金融勘定系元帳のリレーショナル データベースエミュレーション手法

富士通エフ・アイ・ピー 株式会社

■ 執筆者Ｐｒｏｆｉｌｅ ■

石　田　淳　一

1982 年　富士通エフ・アイ・ピー（株）入社
2004 年　現ビジネス SI 事業部
ファイナンシャルシステム部
プロジェクト担当課長
（入社以来，金融系システムの設計／開発に従事）

江　口　　淳

1991 年　富士通エフ・アイ・ピー（株）入社
2004 年　現ビジネス SI 事業部
ファイナンシャルシステム部
（入社以来，金融系システムの設計／開発に従事）

■ 論文要旨 ■

メインフレーム市場の縮小は，金融系システムにも大きな波紋を投じている．周辺装置として導入を開始したオープン系マシンの適用範囲が今では，基幹業務システムにまで進出しそうな勢いである．ハード機器の価格低下に加え，性能・信頼性の向上には目を瞠る物があり，大規模を誇る金融系基幹業務システムといえども，ハード面からは十分に適用可能な状況が整いつつある．一方，複雑な金融業務仕様が蓄積されているソフト資産の取扱いについては，既存品質の絶対保証やソフト開発費用抑制の観点から，オープン系特有の言語仕様を用いた再構築には，慎重論を唱える顧客も多い．そこで，稼動中の業務ソフト資産を可能な限り流用する金融系基幹業務システム移行に関して，最大の課題となる元帳ファイル（ネットワークデータベース）アクセス方式の継承を検討し，ソフト資産流用を守りながら，オープン系マシンへのダウンサイジング実現の検討を行った．

■ 論文目次 ■

■ 図表一覧 ■

１．はじめに

富士通エフ・アイ・ピー㈱では，IT アウトソーシングサービス・ｗｅｂサービス・システムインテグレーションサービスの三つを柱にサービスの提供を行っている．

私たちが所属するファイナンシャルシステム部では，長年にわたり，メインフレームでの勘定系大規模バンキングシステム開発のノウハウを蓄積してきた．

しかし，メインフレームからオープンシステムへのプラットフォーム移行が進む中，大規模バンキングシステムも例外ではなく，当社としても，メインフレームでのノウハウを活かし，オープン系の開発を行っている．

現在，情報処理産業のニーズとして，メインフレームからオープンシステムへのプラットフォームの移行が進む中，大規模バンキングシステムでも，システム導入コストの面から，オープン化の必要性及び実現が求められている．

また，プラットフォームに，オープン化の必要性が求められる中，ソフト面でも，業務アプリケーションの開発コスト削減／安定運用中の業務アプリケーションの新規構築に伴う危険回避などから，現行業務アプリケーション資産の流用が求められている．

現行のメインフレームをオープンシステムに移行し，業務アプリケーション資産の流用を行う場合には，業務アプリケーションは，プラットフォームの変更を意識していないため，現行システムでアクセス方法に則って，アクセス依頼を行う．

しかし，現行システム資源の中では，移行先資源にマッピングできない資源が存在するため，あたかも，移行先に現行システム資源があるように，業務アプリケーションに隠蔽する必要がある．

本論文では，現行システムでネットワークデータベースを利用して，構築されている勘定系元帳データベースに対して，業務アプリケーションが，ネットワークデータベースに対して遵守しているアクセス規定を，リレーショナルデータベースへのアクセスでも実現できる，エミュレーション手法について論じる．

２．目的

メインフレーム上で稼動している金融勘定系システム内の業務アプリケーションと元帳ファイルを管理するデータベースマネジメントシステム（以下，DBMS という．）との関係を**図１**に示す．

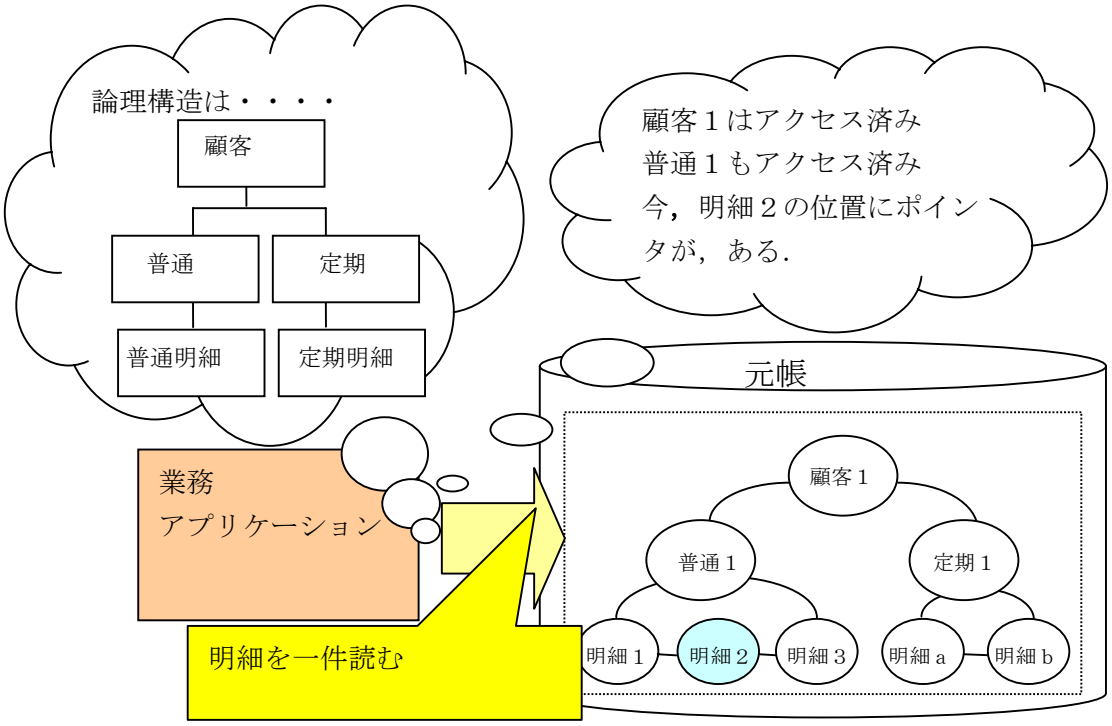


図１　業務アプリケーションデータベースアクセスイメージ

- 業務アプリケーションとDBMSは論理構造を媒介として必要な情報アクセスを実現する
- 業務アプリケーションは，DBMS 規則に従ったアクセス順序を遵守する
- DBMS は，業務アプリケーションからのアクセス要求実現を果すとともに，データベースアクセスに必要となる管理情報を動的に変更し，安全な情報アクセスを保証する

業務アプリケーションは，明示的なアクセス命令記述とは別に，暗黙的にアクセス対象データベースの論理構造を刷り込んだ状態を持ち，必要なレコードへ辿り着くためのアクセス順序は，論理構造を正しく理解する事で解決している．

またデータベース状態（レコード最新位置，排他情報など）管理機能は，データベースの安全性確保と，業務アプリケーションの任意アクセス開放を目的として DBMS が，その機能を担っている．

しかし，業務アプリケーション資産流用を前提として，プラットフォームをオープン系に移行する場合，オープン系ファイル製品では，トランザクション保証が唯一可能なリレーショナルデータベースを元帳ファイルとして選択するしかなく，製品機能の相違から，ネットワークデータベース機能をそのまま移植することは，不可能である．

そこで，業務アプリケーションがあたかも，ネットワークデータベースにアクセスしているように，階層型を踏襲したリレーショナルデータベースの実装及びアクセスを実現するためのエミュレーションが必要となった．

３．ネットワークデータベースの特徴

リレーショナルデータベースを元帳ファイルとして採用し，且つネットワークデータベース的な構造・管理概念を実現するために，まずネットワークデータベースの特徴を解明する必要がある．

３．１　階層構造

勘定系元帳ファイルで採用されている代表的な論理構造を**図２**に示す．

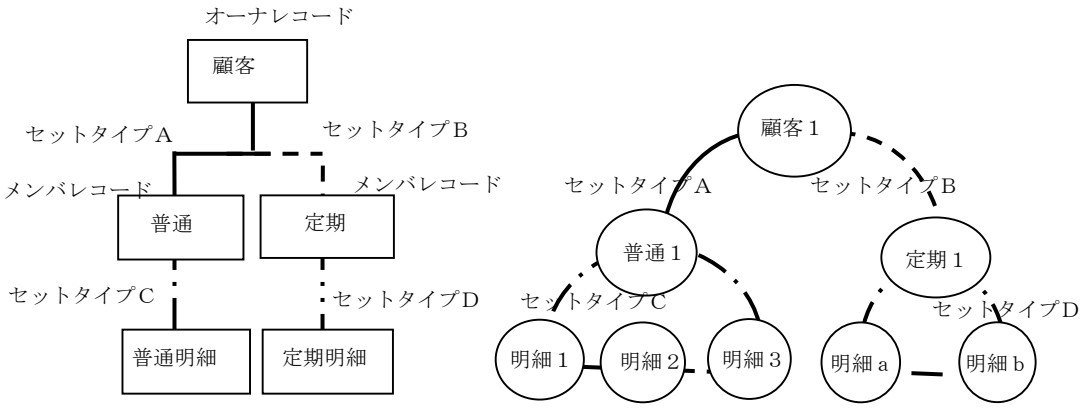


図２　階層構造図

・元帳ファイルは，複数階層構造（一般的には４階層程度）を保有する
・上位レコードタイプは，下位に複数のレコードタイプを保有（1：N）する場合がある
・一つの下位レコードタイプは，一つの上位レコードタイプを保有（1：1）する
・キー指定による直接アクセスは，最上位レコードタイプにのみ許される

３．２　レコード間の関係付け制御

レコードタイプ内に保有するレコード数は増減が自由なため，レコード間の順序関係はリング構造で維持する場合が多い．

また，レコードアクセス順番の保証機能はネットワークデータベースの場合，当該レコードの挿入処理実施時点で解決する．レコード間の関係付け／順番管理機能概念を**図３**に示す．

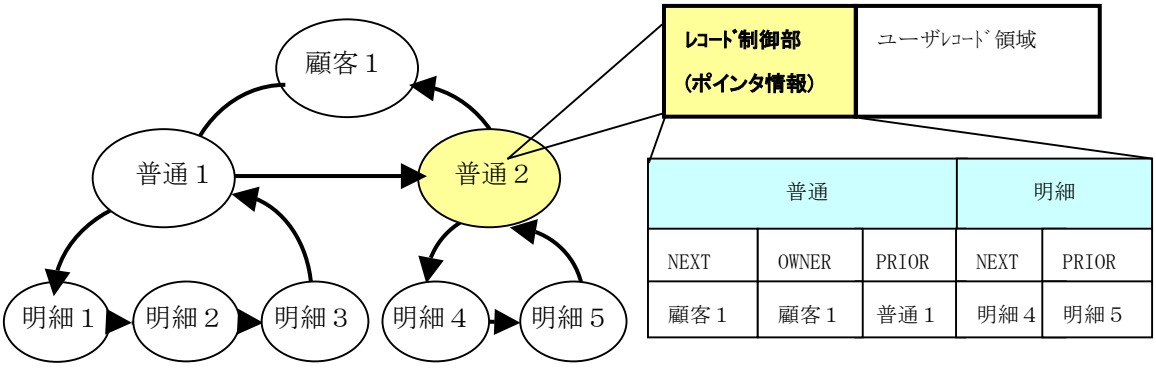


図３　ポインタ情報

４．リレーショナルデータベースへのエミュレーション手法

表間の関係（リレーション）解決を主たる機能としている，リレーショナルデータベースを採用し，ネットワークデータベース的な階層構造方式を実現するために，以下の事が重要な課題となる．

（１）各レコードタイプ間で階層関係を解決し，且つレコード順番保証を実現する

（２）キーを保有しない下位レコードのアクセスを実現する

４．１　データベース設計による実現手法

リレーショナルデータベース設計においての実現手法を以下に示す．

４．１．１　階層概念

レコード（行）に，ネットワークデータベースと同様な階層概念を持たせるため，**図４**に示すように，３層構造の階層の場合は，定義上は３列項目にてキーを構成し，下位レコードは上位レコードキーを継承する事で階層管理をデータベース設計上で実現する．

また，それぞれのレコードタイプを特定するために，キー項目にレコード種別を付与する．

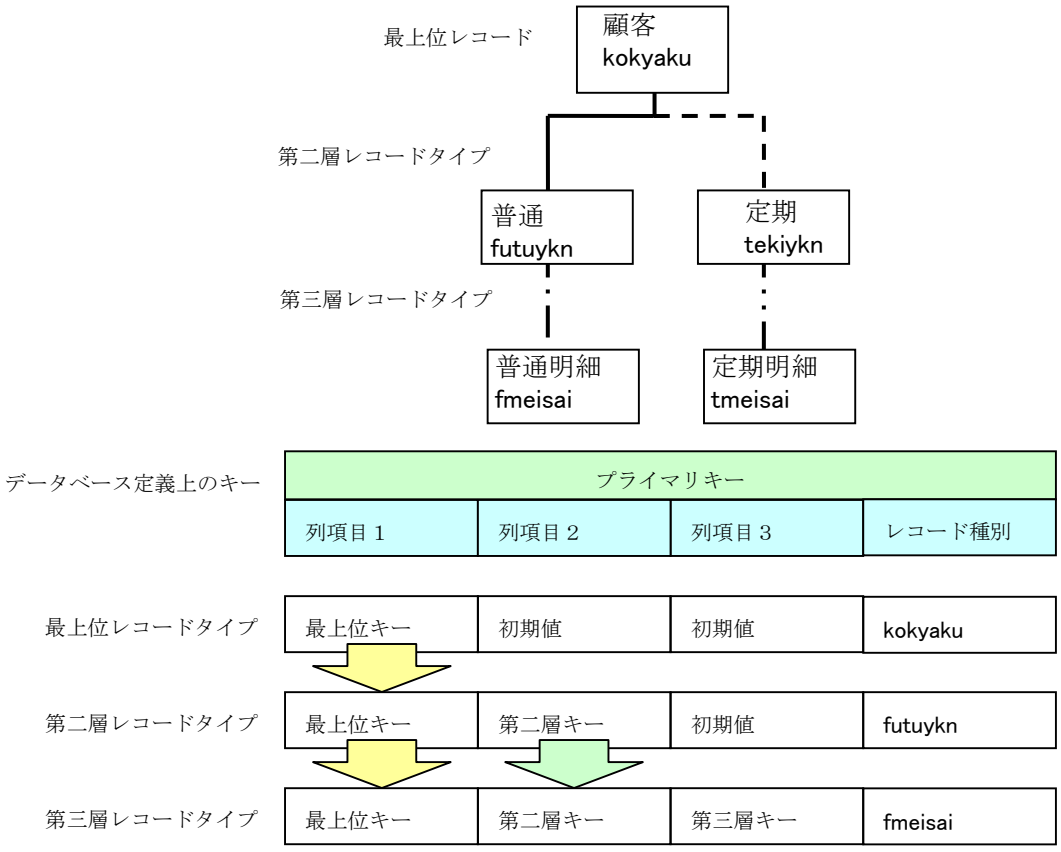


図４　階層概念

４．１．２　順序性保証

ネットワークデータベースでは，挿入時にレコード順序を確定するのに対して，リレーショナルデータベースでは，検索時点でレコード順序を確定する．

ネットワークデータベースの機能で，挿入位置をレコード発生順とした場合（FIRST 指定,LAST 指定），すなわち SORTED KEY 指定を行わない場合は，**図５**に示すように，レコード順序保証キーの内容として，格納時間を選択した．

レコード挿入時に格納時間を設定し，検索時の並べ替え条件（ＯＲＤＥＲ条件）で「昇順」を指定すれば「NEXT 指定」，「降順」を指定すれば，「PRIOR 指定」と同様のレコード検索を実現する．

図５　順序性保証

４．２　データベースアクセスサブルーチンでの実現

リレーショナルデータベースの構造設計を工夫することで，ネットワークデータベースの階層構造を継承すること及び順序性保証について継承実現を果した．

しかし，実際に業務アプリケーションからデータベースの階層構造／順序保証に則ったアクセスを行うためには，データベースの構造設計だけでは不足機能があるため，データベースアクセスサブルーチン（以後，データベースアクセス SR と表現する）にて機能補完を行う必要がある．

４．２．１　オーナレコードアクセス

図６に示すように，オーナレコードへのアクセスは，前述した，キー部の階層概念とレコード種別の付与によって，直接検索を行う．

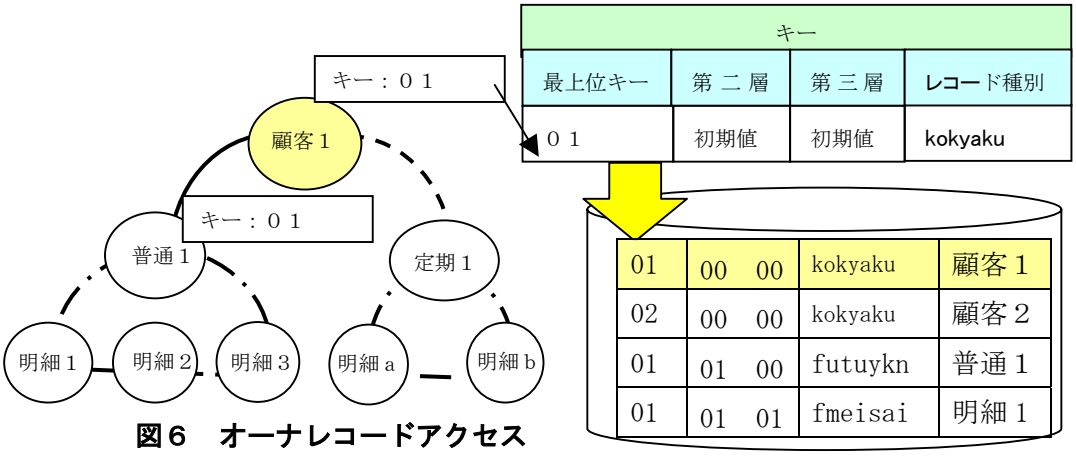


図６　オーナレコードアクセス

４．２．２　メンバレコードアクセス

ネットワークデータベースでは，2 階層目以降のメンバレコードへアクセスを行う場合，オーナレコードは読込み済の状態でなければならず，ＤＢＭＳは，オーナレコードに設定されている下位レコードポインタ情報に従い，メンバレコードのアクセスを行う．

メンバレコードに対するアクセスは，ポインタ情報を元に行うため，業務アプリケーションからキー指定での直接検索はできない．

リレーショナルデータベースは構造上，各レコードにキーを保有するが，業務アプリケーションからメンバレコードへのキー指定検索がないため，アクセスインターフェイス領域にアクセスレコードを特定するキー情報は設定されない．

そこで，データベースアクセスサブルーチン（以後，データベースアクセス SR と表現する）で，メンバレコードキーの組立機能が必要となる．

また，使用するアクセス命令には，**図７**に示すように，指定条件を満たす複数レコードをアクセスする，カーソル系 SQL 文でアクセス制御を行う．

指定条件を満たす複数レコードとは，キー及び同一レコード種別で抽出したレコードの集合である．

抽出したレコードを１件づつ業務アプリケーションに受け渡すことで，メンバレコードアクセスを実現する．

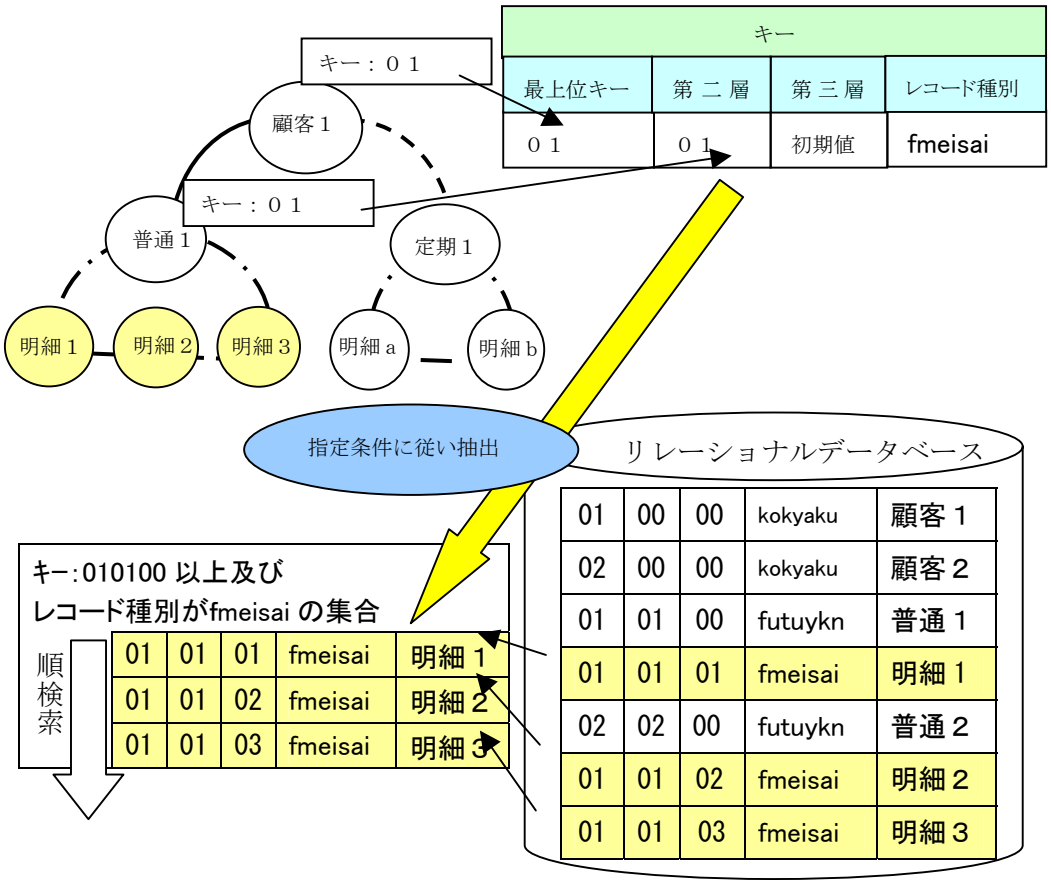


図７　メンバレコードアクセス

４．２．３　ポインタ管理

ＳＱＬ命令によって，レコード順序を保証するアクセスは実現できたが，DBMS が行っているポインタ管理及び下位レコードのキー組立を実現するために必要な上位レコードキーの保有が実現できていない．

DBMS が行っているポインタ管理には，以下のルールに則っている．

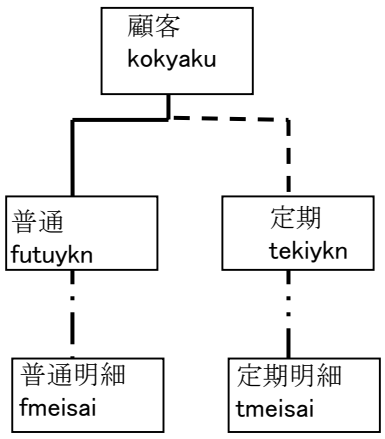
・ポインタはレコードタイプ単位に一意である
・上位ポインタが移動すると配下レコードのポインタは無効である
・ポインタ情報の有効性は，トランザクション内のみである

（１）ポインタはレコードタイプ単位に一意である

DBMS のポインタ管理と同等機能を実現するために，データベースアクセス SR 内部にカーソル状態管理表を作成した．

カーソル状態管理表とは，データベースアクセス SR が，業務アプリケーションから依頼された内容（レコードタイプ名，命令種別など）から，カーソル系 SQL を発行する場合に，**図８**のような制御領域にカーソルのオープン状態やプライマリキー情報をレコードタイプ単位に保有することで，ポインタ管理と同等な機能実現を行う．

論理構造体



カーソル状態管理表

識別子	ﾚｺｰﾄﾞﾀｲﾌﾟ名	ｶｰｿﾙ名	ｵｰﾌﾟﾝ状態	検索キー情報
0101	顧客	ｶｰｿﾙ１	済	01
020101	普通預金	ｶｰｿﾙ2	済	01
03010101	普通明細	ｶｰｿﾙ3	済	03
020102	定期預金	ｶｰｿﾙ4	未	
03010201	定期明細	ｶｰｿﾙ5	未	

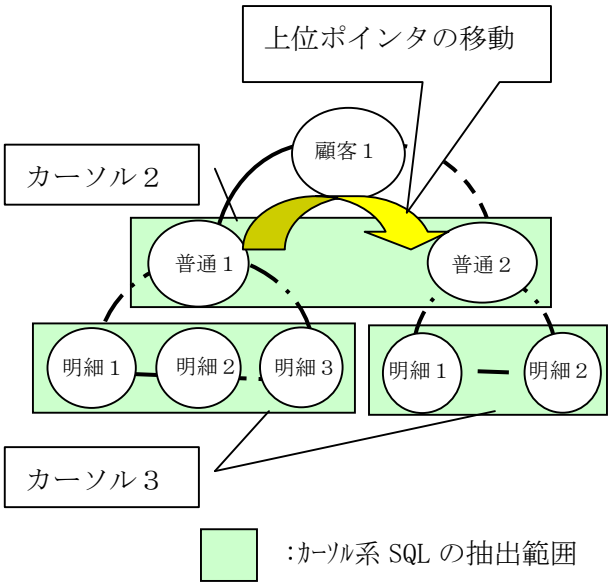
図８　カーソル状態管理表（例）

（2）上位ポインタが移動すると配下レコードのポインタは無効である

図９に示すように，上位ポインタが移動した際（同一レコードタイプで，別キー）は，カーソル状態管理表中の当該レコードタイプキー情報を，最新情報に変更する．

また，オープン状態がオープン済となっている下位階層カーソルに対しては，カーソルクローズ命令を発行し，カーソル状態管理表のオープン状態を初期状態（クローズ中）に戻す．

カーソル状態管理表で，レコードタイプごとのキー情報とカーソルのオープン状態を管理することで，ネットワークデータベースが行っている上位ポインタが移動すると配下レコードのポインタを無効とする機能を実現する．



カーソル状態管理表

識別子	ﾚｺｰﾄﾞﾀｲﾌﾟ名	ｶｰｿﾙ名	ｵｰﾌﾟﾝ状態	検索キー情報
0101	顧客	ｶｰｿﾙ１	済	01
020101	普通預金	ｶｰｿﾙ2	済	01→02
03010101	普通明細	ｶｰｿﾙ3	済→未	03→
020102	定期預金	ｶｰｿﾙ4	未	
03010201	定期明細	ｶｰｿﾙ5	未	

図９　上位ポインタ移動時のカーソル状態管理表

（3）ポインタ情報の有効性は，トランザクション内のみである

トランザクション終了時点（コミットまたはロールバック命令）で，オープン中のカーソルは無条件にクローズされる（SymfoWARE）ため，ポインタの有効範囲をトランザクション内のみとすることを可能とする．

ただし，カーソル制御表内の情報（オープン状態／キー情報）は，設定されたままなので，トランザクション終了後から次のトランザクションの実行されるまでに，カーソル制御表の初期化が必要となる．

５．適用効果

今回の手法を適用することで，データベース機能相違の隠蔽を実現し，金融勘定系取引で必要なデータベースアクセスについては，大凡エミュレートできた．

ファイルアクセスに関する限り，業務アプリケーションの資産流用率を高い次元で確保し，システム導入コスト（業務アプリケーション開発費）の削減に大きな効果を得た．

６．今後の課題

業務アプリケーションの資産流用を最前提としつつ，オープン系マシンへのプラットフォーム移行を実現するために必要となるネットワークデータベースで実装される金融勘定系元帳データベースをリレーショナルデータベースにエミュレートする手法について論じてきたが，今後の課題も残った．課題について，以下に示す．

６．１　アクセス命令の SQL へのマッピング

表１の例に示すように，アクセス命令で SQL への機能のマッピングが難しいものがある．

表１　アクセス命令で SQL への機能のマッピング困難な例

ネットワークデータベース表記	SQL 表記方法	理　　由
FIND 命令によるデータベースポイントの位置付け	SELECT 文または CURSOR 制御による FETCH 文を代用	位置づけ命令は存在しない
CET　CURRENT 命令による直接レコード検索	既存業務アプリケーションアクセスシーケンス修正	ROWID の単純適用の妥当性が不明確

６．２　サブルーチンの肥大化

大規模システムでは，開発効率化等の観点からデータベースアクセス機能を共通（サブルーチン）化することが一般的である．

しかし**図１０**に示すように，SQL 命令中にはテーブル名，キー項目等の可変部分（網掛け部分）が存在する．

静的 SQL 文で記述を行うと，プログラム中に明示する部分が非常に多くなるため，アクセスレコード種類増加に比例して，サブルーチン規模が肥大化する．

サブルーチンの肥大化は，システムメモリ空間の圧迫/メンテ性の低下につながる．

```
EXEC　SQL
        SELECT　　データ部　　INTO　:R業務データ領域　FROM　普通預金
            WHERE　　店番号　　=　:R店番号
                AND　　口座番号　　=　:R口座番号
                AND　　レコード種別　　=　:Rレコード種別
END-EXEC
```

図１０　SQL 記述例

６．３　動的 SQL 化

サブルーチンの肥大化を防ぐために，**図１１**に示すように，動的 SQL 命令を活用する方法もあるが，SQL 文ごとに構文チェック／解析処理等の DBMS オーバヘッドが入るため，性能劣化は否めない．

```
MOVE   SQL命令   TO   "SELECT データ部,・・・・・・・・・・・      "
EXEC   SQL
          PREPARE  :SELECT_OWNER     FROM    :SQL命令
END-EXEC.
EXEC   SQL
          EXECUTE    SELECT_OWNER   INTO  :R業務データ領域
                USING  :R店番号,R口座番号,:Rレコード種別
END-EXEC.
```

図１１　動的 SQL 記述例

金融勘定系取引での単体レスポンスにおける，静的 SQL，動的 SQL でのそれぞれの処理時間の例を**表２**に示す．

表２　処理時間比較（例）

	平均レスポンス時間（ms）	差分（ms）
静的 SQL	37.76	31.84
動的 SQL	69.60	

７．おわりに

業務アプリケーション流用を前提条件として，オープン系マシンへ移行する際に最大課題となる元帳ファイルアクセスの継承については，
（１）リレーショナルデータベーステーブル構造設計での階層・順序性保証領域の設定
（２）アクセス機能時点でのキー生成機能付与など
によって解決する事ができ，目的は達成した．

今回の手法は，メインフレームのネットワークデータベースがオープンシステムに存在しないため，ネットワークデータベース機能を継承する形で，リレーショナルデータベースへの移植を行った．

ここでのエミュレート手法は一つの実現例であり，今後もっと，リレーショナルデータベース本来の機能を活かした手法の模索も必要である．