データベース・システム 87－2
（１９９２．３．１６）

# ISO IRDS の実装と機能の吟味

岩崎　一正、穂鷹　良介

筑波大学

本稿では、ISO/IEC 10728: Information Resource Dictionary System Services Interface の仕様を検討した結果みつかったいくつかの不必要な制約を指摘している。
・full context の定義の不備
・対応範囲の限定されたvalidation 制御
等。

本稿では、これらの制約を取り除き、SQL 依存の概念を置き換えるためにconstruct / component 概念を導入する。この変更提案にもとづいたシステムをSUN-３ワークステーション上にSybase DBMS を利用して現在構築中である。

# Examination of ISO IRDS implementation and specification

Kazumasa IWASAKI, Ryousuke HOTAKA

University of TSUKUBA

This paper investigates the specification of ISO/IEC 10728: Information Resource Dictionary System Services Interface and points out several unnecessary restrictions,
(1) restricted scope of full context
(2) insufficient control of validation status for various kinds of constructs

This paper removed these restrictions and introduced construct / component concepts to replace unnecessarily SQL dependent concepts. The proposed system is being implemented on SUN-3 workstation using Sybase DBMS.

## 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、その時点では検討対象から外していたWorking Set 間の reference path 概念が、仕様の記述におかしな点があるものの、重要な概念であると認識されたことや、schema group と呼ばれる新しい種類のデータグループ化概念が導入されたことに伴う混乱が見受けられるなど、新たに仕様を吟味仕直す必要があった。

## 2.DIS案におけるバージョン管理概略

ISO 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。DISとなったサービスインタフェースの仕様は、情報資源辞書のメタデータベースのデータ構造とそれを操作するための各種サービス（プログラムインタフェースを含む）を対象としている。

個々の設計データは何らかの object type に属する object として扱われ、各 object は ird object key という属性の値によって識別される。例えば、IRD定義レベルペア（情報資源辞書のメタデータベース）では、IRDの設計データをSQLデータモデルを用いて記述するため table object type に属する **t1, t2** (table object) とか schema object type に属する **s1, s2** (schema object) などが現われる。設計データのバージョンに対応するために、また幾つかの object を一纏めに扱うことを可能とするために Working Set という概念が用意されている。Working Set を用いて object のバージョンを考えるため object version は ird object key と Working Set の識別子（working set keyの値）の対によって識別される。

設計データ間の相互関係、例えば表と列や表の列と定義域の間の関係を表現する場合、参照を行う object version に参照先の object version の識別子を値として持たせる。二つの object version が異なる Working Set に属する場合には更に Working Set の間にも reference path と呼ばれる参照関係を定義する（図１）。

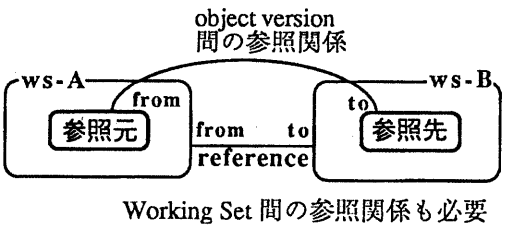



図１　DIS案における object version 間の参照の定義

Working Set はバージョンを考える場合、既存の Working Set を下敷きとしてその上に積み上げる様にして作ることができる。新たに作られる Working Set **ws2** で作業する時には、下敷きにした Working Set **ws1** の内容を **ws2** の内容で上書きしたもの（the materialization of **ws2**）が対象となる。このような積み重ねは何層でも可能で一つの Working Set に複数の Working Set を積み重ねることもできる（図２）。

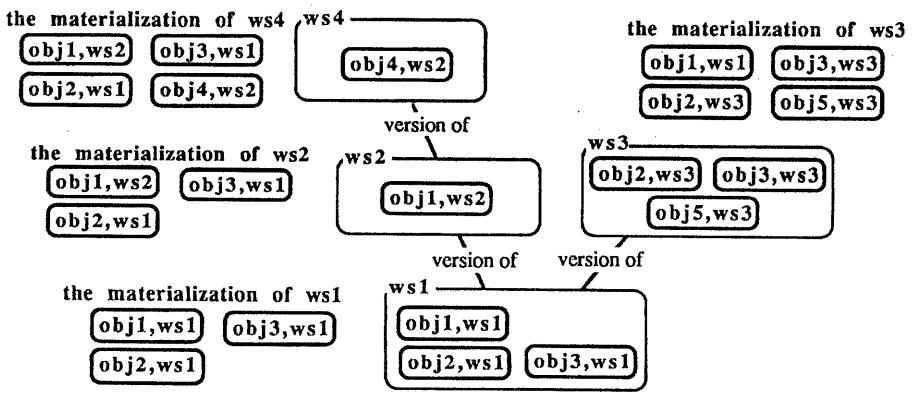
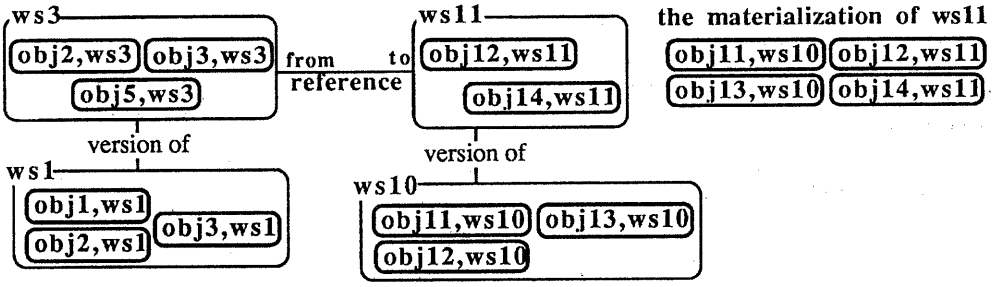



図２　version path と materialization of Working Set

積み上がった Working Set までの Working Set の階層を version path と呼ぶ。また、下敷きとした Working Set が reference path を通して参照している Working Set があれば、新規作成される Working Set でも同様の参照を行えるよう reference path の定義が行われる。

Working Set は content status と呼ばれる状態を持ち、各 content status は uncontrolled, controlled, archived の三つのクラスのいずれかに属する。uncontrolled クラスに属する content status にある Working Set の内容は更新することが可能であり、controlled, archived クラスに属する content status にある Working Set の内容は基本的に更新できない。content status の変更はサービスコールによってユーザが任意に行うことができる。

IRDS を利用して設計データを扱う場合には、context を設定する必要がある。context の設定は、Working Set を指定することによってなされ、single context と full context のいずれかを指定する。single context の場合は、指定した Working Set の materialization のみが作業対象となる。full context の場合には、single context の範囲に reference path で参照関係を定義した Working Set の内容を加えたものが作業対象となる（図３）。




ws3 に full context を設定した場合の作業対象は、

obj1,ws1 obj2,ws3 obj3,ws3 obj5,ws3 obj12,ws11 obj14,ws11

図３　DIS案 full context

DIS 案では、このようなバージョン管理機能を含めて、提供されるサービスや概念が IRD 定義レベルペア、IRD レベルペアの二種類のメタデータベースシステムにできるだけ共通のものとなるようにすること、レベルペアに無差別に適用できるようにすること（レベルパラレリスムの達成）、を基本方針としている。この規格案で規定するデータ構造は、両レベルペアに共通して現われるデータ構造と IRD 定義レベルペアにおいて IRD を設計・管理するためのデータ用のデータ構造（SQL の Information Schema に類似のもの）と IRDS を運用するためのデータ構造（ユーザ管理や IRDS 特有の機能をサポートするためのもの）であり、これらの記述は SQL2 規格（案）を用いてなされている。

## 3.DIS案における混乱と問題点

（問題点１）full context の定義不備

reference path で参照するつもりの Working Set で context を設定したときの作業対象と full context を設定して参照した場合の作業対象が異なってしまうことがある。例えば図３では、**ws11** で context を設定したときには、**(obj11, ws10), (obj12, ws11), (obj13, ws10), (obj14, ws11)** が見えるのだが、**ws3** で full context を設定した場合には reference path を通しては **(obj12, ws11), (obj14, ws11)** しか見えないことになってしまう。

また、object version 間の参照が、異なる Working Set の間をまたがる場合には参照の行われる向きに合わせた Working Set 間の reference path を定義する必要があるという制約があるが、reference path の向きと full context を設定するための参照の方向が一致しないときに困る。

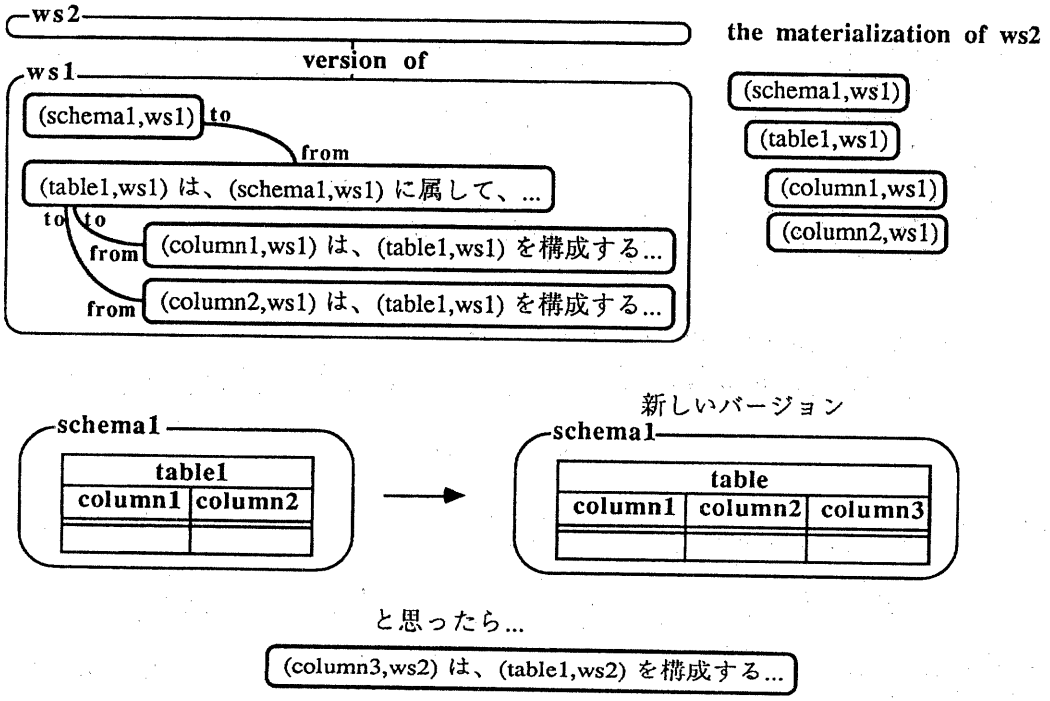
（問題点２）construct / component 関係の表現のSQL依存性

IRD 定義レベルペアの IRD を設計・管理するためのデータ構造に関する仕様を見ると、table object と column object, あるいは schema object と table object の様な「後者は前者を構成する」とい

う関係にある object type に於て、構成物側の方の定義には、construct / component 関係を表現・維持する目的とおぼしきSQL の制約が設けられている。

(1) construct object の更新に関する制約
一例として、table object (version) type の定義には、所属する schema object version を削除された場合には所属している table object version も削除される様にする制約が盛り込まれている。

(2) construct / component 関係にある object version 同士のバージョンの一致
一例として、table object (version) type の定義には、所属する schema object version と同じ Working Set に属する（同じバージョンとなる）様にする制約が盛り込まれている。
このような制約により、schema とそれに属する table 、table とそれを構成する column は互いに同じ Working Set に属することを強いられる。従って、図4のような状況で、**ws2** を context として新たな column を **table1** に付け加えるためには、まず **schema1**、次に **table1** の **ws2** バージョンを作成しておく必要が生ずる。




を ws2 で追加する必要があるが、(table1,ws2)が在らねばならない。
同様に、(schema1,ws2) も在らねばならない。

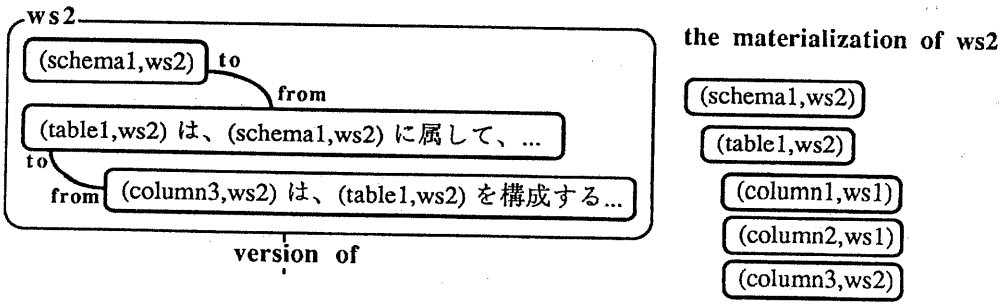



図4　DIS案の construct / component 関係

(2)の様な制約は、schema よりも細かな単位で Working Set を使うことを妨げるなどユーザの自由度を奪うものである。また、これらの制約は、IRD定義レベルペアに現われるobject type については、その定義と共にSQLの制約として陰に表現されているが、IRDレベルペアにおける construct / component 関係を定義する場合には、同じことを陽にデータ構造（table constraint, referential constraint など）に格納することとなる。

（問題点３）schema group の validation 制御

schema group object は、いくつかの schema object から構成される複合オブジェクトであり、それら schema object の直接間接の構成内容に基づいてDBスキーマを作成する目的に使用される。このため schema group object type は、その直接間接の構成内容の整合性を反映するための属性 activation status を持つ。

DIS 案では、activation status 属性の値としてACTIVE, VALIDATED, UNVALIDATED を許している。add object, modify object, delete object サービスによって、整合性の検査を行った後 VALIDATED 状態になった schema group object の構成を変更した場合には、この影響によって validation 状態が不定となったものとして、activation status 属性の値が UNVALIDATED 状態となるよう自動的に変更される、ことになっている。しかし（図５）に示したような、schema group とそれを構成する部分がそれぞれ異なる Working Set に属するような状況では context を与えないかぎり **ws10** の中の（**schema1,ws10**）がどの schema group によって使用されているかを簡単には確定できない。また schema group の様な構成物の整合性をモニタするような属性を持つ複合オブジェクトは、IRD レベルペア（応用データベースなどを設計するためのDB）でも色々なものが考えられるので一般化して扱うべきと思われる。

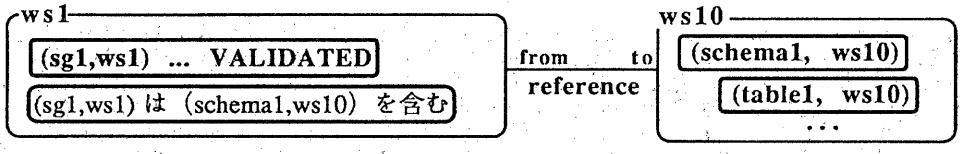



図５　schema group の validation 制御

## 4.DIS案に対する変更提案

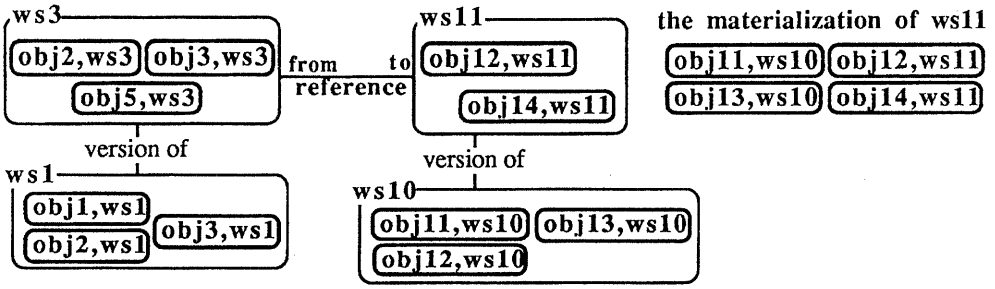
前項で指摘した問題点を避け、
「必要以上にきつい制約は設けず、自由度を確保する」
「context を設定することが作業対象を決定することである」
「Working Set は作業対象を区切るための単位となる」
「複合オブジェクトの構成等については、作業対象が確定してから考慮する」
を方針として以下に述べるような修正案を呈示する。

### 4.1.context 概念に関する変更提案

#### (1) full context の定義の変更

「指定された Working Set の materialization に、もしあれば、指定された Working Set が参照している Working Set の各々で full context を設定した場合に作業対象となるものを加えたものを full context の作業対象とする」。

。この新しい定義によれば、参照先に指定してある Working Set で full context を設定した場合の作業対象と reference path を通して取り込まれる作業対象が同じになる。図６は、図３と同じ状況下で新しい定義による full context を設定した例である。DIS案では見えなかった(**obj11,ws10**), (**obj13,ws10**) も見えるようになるため、**ws3** から **ws11** を参照する reference path を通して見えるものは **ws11** で (この例ではsingle) context を設定したとき見えるものと同じになっている。




ws3 に full context を設定した場合の作業対象は、

obj1,ws1 obj2,ws3 obj3,ws3 obj5,ws3 obj12,ws11 obj14,ws11 obj11,ws10 obj13,ws10

図6　変更提案版 full context の例
（網の掛かった部分も見えるようになる図3参照）

(2) reference path の定義の変更

schema group と schema あるいは schema と table な度をそれぞれ別の Working Set に入れてバージョン管理する場合を考えると「Working Set 間の参照の向きと object version 間の参照の向きを一致させる」というのは邪魔な制約となる。そこで、このような状況を解決できるように「reference path は任意の向きで Working Set 間に定義できる」ものとする。更に、reference path を通して同じオブジェクトの複数バージョンが作業対象に選ばれることを避けるために、「互いにバージョン関係にある二つ以上の Working Set を同時に参照先に選ぶことを禁止する」という制約を設ける。

(3) dynamic reference の導入

ある object version がある object の最新バージョンを参照する、ような参照関係を定義できると便利なことがある。

そこで、「参照を行う object version で、参照先として object key のみ指定することを許し、その場合 context 設定後に実際に参照先の object version が特定される。また、参照先として object version を指定してあっても context 設定後の作業対象に入ってなければ参照関係は成立しない（無視される）」ものとする。

図7の例では、**ws10** で full context を設定した場合には **table1** の属する schema は（**schema1,ws10**）となり、**ws11** で full context を設定した場合には **table1** の属する schema は（**schema1,ws11**）となる。

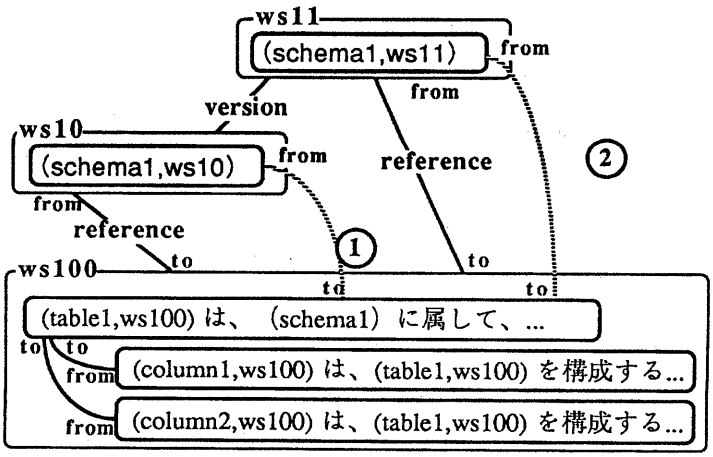



図7　dynamic reference を採用した場合の参照の解決

(4) 整合性制約の context への依存

整合性制約は、複合オブジェクトや複数のオブジェクト間の整合性を問う場合にチェックされるべき条件であり、チェックの対象と条件は指定された context に依存すると考える。
例えば図7では、「表 (table1,ws100) がどのような列を持ち、その構成は整合的か」というような整合性制約を評価する場合には、**ws100** で context を設定すれば充分であるが、同じものの利用（所属）が見る context によって異なるため、表 (table1,ws100) を含む schema レベルでの整合性を検討する場合には、**ws10** または **ws11** で full context を設定しなければならない。

4.2.construct / component 概念の導入による仕様の明確化

(1) construct と component 概念の導入

DIS案は、SQLを用いて表現することが強く意識されているために、例えば schema と table, table と column の間の関係を foreign key による参照と delete 時 cascade（schema を消したら属する table も消える）制約の存在として捉えているが、これは一般的に考えると構成物（construct）とその構成要素（component）と言う見方で捉えたほうがはるかに自然でかつ一般性がある。

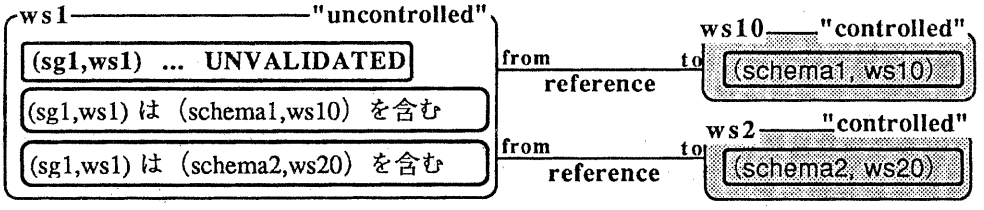
(2) construct の validation 制御

DIS案は、schema group という特別な construct についてのみ validation の制御が定められているが、validation は construct ならばどのようなものでも（例えば段階的なチェックをするときなど）行われる可能性がある（SQLの例では schema 毎のvalidation も考えるかもしれない）。
したがって、このような徹底しないチェックをサービスインタフェースとして提供するよりは、construct の validation 制御はサービスインタフェースの範疇では行なわないこととする。サービスインタフェース仕様に関するかぎり、一つ以上の construct を validate した後にそれらの component に対する変更を禁止するのは、制度は少し落ちるのだが、関連する object の属する Working Set の凍結機能を利用する形のみで行なわれることと想定する。

(3) Working Set 単位の凍結

(2)の想定の基に construct の validation の状態を維持しやすいように reference path に関する以下の制約を追加する。
「content status が controlled クラスであるような Working Set ws に対して、
・ws から新しく Working Set を参照するような reference path の追加を禁止する。
・ws が参照元になっている reference path の変更を禁止する。」

例えば、この reference path に関する制約の追加と controlled クラスの Working Set は一般的に更新できないことを利用して、construct と component を別々の Working Set（のバージョン系列）に入れてこれらの間を reference path で結ぶようにすれば、construct の構成状況に関する属性の値が検査済みになったときの状態を維持できる（図8）。




schema group 用 validation ルーティンによって (sg1,ws1) の activation status を VALIDATED に変更出来たら（検査済みの状態になったら）、ws1 を "controlled" クラスの content status に変更するとことにより(sg1,ws1) の validation 状態を凍結できる。

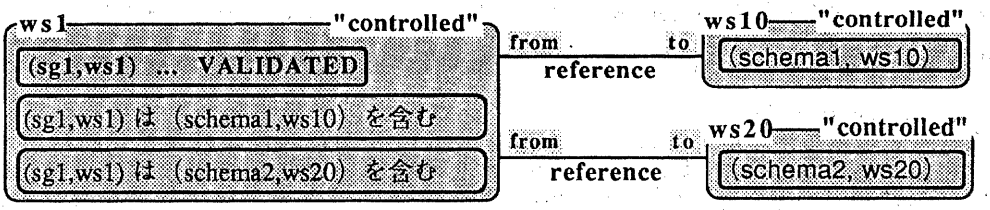



図８　construct の構成の凍結

図のような状況では、参照先の Working Set の内容を変更するには、まず、参照元である **ws1** の content status を uncontrolled クラスのものに変更してからでなければ、参照先の **ws10, ws20** の content status を uncontrolled クラスのものに変更することが出来ないため、誤って construct の構成内容を変更することは出来ないことになる。

## 5.まとめ

前項に示した提案に基づいて、IRDS サービスインタフェースを提供するプログラムを Sun-3 ワークステーション上に、Sybase DBMS のデータベース管理機能を利用する形で構築中である。

DIS 案は依然として各国の投票結果待ちで1992年5月の編集会議を経てその仕様が確定される見込である。DIS案が我々の本稿の提案にも拘わらずそのまま規格となったとしても、現在作成中のプログラムは若干の手を加えることでISO規格を満足できるものとなり得、かつ将来の機能拡張にも耐えることができると思われる。

## 謝辞

本研究を行うにあたって、活発な議論を頂いた方々、特にDIS案の実装面からの検討に関して貴重なご意見を頂いた管理工学研究所の方々に御礼申し上げます。

## 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、情報処理学会第４０回全国大会講演論文集（２）
[3] 岩崎他：ISO IRDS1の実装と評価、情報処理学会第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。