データベースシステム 109－4
（1996．7．24）

メタ制御による情報ベースのブラウズイング法

塩谷 勇　三浦 孝夫
産能大学
259-11 神奈川県伊勢原市上粕屋 1573
{shioya,miura}@mi.sanno.ac.jp

この論文では，情報ベースのブラウズイング支援メカニズムの一つの方式を述べる．情報ベースの検索では質問系列の最終的な答え以外は必ずしもすべての答えを求める必要がなく，質問の部分解と計算中の解をつぎの質問計算に如何に反映させるかが問題である．ここでは，第一階言語を質問言語とするブラウズイング支援方式について報告する．

A Browsing Method for Information Base based on Meta-Control

Isamu SHIOYA and Takao MIURA
SANNO College
Isehara-shi, Kanagawa 259-11 Japan

This paper describes a new method to support browsing for information bases. The browsing are sequences of queries, and we do not ask to compute all the solutions except final step. Then, we suspend the current evaluation, and start the evaluation of the next query. We propose the method how the part of solutions and their intermediate solutions for the current are available to evaluate the next.

1 はじめに

情報ハイウエイの発達は新たな情報処理の技術を必要としている．従来のデータベースは利用者の意図した情報が格納され，利用者があらかじめ検索の意図を確定して，それをスキーマに基づいて質問によって表現し，大量のデータの検索をデータベース技術を背景に可能しめている．格納されている情報の意図は一意，すなわち宣言的に意味が一意に決まる特徴がある．

一方，情報ハイウエイの情報は自然発生的なデータが時間とともに変化し，大量の情報が分散して格納されている．この情報は場合によって矛盾や欠落を含み，情報が蓄積される段階で分類や属性が変化する(スキーマ進化)というこれまでの多くのデータベースでは想定しなかった所謂情報ベース特有の性質を有し，従来のデータベース技術では対応しきれない新たなデータベースの必要性に直面している．

特に，利用者の意図する情報を自然発生的に分散している情報から質問に如何に反映させるかのメンタルなボトルネックがある．これまでの既知の確定した情報(スキーマ)から必要とする情報を抽出するデータベースとは異なるアプローチを必要とする．TSIMMIS[5]では，分散管理されている情報をOEMと呼ばれる共通のモデルに変換することで，スキーマ進化の問題を扱っている．

情報ベースに対する検索機構はデータベースパラダイムとは大きく異なる．情報ベース上の検索では，一般に部分的にスキーマは知っているが全体のスキーマは知らず，情報へのアクセスはブラウズイングと呼ばれる[1]．ブラウズングはインスタンスによる視覚等の情報から対話的に検索が行われ，得られた結果を元に，検索意図を確定する検索スタイルである．ここでは従来のデータベース検索と逆の意志決定がなされている．これは情報の収集，整理，分析のサイクルから得られた情報を活用し，また収集分析された情報に基づいて新たなサイクルを繰り返しが基本的流れである．過去に情報フィルタリングと呼ばれる検索法やブラウズイングを支援する様々な方式が提案されているが，インスタンスの検索をするツールにすぎない．このため，検索を効率よく実行するのみならず，検索構築をも対象とするシステムの構築が必要である．本質的には，スキーマ知識自体も検索対象であり，その結果を検索機構に反映させる技術が求められている．

この論文では，情報ベースの検索(ブラウズイング)を支援するメカニズムの一つの方式を述べる．検索は質問系列 $Q_1, Q_2, ..$ であり，最終的な答え以外はすべての答えを計算する必要は必ずしもなく，質問 Q_i の得られた部分解から，Q_i の処理を停止してつぎの質問 Q_{i+1} を計算する必要がある．ここでは，すでに得られている答えと処理中のものを Q_{i+1} の処理に活用する情報ベース検索の支援方式を提案する．質問に基づいて計算された結果と計算プロセスの情報をメタ情報として，質問に反映させる情報ベースのメタレベル及びオブジェクトレベルの検索支援ができる．

2 情報ベースの検索

2.1 ブラウズイング

ブラウズイングはデータベースのインスタンスから新たな質問を見いだすこと，つまり，質問 Q_i からインスタンスを視覚的に検索して，インスタンスを含むスキーマを確定して，質問の意図を捕らえて，新たな質問 Q_{i+1} を作り出すことである．この過程で利用者の意思決定が対話的に確定される．このため，利用者の意図をくみ取る知的な操作が望まれている．質問 Q_i の答え A_i の系列は一般に非可逆的であり，以前になされた質問から自動的には演繹されるものではない．すなわち，モデルと推論のメカニズムで機械的に質問の意図を演繹する方式は適さない．

情報ベースの検索に於いて，データベースパラダイムとのミスマッチがデータベース技術を情報ベースに利用できない最大の問題である．ブラウズイングによって得られた結果の検索意図または質問への反映には，2つの場合が考えられる．

1. 利用者にとって，質問の意図が明確であるが，スキーマが不明な不完全質問
2. 利用者にとって，スキーマが既知であるが，質問の意図が不明である不明確な質問

不完全な質問は利用者が質問を十分し絞り切れないため，スキーマを見つけて質問の意図と合致する指定項目(スキーマ)を対話的に埋めて，質問の意図を確定していく．質問によって得られる答えは増加，減少する．この際，検索パスは必ずしも一意に定まらないため，非決定的な選択が必要となる．検索の戦略をメタレベルの言語 $\mathcal{LOC}$[3]で記述する方式も提案されている．

2.2 不明確な質問

不明確な質問(意図が不明)の場合，この場合，インスタンスをブラウズイングしながら検索条件を探す．すなわち，質問の意図をインスタンスのブラウズイングしながら質問に置換える．この場合，利用者の質問の意図自身が曖昧であるから，不完全な質問の場合のように，インスタンスに注目して質問の答えを絞り込む方法は用いることができず，質問から利用者の意図を絞り込み必要がある．

不明確な質問の処理は質問からインスタンスを得て，新たな質問を行う．利用者はスキーマを知っているが，どのように質問を作るべきかを確定していく．このため，不完全な質問の場合と異なり，質問の解のインスタンスが必ずしも，単調に増加する，減少するものではない．検索の意図を知的にするためには，質問処理プロセスを知的にくみ取る，すなわち，質問の答えと計算中の答えをつぎの質問の処理にどのように反映させるかが重要な問題となる．

この論文では，不明確な質問の場合のブラウズイング支援方式のみを述べる．

2.3 ブラウズイングの要件

ブラウズイングに於いて，質問の度に答えをすべて求めて提示するのではなく，求められた答え(部分的な答え)を順に提示する．複数の可能性があれば並列に評価して答えを利用者にフィードバックし，最終的な答え以外は，必ずしも全部の答えが必要ない

質問の系列 $Q_1, Q_2, ..$ とする．現在の質問 Q_i の答えと計算中の答えを，つぎの質問 Q_{i+1} の処理への反映は以下の場合がある．

1. **Query Pushing**：質問 Q_i で得られている答え，または，処理中の答えを質問 Q_{i+1} の処理に活用する
2. **Query Popping**：質問 Q_i の計算中を停止して，Q_{i+1} の計算に備える．
3. 上記以外：この場合は以前の答えや処理中の答えを利用できない

3 Query Pushing

3.1 Query Pushing の要件

以下では，第一階言語またはその部分クラスを質問言語とする．質問の系列 $Q_1, Q_2, ..$ に対して，Q_i が Q_{i+1} の Query Pushing であるための要件は，

1. 質問 Q_i の処理で現在までに得られている答え A_i を質問 Q_{i+1} の処理で利用でき，かつ
2. 処理中の答えを Q_i の処理プロセスを引き継いで Q_{i+1} の処理で利用できる．

ここでは，Query Pushing であるための十分条件として，以下の 2 つの場合について述べる．

1. **[P1]** 質問 Q_i が Q_{i+1} の部分式に含まれる
2. **[P2]** Q_{i+1} をある形式に変換した一部に Q_i を含む

例 3.1は **P1** の例であり，質問 Q_i の答えの一部 A'_i と処理中のプロセス Q_i^P, Q_i が $Q_{i+1} = f(Q_i)$ の一部に現れるならば，以下で計算できる．

$$A_{i+1} = f(A'_i) \cup Q_i^P \| f(\cdot)$$

ここで，$x \| y$ は x の答えを y に引き継ぐ演算子．従って，**P1** は Query Pushing である要件を満足する．質問言語は制約されないが，部分式による Query Pushing の定義は応用上制限が強く，例 3.4は **P1** の意味での Query Pushing でない．

例 3.1 つぎの質問 Q_i と Q_{i+1} を考える．Q_i は Q_{i+1} の部分式であるから，**P1** を満足する．

$$Q_i(yz) = \exists x(p(xyz) \wedge x = a)$$
$$Q_{i+1}(z) = \exists y \exists x(p(xyz) \wedge x = a \wedge y = b)$$

Q_i の処理で得られている一部の答え A'_i と処理中の答え $Q_i^P(yz)$ とすると，Q_{i+1} の答え A_{i+1} は以下で計算できる．

$$A_{i+1} = \{(z) \mid (yz) \in A'_i, y = b\} \cup \{(z) \mid Q_i^P(yz) \| y = b\}$$

例 3.2 つぎの質問 Q_i と Q_{i+1} を考える．例 *3.1*と同様に，Q_i は Q_{i+1} の部分式であるから，**P1** を満足する．

$$Q_i(xyz) = p(xyz) \wedge x = a$$
$$Q_{i+1}(xyz) = (p(xyz) \wedge x = a) \rightarrow q(xyz)$$

Q_{i+1} の答えは以下で計算できる．

$$A_{i+1} = \{(xyz) \mid (xyz) \in A'_i, q(xyz)\} \cup \{(xyz) \mid Q_i^P(xyz) \| q(xyz)\}$$

例 3.3 つぎの質問 Q_i と Q_{i+1} を考える．

$$Q_i(xy) = p(xy)$$
$$Q_{i+1}(xyz) = p(xy) \wedge q(yz)$$

つぎの質問 Q_i と Q_{i+1} を考える．

$$A_{i+1} = \{(xyz) \mid (xy) \in A'_i, q(yz)\} \cup \{(xyz) \mid Q_i^P(yz) \| q(yz)\}$$

例 3.4 つぎの質問 Q_i と Q_{i+1} を考える．Q_i は Q_{i+1} の部分式でない．選言標準形に変換すると，**P2** を満足する．

$$Q_i(y) = \exists x(p(xy) \wedge x = a)$$
$$Q_{i+1}(y) = \exists x(p(xy) \wedge (x = a \vee x = b))$$
$$= \exists x((p(xy) \wedge x = a) \vee (p(xy) \wedge x = b))$$

Q_{i+1} の答えは以下で計算できる．

$$A_{i+1} = \{(y) \mid (y) \in A'_i\} \cup \{(y) \mid p(xy) \wedge x = b\} \cup \{(y) \mid Q_i^P(yz) \| q(yz)\}$$

P2 が Query Pushing の要件を満足するか否かは標準形に依存する．以下では，質問言語を制限した Query Pushing である部分クラスを導入する．**P2** の場合は，質問言語を構文で制限するが，Q_i と Q_{i+1} の依存関係が弱まり，適用範囲が広がり，例えば例 3.4に適用できる．

3.2 部分クラス L_0

L_0 を以下の条件を満たす第一階言語の部分クラスとする．

1. 情報ベース M は正の確定的な情報のみ保持
2. 述語記号は基礎関係，$=$, $\neq$
3. すべての変数は基礎関係に出現
4. Safe：負リテラルのすべての変数は$\wedge$ で結ばれた正リテラルにも出現
5. 限定作用素なし
6. 結合子 $\vee$, $\wedge$, $\neg$
7. 否定は各原始論理式の前に高々一回出現

言語 L_0 で書かれた質問 Q_i, Q_{i+1} を選言標準形に変換して，Q_{i+1} の形式に Q_i が含まれる，すなわち，Q_i, Q_{i+1} の選言標準形を以下とする．

$$Q_i(w_1)..(w_m) = (W_1 \vee .. \vee W_n)$$
$$Q_{i+1}(y_1)..(y_{m'}) = (V_1 \vee .. \vee V_{n'})$$

ここで，W_i, V_j はリテラルの $\wedge$ による結合．以下の条件を満足するならば，Q_i が Q_{i+1} の **P2**-Query Pushing であると呼ぶ．

各 $W_j = L_1 \wedge .. \wedge L_l$ に対して，$V_k = L_1 \wedge .. \wedge L_{l'}$ が存在して，$l \leq l'$．L_i はリテラル．

例 3.4は **P2**-Query Pushing の条件を満足する．

Q_i が L_0 で表された Q_{i+1} の **P2**-Query Pushing としよう．Q_i と Q_{i+1} の選言標準形を以下とする．

$$Q_i = (w_1)..(w_m)(W_1 \vee .. \vee W_n)$$
$$Q_{i+1} = (y_1)..(y_{m'})(V_1 \vee .. \vee V_{n'})$$

Q_i が Q_{i+1} の **P2**-Query Pushing であるから，A_{i+1} は以下で計算できるから，Query Pushing である．

$$A_{i+1} = \{(y_1..y_{m'}) \mid (w_1..w_m) \in A_{w_i}, (y_1..y_n) \| V'_{l'-l}\} \cup \{(y_1..y_{m'}) \mid V_{n'-n}\}$$

P2-Query Pushing であるか否かは，W_i, V_j を正負の各リテラルについて，辞書式に並び替える $(O(n \log n))$．従って，W_i が V_j の部分式である否かは，$O(n)$ で可能．W_i をどの V_j に対応させるかの組み合わせは n^2 あるから，**P2**-Query Pushing であるか否かは，$O(n^2)$ かかる．ここで，n は論理式の長さ．以上から，つぎの定理が得られる．

定理 3.1 言語 L_0 に於いて，Q_i が Q_{i+1} の ***P2-Query Pushing*** ならば，Q_i が Q_{i+1} の *Query Pushing* である．また，Q_i が Q_{i+1} の ***P2-Query Pushing*** であるか否かは $O(n^2)$ で判定できる．n は論理式の長さ．

例 3.5 つぎの質問 Q_i, Q_{i+1} を考える．

$$Q_i = \neg p(xy) \wedge q(xy)$$
$$Q_{i+1} = (\neg p(xy) \vee r(xy)) \wedge q(xy)$$

$Q_{i+1} = (\neg p(xy) \wedge q(xy)) \vee (\neg p(xy) \wedge q(xy))$ であるから，***P2-Query Pushing*** である．

3.3 部分クラス L_1, L_2

言語の L_0 に存在記号を許す言語を L_1 とする．L_0 に於いて，否定は原始論理式の前に高々一回のみ出現するから，Q_i を冠頭標準形に変換しても，$\forall$ に変換されることはない．$\exists x(F \vee G) = (\exists xF) \vee (\exists xG)$ から，**P2**-Query Pushing ならば，Query Pushing である．冠頭標準形に於いて，変数の対する 存在記号の変数の対応を検査すれば良い．変数の個数 m に対して，$O(m^2)$ で検査できるから，$O(n^2)$ で検査可能．従って，**P2**-Query Pushing であるか否かは $O(n^4)$ で可能．

定理 3.2 言語 L_1 に於いて，Q_i が Q_{i+1} の部分式であるか否かは $O(x)$ で決定可能であり，Q_{i+1} が Q_i の *Query Pushing* である．また，Q_i が Q_{i+1} の ***P2-Query Pushing*** であるか否かは $O(n^4)$ で判定できる．n は論理式の長さ．

言語の L_0 に全称記号を許すが，$\vee$ を許さない言語を L_2 とする．以下の性質から，同様に成立する．

$$\forall x(F \wedge G) = \forall xF \wedge \forall xG$$

定理 3.3 言語 L_2 に於いて，Q_i が Q_{i+1} の部分式であるか否かは $O(x)$ で決定可能であり，Q_{i+1} が Q_i の *Query Pushing* である．また，Q_i が Q_{i+1} の ***P2-Query Pushing*** であるか否かは $O(n^4)$ で判定できる．n は論理式の長さ．

$\forall x(F \vee G) \neq (\forall xF) \vee (\forall xG)$ が成立しないから，$Q_{i+1} = \forall x(F \vee G)$ の形式の質問に対しては，Q_i がいかなる形式の論理式の場合でも，Query Pushing とはならない．

Q_i が Q_{i+1} の Query Pushing, かつ Q_{i+1} が Q_{i+2} の Query Pushing ならば，Q_i は Q_{i+1} の Query Pushing となる．

例 3.6 つぎの質問 Q_i, Q_{i+1} を考える．

$$Q_i = \exists x(\neg p(xy) \wedge q(xy))$$
$$Q_{i+1} = \exists x((\neg p(xy) \vee r(xy)) \wedge q(xy))$$

$Q_{i+1} = \exists x(\neg p(xy) \wedge q(xy)) \vee \exists x(\neg p(xy) \wedge q(xy))$ であるから，***P2-Query Pushing*** である．

3.4 L_1 の拡張

Q_i が Q_{i+1} の部分式であるという条件は緩めることができる．質問言語 L_1 を考える．これはつぎの 2 つの場合がある．

1. Q_i の質問変数が Q_{i+1} の存在記号で限定
2. Q_i の変数が存在記号で限定され，Q_{i+1} で質問変数

最初の場合は質問変数に答えが返されるから，$\|$ でその答えを Q_{i+1} で渡すことができるから，Query Pushing であることはあきらか．第 2 の場合は，質問の答えに基礎関係の質問を加えることで，基礎関係へのアクセスを減らすことができる (例 3.7)．

例 3.7 つぎの質問 Q_i, Q_{i+1} を考える．

$$Q_i(xz) = \exists y(p(xy) \wedge q(yz) \wedge x \neq z)$$
$$Q_{i+1}(xw) = \exists y \exists z(p(xy) \wedge q(yz) \wedge r(zw) \wedge x \neq z \wedge y \neq w \wedge x \neq w)$$

この場合，A_{i+1} は以下で計算できる．

$$A_{i+1} = \{(xw) \mid (xz) \in A'_i, r(xy), Q_i^P(xz) \| r(zw) \wedge y \neq w \wedge x \neq w\}$$

3.5 Query Pushingの最適化

P2-Query Pushingでは，冠頭宣言標準形に変換するため，論理式を最小の条件の分解して，$\vee$ で結合するため，同じ質問が何回も評価される．Query Pushingの処理に於いて，以下の最適化を行うことができる．

1. 基礎関係へのアクセスの減少
2. 答えをキーに基礎関係をアクセスする

以下の質問があるとしよう．

$$Q_i(y) = p(xy) \wedge x = a$$
$$Q_{i+1}(y) = p(xy) \wedge (x = a \vee x = b)$$
$$= (p(xy) \wedge x = a) \vee (p(xy) \wedge x = b)$$

Q_{i+1} では $p(xy)$ が $\vee$ をまたいで，同じ原始論理式が現れるから，同じ計算を行わないで，$p(xy)$ の処理プロセスの結果を受けとることを $p(xy)^P$ で表す．

$$Q_{i+1}(y) = (p(xy) \wedge x = a) \vee (p(xy) \wedge x = b)$$
$$(p(xy) \wedge x = a) \vee (p^P(xy) \| x = b)$$

このように変換することで，基礎関係へのアクセスの回数を減らすことができる．

3.6 探索の制限

限定作用素が存在する場合は，変換することによって限定作用素の探索の範囲を制限することができる．以下の場合がある．

1. $\forall x(F \wedge G) = \forall x(F \wedge (F \rightarrow G))$
2. $\forall x(F \wedge G) = (\forall xF) \wedge (\forall xF)$
3. $\exists(F \wedge G) \Rightarrow (\exists F)$
4. $\exists(F \vee G) = (\exists F) \vee (\exists G)$
5. $\forall(\neg F) = \neg\exists F$
6. $\exists(\neg F) = \neg\forall F$
7. $\forall(F \vee G)$：この場合は最適化が困難である．

一方，$F \wedge G = F \wedge (F \rightarrow G)$ であるから，$\forall x(F \wedge G) = \forall x(F \wedge (F \rightarrow G))$ に変換できるから，G の部分は F の検査しもののみを計算することができる．同様に，$\exists x(F \wedge G) = \exists x(F \wedge (F \rightarrow G))$

4 Query Popping

Query Popping は Q_i が Q_{i+1} の Query Pushing でないため，Q_i の計算を中止して，Q_{i+1} に備える．Q_i をさかのぼって，Q_{i+1} の Query Pushing となる Q_k $(k \leq i)$ を見つける問題である．

Query Popping の問題は質問 Q_i を記憶する必要であるが，加えて，Q_i の答えを全て記憶くしているか否かによって，方針が全く異なる．

過去の全ての答えを記憶していれば，前にさかのぼって Query Pushing となる Q_i や Q_{i+1} を Q_k $(k \leq i)$ で計算する方法を探せば良い．しかし，答えを記憶していない場合は，一般には再計算を必要とする．

ブラウズイングしながら，つぎの質問列を入力したとしよう．

1. $Q_1 = p(xyz) \wedge x = a$
2. $Q_2 = p(xyz) \wedge x = a \wedge y = b$
3. $Q_3 = p(xyz) \wedge x = a \wedge y = b \wedge z = c$
4. $Q_4 = p(xyz) \wedge x = a \wedge y = d$

Q_1 の答えが記憶されていれば，Q_4 の計算のために，Q_1 まで戻ればよい．さもなければ，Q_4 の計算のために，Q_1 に対応する計算は再計算をせざるを得ない．計算中の答えを保持している部分については，その結果を Q_{i+1} の処理に引き継ぎことも考えられるが，すでに計算された部分については計算の状態まで管理する必要があり，データベースの目的とミスマッチが生じる．

5 メタ制御とブラウズイング

ブラウズイングから質問の意図を捕らえて，スキーマを確定する．検索は一般に並列に行われ，どのスキーマを確定したものとするかは，非決定的な選択が存在する．これをメタのレベルからプログラムで支援する．また，ネットワークの負荷などの非決定的な要因をメタのレベルに反映させる Reflective メカニズムが必要となる [2].

6 おわりに

情報ベース上の一つのブラウズング法を提案した．検索の意図を確定するための知的な支援を行う方式を提案した．この問題は質問 Q_i が質問 Q_{i+1} の部分式に出現するか否かを構文的に検査する必要がある．この論文では，変数名の書き換えや変数の定数による束縛を考慮にいれていないため，**P2**-Query Pushing であるか否かは多公式時間で検査できるが，一般には部分グラフの同型問題を含み，多公式時間となるアルゴリズムは知られていない．この論文では，同一の利用者が質問をするから，質問式は同じ部分式については同じ変数を用い，具体化は等号で置換えると仮定している．

情報ハイウエイ特有のスキーマ進化の問題はインスタンスを見てスキーマを書き換える必要がある．この問題については [4] を参照．

参考文献

[1] F.G. Halasz : Reflections on Notecards: Seven Issues for the Next Generation of Hypermedia Systems, Comm. of ACM, pp. 836-852 (1988).

[2] P. Maes and D. Nardi : Meta-level Architectures and Reflection, North-Holland (1988).

[3] T. Miura and I. Shioya : Determining Database Query Paths by a Logic Language, OOER'95 (1995).

[4] T. Miura and I. Shioya : Mining Type Schemes in Databases, DEXA'96 (1996).

[5] Y. Papakonstantinou, H. Garcia-Molina and J. Ullman : MedMaker : A Mediation System Based on Declarative Specifications, Techincal Report.