ソフトウェア工学 96−21
（1994．1．21）

ラウンドトリップ的な設計作業を可能とする開発支援環境の検討

佐藤 友康　外山 勝保

NTTソフトウェア研究所

データベース設計では，仕様変更・不完全などにより概念設計と論理設計を頻繁に行き来（ラウンドトリップ）して設計が進められる．しかし従来は，変更を行うと影響波及範囲や変更条件が不明などのため，変更に柔軟に対応できなかった．本稿では，具体的なラウンドトリップ作業を支援するという立場から，リポジトリスキーマを設計する．その特徴は，影響波及範囲を狭めるために仕様書（ドキュメント）をその構成要素（オブジェクト）に分解し，ドキュメント単位とオブジェクト単位で管理する．そしてさらにオブジェクト単位での変更履歴を管理し，オブジェクトが変更された背景をあわせて管理することで，さらにその効果を上げる．

Software Engineering Environment to support round-trip activities of database design

Tomoyasu SATO　Katsuyasu TOYAMA

Software Laboratories
Nippon Telegraph and Telephone Corporation
NTT Shinagawa TWINS BLDG., 1-9-1 Kohnan, Minato-ku, Tokyo 108-19, Japan

Database design proceeds back and forth between the Conceptual Design phase and the Logical Design phase because of requirements change or error detection. Our current software engineering environment, however, cannot support the frequent changes in specifications.

This paper describes the repository schema designed for supporting the frequent changes of specification in relational database design. The repository manages the design information in two levels: one is document and the other is object. In addition, it also stores design history of objects with rationale, which helps to control change propagations.

1．はじめに

ソフトウェア開発作業において，生産性を向上するためにCASEツールが導入されている．1つのCASEツールは一部の工程だけを支援しており，全工程を支援するためには異なる工程を支援するCASEツールを組み合わせ利用をすることになる．各CASEツールは他のツールとの関連を考慮しておらず，データを個々に管理しているため，現在我々が構築している開発支援環境[1]では，CASEツール間のデータのやりとりは，上流から下流へのデータの引き渡しを可能とするデータ交換形式を規定してデータを流通させている．しかし，仕様変更の多い実作業においては，仕様変更が生じると，設計したデータを再投入する必要が生じてしまっている．実作業では仕様変更は不可避であり，より柔軟に変更に対応する必要がある．

データの共有を行うためにはリポジトリを用いてデータを管理することが提案されている[2]が，変更に柔軟に対応するためには，どのようにリポジトリを用いるかが重要となる．

本稿では，このような観点に立ち，リレーショナルデータベース設計の実作業を支援するリポジトリの設計について述べる．

2．リレーショナルデータベース設計

2．1　設計作業対象，方法

リレーショナルデータベース(RDB)の設計には3段階あり，概念設計，論理設計及び物理設計がある．我々が構築しているRDBの開発支援環境では，RDBの設計は下記の作業によりそれぞれ成果物をまとめていく．

(a)概念設計

現実の業務で管理，あるいは処理される具体的な事物（実体）とそれらの関連をあらい出す．また，各実体のデータ項目を決定する．最終成果物は実体関連図（ER図）とデータ構造図（DS図）である．

(b)論理設計

概念設計を元に，実体からテーブルを設計し，データ項目からカラムの詳細設計を行う．そして，システム性能要件に基づき，テーブルやカラムの統合，分割を行うなど，テーブル構造を見直す．最終成果物は各カラムの名称，型，桁数などを定義したカラム一覧表である．

(c)物理設計

論理設計を元に，具体的なデータの格納方法，記憶媒体上でのファイルの実現方法，メモリでの展開方法を設計する．

物理設計は利用するRDBMSや計算機資源などにより設計方針が異なるため，利用環境を特定して検討しなければならないのに対し，概念設計と論理設計はこのような環境には影響されず，設計方法やそこで扱われるデータの収集が行い易い．そこで本稿では概念設計と論理設計を対象とする．

また，各設計段階での最終成果物として実際に作成される仕様書には他にも存在するが，本稿では上記の主な仕様書（実体関連図，データ構造図，カラム一覧表）を対象とする．

2．2　設計作業の実際

通常の作業順序は，概念設計が終了してから，これを元に論理設計が行われる．しかし，実際の作業では，概念設計と論理設計は行き来することが常である．例えば，概念設計がある程度まで進むと論理設計が進められ，その途中で概念設計へ戻り，そしてまた論理設計が進められるというように行われる．あるいは論理設計が進行している時点で要求変更や設計のあやまり，不完全性などにより，概念設計へ戻り，変更が行われ，そしてその変更に対応して論理設計を手直しする．本稿では，このような行き来する作業をラウンドトリップと呼ぶ．

2．3　問題点とその原因

仕様変更による設計作業のラウンドトリップにより，様々な問題が生じる．例えば，データ項目を決定するデータ定義表を元にして作成されたカラム仕様書や，その他の仕様書があるとする．データ定義表の一部が変更されると，カラム仕様書やその他の仕様書の何を修正すべきかがわからず，その箇所を

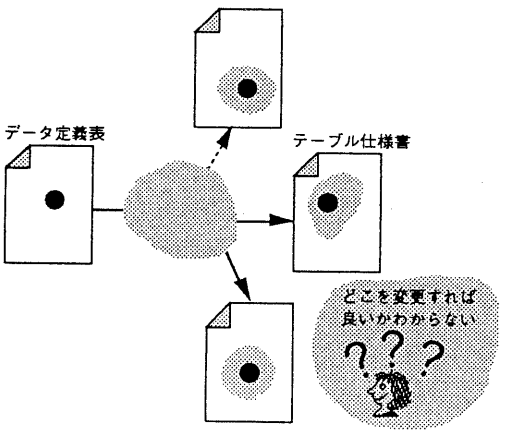

図1．仕様書変更の変更伝搬による負担

探すことについて検討しなければならないということもある（図1）．ラウンドトリップによる問題点には，以下のようなものがある．

(1)仕様変更に対し，その影響波及によって変更すべき設計データを見落とし，変更漏れが生じる．
(2)一部の設計データが修正した場合でも，投入の既投入のデータがすべて使えなくなることがある．
(3)設計者の設計内容や設計変更に対する理解が曖昧であるため，どの部分をどのように修正するかという検討事項に誤りが生じることがある．

以上の問題点は，以下の点が原因と考えられる．第1は，設計データ間の関係が不明確であるため，変更による影響波及範囲が明確にできないことである．そのために，(1)のように波及解析が不完全になり，変更漏れが生じたりするのである．また，(2)は，変更の必要がないデータも変更前の他のデータとの関係を遮断されたり，あるいは削除されたりして再利用できなくなるのである．

第2の理由は，仕様変更の背景が不明確であることである．仕様変更には様々な状況に応じて行われる．例えば，あるデータ項目名を変更した経緯が，変更された時点では理解されている．しかし，その変更を他の設計者が見ると，その変更を行った担当者にその背景を尋ねたりしなければならない．あるいは変更を行った設計者が後に見たときにどういった経緯で変更されたのかということがわからなくなっていたりすることもある．このように，(3)は仕様変更の背景が設計者の記憶や仕様変更決定時の議事録に頼らなければならず，これらと変更前後のデータを設計者により結合されなければならない．その結合が明確に行われないことが多く，理解が曖昧になる．

3．問題点の解決策

3．1　設計方針

前章で述べた，設計データ間の関係の明確化と仕様変更の背景の明確化のために，リポジトリのスキーマの設計，および設計データの管理方法について検討する．

設計データの本質は仕様書ではなく，それを構成する仕様書構成要素である．ここで仕様書をドキュメント，構成要素をオブジェクトと呼ぶ．ドキュメントはオブジェクトを独自の形式で表現するものである．従ってドキュメント単位で管理を行っていたのでは，あるドキュメントで一部のオブジェクトが変更されても，他のドキュメントで本質的に同一であるオブジェクトや，または関係するオブジェクトの管理が複雑化し，反映できない恐れがある．従って変更による影響波及範囲を明確にするためには，データをオブジェクト単位で管理し，それらのデータ間の関係を明確化することである．また，仕様変更の背景を明確化するために，データがいつ，誰が，何を，どこで，どのように，なぜ変更されたかを細かく管理する必要がある．このようにして変更に影響がないデータにはアクセスしない様にすることが必要である．そこで以下のような管理手法を採用する．

・粒度を考慮した設計データ管理
・オブジェクト単位での変更管理

3．1．1　粒度を考慮した設計データ管理

ドキュメント（粗い単位）で管理していたのでは，ドキュメントの一部を変更しても，それに関係するドキュメントが判明するだけで，そのどこを修正すればよいのかはわからず，直接関係しないデータまで修正対象として検討しなければならない．そこで，ドキュメントをオブジェクトに分解し，オブジェクト単位（細かい単位）で管理する．オブジェクト単位で管理すると，変更による影響波及は，その変更されたオブジェクトに関係するオブジェクトのみを変更すれば良いことになり，ドキュメント単位の管理に比べ，波及範囲を狭めることができる．そこでオブジェクトの関係を明確にすることで本当の影響波及範囲を求めることができる（図2）．

これを利用して，ドキュメント単位の管理とオブジェクト単位の管理を行う（図3）．オブジェクト単位でのオブジェクト同士の関係を明確にすることで，同一ドキュメント内外のオブジェクトの関係が

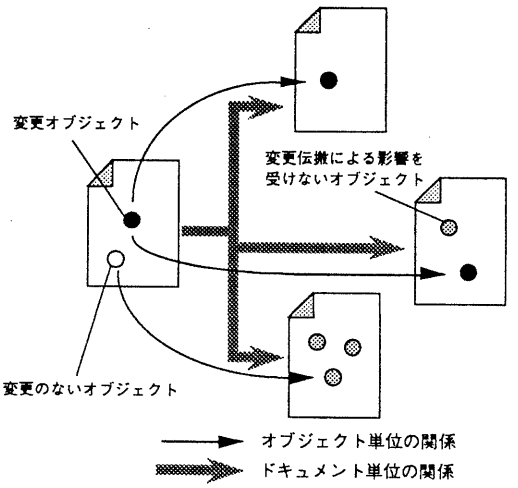

図2．設計データの粗さによる変更影響伝搬

明確になる．そしてオブジェクトとドキュメントの関係を明確にし，ドキュメント間の関係も明確になる．また，異なるドキュメントには，同一のオブジェクトが存在することがある．このとき，このオブジェクトをそれぞれのドキュメントで共有することで，1つのドキュメントでそのオブジェクトを変更すると，それを共有するドキュメントにも反映されることになる．このようにすることで，データ間の関係の簡略化が行える．

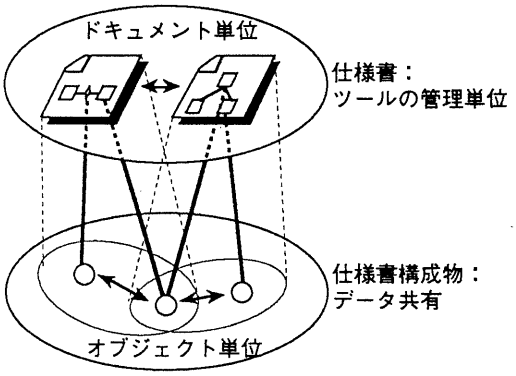

図3．リポジトリの2層構造

3．1．2　オブジェクト単位での変更管理

3．1．1で述べたオブジェクト単位の管理にいつ，誰が，どこで，何を，どのように，なぜ変更したか(5W1H)を明確にするために，オブジェクト単位の変更履歴を記録する変更管理を導入する．このことにより，ドキュメント単位での変更過程も把握することができる．変更による影響波及は，オブジェクト単位では自動的に伝播されるのではなく，基本的に設計者の判断に任される．オブジェクト単位の管理では，変更による影響波及を，変更のあったオブジェクトに関係のあるオブジェクトに限定できるが，オブジェクト単位での関係があっても変更を行う必要のないオブジェクトも存在する．このときは設計者がその時点で波及範囲を決めたことになるが，そのときどういった背景で中止したのかを管理する．これにより，後に設計者がこれを参照したときに，なぜこのようなことを行ったのかが明確になる．また設計作業には，以前に行った作業と同様の作業が多く発生するものである．そして，変更作業が多くのデータを変更しなければならない場合も少なくない．このように以前と同様の作業を行ったときは，その前例として前回の変更の範囲をその理由と共に設計者に提示することができ，変更範囲に関する設計者の判断を支援することができる．

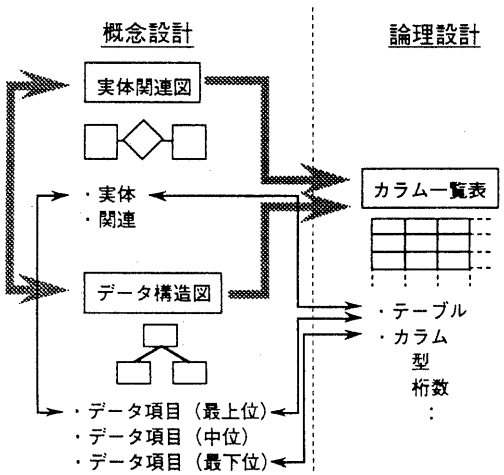

図4．データの対応

3．2　データモデル

(1)ドキュメントの構成要素

各ドキュメントにおけるオブジェクトの対応関係は図4の通りである．各ドキュメントは対応するオブジェクトを共有する．

(2)スキーマの設計

3．1で述べたことを考慮して，リポジトリのスキーマを設計する．図5にリポジトリスキーマをERモデルで示す．ここでは，ドキュメントやそのオブジェクトを長方形で表現し，これを「メタ実体」と呼ぶ．また，メタ実体の間に存在する関連を長方形を結ぶ矢印で表現し，これを「メタ関連」と呼ぶ．「メタ」を付加するのは，概念設計で用いる実体関連図と区別するためである．メタ関連は三角で型（依存，参照）と方向を示す．属性が“d”であるものは依存関係を示し，“r”は参照関係を示す．三角の方向は，頂点が向かっている方向が関係の方向である．また，矢印の数でカーディナリティを示し，矢印が2つあるものは，M（多）を，1つのものは1を表す．例えば，ERDとTableの関係は，1対Mで，依存関係を示し，ERDが存在しなければTableは存在し得ないことを表す．また，EntityとChangeEntityの関係は，M対Mで，両方向に参照するという関係にあるということである．各メタ実体の下にある項目は，そのメタ実体で管理する基本的な情報（メタ属性）である．

■粒度を考慮した設計データ管理

図5において，メタ実体であるERD（ER図），DSD（DS図），Table（カラム一覧表）は各ドキュメントを管理する．一方，Entity（実体），

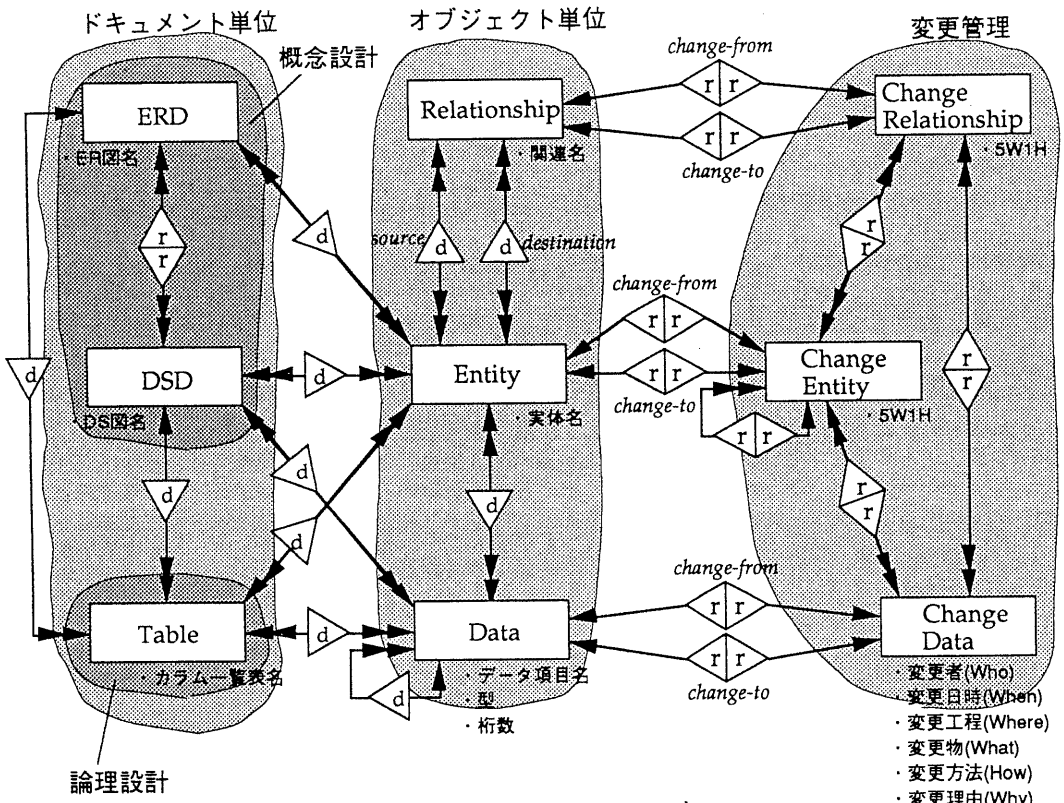

図５．リポジトリスキーマモデル

Relationship（関連），Data（データ項目）は各ドキュメント中に存在するオブジェクトを管理する．そしてこれらがドキュメント単位とオブジェクト単位に分かれている．ドキュメント単位では，概念設計にERDとDSDがあり，ドキュメントの対応関係をつけるため，それらは相互参照関係にある．また，論理設計にはTableがあり，ERDとDSDに存在依存している．これは，通常は概念設計が行われなければ論理設計が行われないので，これを明確化したものである．また，１つの概念設計に対して論理設計が行われるため，ERD，DSDとTableは1:Mの関係である．オブジェクト単位では，ER図やDS図において，実体が存在しなければそれに関係する関連やデータ項目は存在しない．そこでRelationshipとDataはEntityに依存関係を持つ．実体は多くの他の実体と関連を持ち，関連は２つ以上の実体につながる．そして実体は多くのデータ項目を持ち，データ項目は複数の実体に属するため，EntityとRelationship，Dataとのメタ関連はM:Mになる．また，EntityとRelationshipは，関連の方向を示すために，始点(source)と終点(destination)のメタ関連を持っている．このメタ関連が両方とも成立しなければ，Relationshipは存在し得ない．DataからData自身へのメタ関連は，データ構造図を表現するものであり，データ構造図の下位データが上位データに存在依存することを表す．

次にドキュメントとオブジェクトの関連について説明する．各オブジェクトはドキュメントに存在依存しており，各オブジェクトが対応ドキュメントの構成要素であることを明確化している．ER図の基本的な構成要素は実体であるから，ERDとEntityはメタ関連を持つ．関連もER図の構成要素であるが，RelationshipがEntityに存在依存しているため，Entityを通してERDとRelationshipの関連を確認できるため，特にこれらの間のメタ関連を設けない．図４に示すように，DS図の構成要素であるデータ項目の最上位のものは実体と同一であるため，DSDとEntityがメタ関係を持っている．DS図の最上位以外のデータ項目は，Dataで管理されているため，DSDとDataがメタ関連を持つ．Tableに関してもDSDとEntity，Dataのメタ関連と同様である．複数のドキュメントに共有されるオブジェクトは，１つのドキュメント

が存在すれば，そのオブジェクトは存在し得る．

■オブジェクト単位での変更管理

図5右はオブジェクト単位の管理に対し変更管理を行っている．オブジェクト単位のメタ実体に対し，変更履歴を記録するメタ実体(ChangeEntity, ChangeRelationship, ChangeData)が存在する．元データ(change-from)と更新データ(change-to)との関係を持つために，各オブジェクトとの間に相互参照関係を持つ．これらのメタ実体で，変更前のデータに対して，いつ，誰がどこで何をどのように，なぜ変更したのか(5W1H)を管理する．

次に変更管理のメタ実体間の関連について説明する．それぞれのメタ実体は，相互参照関係があり，あるオブジェクトの変更でその影響が波及して引き起こされた変更には関係が持たれる．例えば，ある実体が追加されたために，これに伴い関連も追加されたとする．このとき，実体の追加履歴と関連の追加履歴は関連を持つ．そこでその関連性を管理する．このようにして変更による影響波及を管理する．また，ChangeEntityにChangeEntity自身へのメタ関連が存在する．これは，ある実体が変更されたときに，それと関連を持つ実体がその関連を通して，関連には何の影響も及ぼさずに実体に影響が波及するが考えられるからである．

次に，ドキュメント単位の変更管理について述べる．ドキュメント単位では版管理を採用する．版管理は，あるデータの更新を管理できるが，その背景などを管理することはできない．しかしオブジェクト単位の変更管理により，変更前後のオブジェクトの関係が明らかであり，それを含む変更前のドキュメントの版更新による変更がオブジェクトの変更内容になるので，ドキュメント単位で管理する必要がないためである．

3．3　例

3．2で述べたスキーマの使い方について，例を用いて考察する．図6は，RDBが設計されたときの過程である．紙面の都合上，それぞれ主要な部分のみを示している．(a)はドキュメントで見える状態である．(b)は(a)に対応し，(a)の状態で設計データがリポジトリ上で格納／処理される状態である．点線の四角で囲まれているのが1つのメタ実体であり，その中の実線の四角で囲まれているのがメタ実体名を示し，それ以外はそのメタ実体に含まれるインスタンスを示す．(b)中段にあるDataのインスタンスの括弧で囲まれているのは，エイリアス名である．例えば，中段のDataのインスタンスである県コードのエイリアス名は県名であることを示す．インスタンス間を結ぶ矢印は，スキーマに基づいたインスタンスの対応関係を示す．例えば，(b)上段において，Dataの県コードはEntityの都道府県に属することを示す．

■粒度を考慮した設計データ管理の効果

(a)において，最上段は概念設計を終了して論理設計まで進んでいる状態である．そのリポジトリ上のデータの状態が(b)の最上段である．データ構造図で表現された工程管理と伝送路の構成要素である工程番号は同一のものであるため，リポジトリ上ではデータ項目である工程番号が実体である工程管理と伝送路に共有されている．ここで仕様変更が生じ，工程管理のデータ項目である県名は決まった値が入るものであるから，定数テーブルを作成し，これを参照するという仕様に変更された．但しカラム名は変更できないという要求があった．そこで概念設計に戻り，仕様に合わせて変更が行われた（(b)中段）．このとき，Dataにおいて，工程番号のリンクが県名から県コードに張り替えられる．工程番号は工程管理と伝送路に共有されているため，工程番号においてこのような変更を行えば，伝送路にも自動的に反映される．また，県コードを参照するため，そのデータ項目と関係がある都道府県とインスタンスを共有することになる．そこで，都道府県と工程管理の間に関連を持たなければならないため，このことを設計者に提示することができる．そこで，関連δが作成される．これらの履歴は背景と共に変更管理に記録され，ChangeDataとChangeRelationshipに記録される．そして，いまの変更が関係を持つ．こうして概念設計が終了し，論理設計へ進められる．ここでカラム名が変更できないという要求が存在するため，Dataのインスタンスである県コードにエイリアスを作成し，カラム名を県名のまま保持している．このこともChangeDataに記録されている．他のデータはそのまま残っており，再利用可能である．こうして(a)中段が完成した．このようにして，オブジェクト単位の管理の効果が現れている．

■オブジェクト単位の変更管理による効果

さらに仕様変更が生じ，工程種別も定数テーブルから参照することになった．そこでデータ項目として工程種別コードを参照しようとするが，該当するデータ項目が存在しないため，実体として工程種別定数を作成し，そしてデータ項目として工程種別コードを作成した．そして工程番号は工程種別から工程種別コードにリンクが張り替えられ，関連εも作成された．それに応じて論理設計にも影響が波及する．以前の県名が変更された例を見ると，県名はカラム名を変更しなかった．設計者はこの事例を見

ることにより，カラム名を変更しないことがわかる．しかし，このときどういった背景で変更しないのかということが不明であり，以前の変更を行った時の議事録などをたどらなければならない．ここでは，ChangeDataにその背景を記録しているため，それを設計者に提示することにより，設計者はこれらが理解でき，以前の変更と同様に工程種別コードにエイリアス名（工程種別）を作成し，カラム名を工程種別から変更しない（(b)下段）．これで(a)下段が完成する．このように，変更履歴を管理し，その背景を同時に管理し，提示できることで，設計者がその場の変更するかしないかの判断を促すことができ，これによりオブジェクト単位の管理では止めることができない変更による影響波及を狭めることに効果があることが言える．

以上のように，オブジェクト単位の管理によって，変更による影響波及範囲を狭めることができる．オブジェクト単位の管理に変更管理を行うことによって，設計者の判断を促進し，オブジェクト単位の管理に対してさらに影響波及範囲を狭めるという効果が現れていることがわかる．

4．実験による評価

設計したリポジトリの有効性を評価する．実験プラットフォームとしてPCTE(Portable Common Tool Environment)[2]を用いる．まず，設計したリポジトリスキーマをPCTEスキーマに適合化し，市販の環境に実装する．作業人数は，複数人の手によるものが多いが，簡単のため，１人で行う作業とする．そして例題に，以前に手作業で構築されたRDBを，リポジトリを利用して再構築する．RDBの規模はテーブル数が13テーブルである．

5．関連研究

リポジトリの設計変更に対応できるものが求められている中で，多くの興味深い研究が行われている．西岡[3]は，設計データを細粒度まで分解し，データ間の関係として，部分，参照，記述，汎化の４種類を定義して各データの独立性を高めている．鰺坂[4]は，ソフトウェアプロダクトとプロセスを統合的に管理している．岡[5]は，仕様書構成要素の関係を９種類に分類し，これらの関係に基づいて一貫性確保を試みている．そして，関係に属性を付加し，変更影響波及範囲を明確化しようとしている．村瀬[6]は，設計履歴をペトリネットによってモデル化し，ドキュメント単位で設計変更支援を行おうとしている．

本研究は細粒度のオブジェクト管理に変更管理を併せて管理することで影響波及範囲限定を行い易くしている．また，関係の種類の数を必要最小限とし，関係の簡略化に努めている．

6．まとめ

本稿では，仕様変更が頻発する実作業に柔軟に対応できる開発支援環境を構築することを目的とし，リポジトリスキーマの設計を行った．今後は設計したリポジトリの評価をフィードバックし，実作業をさらに分析を進め，改良を加えていく．そして，他の設計作業についても順次手を広げ，最終的には全工程を支援する予定である．RDBの概念設計，論理設計では２種類の関係（依存関係，参照関係）で十分であった．しかし，他の工程へ拡張した場合には，プロセス設計などデータの関係が非常に複雑に絡み合った部分もある．そのため関係の種類を増やす必要があるかもしれない．そしてこれに対応できる，できるだけ簡略化できるモデルを考える必要がある．リポジトリとツールの間でデータやメッセージをやりとりするのにメッセージ機構は不可欠である．今後は，このメッセージ機構も統合することも今後の課題である．

謝辞

最後に，様々なご助言をいただいた伊集院主幹研究員，本研究の機会を与えていただいた石川グループリーダ，そしてご協力をいただいた多くの方々に感謝致します．

参考文献

[1]伊集院，吉田：“マルチベンダCASEツール統合化実験とその評価”，NTT R&D，Vol.42，No.9，pp.1147-1154，1993

[2]R. Rock-Evans：“Repositories and Frameworks: a Detailed Product Evaluation”，Ovum，1993

[3]西岡，平田，渡邊：“TENSE OMSにおける細粒度情報モデル”，情報処理学会論文誌，Vol.34，No.3，pp.501-509，1993

[4]鰺坂，松本：“ソフトウェアエンジニアリング・データベースKyotoDBの設計と実現”，情報処理学会論文誌，Vol.33，No.11，pp.1402-1413，1992

[5]岡，山本，磯田：“設計情報リポジトリにおける一貫性管理機能”，情報処理学会ソフトウェア研究会91-SE-80，pp.43-50，1991

[6]村瀬，小野，他：“設計履歴とペトリネットを用いたソフトウェア変更支援”，情報処理学会ソフトウェア工学研究会90-4，pp.25-32，1993

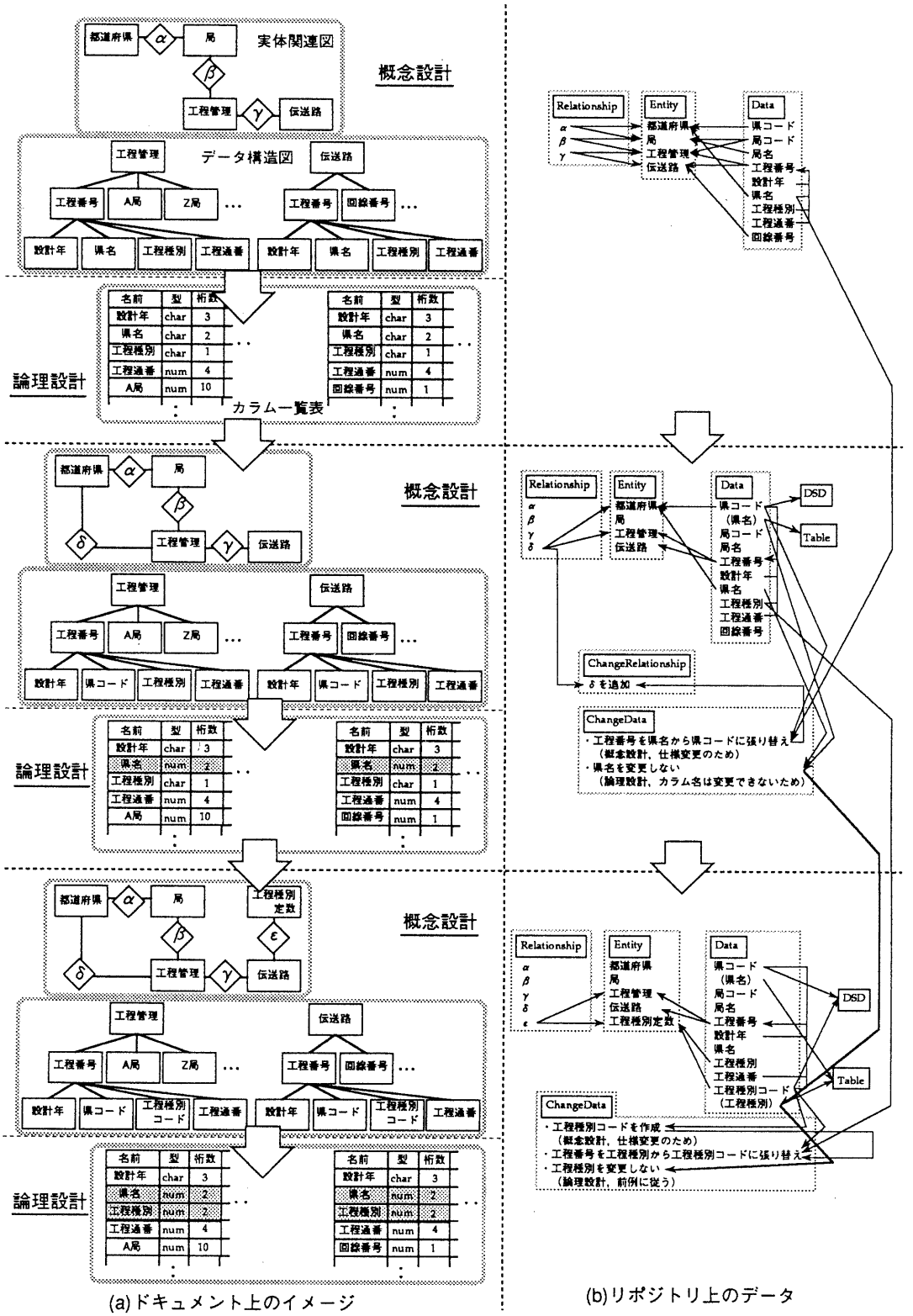
概念設計
実体関連図
都道府県
局
工程管理
伝送路
データ構造図
工程番号
A局
Z局
回線番号
設計年
県名
工程種別
工程通番
論理設計
カラム一覧表
名前
型
桁数
char
num
A局
回線番号
Relationship
Entity
Data
県コード
局コード
局名
(県名)
DSD
Table
ChangeRelationship
δを追加
ChangeData
・工程番号を県名から県コードに張り替え
(概念設計，仕様変更のため)
・県名を変更しない
(論理設計，カラム名は変更できないため)
工程種別定数
工程種別コード
(工程種別)
・工程種別コードを作成
(概念設計，仕様変更のため)
・工程番号を工程種別から工程種別コードに張り替え
・工程種別を変更しない
(論理設計，前例に従う)
(a)ドキュメント上のイメージ
(b)リポジトリ上のデータ

図6．変更事例