Rocket

ホワイトペーパー

入れ子リレーショナル・データベース

目次

入れ子リレーショナル・データベース

はじめに

概要

このホワイトペーパーは，入れ子リレーショナル・データベースの技術的優位性について論じる．特に，リレーショナル・データベースの準拠する第一正規形（属性の原子性）の制約からの解放にフォーカスする．入れ子リレーショナル・データベースは，技術的には非正規形（NF2）データベースと呼ばれるが，一般的にはマルチバリュー・データベース，あるいは拡張リレーショナル・データベースとして知られる．

最初に，Oracle，MySQL，SQL Serverなどの伝統的なリレーショナル・データベースとRocketマルチバリュー・データベースの違い（主に，表を入れ子にすることで複雑なデータ構造をそのまま格納できる能力）について説明する．次に，この技術がリレーショナル・データベースのユーザーにどの様な恩恵をもたらすかについて概論する．そして，入れ子リレーショナル・データベースに関して，よくある疑問に答えることで締めくくる．また，巻末に，入れ子リレーショナル・データベースの参考文献を紹介している．

リレーショナル・データベースの基礎技術

Rocket UniVerse™，Rocket UniData®，Rocket D3®といったRocket Softwareのマルチバリュー・データベース（入れ子リレーショナル・データベース）製品を含め，広く使われているリレーショナル・データベース・システムは，1969年にE. F. Codd氏により最初に提唱され，その後に進化を遂げたリレーショナル・モデルに準拠している．リレーショナル・モデルは他のデータ・モデルと異なり，厳密な数学的定義を伴い，それは本稿の論ずる範囲を超えるが，その基礎技術は以下のように要約できる．

- データベース・システムは，ユーザーにとってのデータの論理的な見え方と，実際に格納されるデータの物理的な構造を明確に区別する．ユーザーは，データのアクセスや管理の際に，そのデータの物理的構造について知る必要がない．
- データのシンプルな論理構造は，データベースの専門家でなくとも容易に理解することができる．データは表の形式で表現され，表の各列は同じ属性を持つ値の集合である．各列の値を行で切り出したものを組（タプル）といい，タプルの集合が関係（リレーション）を構成する．各リレーションは1つ以上のキー属性を持ち，キーの値によりタプルが一意に識別される．

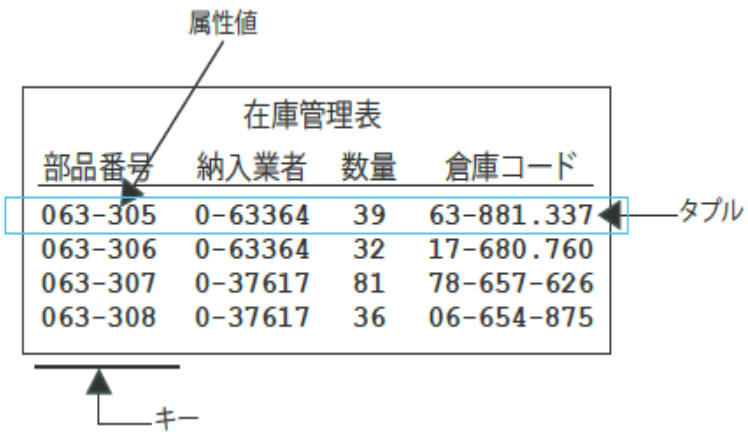

図 1. リレーショナル・モデルは表というシンプルで論理的な構造を持ち，属性値が列と行の中に保持される．行で切り出した値の集合はタプルと呼ばれ，0個以上のタプルがリレーションを構成する．これがモデルの名前の由来である．

- 表の行や列へのアクセス（選択，射影）や，同じ属性の列を持つ表同士の結合を行うために，高級言語が提供される．SQLは，米国国家規格協会（ANSI）が標準化した言語であるが，多くのベンダーがこれを拡張している．

Codd氏はデータ・モデルだけでなく，リレーショナル・データベースにおける

- 表の操作に必要な関係演算，データのアクセス方法，
- セキュリティに関するルール

を提唱している．入れ子リレーショナル・データベースを含めて，全ての商用リレーショナル・データベースは，関係演算を基礎技術としている．それに準拠していないデータベースでは，リレーショナル・データベースとは異なる実行結果が得られるであろう．全ての商用リレーショナル・データベースは，関係演算のルールに概ね準拠している（100%準拠である必要はなく，また100%準拠のリレーショナル・データベースも存在しない）．同じデータに対して同じSQL文を実行すれば，どの商用リレーショナル・データベースでも一貫した結果が得られるのである．

リレーショナル・データベースの正規化

正規化は，リレーショナル・データベースの理論を特徴付けるものである．正規化は，データベースの中でデータをでき得る限りコンパクトで容易に扱い，一貫性のある結果を保証することを目的とする．正規化は，リレーショナル・データベースのスキーマ（設計）を定義するためのガイドラインである．それは，単純に言えば，如何にデータベースを表へ分割し，如何に表を結合するかである．正規化の主な目的は，以下の2つである．

1. データの重複を最小限にする
2. データベースに変更を加えるときに，影響を受ける属性を最小限にすることで，データ管理を簡素化し，エラー発生を予防する

ネストした表

顧客番号	顧客名	注文番号	部品番号	数量
AA2340987	Zedco, Inc.	93-1123	037617	81
			053135	36
		93-1154	063364	32
			087905	39
GV1203948	Alphabravo	93-2321	006776	72
			055622	81
			067587	29
MT1238979	Trisoar	93-2342	005449	33
			036893	52
			06525	29
			090643	33

図2.この表では「部品番号」と「数量」が階層構造になっており，繰り返しグループと呼ばれる．この点で，この表は第一正規形（1NF）ではない.

5つある正規形のうち，3つはCodd氏により最初に定義され[3,4,5]，2つは後に他の研究者により追加された．データベースが***第一正規形（1NF）***であるためには，属性は原子性（アトミック）を備えなければならず，1つの属性が複数の項目を含んでいたり，ましてや繰り返しグループを入れ子表として含むことはできない．つまり，第一正規形のデータベースでは，表を入れ子構造にすることができない．図2は，第一正規形ではない入れ子の表を表している．図3では，表の入れ子を避けるために，データを顧客表と注文表に分割し，顧客と注文の間にリンクを形成している

第一正規形の遵守はデータベースの設計上の話であり，実際に遵守するかどうかは実装の段階で判断される．通常は，設計フェーズではデータベースは完全に正規化されるが，実装フェーズではパフォーマンス上の要件を満たすために，逆に第一正規形を崩すことがある．パフォーマンス改善のために表の結合を避けようとすると，データベースが属性の非原子性を許容しない限り，ユーザーは同じ属性の列を複数の表に持つ実装をしなければならない．これでは，シンプルで分かり易いSQL文で正しい結果を得ることが非常に難しくなる．入れ子リレーショナル・データベースでは表の入れ子を許しているので，例えば「注文」，「部品」，「数量」などの階層関係にある列を入れ子に（マルチバリュー化）することで，表の結合を行わない簡素なSQL文で正しい結果を得ることができる．

第二正規形から第五正規形（以降，*上位正規形*）は，特定の条件に応じてデータベースがとるべき正規形を定義している．例えば，第二正規形は，(1)表が複数のキー属性を持ち，(2)その一部のキー属性のみに対して完全関数従属する属性がある場合，(3)その属性は別の表へ分割されるべきであるとしている．図4は，図3の表を第二正規形（2NF）にするために変更したものである．図4で分かるように，表を第二正規形にした場合，データベース内のデータのサイズ，および設計変更時に影響を受ける属性の数を削減することができる．図3では，「*顧客番号*」は「注文表」の全ての「部品」にそれぞれ付けられている．図4では，「*顧客番号*」は「注文番号」にだけ対応している．もし，顧客に割り当てられている番号を変更することになっても，変更箇所はかなり少なくなる．

顧客表

顧客番号	顧客名
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

注文表

注文番号	部品番号	数量	顧客番号
93-1123	037617	81	AA2340987
93-1123	053135	36	AA2340987
93-1154	063364	32	AA2340987
93-1154	087905	39	AA2340987
93-2321	006776	72	GV1203948
93-2321	055622	81	GV1203948
93-2321	067587	29	GV1203948
93-2342	005449	33	MT1238979
93-2342	036893	52	MT1238979
93-2342	06525	29	MT1238979
93-4596	090643	33	MT1238979

図 3. 第一正規形にするために，図2の表の繰り返しグループを別表へ分離する.

ただし，部品の種類が極めて少ない場合には，この第二正規形では「注文表」と「注文顧客表」の各行がほぼ1対1に対応するため，別表へ分割する意味は薄れてしまう．また，パフォーマンスも低下する．さらに，同じ「注文番号」が2つの表に格納されているため，データ量の削減にも寄与しない．そのようなケースでは，図3の方が図4よりも最適と言える．

第三，第四，第五正規形（3NF，4NF，5NF）も同様に，レベルが上がるにつれて更に厳しい要件を設けており，ストレージ・サイズの削減と設計変更の影響の縮小に貢献する．（上位正規形の詳細な定義については，参考文献の[3,4,5]を参照）

顧客表

顧客番号	顧客名
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

注文顧客表

注文番号	顧客番号
93-1123	AA2340987
93-1154	AA2340987
93-2321	GV1203948
93-2342	MT1238979
93-4596	MT1238979

注文表

注文番号	部品番号	数量
93-1123	037617	81
93-1123	053135	36
93-1154	063364	32
93-1154	087905	39
93-2321	006776	72
93-2321	055622	81
93-2321	067587	29
93-2342	005449	33
93-2342	036893	52
93-2342	06525	29
93-4596	090643	33

図 4.「注文顧客表」を作成して，「顧客番号」を「注文表」から切り離すことで，第二正規形の要件を満たした．

第一正規形への準拠は，データベース・ソフトウェアの仕様で決まるのに対して，上位正規形への適合の可否はデータベースの設計に依存する．どんなリレーショナル・データベースを使用しようが，上位正規形に適合させるかどうかは設計者が決める．データベース・ソフトウェアが，設計が上位正規形を遵守しているかどうかを確実に判断することは不可能であり，データベース・システムにより強制されることはない．いずれにせよ，正規形に適合していないからといって，設計が最適でないとは言えない．

表間の関連

図4は，正規化のテクニックである表間の関連（関連情報の表）の例である．「注文顧客表」は，データそのものと言うより，表と表の間の関連情報（このケースでは，顧客表と注文表の関連）を格納している．

関連情報を格納する表を作ることには，2つの欠点がある．(1)元のデータ構造には含まれていない項目を持つ表を追加すると，設計が複雑になる．(2)データベース内の表の数が増える．よって，余計な処理が必要となり，保守やストレージ要件に影響を与え，システム・レスポンス・タイムが悪化する．更に，SQL文がより複雑になるために，クエリー等の操作が煩雑になる．

第一正規形リレーショナル・データベースの限界

どのようなリレーショナル・データベースでも，上位正規形に適合させるかどうかは，データベースの設計者の判断に完全に任されている．設計の最適化に関し，データベース・ソフトウェアは，正規化の制約を科すことはしない．しかし，格納される値が原子性を持つことを求められるデータベースでは，設計者には第一正規形に準拠すること以外の選択肢はない．上位正規形への準拠は，より複雑になるという欠点はあるものの，設計は概ね最適なものになる．しかし，第一正規形のデータベースでは，しばしばストレージ使用量が増大し，保守は難しくなる．更に重要なことは，設計が複雑になるだけでなく，処理の負荷も増えることである．*図2*と*図4*を比較すると：

1. 表の数は，関連の表も含めて1から3へ増えている
2. 正規化により「注文番号」と「顧客番号」の値を注文毎に2回格納することになった
3. 図1の形のレポートの作成には3つの表の結合が必要である．表の結合は，CPUの負荷が大きい処理である

リレーショナル・データベースでは，第一正規形で分割された表間の関連を結合処理によって解決するが，そのパフォーマンスへの影響は，リレーショナル・データベースの使用を断念させるに十分なほど甚大であることがある．例えば，機械部品の設計を行うCAD/CAMシステムでは，第一正規形のリレーショナル・データベースは受け入れられないことが一般に知られている（参考文献[11]）．

理由の1つは，CAD/CAMのデータは，その性質上階層構造となることである．所定のレスポンス・タイム内で部品情報を端末に出力するには，データベース内の階層構造を素早くトラバースしなければならない．複雑なツリー構造の関連を持つ表から目的のデータを取り出すには，数百，数千の表結合が必要である．そのような結合処理を，許容できる時間内に処理することは，単純に不可能である．これは，第一正規形データベースがCAD/CAMデータの処理に採用されない理由の1つである．そのようなシステムでは，高いパフォーマンスを実現するために階層型データベースを独自に開発・保守しなければならず，高いコストを伴う．

パフォーマンスの観点から離れても，多くのアプリケーションにとって，第一正規形リレーショナル・データベースには実用面での制限がある．階層型やネットワーク型データベースのスキーマを第一正規形へ正しくトラバースすることは可能ではあるが，実際にやるとなると容易な事ではない．機械部品を例に取ると，階層型のデータ構造は，自然な形でコンパクトに部品データを格納することができる．しかし，階層型のスキーマの第一正規形への変換（マッピング）は，直感的な分かり易さからは程遠く，複雑で混乱し易く，扱い難い，相互に関連した表のかたまりになり，それらの表間の関連を格納する表も大量に必要となる．現実的には，そのようなスキーマを実装することは不可能である．同様の問題は，他の多くのアプリケーションのデータでも見られる．

第一正規形の制約からの解放

第一正規形の制約による様々な限界を考えるとき，「根底にあるリレーショナル・モデルを壊すことなく，リレーショナル・データベースから第一正規形の制約を取り去ることはできないのか？」という疑問が浮かぶ.

前述の通り，リレーショナル・モデルでは厳密な数学的定義により，それをきちんと実装したデータベースであれば，予想通りの正しい結果が得られることが保証されている．リレーショナル・モデルから第一正規形の制約を取り去る検証が既に行われてきており[1,7,9,10,12,13]，その結果，モデルの堅牢性を何ら棄損しないことが証明されている．言い換えると，リレーショナル・データベースが上位正規形に適合する限り，第一正規形を除去することによるデータの不正や不整合は発生しない.

第一正規形の制約が取り払われると，すなわち入れ子の表が許されると，データベースは技術的には*非正規形*（NF2）と称されるが，一般的には入れ子リレーショナル，*拡張リレーショナル*，あるいはマルチバリューDBMSと呼ばれる．データベース理論の研究者は，入れ子リレーショナル・データベースが紛れもないリレーショナル・データベースであると認識している．入れ子リレーショナル・データベースについては多くの文献が書かれているので，その数学的根拠について学びたい方は巻末の「参考文献」を参照されたい.

入れ子リレーショナル・データベース

Rocket Softwareは，業界でもトップクラスの入れ子リレーショナル・データベース（マルチバリュー・データベース）製品を開発している．入れ子リレーショナル・データベース管理システムは，第一正規形を前提としたリレーショナル・データベース管理システムでは不可能な機能を拡張したものである．入れ子リレーショナル・データベースのためのデータ・クエリー言語として，SQLもサポートされている．入れ子リレーショナル・データベースの先進機能をサポートするため，SQL標準を完全にサポートした上で，それを拡張した機能を実装し，第一正規形データベースでは不可能だったデータの操作を実現している.

重要なことは，第一正規形を前提としたアプリケーションやツールとの互換性のため，入れ子リレーショナル・データベースが第一正規形のビューを作成できる点である．更に，そのようなアプリケーションが入れ子リレーショナル・データベースを採用していれば得られるであろう高パフォーマンスを得るために，データベースを最適化することも可能である．Rocket UniVerse，Rocket UniData入れ子リレーショナル・データベース（マルチバリュー・データベース）は，そのような最適化機能を実装している．つまり，いかなる第一正規形のデータベース・スキーマも，入れ子リレーショナル・データベース上で実装することができる．通常，その様な実装は，第一正規形を前提とした従来のSQL文をそのまま使う必要があるときだけに行われる．データベースのプログラマーや管理者は，入れ子リレーショナル・データベースのシンプルな設計の恩恵を先取りすることを考えるべきである.

入れ子の表

冒頭で述べた通り，リレーショナル・モデルの基本的な信条は，データを物理的構造から明確に切り離し，シンプルで論理的な構造をユーザーへ提供することである．入れ子リレーショナル・データベースは，この考え方を完全に踏襲した上で，表間の関連を排除することによりリレーショナル・モデルが目指すゴールを超越したシンプルな論理的構造を提供する．

顧客表

顧客番号	顧客名
AA2340987	Zedco, Inc.
GV1203948	Alphabravo
MT1238979	Trisoar

注文顧客表

注文番号	顧客番号
93-1123	AA2340987
93-1154	AA2340987
93-2321	GV1203948
93-2342	MT1238979
93-4596	MT1238979

注文表

注文番号	部品番号	数量
93-1123	037617	81
93-1123	053135	36
93-1154	063364	32
93-1154	087905	39
93-2321	006776	72
93-2321	055622	81
93-2321	067587	29
93-2342	005449	33
93-2342	036893	52
93-2342	06525	29
93-4596	090643	33

図5.これらの第一正規形の表を定義，アクセスするSQL文を以下に示す.

SQLの簡易な拡張により，入れ子の表は定義することができる．以下のSQL文（例は，Rocket UniDataのSQL）は，図5に示された第一正規形の表を定義した例である．

```
CREATE TABLE CUSTOMER_TABLE (CUST# CHAR(9)
DISP (“顧客番号”), CUST_NAME CHAR (40)
DISP (“顧客名”));

CREATE TABLE ORDER_TABLE (ORDER_# NUMBER (6)
DISP (“注文番号”), PART_# NUMBER (6) DISP (“部品
番号”), QTY NUMBER (3) DISP(“数量”));

CREATE TABLE ORDER_CUST (CUST_# CHAR (9)
DISP (“顧客番号”), ORDER_# NUMBER (6)
DISP (“注文番号”));
```

これに対して，図6の入れ子の表を定義するには，以下のSQL文を使用する．

```
CREATE TABLE NESTED_TABLE (CUST# CHAR (9) DISP (“顧客番号),
CUST_NAME CHAR (40) DISP (“顧客名”),
ORDER_# NUMBER (6) DISP (“注文番号”) SM (“MV”) ASSOC (“ORDERS”),
PART_# NUMBER (6) DISP (部品番号”) SM (“MS”) ASSOC (“ORDERS”),
QTY NUMBER (3) DISP (“数量”) SM (“MS”) ASSOC (“ORDERS”));
```

Rocket入れ子リレーショナル・データベース（マルチバリュー・データベース）では，繰り返しグループの属性を入れ子の表として作成することができる．上記のSM句は，その属性が子表の繰り返し要素（*MultiValued* - MV），孫表の繰り返し要素（*Multi-Subvalued* - MS）であることを示している．ASSOC句は，その属性が入れ子の表のカラムであることを示している．Rocketマルチバリュー・データベースでは，親表の中に複数の子表を作成することが可能である．

ネストした表

顧客番号	顧客名	注文番号	部品番号	数量
AA2340987	Zedco, Inc.	93-1123	037617	81
			053135	36
		93-1154	063364	32
			087905	39
GV1203948	Alphabravo	93-2321	006776	72
			055622	81
			067587	29
MT1238979	Trisoar	93-2342	005449	33
			036893	52
			06525	29
			090643	33

図 6.この入れ子の表は，図5の表と等価である．この表を定義するSQL文が上記のものである．この表へアクセスするSQL文を次に示す.

以下は，図5の第一正規形の表からデータを読み取る，標準的なSQL文である．

```
SELECT CUSTOMER_TABLE.CUST#, CUST_NAME, ORDER_TABLE.ORDER_#, PART_#, QTY
FROM CUSTOMER_TABLE, ORDER_TABLE, ORDER_CUST
WHERE CUSTOMER_TABLE.CUST_# = ORDER_CUST.CUST_# AND ORDER_CUST.ORDER_# = ORDER
_TABLE.ORDER_#;
```

これに対して，図6の入れ子の表からデータを読み取るSQL文は，以下のように非常にシンプルであり，上記のSQL文と同じ出力を得る．

```
SELECT * FROM NESTED_TABLE;
```

もちろん，入れ子の表から読み取ったデータを，第一正規形のフォーマットで出力することも可能である．そのためには，SELECT文でUNNEST句を使用する．以下のSQL文を実行すると，図5の注文顧客表が得られる．

```
INSERT INTO ORDER_CUSTOMER (CUST_#, ORDER_#)
SELECT CUST#, ORDER_# FROM NESTED_TABLE UNNEST ORDER_#);
```

以下のSQL文は，注文顧客表を第一正規形のデータベースへエクスポートするためのASCIIテキストを出力する．

```
SELECT CUST#, ORDER_# FROM NESTED_TABLE UNNEST ORDER_# TO ASCII_TABLE;
```

暗黙の表間の関連

前述の通り，（図5の注文顧客表のような）表間の関連情報を格納する表は，入れ子リレーショナル・データベースでは必要ない．それが必要となるのは，第一正規形でデータベースを設計しなければならないときだけである．図6では，顧客と注文の間の関連は存在してはいるが，入れ子リレーショナル・データベースでは，その情報は*暗黙的*である．

表を入れ子にすることで，表間の関連は暗黙的に示される．それは，入れ子リレーショナル・データベースでは自動的かつ効率的に保守される．表間の関連を表として明示的に定義することは，第一正規形の必然的な結果である複雑性，余計な保守，潜在的なエラーを伴うことになる．表の入れ子構造による表間の関連の暗黙的定義は，データベースの設計をよりシンプルにし，データ間の関連を直接反映させることができる．

入れ子リレーショナル・データベースを設計しているときでも，時には明示的な表間の関連の表が必要となることがある．そのような場合は，第一正規形データベースとして，必要な表を定義すればよい．

入れ子リレーショナル・データベースの技術的優位性

以降のセクションは，特定のタイプのアプリケーションにとっての，入れ子リレーショナル・データベースの技術の優位性について論ずる．とは言え，入れ子リレーショナル・データベースは，繰り返しグループのデータを格納するあらゆるデータベースに対して，以下の圧倒的優位性を持っている．

- データベースの設計がよりシンプルであり，定義し保守する表がより少ない
- 表間の関連がデータベースのスキーマにダイレクトに反映されるため，設計がより簡単になり，第三者がより理解し易くなる
- 処理時間が長く，CPU負荷の高い表の結合を回避できる
- 表間の関連を保持する冗長なデータを回避し，ストレージ容量を削減できる
- データへのアクセスがよりシンプルで，より分かり易い．SQL文はよりシンプルで，より直接的である

更に，入れ子リレーショナル・データベースのユーザーが，これらのメリットを享受するのと引き換えに失うものは何もない．

- 入れ子リレーショナル・データベースは，真にリレーショナル・データベースである．数学的に等価なモデルに立脚しており，第一正規形データベースの全ての利点を活かせる．
- 入れ子リレーショナル・データベースで，第一正規形に準拠したデータベースを実装することもできる
- ANSI標準に準拠したSQLを備えており，既存の全てのSQLをサポートできる

入れ子のデータベースと第一正規形のデータベースの間の変換は，容易に行うことができる．

データベースの変換

情報システムをメインフレーム上で構築している企業の多くは，それらをオープン系のクライアント／サーバー環境やWebパラダイムへシフトさせる必要に迫られている．それは多くの困難を伴うが，その最たるものは，大規模でミッション・クリティカルなデータベースをIBM IMSのような階層型データベースやネットワーク型データベースから移行させることである．

IMSとIDMSは，最新の第一正規形リレーショナル・データベースに比べても，2つの明確な優位点を持つ．

1. 企業のビジネス・データの関係性を容易にデータベースへマッピング
2. データベース処理が高速

入れ子リレーショナル・データベースは，これらの優位性をリレーショナル・データベースでも提供することができるので，データベースの移行を容易ならしめることができる．次のセクションでは，移行について具体的に述べる．

階層型データの入れ子リレーショナル・データベースへのマッピング

ビジネス情報は，本質的に階層構造を持つ傾向にある．組織の構造は典型的な階層構造であり，製品は部品や部品組立品の階層で構成されている．また，市場は階層的な地理的セグメントに分けられる．このような構造を，IMSのような階層型データベースへマッピングすることが容易であることは明白である．設計者は，単純に階層化されたデータをデータベースの階層構造に当てはめるだけである．そして，第一正規形のリレーショナル・データベースのフラットな表の構造に，階層を当てはめることが困難であることも，また明白である．なぜなら，これまでのセクションで見てきたように，表間の関連の表が大量に必要となるからである．

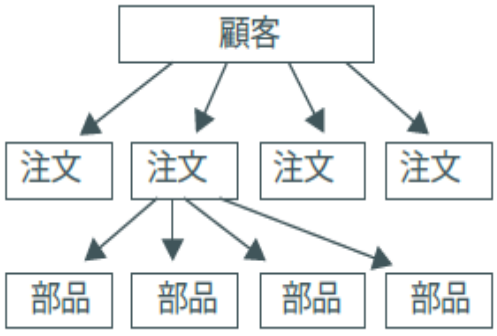

図 7.旧来の受注データベースは，本質的に階層構造を持ち，入れ子リレーショナル・データベースへ自然にマッピングすることができる．

入れ子リレーショナル・データベースが提供できる入れ子の表を利用すると，階層型のデータをリレーショナル・データベースへ直接マッピングすることができる．図7に示した顧客注文データベースは，階層構造である．これは入れ子リレーショナル・データベースへはダイレクトに簡単にマップすることができるが，第一正規形データベースへのマッピングは非常に困難であり手間がかかる．

レガシー・システムから移行した後のパフォーマンス

レガシー・システムで使われているデータベースは，独自開発であり，保守コストがかかり，変更が困難であり，プログラミングが難しいが，その一方高速である．レガシー・システムのデータベースを第一正規形のリレーショナル・データベースへ移行すると，元のメインフレームの応答スピードを提供できなくなることが多いが，その原因は表の結合処理にある．結合処理は，IMSのような階層型データベースでは必要なかったものである．データベースのデータ間の関係は，階層構造として設計されてハード・コードされているので，その応答は非常に高速である．いかなるリレーショナル・データベースでも，たとえ入れ子リレーショナル・データベースであっても，結合は時間のかかるCPU負荷の高い処理である．

然しながら，入れ子リレーショナル・データベースでは，既に述べたように，表間の関連を入れ子構造により暗黙的に定義するため，表の結合を劇的に減らすことができる．ある意味では，暗黙的な表間の関連はハード・コードされたIMSの階層構造に似ており，それにより表の結合の必要性を除去し，非常に高速なデータ・アクセスを可能とする．一方で，IMSの階層構造とは異なり，暗黙的に定義された表間の関連は容易に変更することができる．*すなわち，入れ子リレーショナル・データベースは，階層型データベースのパフォーマンスの優位性と，リレーショナル・データベースの柔軟性と使い勝手の良さを併せ持っていると言える．*

入れ子リレーショナル・データベースは，メインフレームのプロプラエタリなデータベース・システムと同等のパフォーマンスを保証することはできないが，第一正規形のデータベースよりも遥かに高いパフォーマンスを保証することができるのである．

附録A：
Q&A

入れ子リレーショナル・データベースと業界

ベンダーの大勢は，入れ子リレーショナル・データベースへ傾いているというのは本当ですか？

第一正規形のリレーショナル・データベースの限界，特にデータ構造の複雑性のため，商用リレーショナル・データベースは入れ子リレーショナル・データベースの技術を採用し始めていますが，Rocket Softwareは競合他社に先んじて長年の経験を有しています．

他のデータベース・ベンダーは，既存のデータベース・エンジンに，表を入れ子にする機能を追加することはできないのですか？

ほとんどのデータベース・ベンダーは，属性が常に原子性を持つことを前提として，製品を設計し，実装しています．Rocket Softwareのマルチバリュー・データベースは，最初から表の入れ子構造をサポートするように設計されています．この技術を既存のリレーショナル・データベースへ追加するには，アーキテクチャの完全な見直しか，機能のエミュレーションを行う他はありません．更に，Rocket Softwareのマルチバリュー・データベースは，他社の製品とは異なり，入れ子の表が1つまでという制限がありません．

入れ子リレーショナル・データベースは，正しいリレーショナル・モデルに準拠しているとして，学術的に認められていますか？

はい．入れ子リレーショナル・データベースを検証した多くの文献が出されています．（巻末の参考文献リストを参照下さい）

入れ子リレーショナル・データベースの技術

入れ子データベースは，全てリレーショナル・データベースですか？

「拡張」や「入れ子」という言葉は，通常リレーショナル・データベースだけで使われていますが，非原子性の属性を格納できるデータベースは，どのようなものでも入れ子データベースであると言えます．例えば，CODASYLデータベースは入れ子データベースです．

入れ子リレーショナル・データベースとネットワーク型データベースの違いは何ですか？

CODASYLデータベースのようなネットワーク型データベースは，リレーショナル・データベースではありませんが，入れ子データベースであると言えます．リレーショナル・データベースとそうではないものの違いは，前者が厳密な数学的理論を元に，標準化されたアクセス言語（SQL）を提供していることです．他のデータベースは，概してそうではありません。

入れ子リレーショナル・データベースは，第一正規形のデータベースよりも使用が難しいですか？

その逆であると言えます．入れ子リレーショナル・データベースは，設計，実装，使用が容易です．ネストした関係を格納できるということは，アプリケーション・データの関係を，より直接的にデータベースのスキーマにマップできるということです．そして，表間の関連が自動的かつ透過的に保たれるため，それを格納するためだけの不要な表の数を減らすことができ，データベース全体が意味も無く複雑になることを防ぎます．また，入れ子の表をサポートするために機能拡張されたSQLにより，よりシンプルで短いSQL文を書くことができます．

入れ子リレーショナル・データベースの互換性

入れ子リレーショナル・データベースで既存のSQL文を使うことはできますか？

第一正規形データベースのスキーマと入れ子リレーショナル・データベースのスキーマが一致するのであれば，SQL文は変更なしに実行でき，同じ結果が得られます．もし，入れ子リレーショナル・データベースの利点を活かして，入れ子の表を作成するのであれば，入れ子用に拡張したSQLの箇所のみを修正する必要があります．その結果多くの場合，SQL文はよりシンプルでコンパクトになります．

入れ子リレーショナル・データベースは，第二，第三，第四正規形に適合することができますか？また，それは必要のあることですか？

入れ子リレーショナル・データベースは，上位正規形の全てに適合することができます．それら正規形を適用するときの要件は，第一正規形と変わりありません．

入れ子リレーショナル・データベースを，従来の入れ子ではないリレーショナル・データベースへ変換することはできますか？

入れ子リレーショナル・データベースは第一正規形に準拠する必要がありませんが，それを妨げるものでもありません．もし，互換性のために必要なのであれば，他ベンダーの第一正規形データベースと全く同じスキーマを，入れ子リレーショナル・データベースで作成することができます．

入れ子リレーショナル・データベースの入れ子の表を第一正規形へ変換するときは，拡張SQLのUNNEST句を使用することにより，入れ子データベースを容易に第一正規形へ変換したり，第一正規形のビューを作成したりすることができます．

Bibliographical References

1. Abiteboul, S., and Bidoit, N. “Non First Normal Form Relations to Represent Hierarchically Organized Data,” Proceedings of 2nd ACMSIGACT/SIGMOD Symposium on Principles of Database Systems, March 1984, pp 191–200.
2. Arisawa, H., Moriya, K., and Miura, T. “Operations and the Properties On Non-First-Normal-Form Relational Databases.” Proceedings of 9th International Conference On Very Large Data Bases. October 1983, pp 197–204.
3. Codd, E. “A Relational Model of Data for Large Shared Data Banks,” Communications of the ACM. Vol. 13, No. 6, June 1970, pp 377–387.
4. Codd, E. “Further Normalization of the Data Base Relational Model,” Database Systems. Edited by Rustin, R. Prentice-Hall, pp 33–64.
5. Codd, E. “Relational Completeness of Data Base Sublanguages,” Data Base Systems. Edited by Rustin, R. Prentice-Hall, pp 65-98.
6. Dadam, P., and Linnemann, V. “Advanced Information Management (AIM): Advanced Database Technology for Integrated Applications,” IBM Systems Journal, Vol. 28 No. 4, 1989 pp 661–681.
7. Fischer, Patrick C. and Thomas, Stan J. “Operators for Non-First-Normal-Form Relations,” Proceeding of IEEE COMPSAC ‘83, Chicago, November 1983, pp 464–475.
8. Garnett, L. and Tansel, A. “Equivalence of the Relational Algebra and Calculus for Nested Relations,” Computers Math. Applic.Vol. 23, No. 10, 1992, pp 3–25.
9. Kitagawa, H., and Kunii, T.L. The Unnormalized Relational Data Model for Office Form Processor Design. Tokyo: Springer-Verlag, 1989.
10. Makinouchi, A. “A Consideration On Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model,” Proceedings of 3rd International Conference on Very Large Data Bases, October 1977, pp 447–453.
11. Roth, M., Korth, H., and Silberschatz, A. “Extended Algebra and Calculus for Nested Relational Databases,” ACM Transactions on Database Systems, Vol. 13 No. 4, December 1988, pp 389–417.
12. Schek, H., and Pistor, P. “Data Structures for an Integrated Data Base Management and Information Retrieval System,” Proceedings of 8th International Conference on Very Large Data Bases. September 1982, pp 197–207.
13. Schek, H., and Scholl, M. “The Relational Model with Relation-Valued Attributes,” Information Systems, Vol 11, No. 2. 1986, pp 137–147.

rocketsoftware.com
info@rocketsoftware.com
US: 1 855 577 4323
EMEA: 0800 520 0439
APAC: 612 9412 5400
twitter.com/rocket
www.linkedin.com/company/rocket-software
www.facebook.com/RocketSoftwareInc
blog.rocketsoftware.com