データベースシステム 103－2
（１９９５．５．２５）

常時検索機能を持つデータ主導情報システム構築ツール

金玄坤，　穂鷹良介

筑波大学社会工学研究科

本稿では，データ主導アプローチにもとづいて，情報システムの分析，設計，ディバギングなどできる限り広い範囲をプロトタイピングによりカバーできる構築ツールISDDのアイディア及びその開発について議論する．

ISDDは常時検索機能を導入することによってディバギングの為のヒューマンインターフェースを改善し，結果的に効率的なディバギングを可能にすることに特徴がある．本稿では常時検索の意義及びそれを実現するためのUpdate Propagationとその制御ロジックについて述べる．そして現在まで行ってきたISDDシステムの実装に関して説明する．

A Data-Oriented Information System Development Tool ISDD With Constant Query Interface

HyeonKon Kim,　Ryosuke Hotaka
email: kim@wiz.sk.tsukuba.ac.jp, hotaka@shako.sk.tsukuba.ac.jp

Institute of Socio-Economic Planning, Univ. of Tsukuba

This paper presents the basic idea of a data-oriented information system development tool named ISDD that aims by prototyping to cover as wide a range of development cycle as possible, including analysis, design, and debugging.

ISDD is characteristic in that it has improved the human interface for debugging by introducing the concept of constant query. The meaning and role of constant query is discussed. Update propagation and its control logic to realize constant query is also explained.

1. はじめに

現在情報システム構築をめぐって数多い設計ツールが開発されている．これらの設計ツールのほとんどは情報システム構築の一部の局面だけをカバーするものであるのが普通で，実際の情報システム構築のためには設計ツール間の互換性の問題が起きるのが現状である．

こういう状況を考慮し本稿では，データ主導アプローチにもとづいて，情報システム構築においてできる限り広い範囲をカバーできる構築ツールのアイディア及びその開発について議論する．

具体的には，設計方法論とは独立に，情報システムの分析，設計，ディバギングなどシステム構築及び利用の全過程をプロトタイピングによりカバーできる構築ツールISDD (Information S ystem Designer's Desktop)の開発に関して述べる．

勿論現在もシステム構築の殆どの局面がカバーできる構築ツールはある．しかしこれらのツールは各局面をサポートするサブツールのただの集まりとしてできているのが普通である．それとは対照的にISDDはデータ主導でシステムを構築していくという基本原則を守りながら，それを実現するメカニズムとして常時検索機能という考え方を導入し，単一のツールでシステム構築の各局面をサポートする点で他のツールとは異なる．

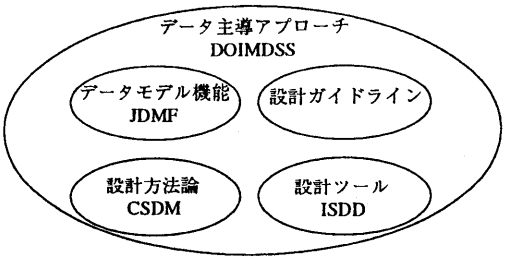
本論文で提案する構築ツールISDDの背後にはデータ主導アプローチDOIMDSS(Data Oriented Information Modeling Design Supporting System)がある [1,5].

DOIMDSSとは，「情報システムの本質はデータ側にあり，データとデータの間の変換の実現としてプロセスが介在する形で情報システムが出来上がる」という認識にもとづいて，データの把握を先行させていくことによって情報処理の単純化と一意なシステム設計をめざす情報システム設計方法論である．

その目的の為にDOIMDSSでは，データを規定する堅固な基礎としてデータモデル機能JDMFを用いると同時にデータ設計の一意性を高めるための設計ガイドラインを提示している．そしてこれらを理論的なベースにしてDOIMDSSでは，データ主導の具体的な情報システム設計手順をもつ論理設計方法CSDM (Conceptual System Design Method)がある [3].

ISDDはDOIMDSSプロジェクトの一部として位置づけられるもので，DOIMDSSの思想をコンピュータ上で実現し実際のシステム構築をサポートする設計ツールである．ISDDの位置づけを表すと図1のようになる．

図1 ISDDの位置づけ



以下では特に，ISDDの主要特徴である常時検索機能及びそれを実現するUpdate Propagationのロジックに焦点をあわせて議論する．その後ISDDシステムの実現について説明する．

2. 常時検索の意義

2.1 常時検索の定義と目的

常時検索とは，あらかじめユーザが必要とする全ての検索画面が定義され，システムの開発中も含めて常時その検索結果が表示される状態になっているものである．

通常の検索の場合は，データベースに変化があっても該当検索画面をユーザに再表示させる操作をする前まではその変化はデータベースの中に閉じて起こる．それとは対照的に常時検索の場合は，ほしい検索画面が全てユーザの目の前に開いてあって，データベースの変化があると同時に関連した全ての検索画面の内容もその変化を反映し変わる．そういう意味で常時検索はRead While Changingであると言える．

常時検索の一番原始的な形として株価情報の検索画面の例があげられる．顧客の目の前に各会

社の株価を表す画面が表示され，その会社の株式取引の情報の変化にもとづいてその画面の内容もどんどん変わっていく．

では常時検索概念は情報システム構築においてどういう役割を果たせるのか．まず結論から言うと，常時検索機能を導入することによってシステム開発中に必要不可欠な作業であるディバギングの為のヒューマンインターフェースを改善し，結果的に効率的なディバギングを可能にする．

一般的に，データベースの内容というのは目的と認識が違ういろいろな角度から見られる．そして一カ所の所でバグがあれば修正をするけど，その修正によってまた他の所で副作用が生じるのが普通である．

エラーを直すときは関連したある一つの検索画面で眺めて原因を直すのが普通である．そしてそのエラーの修正によって処理内容が変わることになる．しかし，その処理は他の複数個の検索画面にも関連づけられているのが一般的であるので，変わった処理の内容が関連したほかの検索画面全てに対して有効かどうかの保証はない．元の検索画面の内容だけが正しくなるだけではだめで，関連した他の画面も全部正しい状態を保たなければならない．

この点から考えると，ほしい全ての検索画面を全部表示しておいて常時検索機能を付与することは正しいディバギングのために非常に有効な役割をする．

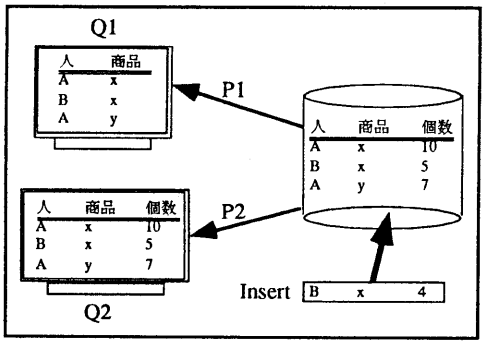
2.2 例

ここではディバギングの目的のために常時検索機能が有効であることを示す例を考える．

情報システム開発においてバグは多様な理由から起きる．例えば，データベースの内容の追加，更新，削除などに伴ってエラーが起きるケース，クラスの定義などに関する認識の差が原因でエラーが起きるケース，検索結果の導出ロジックの変更によってエラーが起きるケースなどが考えられる．以下ではこれらのケースに関する例を提示する．

まず最初にデータベースの変化によってエラーが起きるケースを考えてみる．例えば，図2で示すように商品購入というデータベースがあり，それに関連した二つの検索画面Q1（だれが何を購入したか）とQ2（だれが何をどのぐらい購入したか）があると仮定する．そしてQ1，Q2は各々導出ロジックP1，P2を持ち，P2には商品購入データベースに人と商品が同一である場合はそのようなケースを全部集めて個数だけを合算するというロジックがないというバグがあったとする．その時，図で表したようにデータベースに4番目のインスタンスを入れると，Q2でBがxを購入したのが5から9に変わるべきなのに変わらないことになり，間違った検索結果を出してしまう．

図2　常時検索の例1

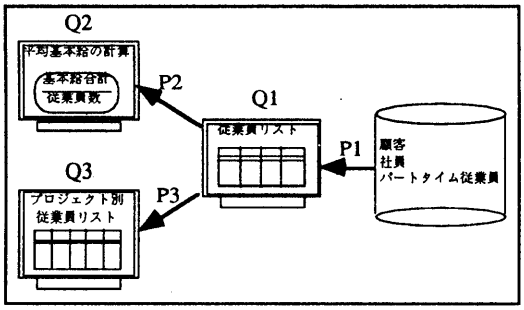


こういう例から考えると，常に全部のデータを出しておいて矛盾があるかどうかチェックできる常時検索機能は，プログラムのディバギングに有効であることが分かる．

次は，認識の差と導出ロジックの変動の両者が原因でエラーが起きるケースをあげる．例えば，ある会社の次のような情報システムを考えてみる（図3）．データベースの中に顧客，社員，パートタイム従業員などの情報が蓄積されている．そして従業員全部のリストを出す画面Q1と会社全体の平均基本給を計算する画面Q2，そしてプロジェクト別従業員リストを出す画面Q3の三つの検索画面が必要であるとする．またQ1，

Q2, Q3は各々データの導出ロジックP1, P2, P3を持つが, P2とP3にはP1が共通のプロセスとして入っていると仮定する.

図3　常時検索の例2



最初にQ2の結果にはパートタイム従業員は排除されているとする. その場合, 一般的に平均基本給の計算には正社員だけを考慮するのでQ2の結果は正しくなる. しかし, Q3の為には各プロジェクトに属する全ての従業員を入れる必要がある.

それで今度はQ3の目的の為にQ1の従業員リストにパートタイム従業員も入れるようにプロセスP1を変更したとする. そうするとQ3は正しくなるが, 今度はQ2で平均計算のときパートタイム従業員の数だけ社員数が増えて計算されるのでそれだけ平均基本給が低くなりQ2の結果が間違ってしまう.

こういうバグは情報システム開発の途中よく起きる問題である. もしこの場合, Q1, Q2, Q3などユーザに必要な全ての画面を全部ユーザの目に見えるようにして常時検索を行っている状態にすると, 効率的なディバギングが可能になる.

3. ISDDおける常時検索の実現

ここではISDDにおいて常時検索機能を実現していくとき, スプレッドシートにおける常時検索の考え方を活用したアイディアと実際の常時検索機能の実現ロジックであるUpdate Propagationについて議論する.

3.1 スプレッドシートにおける常時検索機能の活用

スプレッドシートの場合は, 我々の常時検索機能に近い思想を持っている. 故に, 情報システムの構築における常時検索機能の実現の為にスプレッドシートにおける常時検索の考え方を十分活用した.

スプレッドシートでは, まず全てのデータがスプレッドシート上に現れている. そしてスプレッドシートでは全てのDerived DataがSynchronizeされている.

例えば, 図4のようなスプレッドシートがあると仮定する. ここでA1からC4までのデータは自立性をもつベースデータである. それに対してD2とC5はDerived D ataである. Derived Dataの場合は常にデータと一緒にそのデータの導出ロジックが定義される. そして各々のDerived Dataを選択するとそのデータの導出ロジックが別の定義情報欄に表示される.

図4　スプレッドシートの例

	A	B	C	D
1	0	0	1	
2	5	6	7	18
3	3	2	0	
4	5	3	2	
5			10	

例えば, D2をクリックすると定義情報欄にはD2のデータの導出ロジック `=SUM(A2:C2)` が表示される. 同じにC5をクリックすると定義情報欄には `=SUM(C1:C4)` が表示される.

そして, ベースデータを変更すると, そのデータと関連を持つ全てのDerived Dataもその変化を反映し変わっていく. 例えば, 図5で示すように, もしC2の値を7から9に変更したら, D2の値は18から20に, C5の値は10から12に自動的に変わる.

図5　スプレッドシートにおける常時検索の例

	A	B	C	D
1	0	0	1	
2	5	6	9	20
3	3	2	0	
4	5	3	2	
5			12	

我々はスプレッドシートの考え方をデータ主導情報システム構築ツールISDDに次のように取り入れたい．

まず，データベースを含む情報システム全体を一つのスプレッドシートであるように考える．スプレッドシートでベースデータとDerived Dataを含む全てのデータをスプレッドシート上に表したことと同じに，ユーザが必要とする全ての画面を全部ISDD上に置いておく．

データベースに対するQueryは，結局そのデータベースからDerived Dataを用意することであると言える．故に，スプレッドシートの全てのDerived Dataには常にそのデータの導出ロジックが定義されることと同じに，全ての検索画面にもデータベースからその検索画面上のDerived Dataが導出されるロジックを定義する．

そしてDerived DataのSynchronizationを常に保つようにする．スプレッドシートでベースデータに変化が起こったら同時に関連したDerived Dataの内容も変化することと同じに，ISDDでの検索のロジックも常時検索機能を入れて常に検索を行っているように工夫する．

3.2　Update Propagationとその制御

前述したように本稿では情報システム構築において常時検索機能を活用することを提案している．

ところで，本当の意味での常時検索とは常時データベースの状態をチェックし検索画面に反映することであるが，これは非効率的である．それで現実的には常時検索と論理的に同一な機能を持たせながらも効率を維持することが必要である．

即ち，データベースの内容が変わったときだけ，その変わった内容だけを関連された検索画面のところで変わればいいのである．これがUpdate Propagationの考え方である．

普通の検索の場合にはデータベース内容の更新はデータベースに限って行われ，検索画面の再表示指示が発生するときまでは更新の結果が検索画面には反映されない．故に，普通の検索の場合はUpdate Propagationの問題は起こらない．効率を考慮し現実的に常時検索機能を持たせようとするとき，自然にUpdate Propagationが必要になる．

ところでUpdate Propagationによる常時検索機能を実現しようとすると，どういうロジックでUpdate Propagationを実現するべきかの問題を解決しなければならない．

極端には各々のUpdateのメソッドの中で関連した検索画面へのUpdate Propagationのロジックを全部書く方法が考えられる．しかしこの方法はUpdate Logicにあまりにも重い負荷がかかるだけでなく，全ての検索画面を事前に全部予想しなければならないので理論的に不可能である．

故に，常時検索機能を実現する目的のための方法を新たに工夫する必要がある．そのためには基本的にUpdate LogicとRetrieve Logicとは別途にUpdate Propagation Logicを管理するメカニズムが必要である．

本稿では，次に述べるようにUpdate Propagationを管理するメタデータベースを作成することによって解決する．そのときUpdate Propagationの具体的な制御の方式としては次のようなものが考えられる．

1．属性値レベル：各々の検索画面は，その中に現れるインスタンスの属性値と直接な関連を持ち，結果的にUpdate Propagationがそのインスタンスの属性値と直接つながって行われる．

2．インスタンスレベル：各々の検索画面は，その中に現れるインスタンスと直接な関連を持ち，結果的にUpdate Propagationがそのインスタンスと直接つながって行われる．

3．クラスレベル：各々の検索画面は，その中

に現れるインスタンスが属するクラスと関連を持ち，結果的にUpdate Propagationがクラスレベルで行われる．

上記の代替案の中で代替案1は理想的であるが非効率的である．故に，代替案2と代替案3のどちらかを選択して使うことが合理的である．そのとき，代替案の選択基準としては各々の検索の結果として現れるインスタンスの数を使うのが考えられる．即ち，適切な数のインスタンス以下ならば，インスタンスと検索画面との関連を維持する（代替案2）．そしてインスタンスの数が適切な基準を越えて多ければクラスレベルにする（代替案3）．本稿ではインスタンスの数10を基準にしているが，これは設計者が任意に決めることができる．

図6と図7では代替案2と代替案3の例を各々表している．まずクラスレベルUpdate Propagationの例を説明する．この場合は図6で示しているように，各々の検索画面が出来上がるときその画面の名前(ViewName)とその中に現れるインスタンスが属するクラス名(ClassName)を属性としてもつUpdate Propagationのインスタンスが登録される．そしてデータベースに変更が起こったらこのUpdate Propagation表で関連したクラスがあるかを探索しそれにつながっている全ての検索画面の内容も更新する．

図6　クラスレベルUpdate Propogation

ViewName	ClassName
Student128	Student
Student129	Student
Customer130	Customer

これに対して図7はインスタンスレベルのUpdata Propagationを表す例である．各々の検索画面が出来上がるときその中に現れるインスタンスの数が10を越えないと，その画面の名前(ViewName)とその中に現れるインスタンスが属するクラス名(ClassName)と同時にそのインスタンスの集合(InstancesInView)を属性としてもつUpdate Propagationのインスタンスが登録される．そしてデータベースに変更が起こったらこのUpdate Propagation表で関連したインスタンスがあるかを探索しそれにつながっている全ての検索画面の内容を更新する．

図7　インスタンスレベルUpdate Propogation

ViewName	ClassName	InstancesInView
Customer131	Customer	{<Customer>,<Customer>... }
Customer132	Customer	{<Customer>}

実際の実装システムではクラスレベルのUpdate Propagationとインスタンスレベルの Update Propagationの情報は両者ともにUpdate Propagationというメタデータベース表のなかに蓄積される．ただクラスレベルのUpdate Propagationのインスタンスの場合は属性IntancesInViewの値がヌル値になる．

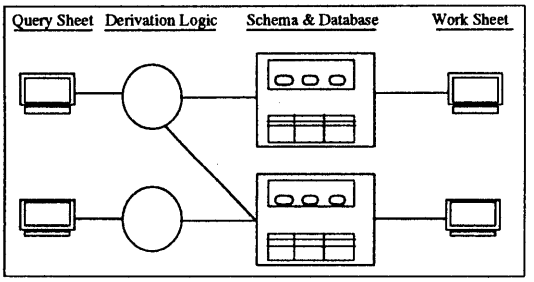
4. ISDDシステムの実装

データ主導アプローチDOIMDSSの考え方にもとづいて常時検索機能を持つ実際の情報システム構築ツールISDDシステムを開発中である．本稿では現在まで行った実装作業を報告する．

ISDDシステムの実装はマッキントシュ上でオブジェクト指向ビジュアル言語Prographを使って行った[7,8]．Prographは効果的かつ豊富なディバギング機能を備えておるし，ディバギングのためのヒューマンインターフェースがよくできていて本システムの開発には適切な言語であった．

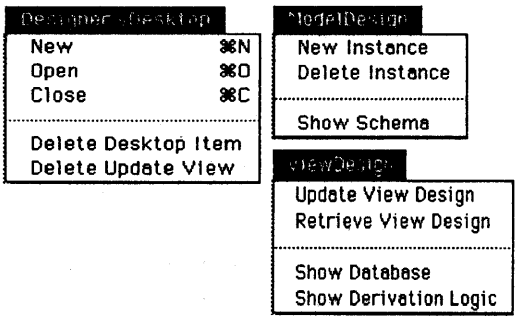
ISDDシステムのアーキテクチャは図8のように表現することができる．検索画面であるQuery SheetはQuery Fieldからのみ構成され，データベースからの検索結果が表示される．更新画面であるWork Sheet はQuery Field とユーザが新規に投入するデータが表示される D ata Entryから構成され，データの追加，修正，削除などデータベースの更新作業が行われる．

図8　ISDDシステムのアーキテクチャ



ISDDシステムでのコマンドは基本的にメニュー方式で実行される．そして情報システム開発のための作業は，開発作業台であるISDD上で行われる．ISDDシステムのメニューの構成は図9のようになっている．

図9　ISDDシステムのメニュー構成



Designer's Desktopメニューは開発作業台自体に関するもので，次のようなメニュー項目をもつ．

New, Open, Close：作業台ISDDの新規生成，既存のISDDのオープンとクローズを各々行う項目である．ISDD上には必要な検索画面，更新画面，スキーマ図など全ての開発作業の結果が表示されるので，開発システムの規模を考慮して作業台の大きさは15,000 x 15,000まで拡張可能である．

Delete Desktop Item, Delete Update View：ISDD上に表示した開発作業の結果物をISDDから削除する機能である．開発作業は設計と設計結果の修正の繰り返しであるので，それをサポートするための機能の一つである．

Model Designメニューはスキーマの設計及びスキーマ設計のベースになるデータモデル機能自体の拡張などの目的のためのもので，次のメニュー項目がある．

New Instance, Delete Instance：クラスを含む全てのオブジェクトをインスタンスとして考えることによって，応用クラスの生成，応用クラスの属性の生成及び応用クラスのインスタンス生成などが全てNew Instanceを選択することによって実行される．それだけでなく，JDMFのCore Modelの自己記述性を活用し，New Instanceによってデータモデル機能自体の拡張も可能である [4]．Delete Instanceは逆にNew Instanceよって生成されたインスタンスを削除する．

Show Schema：New Instanceによって作成したスキーマの設計結果をISDD上に表示する．

View Designメニューは更新画面，検索画面及びその導出ロジックに関する作業を行うもので，以下のようなメニュー項目をもつ．

Update View Design：更新画面であるWork Sheetを作成するもので，基本的には対象クラスを選択することによってWork Sheetを自動生成する．

Retrieve View Design：検索画面であるQuery Sheetを作成するもので，このメニュー項目を選択すると開かれる導出ロジック定義画面で導出ロジックを第一次述語論理式で書くことによってQuery Sheetを自動生成する．検索画面の導出ロジックの一般的な形式は次のように定義される．

<query> ::= { <target list> | <sentence> }

where <target list> ::=

<indvar>[.<attr>][[, <indvar>[.<attr>]]....]

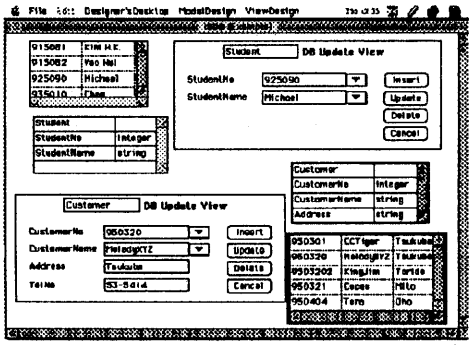
<sentense>は強力な表現力をもつKIFの概念を借用して使う [6].

Show Derivation Logic：生成されISDD上に表示されている特定検索画面をクリックした後このメニュー項目を選択するとその検索画面の導出ロジックの定義画面を表示させる．

Show Database：選択されたクラスの全てのインスタンスの情報をISDD上に表示させる．そしてその画面もQuery Sheetの一つである．

図10はこれらのメニュー項目の実行によってできたISDDの一つの例を表している.

図10　ISDDの例



そして上記の機能によってISDD上に出したスキーマ，検索画面，更新画面などの全ての項目に対して，適切な場所への移動と大きさの変更なども可能である．そしてこれらの項目に対して常時検索機能がサポートされているので，例えばある更新画面でデータベースの内容の変化が起きたとすると，Update Propagationメタデータベースでその変化と関連された全ての検索画面を探しその変化を即時反映させる.

5. おわりに

以上でデータ主導アプローチDOIMDSSの考え方にもとづいて常時検索機能を持たせることによって，情報システムの分析，設計，ディバギングなどシステム構築及び利用の全過程をプロトタイピングによりカバーすることを目的とする情報システム構築ツールISDDシステムのアイディア及びその実装作業について述べた.

特に，常時検索機能を導入することによってディバギングの為のヒューマンインターフェースを改善し，効率的なディバギングを可能にするという考え方に重点をおいて議論した．実際のISDDシステムの実装におけるディバギング作業自体も実は常時検索機能を活用して行なうことにより，その有効性を確認した.

今後もシステム開発作業を継続していく予定である.

参考文献

[1] 穂鷹良介，データ主導情報モデリング設計支援システムIMDSSについて，データベースシステム92-7，情報処理学会，1993.3.

[2] 日本規格協会，データモデル機能JDMF/MODEL-1992，1993.

[3] 金玄坤，データ主導情報システム設計方法論に関する研究，筑波大学社会工学研究科修士論文，1995.1.

[4] Bjorn, M., Kim, H .K., Hotaka, R., A S elf-Descriptive C onceptual S chema M odelling Facility, its Implementation a nd Extension, Proc. of IFIP WG8.1 I nt'l C onference on I nformation System Concepts Towards a Consolidation of Views (ISCO3), March 1995.

[5] Hotaka, R., Bjorn, M., Data Oriented Approach to B usiness I nformation M odeling, P roc. of I FIP TC6/WG6.1 I nt'l C onference o n O pen Distributed Processing, North-Holland, 1994.

[6] Genesereth, Michael R ., Fikes, R ichard E ., Knowledge I nterchange F ormat V ersion 3 .0 Reference Manual, Stanford University, June 1992.

[7] Prograph International, Prograph CPX User Guide, Prograph International, Inc., 1993.

[8] Prograph International, P rograph CPX A BC Reference, Prograph International, Inc., 1993.