6H−6　ISO IRDSの仕様安定化のための提言

岩崎 一正、 穂鷹良介
筑波大学

1.はじめに

本報告では、DIS (Draft International Standard)となった情報資源辞書システム(IRDS)・サービスインタフェース[1]について、バージョン管理の仕様に見られる不都合の指摘とその改善案の提示を行なう。

以前にも同様の主旨の報告[2][3]を行なっているが、以前には考慮の外にあった、Working Set 間の参照経路(Reference Path),およびschema groupと呼ばれるデータ辞書作成メカニズムなどが導入されたことに伴う混乱が見受けられるため、これを改善する必要があった。

2.DIS案におけるバージョン管理と参照経路の概略

DIS案IRDSは、情報資源管理、CASEツールなどに用いるデータ辞書などの設計支援が主要な用途の一つであり、設計データの様々なバージョンを管理・提供する機能を持っている。

設計データの最小単位はオブジェクトと呼ばれている。各オブジェクトは、オブジェクトタイプと呼ばれる種別を持つ(例、tableオブジェクトタイプのオブジェクトt1,t2、schemaオブジェクトタイプのオブジェクトs1,s2)。

いくつかのオブジェクトを属するオブジェクトタイプを問わずひとまとめのものとして扱うことが出来るようにWorking Setという概念が用意されている。

既存のWorking Set を基にして、積み木を積むように、新しいWorking Set を上に重ねることが出来る。

図2.1の様な積み重なった二つのWorking Setがある場合、WS1の内容は、obj1,obj2,obj3であり、WS2の内容は、obj1,obj2(変更済み),obj4 である。

つまり、WS1の上に積み重ねられたWS2の内容は、WS1の内容にWS2で行なった編集作業による変更を加味したもの、となる。

このように、設計データのバージョン管理は、個々のオブジェクト単位では行なわれず、いくつかのオブジェクトを適当な作業単位にまとめたWorking Set を単位として行なわれる。

便宜上、WS1で追加したobj2とWS2で変更したobj2を区別するために、それぞれをobj2のWS1バージョン、obj2のWS2バージョンと呼ぶ。また、このようなオブジェクトIDとWorking Set ID の対で識別されるものをオブジェクトバージョンと呼ぶ。

参照経路は、二つのWorking Set間に参照経路を定義することによって、異なるWorking Setに属するオブジェクトバージョン間の参照関係を定義出来るようにするものである(図2.2)。

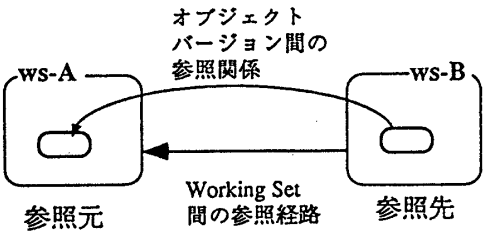

図2.2 ws-Aからws-Bへの参照経路

図2.2のws-Aを考えたときに、ws-Bに含まれるオブジェクトバージョンを参照しているオブジェクトバージョンを含んでいるので、ws-Aを使って作業する場合にwsBの内容も同時に参照できるほうが便利なことがある。

DIS案では、作業対象をws-AでのFullContextと指定することで、ws-Aからの参照経路の先にあるws-Bも含めた範囲を検索対象と出来る。ただし更新対象は、ws-Aの内容のみに限定される。

このようにFull Contextは、いくつかのWorking Setを参照関係を元にひとまとめに扱いたいときなどに威力を発揮する。

3.DIS案におけるバージョン管理の問題点

バージョン管理に関しては以下に述べるように二つの相反する要求がある。

(a)Working Set間の関係が積み重ねだけであれば、とあるWorking Setに於けるオブジェクトバージョン同士の参照を定義するにはオブジェクトIDだけで十分である(自動的に一番上のバージョンが採用される)。

(b)参照経路を用いて異なるWorking Set間にまたがる参照の場合には、参照先のオブジェクトバージョンを特定する必要がある。

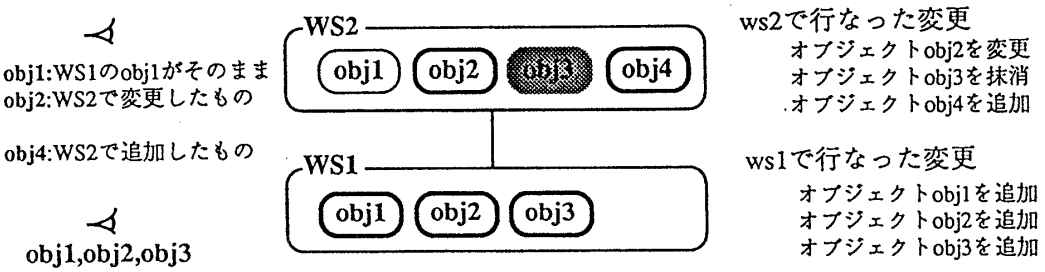

図2.1 Working Set を用いたバージョン管理

A Proposal for improving the specification of ISO IRDS
Kazumasa IWASAKI, Ryosuke HOTAKA
University of Tsukuba

(c)現在では後者の要求に引きずられて「参照関係のあるオブジェクトバージョン同士は、互いに同じWorking Setに属さねばならない。」とする不自然な制約が設けられている。これは、バージョンアップのサイクルの異なるもの(例、列と表、表とスキーマ)を同一のWorking Setに入れることから生ずる混乱である。

例えば、「スキーマオブジェクトs1には、列としてカラムオブジェクトc1,c2を持つテーブルオブジェクトt1が含まれる。」という設計データを考える(バージョン1)。また、他の部分は変更せずに単にテーブルt1の列c1の桁数だけをバージョン2で変更しようとしたとする(図3.1)。

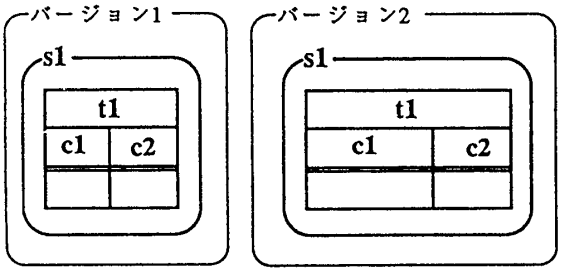

図3.1 設計データのバージョン例

これを、DIS版IRDS上で行なうとすると、設計データバージョン1を入れたws1上にws2を用意し、新しいc1のバージョンのほかに、s1,t1のws2バージョンを登録する必要が生じてしまう(図3.2)。図では省略されているが、同じことはカラムオブジェクトが参照する他のオブジェクト(例、定義域)などにも生じてしまう。

4.constructおよびcomponent概念の導入

IRDSで扱っている対象には、構成物(construct)とそれを構成する要素(component)の二種類がある。
3.で指摘した問題(c)は、constructのバージョンがcomponentのバージョンと一致すべきという制約であり、これは明らかにあるcomponentがあるconstructの複数のバージョンに組み入れられるとき不都合を生じる。

この不都合を解決するために、以下のように考える。
(1)Full Contextは、constructを作るための部品の集まりを決定する。
(2)そのcontextの中で、作られるべきconstructが指定される。
(3)construct自身はcomponentと異なったバージョンを持ち得る。
(4)materializationは、特定のconstructのバージョンを作るという前提に基づいて行なわれる。換言するならば、materializationは、contextとconstructバージョンに依存してなされる。
(5)construct /component概念は相対的なものであり、constructであったものが、他のconstructのcomponentとなることもある。

constructおよびcomponent概念の導入により図3.2で生じた不都合は図4.1のように解決される。ここで、contextをws2としてテーブルt1のws1バージョンをconstructとしてmaterializationを行なえば、そのcomponentは、列c1のws2バージョンと列c2のws1バージョンとなる。

5.未解決の課題

以下のような課題が今だ未解決のまま残されている。
(1)現行のFull Contextの作成方法では参照経路を一段しか辿らないが、再帰的に辿るように拡張されるべきである。
(2)新しく導入したconstruct概念に関連した整合性維持の問題。

参考文献

[1] ISO/IEC JTC1/SC21 WG3 DIS10728:IRDS Services Interface,Draft International Standard,July 1991
[2] 岩崎他：SQLによるIRDSの実現、第40回全国大会講演論文集(2)
[3] 岩崎他：ISO IRDS1の実装と評価、第42回全国大会講演論文集(4)

本研究は、一部筑波大学1991年度学内プロジェクトの支援により成された。

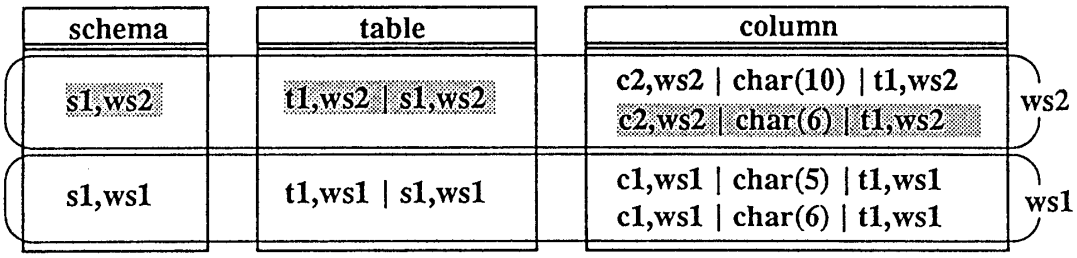

図3.2 不自然な制約から来る不都合の例（網をかけた部分を余計に登録しなければならない）

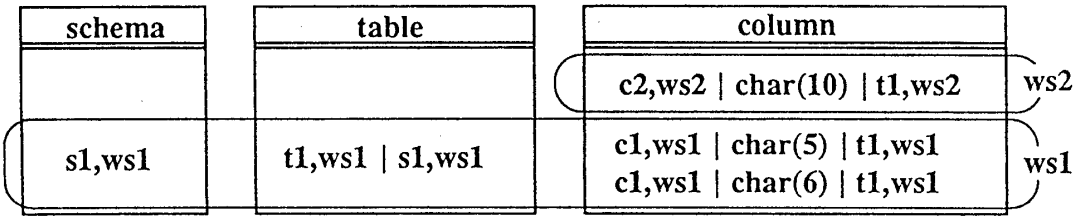

図4.1 constructおよびcomponent概念導入による改良