5J−8

４ＧＬプログラム統合における名前衝突に関する一考察

白井靖人 静岡大学教育学部
上田 穣 日本ウェーブフロント㈱
國井利泰 東京大学理学部

1 はじめに

第四世代言語(４ＧＬ)プログラムは、そのプログラムの扱うファイルの定義を記述した部分と、そのファイルに対する操作内容を記述した部分の二つから構成される。既存の４ＧＬプログラムを統合する場合、各プログラムの扱うファイルをまとめて一つのデータベースで置き換える必要がある。この統合データベースの設計に当たっては、各プログラムのファイル定義部を統合してデータベーススキーマを導出する。本稿では、ファイル定義部統合の際に生じる「名前衝突」の問題を取り上げ、この問題の解決を一部自動化する方法を検討する。

2 ４ＧＬプログラムの統合

2.1 ボトムアップ手法の必要性

1980年代、第四世代言語(４ＧＬ)は自動プログラミングツールとしてＣＡＳＥ(コンピュータ支援ソフトウェア設計)の一環をなすものとして認識されることが多かった[1]。しかしそれとは別に、４ＧＬの持つ「ユーザ自身が手軽に効率的なアプリケーションを作成することができる」という特徴から、特に中・小の企業が従来の業務をシステム化する際の、アプリケーション開発ツールとして４ＧＬを導入することも多く行なわれた。

最近では、データ要求の多様化や処理要求の高度化から、こうして開発された個々のアプリケーションプログラムに代わり、一つの統合システムに移行しようという動きがある。統合化の手法は、新規にシステムを構築するトップダウン手法と、既存のものを一つにまとめるボトムアップ手法の二つに分けられるが、４ＧＬプログラムの統合に当たっては、その導入の背景から、ボトムアップ手法の適用が強く望まれる。データベース設計やソフトウェア設計においては従来トップダウン手法の方が優れているとされ、ほとんどの研究がこの方法に関するものであった。そのため、ボトムアップ手法についてはほとんど検討されてきていないのが現状である[2]。

2.2 統合時の問題点

４ＧＬプログラムの統合では、各プログラムに固有のファイルを一つにまとめたデータベースの設計が中心となる。このデータベースのスキーマは、プログラム中のファイル定義部から第３正規形の形で得ることが可能である[2]。

ファイル定義部を統合してスキーマ設計を行なうという作業は、データベース設計におけるスキーマ統合の作業と類似している。スキーマ統合においては、次ぎに挙げるような項目に注目してサブスキーマ同士を比較検討する[3]。

・名前付け
・属性同士の対応関係
・(型、キー、関連に関連した)構造上の対応関係

多くのファイルでは、キーフィールド(群)の値に非キーフィールドが依存するようになっている。これは、キーフィールドから非キーフィールドへの関数従属性(ＦＤ)が存在することを示しており、各ファイル定義の構造は極めて単純なものとなる。

従って、ファイル定義の統合においては、スキーマ統合の場合と比べて、構造面での対応関係よりも名前付けについての検討が重要となる。そこで、次節以下ではこの点に絞って検討することにする。

3 名前衝突：ホモニムとシノニム

各ファイルはそれを必要とするアプリケーションのみを意識して設計されている。フィールドの名前付けについては、(同一ファイル内においては、各フィールドは異なる名前を持つようになってはいるが)他のファイルにおいてどの様な名前が使われているかといった特別な注意は払われていない。その結果、フィールド名についてファイル間での整合性が失われ、統合作業時にはこの整合性の欠如が名前衝突として表面化する。

名前衝突には、次の二種類がある。

①シノニム(synonym: 同物異名)
同一の対象物に対して複数の異なる名称が与えられた場合、それらの名称をシノニムと呼ぶ。同一のデータ項目が複数のファイルに異なるフィールド名で蓄えられた場合、これらのフィールド名はシノニムとなる。

②ホモニム(homonym: 同名異物)
異なる対象物に対して同一の名称が与えられた場合、それはホモニムと呼ばれる。異なるデータ項目でありながら、実際に蓄えられているファイル中での対応するフィールド名が同じ場合、このフィールド名

On Name Conflicts in the Integration of Fourth-Generation Language Programs
Yasuto Shirai (Faculty of Education, Shizuoka University), Minoru Ueda (Wavefront Japan, Ltd.) and Tosiyasu L. Kunii (Faculty of Science, The University of Tokyo)

はホモニムである。

4　名前衝突の解消

シノニムが出現するのは、本来同じ名称となっているべきフィールドに異なる名称が使われている場合である。統合化においては、これらのフィールドの名称を同じものに付け代えて統一する必要がある。一方、ホモニムの場合は、異なるデータ項目を参照しているフィールドにはそれ独自の別の名称を与えてやることになる。

4.1 型情報の利用

シノニムとホモニムの検出には、各フィールドの型についての情報が有効である。

①シノニムの検出

シノニムフィールドの場合、表現対象となっているデータ項目は同一のものであり、出現値のドメインも共通のものとなっている。従って、シノニムフィールド同士は同一の型、または表示の上では同値の型を持っている必要がある。そこで、そのような型を持つドメイン同士は、シノニムフィールドの候補と考えることができる。

②ホモニムの検出

ホモニムフィールドの場合、シノニムの場合とは反対に異なるデータ項目を表現していることから、ドメイン自体も異なることが予想される。従って、名称が共通でありながら異なる型を持つフィールドは、ホモニムフィールドの候補と考えられる。

型情報によって得られるのは、シノニムまたはホモニムの候補に過ぎない。従って、最終的な決断は設計者に委ねられることとなる。

4.2 ホモニムの除去

前節ではシノニムとホモニムの両方を考慮の対象としたが、ホモニムを除去し、シノニムのみを考慮するようにすることも可能である。

統合対象となるファイルには各々名前が付けられているが、このファイル名には重なりがないものと仮定することができる。仮に、ファイル名にもホモニムが存在したとしても、そのファイルを定義しているプログラムの名前とファイル名とを組にして考えれば、ファイル名のユニーク性を主張するこの仮定の有効性を保証することができる。

そこで、各フィールドの名称を、元の名称の先頭(または末尾)にそのフィールドを含むファイルの名称を追加したものに付け代えることにする。こうすると、ファイル名のユニーク性と各ファイル内でのフィールド名のユニーク性から、全フィールド名のユニーク性が保証されるようになる。全てのフィールドは異なる名前を持つことになるので、ホモニムは存在しないことになる。

このような名前の付け代え作業は人間が行なうとすれば混乱を招き易いが、簡単なツールを用意することで容易に自動化することができる。ホモニムが除去された結果シノニム候補の洗い出しのみを行なえばよいことになるので、これら二つのステップを連続して行い、シノニム検出の最終決定のみを設計者に問うようにすることができる。設計者の決断を仰ぐ際には、元のファイル定義中のフィールド名を用いるようにすれば、設計者側はホモニム除去ツールの存在を意識することもない。

4.3 ＦＤ情報の利用

4.1節で述べたシノニム検出の方法は、フィールドの型情報のみに基づくものであった。型情報のみで判断すると、例えば書籍の「定価」と「発行部数」などのように、全く別のものでありながらもシノニムの候補と判定される場合が多く発生する。

ＦＤ情報を用いると、このような無意味な候補を除外することが可能である。先に述べた書籍の例を考えると、「定価」と「発行部数」に関して次の二つのＦＤを考えることができる。

書籍名 → 定価

書籍名 → 発行部数

このように、同一のものに関数的に従属するもの同士は、たとえ同じ型であってもシノニムの候補とはなり得ない。ＦＤは各ファイルの定義部分から抽出されるものであるから、ファイル定義の性質と考え合わせると、同一のファイルに属するフィールド同士に関しては、シノニム候補の検出を行なう必要はないということになる。

5　まとめ

ボトムアップ手法はその実用的な価値は認識されていながらも、これまであまり検討されてこなかった。しかし、特に４ＧＬプログラムの統合を考える際、ボトムアップ手法の確立が強く求められる。

本稿では、ファイルを構成するフィールドの名称について統合化の際発生する名前衝突の問題を取り上げ、その解決を一部自動化するための方法を検討した。統合化作業においては、最終的な判断は設計者に委ねられるとはいえ、それを支援する意味で一部の機械的な作業を自動化することが強く望まれる。

本稿の検討内容は予備段階のものであり、さらに深く検討する必要がある。特に、ＦＤ情報については、より高度な利用法も可能であると考えられるので、今後の検討課題といえよう。

参考文献

[1] 原田監修：『CASEのすべて』、1991、オーム社

[2] 白井他：「４ＧＬプログラム統合のためのデータベース設計手法」、情報処理学会第43回全国大会講演論文集(5) 2K-7、pp. 413-414

[3] Ozkarahan, Database Management, Prentice-Hall, 1991