データベース・システム　77－6
（１９９０．５．１７）

ISO IRDSの日本提案の実装

岩崎　一正
筑波大学

ISO/IEC IRDS 規格案SEL01に対する改善提案である日本提案の実現について報告する。

日本提案は、IRDS独自の機能を必要最小限に絞り、SQL-DBMSの機能を前提とした改善を提案するものである。規格案が机上の空論に終わらぬように、実現可能であることを重要視している。

本報告では、まず、この提案の概要を説明し、1990年4月初旬現在までに実現されている部分について紹介する。最後に実現されているIRDSの応用プログラムの一例として、会話的にデータ操作を行なうCLIPを紹介する。

Implementation of Japan Proposal for ISO IRDS

Kazumasa IWASAKI
University of TSUKUBA

We present here the implementation of Japan's change proposal for ISO/IEC/JTC1/WG3/IRDS SEL01, working draft of IRDS Services Interface. In this proposal, Added Value Functions of IRDS are redeuced to necessary minmum, and implementaion in SQL-DBMS is assumed.

First, we explain major parts of Japan proposal. Second, we present IRDS1, which are implemented by April,1990. Lastly, we show CLIP,Command Language Inteface Processor, as an example of application programs of IRDS1.

1．はじめに

現在、ISO/IEC JTC1 にて国際規格としてのIRDSをどうするか、についての検討が行なわれている。

IRDS とは、Information Resource Dictionary System の略称であり、情報資源辞書システムと訳される。これは、企業における情報資源を管理運用するための辞書（情報資源辞書）と、これをさらに管理するための辞書からなるシステムである。情報資源辞書を管理する辞書を用意することによって、システム内に幾つもの異なる情報資源辞書を持つことが可能となっている。

IRDS規格案では、これらの辞書と応用データベースとの関係を、レベル対（ＬＰ）という概念で整理している。ＬＰには、上下関係が存在し、幾つかのデータベースが属する。そして、上位のＬＰに属するデータベースは、一つ下のＬＰに属するデータベース（群）を、意味的に、管理するデータを持つと考える。

このようなＬＰを三つ考え、下位の方から順に、応用レベル対(AP_LP),情報資源辞書レベル対(IRD_LP),情報辞書定義レベル対(IRDD_LP)と名付ける。各々のＬＰには、応用データベース（群）、情報資源辞書（群）、唯一つの情報資源辞書の定義辞書が属する、と考える（図１）。SEL01では、IRDD_LPに属する辞書を定義し、IRD_LPに属する辞書（群）の満たすべき形式を定義し、この二つのLPに属するデータベースを操作するインタフェースを定義している。また、IRDD_LPの辞書を用いて情報資源辞書の設計開発を行なえる機能も定義している。

本報告は、IRDS規格案の改善提案である、日本提案の因となった、IRDS規格案の実現、特にそのサービスインタフェースの実現、に関するものである。

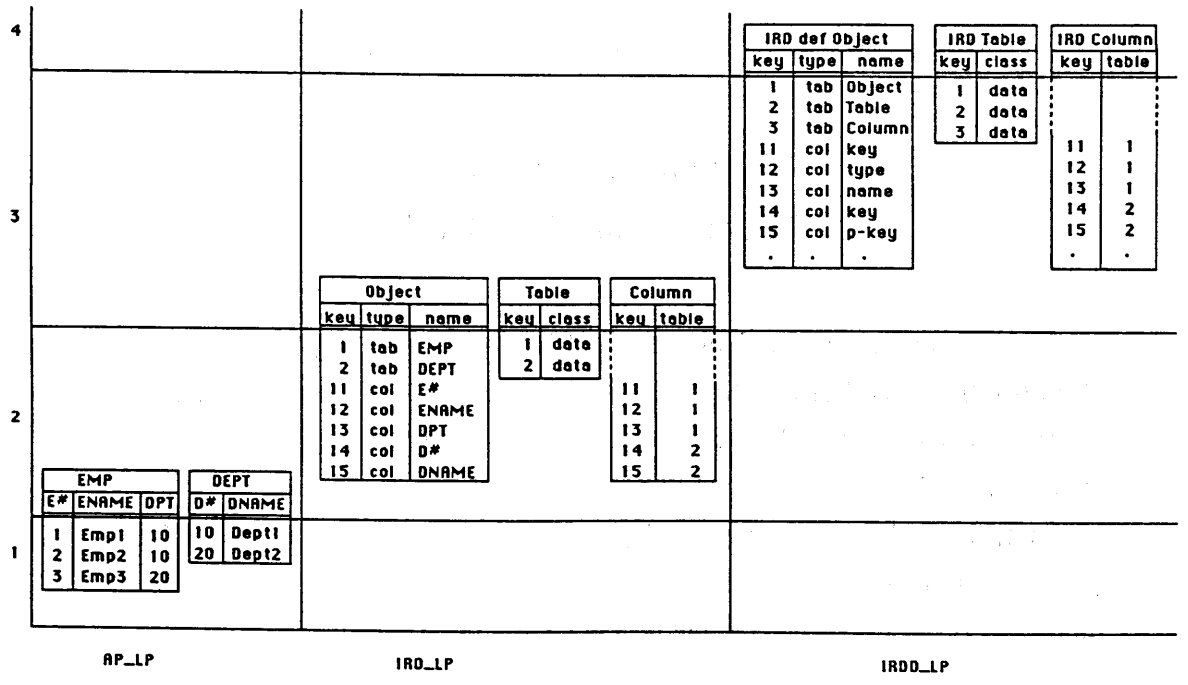

図１．三つのレベル対

2．ISO/IECのIRDS規格案

ここで、IRDS規格案というのはSEL01を指し、これは、主にデータ構造と、サービスインタフェースを規定している。SEL01では、以前の規格案と比べて、データ構造が大きく変わっている。以前は、E/Rデータモデル機能を使って記述されたANSI IRDSのデータ構造をObjectとその間のAssociationに着目した独自のデータモデル機能で記述し直したものであったが、SQLの能力を利用することを前提として規格案の見直しを行なった結果である。その結果、IRDD_LPに属するデータベースの構造の内、情報資源辞書の定義情報を格納する部分については、前提とされたSQL2規格案のメタデータベースと酷似したデータ構造を持つことになった。また、情報資源辞書を設計を支援する機能としてObjectレベルでのバージョン管理、ライフサイクル管理、等を用意している。ここで、Objectというのは、情報資源辞書・応用データベースの定義情報の単位となるものを差し、例えばSQL流に定義する場合、Table Object ,Column Object等が考えられる。IRDSにおけるデータ構造は、基本的に全ての表が、図1に示されるようにObject表の副表となっている。

一方、サービスインタフェースの方は大幅な変更はなく、検討が未だ及んでいないかのようである。

規格案で、規定されているサービスは以下の様に分類される。

1）運用サービス

2）データ操作サービス

3）情報資源辞書サービス

1）は、IRDSの使用開始、終了や、トランザクションのコミット・ロールバック等を行なうサービスである。

2）は、表に対するObjectの挿入、検索、更新、削除やObjectのライフサイクルの変更等を行なうサービスである。

3）は、IRDD_LPのデータベースのみに適用可能なサービスで、情報資源辞書の定義情報に基づいて新たな辞書を追加したり削除したりするサービスである。

このうち2）のサービスは同じ機能を持つものがIRDD_LP,IRD_LPごとに別々のサービスとしてそのインタフェースが規定されている。また、一般の属性とテキスト文字列をいれる属性とで、別々のサービスを用意している。

挿入以外のデータ操作を行なう場合、カーソルという概念を用いるが、これはこの規格案独自のものであり、Object-Associationデータモデル機能に基づいている。カーソルは、Objectの集合に対して定義され、その集合中のObjectを順に一つづつ指し示す目的で用いられる。例えば、Open MetaObject Cursorサービスを用いることによって、IDの値や、名前等の指定した条件を満たすObjectの集合が決定され、カーソルが割り当てられ、そのカーソルIDが返される。結果として得られたカーソルIDを用いて、検索・更新・削除・複製等のデータ操作を行なうことができる。カーソルを用いて検索を行なう度に、カーソルは割り当てられた集合の次のObjectを差していく。カーソルの使用終了は、Close MetaObject Cursorサービスによる。

3．日本提案の概要

日本提案は、SEL01を基にしており、SQLの能力を利用するという方針に従い、実現可能であることを前面に押し出した改善提案である。

IRDD_LPに属するデータ構造の内、情報資源辞書を定義情報を格納する部分は、IRD_LPに作られる、情報資源辞書を定義するSQL-DDL文を生成できるだけの情報を持つような構造に改められ、ずいぶん簡略化された。

IRDSの持つ、情報資源辞書・応用データベースの定義情報の運用管理のための独自の機能（Added Value Functionと呼ばれる）については、

必要最小限ということで

(1) Working Setレベルのバージョン管理

(2) Working Setレベルのライフサイクル管理

(3) スキーマからの複数データベースのインストール

の三つに絞り、これを用意することにしている。AVFについては、原稿執筆時点では、必要なデータ構造やサービス等の細部までは確定していないので概要説明のみにとどめる。

まず、Working Setと言うのは、設計作業の対象となるObjectの集まりを差す。従って、設計のより所としたデータモデル機能によってその構成要素の種類、要素間の整合性チェックの手法等が異なる。

(1)は、Working Set WS1がWS0のバージョンであるとき、WS1はWS0の構成要素に対する変更部分の情報（差分情報）だけを保持し、ユーザがWS1を見た場合WS0の構成要素にWS1の差分情報で変更を施したものが見える、また、WS0の構成要素の変更は、そのバージョンであるWS1にも影響を及ぼす、というような仕組みが検討されている（図２）。

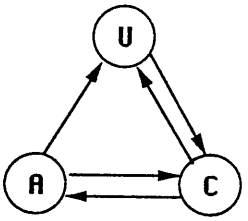

図３．Woking Set の
ライフサイクルフェーズ

(2)は、Uncontroled,Controled,Archivedの三つのライフサイクルフェーズを考え、各フェーズ間の遷移を図３のように規定し、C状態にあるWorking Setは、そのベースとなったデータモデル機能の整合性を満たす、というような仕組みである。

(3)は、スキーマを設計するためのWorking SetがC状態にある場合、そのスキーマを基に複数のデータベースコンテナを実装できる、というような仕組みである。

Working Setの概念は、LPとは独立の概念であるためIRDD_LPだけではなく、IRD_LPのデータベースを使って応用データベースを設計する際にも適用可能である。

サービスインタフェースについてもSQLを活かすように改善され、属性別・LP別に同様のサービスを用意せず、一本にまとめることになった。詳細については、次章で触れる。

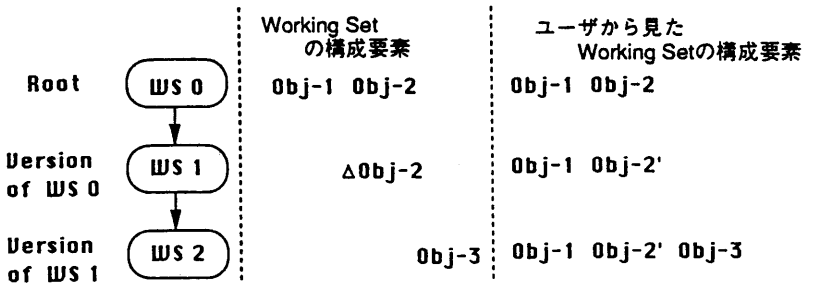

図２．Working Set のバージョン

4．日本提案の実現

市販のSQL-DBMSを利用して、前章で説明したIRDSの日本提案を実現した、以下ではこれをIRDS1と呼ぶ（図４）。IRDS1に属する全ての表をSQL-DBMSの一つのデータベースに含まれることとし、基本的にIRDS１のサービスインタフェースを利用する応用プログラムを通してのみ、このデータベースに対する操作が可能であるようにしている。個々の応用プログラムは、動的SQLインタフェースを利用してＣ言語で記述されたサービスインタフェース・サポートルーティンのオブジェクトとリンクされて作られる。

SQLの能力を利用するというのは、

(1) SQl-DBMSの機能を利用して実現する。

(2)データ操作言語、およびカーソル概念を用いてサービスインタフェースを改善する。

ということである。

(1)は、表作成機能、制約サポート機能、ユーザのアクセス権のチェックなど、SQL-DBMSの持つ能力については、IRDSを実現するに当たって新たに作らず、SQL-DBMSの機能を利用する、ということである。

(2) は、基本的なデータ操作サービスについては、SQL流に改めるということである。

IRDS1のサービスは、

(1) 運用サービス

(2) 基本データ操作サービス

(3) AVFサービス

に分類される。

(1)は、基本的にSEL01のものと変わらない。変更点は、実現の形態がサービスサポートルーティンを応用プログラムごとにリンクする形を採ったため、Open_IRDSサービスのパラメタが簡略化されたこと、SQLの制約のチェックをデータ操作直後に行なうかトランザクション終了時に行なうかの条件設定を行なうサービスが追加された点である。

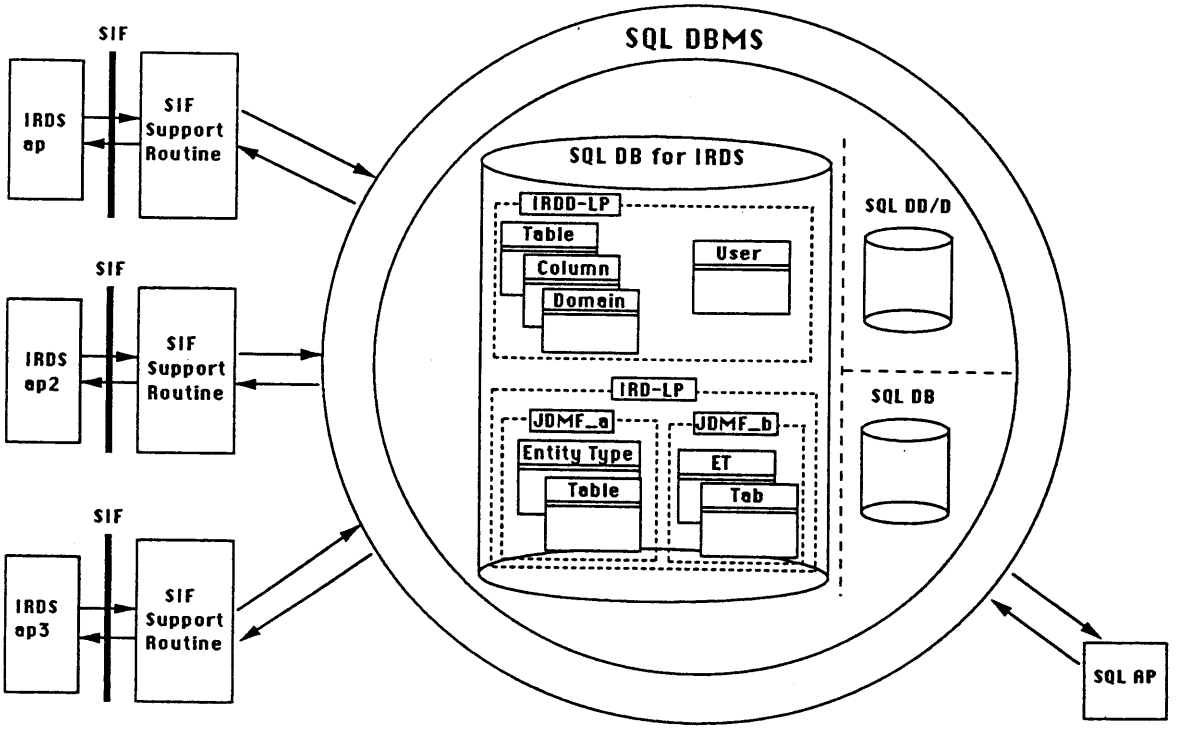

図４．IRDS1 のアーキテクチャ

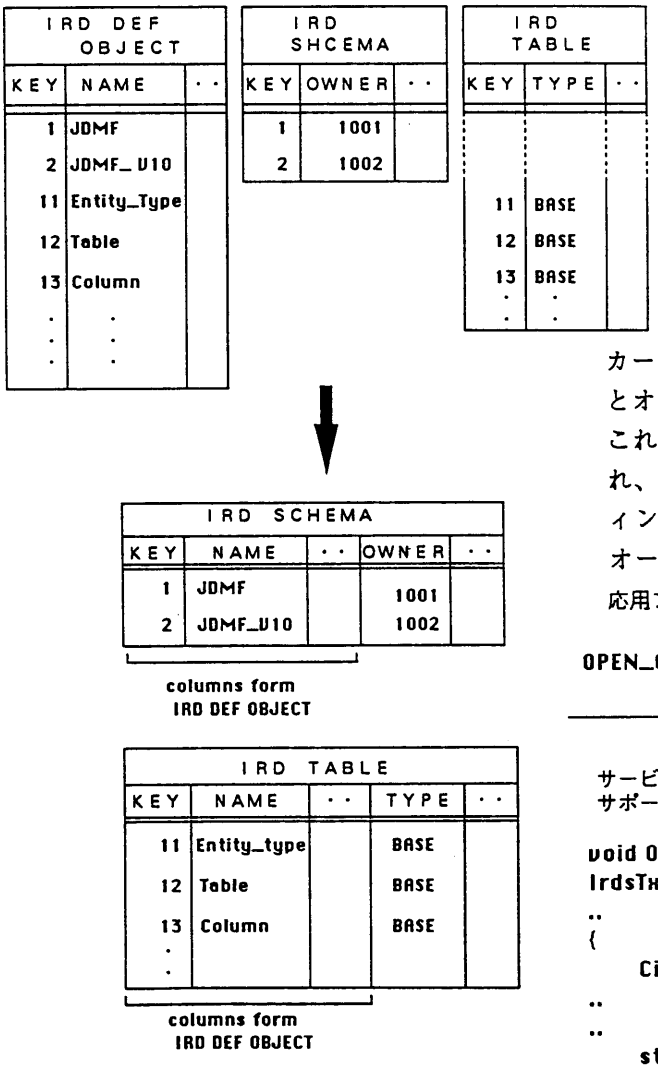

図５．IRDS1 の表の構造

(2)に含まれるサービスは、表に対する行の追加・検索・更新・削除である。Object表と副表との関係を図５に示すような形態の表とし、Object表に相当するViewを用意することにしたため、表の行とObjectが対応するようになっている。検索・更新・削除を行なうサービスは、カーソルを使用するが、これは、SEL01のではなく、SQLのカーソルである。

カーソルの使用に当たっては、カーソルの定義とオープン（使用開始の宣言）が必要であるが、これはOpen Obj Cursor サービスによって行なわれ、サービスインタフェース・サポートルーティン内部では、SQL-DBMSのカーソルが定義・オープンされる（図６）。

応用プログラムでの呼び出し方

```
OPEN_OBJ_CURSOR("Select ...",No,...,*Cid);
```

カーソル定義、ロック有無、..,カーソルID

サービスインタフェース
サポートルーティン

```
void OPEN_OBJ_CURSOR ( SqlStmt, HLock,...,Cid)
IrdsTxt SqlStmt;
..
{
    Cid = get_CursorId();
..                          未使用のカーソルのIDを変数
..                          Cidの値として得る。
    stmt = SqlStmt;
    switch(Cid) {           Cidの値によって、使用する文
     case 0:                を切り替える。
      EXEC SQL PREPARE S0 FROM :stmt;
      break;
     case 1:
      EXEC SQL PREPARE S1 FROM :stmt;
      ...
    }
    EXEC SQL DECLARE C' CURSOR FOR S';
    EXEC SQL OPEN C';
    ...
}
```

C',S'を使っている部分は、実際には、PREPARE文と同様にCidの値によって違う名前をつかった文に分岐するようになっている。

図６．Open_Obj_Cursorサービス

カーソルの定義は、このサービスの入力パラメタの一つであるSQL-DDLのSELECT文に基づいて行なわれる。このサービスは、成功した場合カーソルIDを返すのだが、IRDS1では同時にオープンできるカーソルの数には上限が設けられている。なぜならば、動的SQLインタフェースでカーソルを指定するために使用するカーソル名は、ホスト変数を使って指定することができず、したがって異なるカーソル名を持った幾つかの動的SQL文を用意し、指定によってカーソルを使い分ける以外に、同時に複数カーソルのオープンを実現する方法はなく、用意した動的SQL文の個数が上限となる。

また、更新・削除サービスを行なう場合は、カーソルの対象に更新ロックを掛けるようにパラメタをセットする必要がある。

SQLカーソルのオープンに成功したら、Open Obj Cursor サービスは、そのカーソルを識別するカーソルIDを出力パラメタとして返す。

Open Obj Cursor サービスによって定義されたカーソルは、指定されたSELECT文によって選ばれる行の集合に割り当てられる。カーソルは、Retrieveサービスが呼びだされる度に集合内の行を順に指していき、指す行がなくなればそのことがRetrieveサービスの出力パラメタによって通知される（図7）。使い終わったカーソルはClose Obj Cursorサービスによって開放され、新たに定義して再利用することができる。

Retrieveサービスでは、固定されたインタフェースで様々なカーソルの定義に対応できねばならない。そこで、カーソル定義時点でそのサイズが確定する配列を要素として持つ構造体変数をOpen Obj Cursorサービス実行時に用意し、Retrieveサービスでは、入力パラメタで指定されたカーソルによる検索を行ないその結果を用意された構造体変数に格納し、この変数へのポインタを戻り値として返すようにした（図8）。

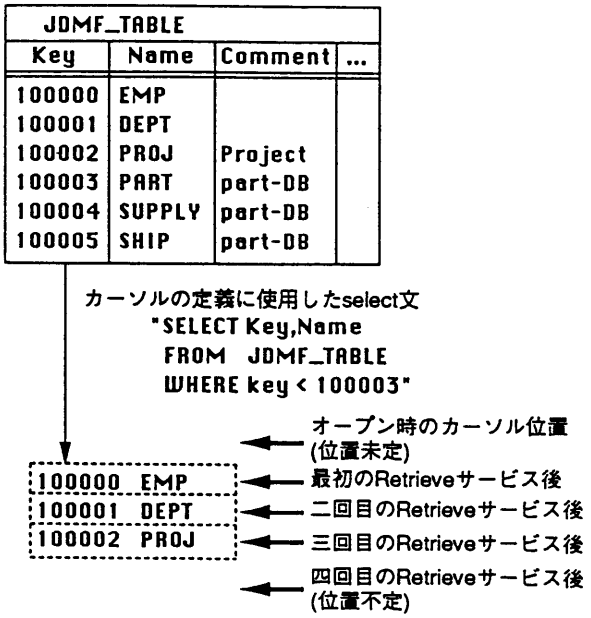

図7．カーソルを用いた検索

応用プログラムでの呼び出し方

```
OPEN_OBJ_CURSOR ("select key,name from
jdmf_table where key < 100003",No,...,*Cid);

for (;;) {   検索用のループ
  CL = RETRIEVE_OBJ (Cid,...);
  ...        オープンしたカーソルのIDを指定して検索を行なう
}
CLOSE_OBJ_CURSOR (Cid,..);
```

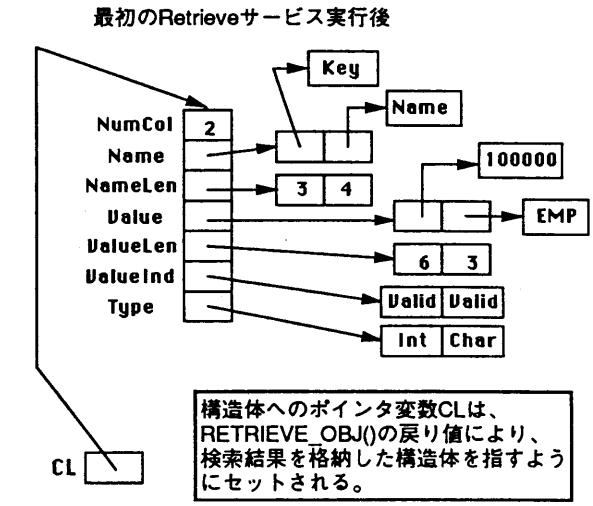

図8．検索結果を格納する構造体

ところが、この構造体変数では検索結果を全て文字列として格納しているため数値データは変換の手間が掛かる。そこで、検索するカラムの個数・各々のデータ型があらかじめ知られているとき便利なように、データ変換をした結果を指定した変数に格納するFetch_Colsサービスが用意されている（図９）。

更新・削除サービスは、カーソルの指している行を対象に行なわれる。更新は、Modify Objectサービスによって行なわれ、このサービスは入力パラメタとして、更新用のロックの掛かったカーソルIDとどのように変更するかを指定するためのUpdate文を要求する。削除は、Delete Objectサービスによって行なわれ、このサービスは入力パラメタとして、更新用のロックの掛かったカーソルIDとDelete文を要求する。どちらのサービスもSQL-DML文を入力パラメタとして指定する必要があるが、操作対象の選択条件を指定するwhere 節のない文である必要がある。これは、本来ならば入力パラメタとしては"UPDATE ... WHERE current of <cursor-name>"（＜＞は、カーソル名が入る）を要求したいところではあるが、サポートルーティンを記述する際に使用しているEXECUTE IMMEDIATE文ではcurrent of <cursor-name>というwhere 節を持ったUPDATE/DELETE文を実行できないからである。そこで、カーソルを用いたUPDATE文をSQLプリコンパイラが翻訳するやり方を真似て、"WHERE rowid = 'カーソルの指す行のシステムID'"というwhere 節を入力されたUPDATE/DELETE文に追加して、その変更された文をEXECUTE IMMEDIATE文によって実行することにしている。SQLインタフェースのEXECUTE IMMEDIATE文を使う理由は、サービスの入力パラメタである文字列をSQL文と見做して実行できるからである。

応用プログラムでの呼び出し方

```
int   tab_key;                    検索結果を取り込むため
char tab_name[30];                の変数を定義する
...
...
OPEN_OBJ_CURSOR ("select key,name from
jdmf_table where key < 100003",No,...,*Cid);

for (;;) {   検索用のループ
  RETRIEVE_OBJ (Cid,...);
  FETCH_COLS (Cid,2,...,&tab_key,tab_name);
  ...              検索後、同じCidを指定して
}                  データを指定した変数に代入させる

CLOSE_OBJ_CURSOR (Cid,..);
```

最初の検索後に実行されたFetch_Colsサービス

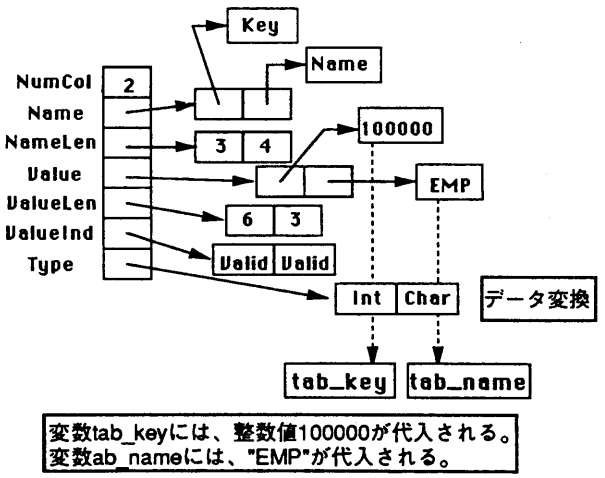

図9. Fetch_Cols サービス

AVFサービスは、AVFをサポートするためのサービスや、基本的なデータ操作サービスを利用したよりユーザフレンドリィなサービスを指す。前者の例としては、Working Setの生成・抹消・ライフサイクルフェーズの変更や、Working Setをスキーマと見做してのデータベースコンテナの実装等が挙げられる。後者の例としては、ユーザの登録、Working Setの構成要素の追加・検索・更新・削除等が挙げられる。
残念ながら、本原稿執筆時点ではAVFサービスの使用は未定であるため、実現もされていない。

5．使用例

今回実現されたIRDS1のサービスインタフェースを利用するには、前章で説明したようなサービスを関数として呼び出すようなC言語のプログラムを作る必要があるが、ユーザの便宜のため、サービスインタフェースを利用した応用プログラムの一つとして会話的なインタフェースを実現したプログラムを用意した。

これは、SQLインタプリタを使うのと同じような感じで、INSERT/SELECT/UPDATE/DELETE文を使用してIRDS1の各表に対して操作を行なうことができる。但し、このプログラムCLIPを起動するときに、IRDD_LPにあるデータベースを対象とするか、IRD_LPにあるデータベースのどれか一つを選択する必要があり、実行中は、選ばれたデータベース中の表のみが操作の対象となる。

参考文献

[1] ISO/IEC JTC1/SC21/WG3/IRDS SEL01,
IRDS Services Interface Working Draft, Revision 10,
1989-12-29

[2] 穂鷹良介、佐藤亮、Aurora Lo、岩崎一正、
情報システム開発に期待されるISO IRDSの役割、
情報処理学会　第４０回全国大会、1990-3

[3] Aurora Lo, 岩崎一正、佐藤亮、穂鷹良介、
Standard IRDS Architecture,
情報処理学会　第４０回全国大会、1990-3

[4] 岩崎一正、Aurora Lo、佐藤亮、穂鷹良介、
SQLによるIRDSの実現、
情報処理学会　第４０回全国大会、1990-3

[5] 佐藤亮、穂鷹良介、Aurora Lo、岩崎一正、
情報システムのライフサイクル・アプローチの
IRDS上の実現について、
情報処理学会　第４０回全国大会、1990-3

```
wiz% clip demo/irds

IRDS Command Language Interface Program (CLIP) Ver.3.0 1990/3

SELECT DB
        1. IRDD only
        2. IRD only
        3. Both

Your Choice ? : 1

IRDS> select key,name,table_class from irdd_table;

    KEY NAME                          TABLE_CLASS
------- ----------------------------- -----------
  30017 table                         data
  30018 column                        data

2 object(s) retrieved.

IRDS> select key,name,table_key from irdd_column;

    KEY NAME                          TABLE_KEY
------- ----------------------------- ---------
  40101 key                             30017
  40102 name                            30017
  40103 key                             30018
  40104 name                            30018
  40105 table_key                       30018
  40111 add_by                          30017
  40112 date_time_add                   30017
  40113 locked                          30017
  40114 add_by                          30018
  40115 date_time_add                   30018
  40116 locked                          30018

11 object(s) retrieved.

IRDS> select c.key,c.name,t.name
  2 : from irdd_column c, irdd_table t
  3 : where c.table_key = t.key;

    KEY NAME                          NAME
------- ----------------------------- ----------------------------
  40101 key                           table
  40102 name                          table
  40112 date_time_add                 table
  40111 add_by                        table
  40113 locked                        table
  40103 key                           column
  40116 locked                        column
  40104 name                          column
  40114 add_by                        column
  40115 date_time_add                 column
  40105 table_key                     column

11 object(s) retrieved.

IRDS> exit;

Transaction committed.

wiz%
```

図１０．CLIPの使用例