DEWS2003 8-B-01

設計意図を表現可能なデータベース設計手法とその設計支援ツール

家富　誠敏†　　有澤　博††

† 横浜国立大学 エコテクノロジーシステムラボラトリー　〒240–8501 横浜市保土ヶ谷区常盤台 79-7
†† 横浜国立大学 大学院 環境情報研究院　〒240–8501 横浜市保土ヶ谷区常盤台 79-7
E-mail: †{eto,arisawa}@ynu.ac.jp

あらまし　データベース設計者は、実世界の情報をモデル化した結果としてデータベーススキーマを設計するが、最終的に設計されたスキーマからは設計者がどのような考え方で実世界を捉えたか、どのような意図に従ってモデル化したかという情報を推測することは困難である。それ故に、他人が設計したデータベーススキーマは時として理解し辛く、設計者以外による高度なデータベースの利用を阻害する要因となってしまっている。これを解決するために、本研究ではデータベース設計時に設計者の意図を蓄積し、これを再利用するデータベース設計手法を提案し、設計意図を視覚的に表示・編集できるインタフェースを実現する。
キーワード　ユーザインタフェース, データの可視化, DB 設計

A Database Design Method expressing the Intention of Designer and its Authoring Tools

Masatoshi IETOMI† and Hiroshi ARISAWA††

† Ecotechnology System Laboratory, Yokohama National University　Tokiwadai 79-7, Hodogaya-ku, Yokohama, 240-8501 Japan
†† Graduate School of Environment and Information Sciences　Yokohama National University　Tokiwadai 79-7, Hodogaya-ku, Yokohama, 240-8501 Japan
E-mail: †{eto,arisawa}@ynu.ac.jp

Abstract　A database designer designs a database schema as a result which modelized the information on the real world. It is difficult to guess the intention of a designer from only the schema. How do the designer caught and modelize the real world? So, the database schema which others designed is difficult to sometimes understand, and it is interfering with advanced using the database. In order to solve this problems, in this research, we propose a method that the intention of a designer is accumulated at the time of a database design and realize the interface which can display and edit a design intention visually.
Key words　User Interface, Information Visualization, Database Design Method

1. はじめに

データベース設計者は、実世界の情報をモデル化した結果としてデータベーススキーマを設計するが、最終的に設計されたスキーマからは設計者がどのような考え方で実世界を捉えたか、どのような意図に従ってモデル化したかという情報を推測することは困難である。

しかし、データベースを高度に利用するためには、表層的なスキーマの情報だけでなく、その設計思想を十分に把握した上で行なう必要がある作業も多く存在する。例えば、スキーマを変更する上では、現状の設計思想を十分理解した上で、変更内容を吟味するのは勿論のことであるし、アプリケーションを製作する上でも、データベースの思想を理解した上でそれを活かして製作することが望ましいだろう。これを実現するためには、設計者だけが知っているデータベーススキーマに関する情報、とりわけ、どのような意図や思想に基づきスキーマが設計されたのかという情報を設計者以外に伝える方法が必要であったが、説明文など言語を用いた伝達では、情報を記述する手間のわりに、理解される情報の量が少なかった。

ところで近年、情報視覚化（IV）[1]〜[7] の研究が盛んに行われるようになると、データベースの設計意図を視覚化して表示することによって、設計者以外の人間に情報を伝えるという

可能性が開けてきた。勿論、意図や思想といった目に見えないものを、そのまま視覚化することはできないが、設計時における設計者の試行錯誤のプロセス（特に、スキーマ設計思想に関わる重要な意思決定のプロセス）を視覚化できれば、設計者の思考の過程を追体験することによって、設計意図を汲み取ることが可能であるだろう。例えば、「設計途中のスキーマでは、ある特定のインスタンスが表現できない問題があった」「初期に考えていた表現法は、ある致命的な表現上の問題があって破棄した」といった情報および、それをどのようにして解決していったかが判れば、完成したスキーマからは決して伺い知ることの出来ない設計段階での設計者の意図を知ることができ、これをスキーマの再設計などに役立てることができるはずである。

しかし、この考えに基づき、設計時の意図として、設計者の試行錯誤のプロセスを記録し、視覚化するシステムの実現を試みる場合、試行錯誤のプロセス、とりわけ試行段階のスキーマにどのような問題点があり、それをどのように解決していったかを如何にして視覚化するかが大きな問題となる。本研究では、設計時におけるスキーマ作成の履歴を保存した上で、重要と考えられる作成途中のスキーマのみを抜き出し、および各段階のスキーマに対する問題点を、設計者が実際にスキーマの検定に用いたインスタンスおよびそれに対する評価（どこに問題があったのか）をセットにして蓄積することにより表現する手法を提案する。これにより、設計者が実際に想像したインスタンスを考えた上で、どこに問題があったのかを把握でき、次の試行段階のスキーマではそれがどのように変更されていったのかを、やはりインスタンスを通して把握することが可能になる。

本稿では、まず提案手法の概念について述べ、次に本提案手法に基づき作成した途中スキーマの履歴や検定用インスタンスおよびそれに対する評価を、スキーマ設計時に蓄積するための設計支援ツールについて述べる。この設計支援ツールでは、設計者はスキーマを画面上で編集するだけで、スキーマ履歴などを自動的に抽出し、設計時の設計者の負担を極力減らす工夫をしている。これにより、設計者は通常の設計を行なった上で、ちょっとした操作を加えるだけで、自分の設計時の履歴と考え方の推移を蓄積することができ、これを必要に応じて再利用することが可能になる。

2. 関 連 研 究

設計意図とは異なるが、データベーススキーマに関する情報をより詳細に蓄積するという観点では、これまでもいくつかの研究が報告されている。代表的なものとしては、オントロジー[8]〜[10] や UML[11] 等の概念を用いて、より高度なデータベーススキーマ記述を可能とするアプローチがあるが、それらは最終的に完成したスキーマに関する付加情報でしかなく、どのような試行のプロセスを経て、そのスキーマが設計されたかといった設計意図を汲み取ることは非常に困難である。

また、設計ツールとして、スキーマを図的にデザインするツール[12]〜[14] も提案されているが、これらは所詮、視覚的デザインを目的としたもので、それらの操作のログを記録しておきスキーマ編集の履歴を後から参照することは出来ても、なぜ、そのような履歴を辿ったのか、そこにはどういった思考・意図があったのかといったことは伺い知れない。

本研究では、このような問題を解決するために、完成したスキーマに関する情報だけでなく、スキーマ設計時の履歴を大局的に記録するとともに、何故、そのような履歴を辿ったのかがわかるように、設計者の設計途中スキーマに対する見解（評価や問題点）を記録しておく手法を提案し、これを実現する設計支援ツールを構築する。

3. 設計プロセスを用いた設計意図の表現

3.1 提案手法の基本概念

本研究では、設計時における設計者の試行錯誤のプロセス（どのような過程を経て、それぞれの段階で何を考えていたか）を蓄積しておき、これを再現し、設計者の思考を追体験してもらうことで設計者以外の利用者にも設計者の意図を掴んでもらうという考えに基づき、設計時の試行錯誤のプロセスを蓄積する手法を考えていく。

その基本的な考え方は以下のようになる。

・スキーマの履歴を保存しておく
・重要な段階の途中スキーマに対する評価を保存しておく
　－検定用インスタンスを入力しておく
　－インスタンスに対する評価を記述しておく

これによって、設計者以外の利用者が、スキーマの履歴を参照しながら、設計の重要な局面・段階において、設計者が作成中のスキーマに対してどのようなインスタンスを想定し、どのような評価を下したか、すなわち「このインスタンスがきちんと表現できるからこれは良くできたスキーマである」とか、もしくは「こんなインスタンスが表現されてしまうスキーマは良くない」といった判断を行なった、という事実を知ることができる。

3.2 設計プロセスの定義

このような考えに基づき、本手法ではスキーマ設計時の設計プロセスを次のような要素の組みによって表現し、これを記録している。

設計プロセス＝｛S, D, I, J｝
－Sは試行時に作成したスキーマの集合
－DはSに属するスキーマ間の依存関係の集合
－Iは各スキーマに対して検定時に想像したインスタンスの集合
－Jは各インスタンスに対する設計者の判断（検定結果）

それぞれの要素の詳細を以下で順に説明する。

3.2.1 スキーマ集合S

設計者が対象をモデル化し、その結果として作成されたスキーマ（設計途中段階のスキーマ）を Si と表記する。i はそのスキーマを作成するまでに設計者が試行した回数であり、最終的なスキーマの設計までに費やした試行の回数を n とすれば、

S1 は始めに作成したスキーマ、

S2 は次に作成したスキーマ、
：
Sn は最後に作成したスキーマ（完成版）

を表す。

この時、最終スキーマ Sn を作成するまでに、途中で作成した全てのスキーマを集めた集合 {S1,S2,S3,...,Sn} をスキーマ集合Ｓと定義する。

3.2.2 依存関係Ｄ

これらのスキーマ集合Ｓの間には、Si を参考にして Sj を作成したという依存関係がある。この依存関係をここでは Si → Sj と表記するが、スキーマ集合Ｓに属する全てのスキーマに対する依存関係をもれなく集めたものを依存関係集合Ｄと定義する。

3.2.3 検定インスタンス集合Ｉ

スキーマ集合Ｓの要素であるスキーマ Si に対して、検定時に設計者が実際に考えてみたインスタンスを Iij と表記する。j は設計者がそのスキーマの対してインスタンスを考えてみた回数であり、スキーマ Si に対して検定した回数を m とすると、

Ii1 が始めに考えてみたインスタンス、
Ii2 が次に考えてみたインスタンス、
：
Iim が最後に考えてみたインスタンス

を表す。

スキーマ Si に対して検定に用いた全てのインスタンスの組 {Ii1,Ii2,...,Iim} をスキーマ Si に対する検定インスタンス組Ｉｉと呼ぶ。またスキーマ集合Ｓに属する全てのスキーマに対する検定インスタンス組をもれなく集めたもの {Ｉ 1, Ｉ 2,..., Ｉ n} = {{I11,I12,...} {I21,I22,...} ... {In1,In2,...}} を検定インスタンス集合Ｉと定義する。

3.2.4 検定結果集合Ｊ

スキーマ Si に対する検定インスタンス組Ｉｉに属する各インスタンスに対して、あるインスタンス Iij を用いてスキーマ Si を検定した結果を検定結果 Jij と表記する。検定結果 Jij は各インスタンス Iij に対して１つずつ存在し、そのインスタンス Iij を考えてみた場合のスキーマ Si に対する設計者の判断（そのインスタンスがスキーマで表現可能か否か？　またそのインスタンスが表現できる or 出来ないことは　スキーマとして問題がないか？　等）を表す。

検定インスタンス集合Ｉと同様に、あるスキーマに対する各インスタンスへの検定結果 Iij の組をすべてのスキーマに対して集めたもの {{J11,J12,...} {J21,J22,...} ... {Jn1,Jn2,...}} を検定結果集合Ｊと定義する。

3.3 設計プロセスの表現例

このように定義される {Ｓ, Ｄ, Ｉ, Ｊ} の組によって、設計プロセスを表現し、それによって設計者の意図を推測できるようにすることが本手法のねらいである。

実際のスキーマ設計を想定して、設計プロセスを表現した例

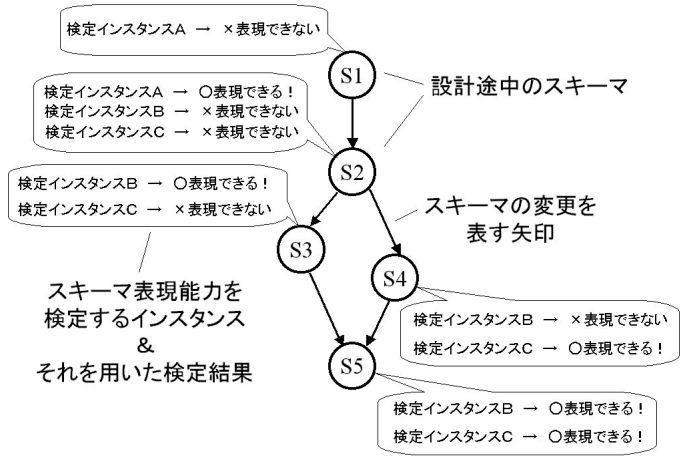

図 1　設計プロセスの表現例

が図 1 である。図 1 では、設計途中で作成された設計上の重要な途中スキーマ Si をノードとして、その間のどのスキーマの作成にどのスキーマを参考にしたのかという依存関係を Jij をノード間のアーク（有向）で図示している。このような情報を蓄積しておくことによって、設計者による設計スキーマの推移を知ることができる。

また、各スキーマを表すノードには、そのスキーマを設計した際に設計者が想像したインスタンスおよびそれに対する設計者の評価が付加されている。これを見ることによって、設計者のそのスキーマに対する考えを推測することができる。

設計されたスキーマの推移と、各スキーマに対する設計者の考えを組み合わせると、どのような考えに基づいてスキーマを改良していったのかを推測することができる。この図の例では、スキーマ S1 にはインスタンスＡが表現できない問題があり、それが次の段階ではスキーマを S2 のように改良することによって解決している。しかし、この S2 ではインスタンス B および C が表現できないという問題があり、これを解決するために S2 の改良版スキーマ S3 とスキーマ S4 を参考にして、S5 のような形のスキーマを設計した、といったような設計者が辿った試行錯誤のプロセスを追うことができ、設計意図を理解する上での大きな助けになると考えられる。

4. 設計支援インタフェースの実現

これまで述べてきた概念に基づき、設計時に設計者の意図（設計プロセス）を蓄積し、これを後から自由に参照することができる設計支援インタフェースを構築した。このシステムでは、設計者はＥＲ図[15] を用いて視覚的に画面上に表示されたスキーマをマウス等を用いて編集することにより、スキーマを設計することができる。このとき、設計者の操作のログから、設計途中スキーマおよびそれらの間の依存関係を半自動的に抽出する。

また、設計者は設計途中のスキーマを検定するために、そのスキーマに対する検定用インスタンスを入力することができる。検定用インスタンスの入力は画面上のスキーマに重ねてインス

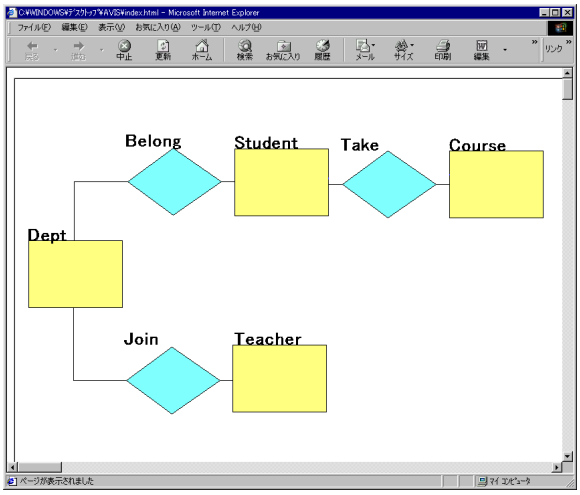

図 2　ＥＲ図によるスキーマの視覚化

コマンド	機能
Make	指定した名前の entity-type を新たに定義する
Delete	entity-type、relationship-set を削除する
Relate	entity-type 間に relationship-set を定義する
Move	entity-type の画面上の配置を移動する

図 3　スキーマ編集コマンド

タンス図を描くことによって行ない、この視覚的に表現されたインスタンス図を見ながら、そのインスタンスを用いたスキーマ検定結果を設計者のコメントやインスタンスの各部に対するカラーリング等を組み合わせて表現し、これを蓄積することができる。

このような設計プロセスを設計時に蓄積しておくことによって、設計者以外の利用者は以下のような操作を行なうことができる。

- 設計時における設計途中スキーマの推移を有向グラフによる表現や実際の設計操作を簡易アニメーションで表現したものによって捉えることができる
- スキーマの推移を見ながら、必要に応じて、各途中スキーマで、どのような検定が行われたのかを視覚化された検定インスタンスおよびそこに付加された検定結果から知ることができる

これによって、設計者の意図をその他の利用者に伝えることが可能になる。それぞれの機能の詳細を以下で順に述べる。

4.1　視覚的スキーマ設計

本インタフェースでは、スキーマ設計をディスプレイ上に視覚化されたＥＲ図 (図 2 参照) を編集することによって行なう。その際用いることができるコマンドは図 3 のものがある。

これらのコマンドを組み合わせることによって設計者は視覚的にスキーマを設計することができる。

4.2　設計中のスキーマ履歴の保存

設計者はＥＲ図をディスプレイ上で描くことによって、スキーマを設計していくが、ある程度の形が出来上がった段階で、途中段階のスキーマを記録し、後術するスキーマの検証を行なうことができる。このように、スキーマ設計時に作成途中のスキーマを要所要所で記録してもらうことによって、設計者が大局的にどのような試行を経て、最終的なスキーマにたどり着いたのかを記録しておく。また、このとき全ての編集作業の履歴はログとして保存しておき、逐一、後から参照することも可能にしている。

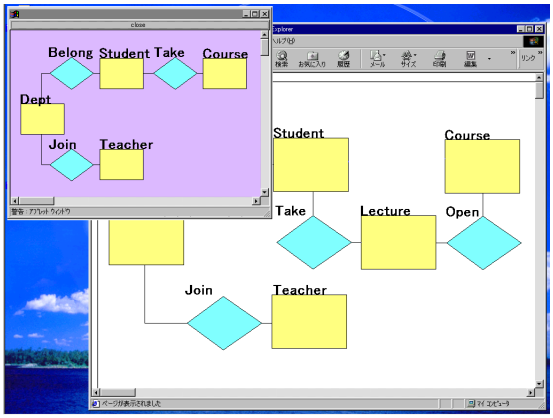

図 4　スキーマ参照ウィンドウ

4.3　設計途中スキーマの依存関係

設計途中のスキーマを保存する際、そのスキーマがそれ以前に作成されたどのスキーマを参考にして、作成されたのかという依存関係を記録しておく必要がある。本インタフェースでは、これらの依存関係をある程度自動的に抽出して記録するために大きく以下の２つの機能を実現している。

- そのスキーマを編集する際の原型となったスキーマに対して依存関係を自動的に記録する
- 既に作成したスキーマを別ウィンドウ（図 4 参照）で呼び出して閲覧できるようにし、ここで呼び出したスキーマと現在編集中のスキーマの間に依存関係を候補として記録する（候補を最終的な依存関係として採用するかどうかは設計者の判断による）

これらの機構によって、半自動的に設計途中スキーマ間の依存関係を記録している。

4.4　スキーマの検証

設計者は設計途中スキーマに対して、そのスキーマの検証を本インタフェース上で行なうことができる。

スキーマの検定は

- そのスキーマに対して設計者が検定時に実際に考えてみたインスタンスを入力
- そのインスタンスを見て、スキーマに問題があるかどうかを判定し、判定結果を記録する

という試行を必要回数繰り返すことによって行なう。

4.4.1　検証用インスタンスの入力

本インタフェースでは、スキーマ検証時にサンプルとしてインスタンスを入力することができる。このインスタンスはスキーマ図に重ねて視覚的に表示される（図 5 参照）。

検定インスタンスの入力コマンドは図 6 のようなものが用意されている。

これらのコマンドを用いて、インスタンスを入力し、スキー

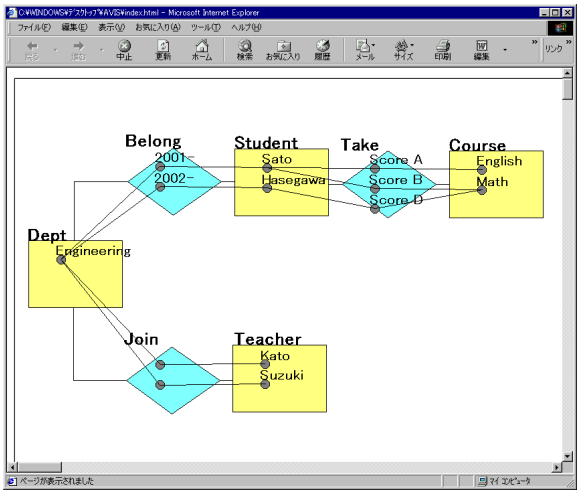

図 5　検定インスタンスの表示

コマンド	機能
Make	指定した entity-type に entity を生成する
Delete	entity を削除する
Relate	entity 間をつながりで結びつける
Cut	entity 間のつながりを切る

図 6　インスタンス入力コマンド

マと照らし合わせて閲覧し、検証する際の参考にすることができる。

4.4.2　判定結果の記録

設計者は、入力したインスタンスを検討して、そのスキーマに問題があるかどうかを判定する。この判定結果を記録しておくために、本インタフェースでは次の２つの機能がある。

- 入力したインスタンスに対して判定のコメントを付加することができる
- 視覚化されたインスタンスの各部分を自由にカラーリングできる

これらの機能を組み合わせることによって、インスタンスの各部分がどのように問題があるのかを記録する。

例えば、カラーリングによって

- インスタンスの一部に青く着色→現在のスキーマでは表現できないが、実際には表現したい内容であることを示す
- インスタンスの一部に赤く着色→現在のスキーマでは表現できてしまうが、実際には表現したくない内容であることを示す

というようにインスタンスの着目すべき部位を視覚的に表現し、そこに詳細なコメント（どうして問題があるのか等）を記述しておくことで、設計者が設計時にどのような考えをスキーマに抱いていたのかを蓄積することができる（図 7 参照）。

4.5　設計プロセスの表示

このように蓄積した設計時の設計プロセスに対して、設計後、必要に応じて設計プロセスを再表示して確認することができる。

設計プロセスの表示には大きく以下の２つの方法がある。

- 設計時のスキーマ変更の履歴をグラフで表示する。スキーマ履歴を図 8 のようなグラフで一括表示し、詳しく知りたいスキーマをクリックすると、そのスキーマ図が表示される（星マークはそのスキーマに付加されている検定インスタンスを表す）。
- 設計時のスキーマ変更の履歴をアニメーションで表示する。設計時の順番通りに、設計開始からのスキーマが設計されていく様子をスキーマ図のアニメーションで表示する。

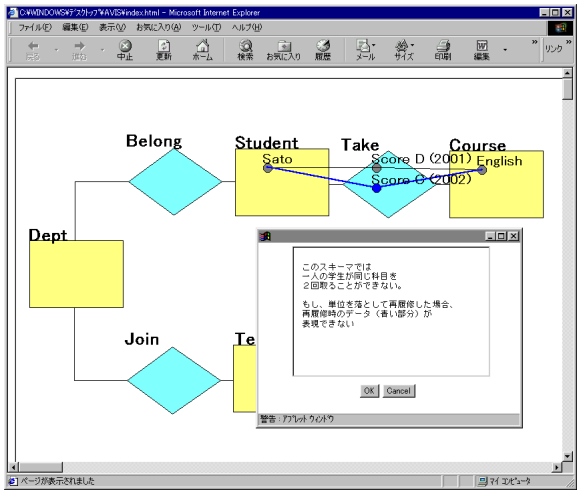

図 7　検定結果の表現

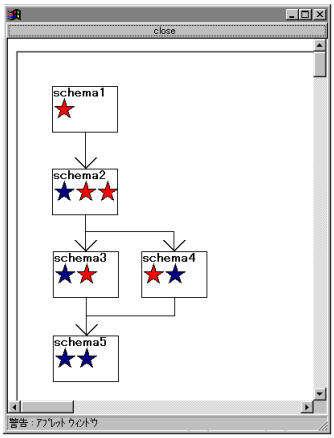

図 8　スキーマ履歴のグラフ表示

これらの機能を使いわけてスキーマ履歴をチェックした上で、各スキーマごとに検定用インスタンスや検定結果（図 7）を参照することができる。これによって、スキーマに対し設計者が何を問題として捉え、どのような改良を加えていったのかを設計者の立場で見ていくことができる。

4.6　インタフェースを用いた設計プロセスの表示

図 9 は実際にデータベースを設計した際の設計プロセスを画面に表示した例である。

画面上中央にあるのが、スキーマ履歴を表したグラフであり、ここでは設計者が schema1 を参考にして schema2 を作成していった場合の例であることを表している。このスキーマグラフ上で、スキーマを表す矩形や検定インスタンスを表す星型をマウスでクリックすると、そのスキーマやインスタンスが画面に図で表示される。この例では、２つの検定インスタンスを画面に表示させているが、画面左が schema1、画面右が schema2

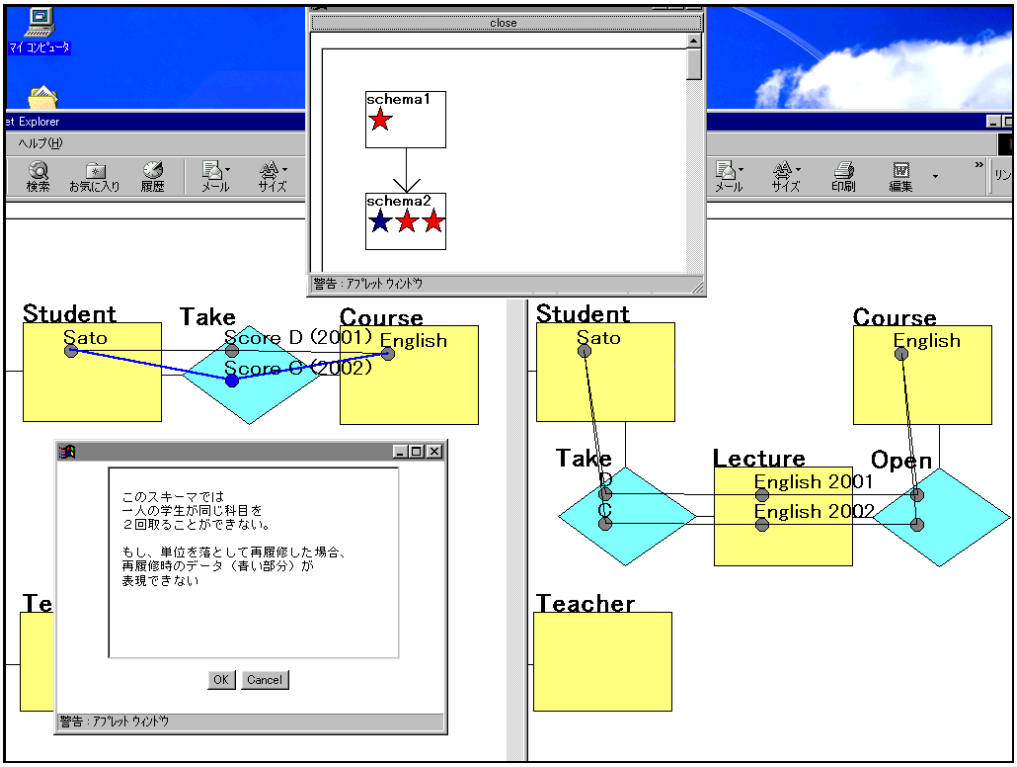

図 9　設計プロセスの表示

の検定インスタンスである。ここでは schema1 に付加された検定インスタンスおよびコメントによって、設計者がどのような問題点に着目していたのかが示され、schema2 に付加された検定インスタンスによって、それが設計者がこの問題をどのように解決したのかが実例で示されている。さらに詳細を知りたい場合には、設計者の行なったスキーマ変更操作を細かくアニメーションで見ていくこともできるようになっている。

5. まとめと今後の課題

本研究では、設計者の設計意図を表現するために、設計プロセスを蓄積・視覚化する手法を提案し、そのためのインタフェースの構築を行なった。

現状では、設計プロセスを蓄積した上で、それを視覚化するだけのインタフェースだが、それ以外の再利用方法、スキーマ修正時に設計者のログを参照して、過去において問題があるとされている変更であった場合にそれを警告するといったような応用についても、今後は研究していきたい。

文　献

[1] Stuart K. Card. : Visualizing retrieved information: A Survey, IEEE Computer Graphics and Applications, Vol. 16, No. 2, pp. 63-67, 1996

[2] Daniel A. Keim, Hans-Peter Kriegel : VisDB: Database Exploration Using Multidimensional Visualization, IEEE Computer Graphics and Applications, Vol. 14, No. 5, pp. 40-49, 1994

[3] 佐野 綾一, 波多野 賢治, 田中 克己 : 自己組織化マップを用いた Web 文書の対話的分類とその視覚化, 情報処理学会研究報告, Vol. 98, No. 57, 98-DBS-116(1)-5, pp. 33-40, 1998

[4] 磯部 成二, 黒川 清, 塩原 寿子, 飯塚 哲也, “視覚化による多次元データ分析システム : INFORVISER,” 情報処理学会論文誌, Vol. 40, No. 5, 1999

[5] 武田 浩一, 野美山 浩, “テキスト情報の可視化を利用した情報検索,” 情報処理, Vol. 41, No. 4, 2000

[6] 古畑 理香, 藤代 一成, 市川 哲彦, 竹島 由里子, “GADGET/IV: 情報可視化の半自動設計支援環境,” 第 11 回データ工学ワークショップ (DEWS 2000), 2000

[7] 尾下 真樹, 牧之内 顕文 : オブジェクト指向データベースの半自動可視化環境, 電子情報通信学会技術研究報告, Vol. 100, No. 31, DE2000-5, pp. 33-40, 2000

[8] Bernhard Thalheim, “Component Construction of Database Schemes,” Conceptual Modeling - ER2002, Springer LNCS 2503, pp.20-34, 2002.

[9] Giancarlo Guizzardi, et al, “On the General Ontological Foundations of Conceptual Modeling,” Conceptual Modeling - ER2002, Springer LNCS 2503, pp.65-78, 2002.

[10] Farshad Hakimpour, Andreas Geppert, “Global Schema Generation Using Formal Ontologies,” Conceptual Modeling - ER2002, Springer LNCS 2503, pp.199-213, 2002.

[11] Sergio Lujań-Mora, et al, “Multidimensional Modeling with UML Package Diagrams,” Conceptual Modeling - ER2002, Springer LNCS 2503, pp.199-213, 2002.

[12] A.Grau, et al, “The TROLL Approach to Conceptual Modelling:Syntax,Semantics and Tools,” Proceedings of the International Conference on Conceptual Modelling, pp.277-290, 1998.

[13] Sudha Ram, et al, “DISTIL: A Design Support Environment for Conceptual Modeling of Spatio-temporal Requirements,” Conceptual Modeling - ER2001, Springer LNCS 2224, pp.70-83, 2001.

[14] Virginie Detienne, Jean-Luc Hainaut, “CASE Tool Support for Temporal Database Design,” Conceptual Modeling - ER2001, Springer LNCS 2224, pp.208-224, 2001.

[15] P. P. Chen, “The Entity-Relationship Model : Toward a Unified View of Data,” ACM Transactions on Database Systems, Vol.1, No.1, 1976.