは，第1フェーズの更新保証処理後であっても第2フェーズの実更新処理が終了する前にプロセッサ障害が生じた場合には，救済単位終了ジャーナルが取得されていないので，他被依頼側ノードのすべてがどのような状態で第1フェーズを終了しているかどうかわからないため，自ノードの第2フ

ェーズの処理内容を確定することができない．これを解決するためには，被依頼側ノードは，依頼側ノードのグローバルな救済単位の処理状態の情報を得て，被依頼側ノードのローカルな救済単位の処理状態と整合をはかる必要がある．そこで DEIMS-3 では整合をはかるために必要な情報として，依頼側ノードでは，救済単位開始処理の依頼時，第1フェーズの終了時，第2フェーズの終了時に処理状態情報のジャーナル取得を可能にしている．ただし，ジャーナル取得回数を削減するために，**図3**に示すように取得の最適化を行っている．すなわち，同一タスクにおける救済単位はジャーナル上では直列化されていることから，救済単位の開始情報は直前の救済単位の終了情報により識別できるため取得を省略する．さらに第2フェーズ時の終了情報のジャーナル取得は，直後の救済単位の第1フェーズ時の処理状態情報のジャーナル取得（依頼側ノード）あるいは更新保証情報のジャーナル取得（被依頼側ノード）と合わせて取得する．これにより各ノードでの救済単位におけるジャーナル取得回数を1回とすることができる．また，図3の例の整合処理は次のようになる．



注）R_i：救済単位
DJ：データベース更新保証ジャーナル
EJ：第2フェーズ終了状態ジャーナル
SJ：第1フェーズ処理状態ジャーナル

図 3 救済単位とジャーナル取得処理
Fig. 3 Rescue unit and journal processing.

依頼側ノードのプロセッサ障害の場合には（期間 T_1〜T_3），依頼側ノードは自ノードのジャーナル情報（R_1 の SJ, R_0 の EJ）でのみ状態を決定することができ，救済単位 R_1 は有効，R_2 は無効にする必要があることがわかる．被依頼側ノードのプロセッサ障害の場合には（期間 T_4〜T_6），依頼側ノードの処理状態の問合せが必要であり，その結果，期間 T_4 では依頼側ノードの R_1 の処理状態に整合させる必要があることが，T_5 では整合を必要としないことが，T_6 では R_2 を無効にする必要があることがわかる[17]．

以上の処理により，いかなる障害に対してもノード間の更新同期の保証を可能にしている．

4. システム構成

4.1 ソフトウェア構成

3章で述べた方式を実現する DEIMS-3 のソフトウェア構成を**図4**に示す．分散形データベース処理のために新規に作成したソフトウェアは，主に言語処理プログラム，実行制御プログラムである．

言語処理プログラムは，グローバルスキーマ定義言語および COBOL 用高水準スキーマ定義言語で記述された各々のスキーマを生成するジェネレータ，アプリケーションプログラムでのデータベース操作のための COBOL 用 HDML を処理するプリコンパイラを追加している．なお，高水準スキーマ定義言語と HDML は集中形データベース処理においても使用できる．

データベースモニタ（DBM）と呼ぶ実行制御プログラムは，従来の集中形データベース実行制御機能に対し，分散型データベース実行制御機能を独立した機能として付加できるようなプログラムで構成した．したがって，集中形データベース実行制御機能を実現するローカル DBM (LDBM) に加え，分散形データベース実行制御機能を実現するグローバル DBM (GDBM) を新規に作成している．

GDBM はシステムジェネレーション時に選択でき

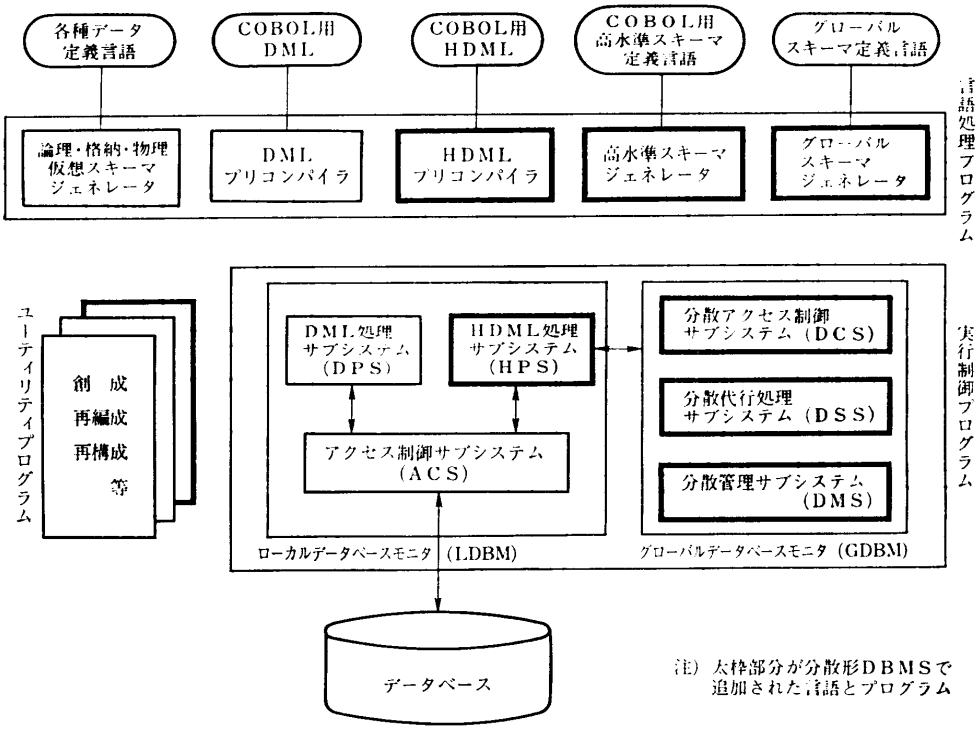


図 4 DEIMS-3 のソフトウェア構成
Fig. 4 DEIMS-3 software configuration.

るソフトウェアであり，次の三つのサブシステムから構成される．依頼側ノードにおいて，他ノードへのデータベース操作の実行制御を行う分散アクセス制御サブシステム (DCS)，被依頼側ノードにおいて，依頼側からのデータベース操作の代行処理を行う分散代行処理サブシステム (DSS)，依頼側，被依頼側の両ノードにおいてシステム状態の管理を行う分散管理サブシステム(DMS)である．

LDBM では，DML 処理サブシステム (DPS)，アクセス制御サブシステム (ACS) に加えて，HDML 処理サブシステム (HPS) を追加している．DPS, HPS はそれぞれ DML, HDML の解釈と実行を行い，ACS はローカルなデータベースアクセスの実行制御として，記憶管理，排他制御，入出力制御，障害制御などを行う．HPS, ACS は分散処理環境では必須である．

4.2 実行時システム構成

実時間システムにおける分散形データベースアクセス時のシステム構成を**図 5** に示す．分散形データベースの処理環境において，新たに必要な処理を以下に述べる[18]．

データベースへのアクセスを行う DBM は，アプリケーションプログラム (AP) が走行するタスクの延長上で走行するプログラムである．したがって，被依頼側ノードでも，依頼側ノードの AP の代行処理が走行するためのタスクを必要とする．これらのタスクは，通常実時間システムの開始処理時に，アプリケーションサイドのサービス管理プログラム (SMP) により，一括して生成しておく必要がある．

さらに，依頼側，被依頼側のノード間で，DCNA のコマンドを授受するためには，DCNA で規定される情報処理フィールド間の論理的な通信路（Fパス）を設定する必要がある．このFパスの設定は，ノード間のトラヒック，障害時のパスの切替えの柔軟性などから，SMP が分散通信プログラムの機能を利用して行う．ここで分散通信プログラムは，ノード間の通信回線，チャネルなどの通信路の物理的特性に依存したアクセス法をさらに仮想化し統一的なアクセスを提供するとともに，設定されたFパスの管理を行う．Fパスはタスク生成時に一意に割付けられ，そのFパス識別子は AP による DEIMS-3 へのデータベースアクセス開始命令の発行契機に，SMP から DEIMS-3 へ渡される．以降，このFパスを用いて DEIMS-3 はノード間の DCNA コマンドの授受を行う．なお，このようなFパスは AP に対しては不可視であり，AP は何ら意識することなしにデータベースの処理が可能

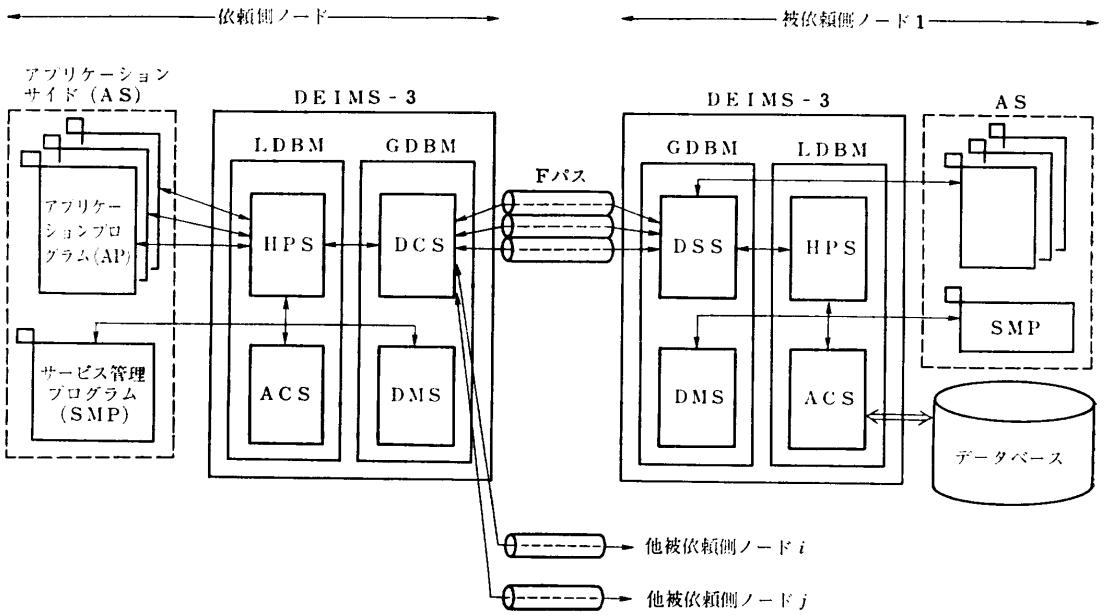


図 5 DEIMS-3 のシステム構成
Fig. 5 DEIMS-3 system configuration.

である．さらに，AP に対し，この不可視化および集中形処理との連続性を保証するために，Fパス障害回復後の救済単位開始時に再送が必要な DBAP コマンド（データベース開始用 OPEN コマンド，資源占有用 LOCK コマンド）を，DBM は常にスタックしている．

5. 評　　価

定形的な業務処理を行う実時間システムでは，スループットとともに，端末における応答時間が性能評価の要因となる．分散形データベースを構成したときは，分散処理の通信時間を含めた応答時間の評価が必要になる．そこで，通信処理の効率化をはかるために導入したユーザインタフェースの高水準化について，通信効率と実行効率を考察し，分散処理時間について評価する．

評価モデルは，設計目標で適用対象として述べたシステムの代表的な三つの処理パターン（検索，追加，更新）とし，一つの依頼側ノードのデータベース操作は，すべて他の一つの被依頼側ノードに依頼されるものとする．その通信回数と，DBM の CPU 処理時間について，HDML による方式と DML による方式との比較結果を**図 6** に示す．

通信回数は，1回の通信で必要なレコードを多く操作できる処理パターンになるほど DML に比べ回数

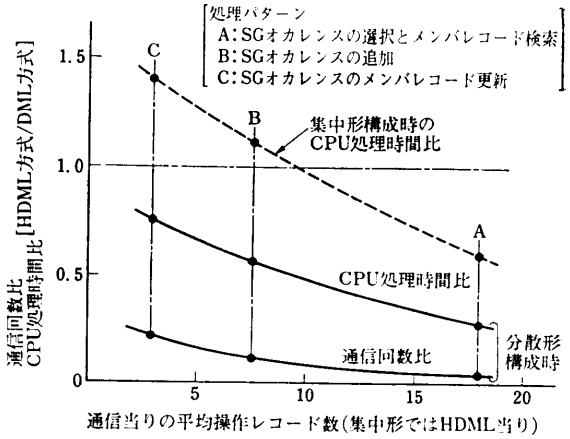


図 6 HDML 方式の評価
Fig. 6 HDML method evaluation.

は少なくなり，約 1/5〜1/20 に削減できている．また，DBM の CPU 処理時間は，通信回数に依存することから，DML と比べ通信回数が少なくなるほど通信処理に要する CPU 処理時間の影響が小さく実行効率がよくなり，約 3/10〜7/10 に削減できている．また実行効率からみると，集中形データベース処理においても，HDML 当り約 10 レコード以上の操作を行うのであれば HDML を適用しうるといえる．それ以下では，高水準化に伴う HDML の解析処理時間が DML に比べ大きくなる．

上記の3処理パターンをミックスしたトランザクシ

ョンモデルの端末との通信処理時間を除く分散処理時間の評価例を**図 7** に示す．図7では，低トラヒックのノード間は回線で，高トラヒックのノード間はチャネルで結合する処理モデルを示したものであり，いずれも DEIMS-3 のユーザインタフェースの高水準化による通信効率の大幅な向上等により，目標とする分散処理時間を達成できている．

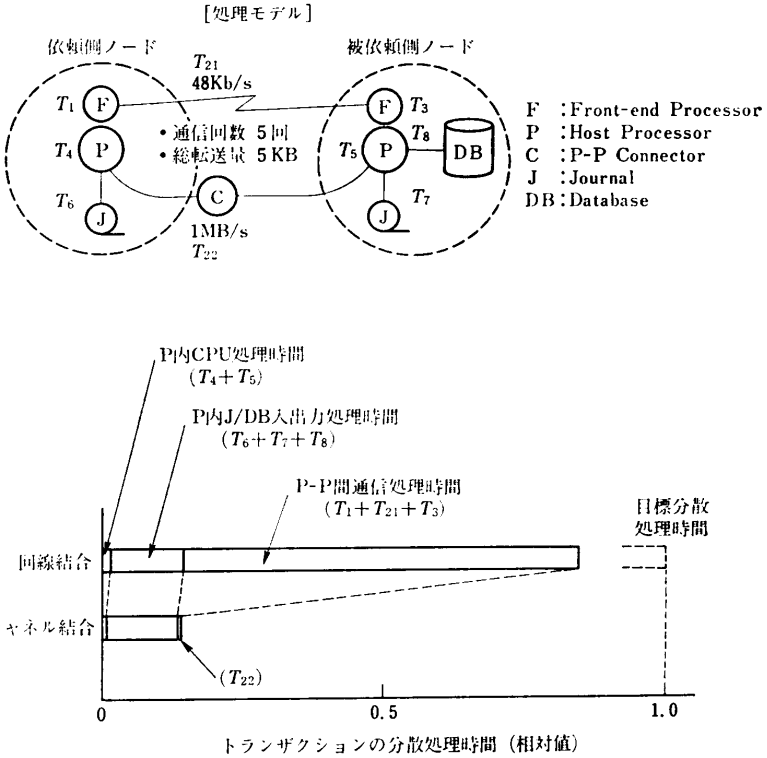


図 7 分散形データベースの処理時間の評価例
Fig. 7 Example of distributed database processing time evaluation.

6. むすび

DEIMS-3 は，異種データベースとの接続への拡張性のあるスキーマ構成，高水準なユーザインタフェースによる通信効率のよい問合せ処理，時間監視による実行効率のよいノード間の排他制御，2フェーズによる高信頼なノード間の更新同期制御を実現することにより，当初の目的である大容量高トラヒックな大規模実時間システムに適用しうる分散形 DBMS を実現することができた．

現在，DEIMS-3 は，DIPS 上で走行し，大規模データベースシステムに実際に適用され稼動している．今後の課題として，異種との接続をはかるためには，本スキーマ構成をベースに，異種データモデル間の変換，分散形スキーマの変更に対する同期管理などの技術を実現する必要がある．

謝辞 本開発の機会を与えていただいたデータ処理研究部高村真司部長，日頃ご指導いただいているデータ応用研究室森道直室長に深謝します．さらに，本開発に参加された岡田静夫，服部文夫，長濱芳寛，戸田博の各氏に深謝します．

参考文献

1) 村上，森：大規模データベースとその実現技術，情報処理，Vol. 23, No. 10, pp. 955-961 (1982).
2) Rothine, J. et al.: Introduction to a System for Distributed Databases (SDD-1), *ACM Trans. Database Syst.*, Vol. 5, No. 1, pp. 1-17 (1980).
3) Williams, R. et al.: R*: An Overview of the Architecture, Research Report, RJ 3325, IBM Research Laboratory San Jose (1981).
4) 増永：米国における最近の分散型関係データベースシステム技術，情報処理，Vol. 25, No. 5, pp. 443-450 (1984).
5) CODASYL Data Base Language Task Group: CODASYL COBOL Data Base Facility Proposal Jan. (1972).
6) CODASYL Data Description Language Committee: Data Description Language Journal of Development (1978).
7) 鈴木：DEIMS-2 について，情報処理学会データベース管理システム研究会，11-3 (1979).
8) 高平他：大容量高トラヒック向データベース管理システム，通研実報，Vol. 28, No. 12, pp. 2811-2826 (1979).
9) 鈴木他：高成長データ向けデータベース管理機能，通研実報，Vol. 29, No. 10, pp. 1645-1658 (1980).
10) Adiba, M. et al.: Issues in Distributed Data Base Management Systems: A Technical Overview, Proc. 4th VLDB, pp. 89-110 (1978).
11) Takizawa, M. and Hamanaka, E.: The Four-Schema Concept as the Gross Architecture of Distributed Databases and Heterogeneity Problems, *J. Inf. Proc.*, Vol. 2, No. 3, pp. 134-142 (1979).
12) Suzuki, K. et al.: Implementation of a Dis-

tributed Database Management System for Very Large Real-time Applications, Proc. IEEE COMPCON Fall, pp. 569-577 (1982).
13) 河津他：DCNA のデータベースアクセスプロトコル，通研実報，Vol. 30, No. 3, pp. 799-823 (1981).
14) 戸田，村田，田中：CODASYL DML の高水準化とその評価，情報処理第 24 回全国大会，pp. 525-526 (1982).
15) 村田，服部，鈴木：分散形データベースのアクセス処理方式について，情報処理第 24 回全国大会，pp. 459-460 (1982).
16) Kawazu, S. et al.: Two-phase Deadlock Detection Algorithm in Distributed Databases, Proc. 5th VLDB, pp. 360-367 (1979).
17) 岸本，田中，服部：分散形データベースシステムにおける救済制御について，情報処理第 24 回全国大会，pp. 465-466 (1982).
18) 鈴木他：分散形データベース管理システム (DEIMS-3) の構成，情報処理学会データベースシステム研究会，37-1 (1983).

(昭和 59 年 10 月 8 日受付)
(昭和 60 年 4 月 25 日採録)