

PCTEの性能に基づく効果的な利用法について

2M-7

佐藤 友康 森保 健治 外山 勝保 伊集院 正

NTTソフトウェア研究所

1. はじめに

ソフトウェア開発環境において、設計情報のデータベースであるリポジトリの存在が重要になっている。その中でPCTE(Portable Common Tool Environment)[1]はリポジトリの国際標準規格の最有力候補である。

多くの開発者が携わる大規模ソフトウェア開発では、リポジトリへのアクセスの多重度が大きいいため、その応答時間などの性能は、実用上重要な要件の1つである。

また、開発作業に複数のCASEツールが導入されるときは、ツール間で設計情報を共有する必要があるため、設計情報を細粒度で管理することが必要条件である。

そこで本稿では細粒度の設計情報管理をPCTEで行う場合の性能評価結果をRDBと比較して示す。さらにこれに基づいてPCTEを利用したリポジトリの構成法について述べる。

2. PCTEの性能

2.1 性能測定

図1のようなER図を複数用意し、その細粒度での管理を想定し、オブジェクトの型定義として実体型、関連型、及び属性型とし、それぞれの型において各構成要素毎にオブジェクトを生成する。そしてオブジェクトの生成、検索を行う時間を測定した。

性能評価結果は、PCTEがRDBと比較して、全評価項目において25～130倍低速であった。またそれは、オブジェクト数が増加するほど性能が悪化する傾向が見られた。

2.2 分析

上記の評価結果が得られたのは、以下の原因によるものと考えられる。評価に用いたPCTEのインプ

リメントは、1個のオブジェクトに対し1個のファイルを生成する。一方、RDBは1個のオブジェクト型に対し1個のファイル（テーブル）を生成し、オブジェクトはレコードとして管理する。そのため、PCTEではオブジェクト数の増加につれてファイル数が増加するが、RDBはオブジェクト数にはよらずオブジェクト型が一定であればファイル数は一定である。図1の例では実体型のオブジェクト数が3個、関連型が2個、属性型が7個となる。つまりPCTEでは全12個のファイルをオープンすることになるが、RDBでは3個のみでよい。このように、PCTEはRDBに比べ、オープンするファイル数が多くなる。

また、PCTEはRDBに比べ、データモデルが複雑である。例えばRDBではオブジェクト間のリンクは属性を持たないが、PCTEではリンクにおいてカテゴリなどの属性を管理する。これは、設計情報の管理目的によっては冗長である場合もある。

このように、PCTEの性能が全般的に低い理由として、取り扱うファイル数が多いことと冗長な管理情報があるためと考えられる。

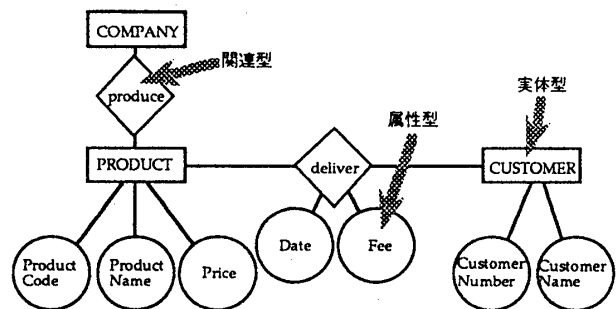


図1. ER図の例

3. 利用法

PCTEは上記のような性能上の問題があるが、RDBと比較して設計情報間の関連の明確化が可能であり、構成管理に必要な機能を備えているという利点がある。そこでPCTEはリポジトリとしての機能を十分備えていると言える。そこでPCTEを利用した有効なリポジトリの構成法を考える。PCTEは設

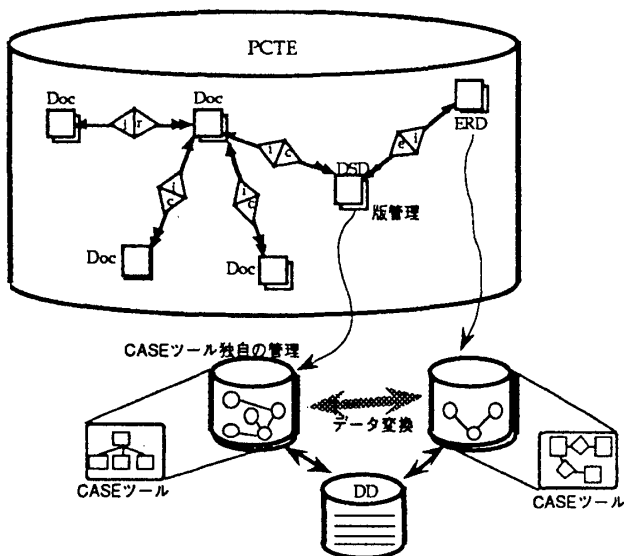


図2. 設計情報管理方式

計情報（オブジェクト）定義に加え、オブジェクト間のリンク属性や版管理情報など、多くの管理情報が存在する。これらはドキュメントの構成管理を行うためには有効であるが、CASEツールの扱う単位であるドキュメントの構成要素のような細粒度の管理には冗長である。このためすべての設計情報をPCTEで管理せずに、管理の要件に応じたデータモデルを組み合わせて利用する方法が考えられる。

具体的な管理方式を図2に示す。PCTEを利用してドキュメントをオブジェクトとして定義し、オブジェクト間の関連を定義する。例えば、実体関連図(ERD)を元にしてデータ項目決定のためにデータ構造図(DSD)を作成する場合は、図2の例のようにERDとDSDの間に存在依存性を表すexistenceリンクを張ることでその関連を表現する。

細粒度設計情報の管理には、CASEツール独自の管理ファイルを採用する。その理由はCASEツール独自の管理ファイルでは、実装において必要情報のみを管理するようにデータモデルを適合させており、ツールが管理するために実用的な性能を達成しているからである。これにより、既存ツールのデータモデルの変更が不必要になり、ツール統合面においても改造量が減少し、ツールの特長を最大限に利用可能となる。

また、ツール間での設計情報の共有のため、ツール間での設計情報の引き渡しをファイルで行う。その際、データ項目を統一的に管理するために、データ辞書(DD)を設ける。DDはデータ項目の一意性を

管理すれば良く、表などの単純な管理方式で良い。

提案する方式で構成されたりポジトリが必要機能を実現可能か考察する。必要機能には、並行作業に対応するための排他制御、ベースラインの設定を行うための版管理、変更追跡性の確保がある。

・排他制御 PCTEで排他処理を行えば、ドキュメント単位で行うことになる。これは関連するドキュメントの作成作業が停止することになる。CASEツールを利用して作成するドキュメントに対しては、設計情報の共有が損なわれることになる。そこで細粒度での排他処理はDDで行い、データ項目単位で排他処理を行う。

・版管理 ベースラインの設定などに必要な版管理は、細粒度で行う必要はなく、ドキュメント単位で行えば良い。そこでPCTEの版管理機能を利用すれば実現可能である。

・変更追跡性の確保 変更追跡性確保には、設計情報の変更前と変更後の関係を明確にしなければならない。そのために、ドキュメントの版管理を利用し、変更前後の差分管理によって明確化する。

また仕様変更による設計情報の変更による波及も考慮しなければならない。ドキュメントでは関連がPCTEのリンクで明確化されているため波及箇所が明らかである。また、細粒度においては、ツール間で変更差分のデータ交換により対応可能である。

4. まとめ

本稿ではPCTEの特性を利用したりポジトリの構成法について述べた。この手法はPCTEではリンク情報を管理し、細粒度の設計情報は、各CASEツールで管理することである。このような管理手法には以下のようなメリットがある。

- ・ツールの特長を最大限に利用可能
- ・PCTEの機能を最大限に利用可能
- ・要件に応じた粒度で管理が可能
- ・オブジェクト数の減少による性能向上

尚、この実現において、PCTEとツールの設計情報のリンクの張り方、そしてツール間の設計情報の引き渡し方法については今後の検討課題である。

参考文献

- [1]L. Wakeman, J. Jowett: "PCTE - The standard open repositories", Prentice Hall, 1993