データベース・システム　59－5
（１９８７．５．１８）

データベース言語　ＮＤＬ

横山　一郎
㈱日立製作所　ソフトウェア工場

データベース言語ＮＤＬは、ネットワーク型のデータベース言語としてＡＮＳＩが草案を作成し、ＩＳＯでの審議を経て国際規格になろうとしている。またそれを受けて、日本でもＪＩＳ原案が作成された。

ＮＤＬはＣＯＤＡＳＹＬデータベースの仕様をベースとしながらも、プログラミング言語からの独立、論理性の強化、ＤＤＬとＤＭＬとの融合、仕様の簡明化などが図られている。特に、レコード中のデータ項目の値によって親子集合を決定する機能、ＦＩＮＤ文の形式の簡略化、適用業務プログラムから発行するＤＭＬを集めて記述する「モジュール」など、特徴点が多い。

このＮＤＬの制定の経緯とその特徴点について述べる。

DATABASE LANGUAGE NDL (in Japanese)

Ichiro YOKOYAMA
Software Works, Hitachi,LTD.
5030, Totsuka-cho, Totsuka-ku, Yokohama, 244 Japan

Database language NDL, a database language for network structured databases, was developed by ANSI. NDL had been reviewed also in ISO, and will soon be an international standard. In Japan, according to these situations, a draft standard of NDL was proposed to JIS.

NDL is based on the CODASYL database specifications, but many points are improved, for example, the independency on programming languages, the increase of logicality, simplification of the specifications.

The special points of NDL may be a function that determines the set occurence according to the value of an item in a record, a simple format of FIND statement, and modules that include DML's that are issued by an application program.

１．はじめに

データベース管理システムの発展とデータベースシステムの普及に伴って、データベース言語の標準化が注目されている。

ネットワーク型のデータベースに関しては従来からＣＯＤＡＳＹＬ仕様があったが、国際規格として認められたものではなかった。

今般国際標準化機構(ISO－－International organization for Standards)において、ネットワークデータベースに関するデータベース言語ＮＤＬ[1](Database language ＮＤＬ)が、国際規格(IS－－International Standards)になろうとしている。

このＮＤＬについて、その制定の経緯ならびにその内容についてのべる。

２．ＮＤＬ制定の経緯

２．１　ＣＯＤＡＳＹＬの活動

データベース言語の標準化の動きは、ＣＯＤＡＳＹＬ(Conference on Data Systems Languages--データシステムズ言語協議会）のデータベース作業班(DBTG－－Data Base Tasking Group)に始まると言ってよいだろう。

このＤＢＴＧの最初の報告書が1969年10月に提出された。これがいわゆるＣＯＤＡＳＹＬデータベース[2]の原型である。1971年に最終の報告書を提出したが、これを境として、データベース定義はデータベース管理者のためのものであってＣＯＢＯＬの一部ではないとの意見が強くなり、その結果、データ操作言語を検討する委員会と、データ記述言語を検討する委員会とが、分かれてしまった。

つまり、スキーマ記述に関する部分の検討は新設のデータ記述言語委員会(DDLC－－Data Description Language Committee)が担当し、ＣＯＢＯＬでのサブスキーマ記述及びデータ操作言語はＣＯＢＯＬのプログラム言語委員会が担当することになった。

その後もＣＯＤＡＳＹＬの活動は続けられたが、データ記述言語(DDL－－Data Description Language)とデータ操作言語(DML－－Data Manipulation Language)とを切り離すことはできないとの反省から、米国規格委員会(ANSI－－American National Standards Committee)が、ＤＤＬとＤＭＬの両者を含むＮＤＬをまとめた。

２．２　ＡＮＳＩとＩＳＯの活動

ＡＮＳＩは1978年5月に、ＤＤＬＣの1978年版報告書をベースとして、ネットワークデータベースのＤＤＬの標準化を検討しはじめた。しかしこのドキュメントにはＤＭＬを含むべきとの意見により、郵便投票で否決された。

そこでＡＮＳＩは、ＣＯＤＡＳＹＬ　ＣＯＢＯＬ委員会の1978年1月版開発報告をベースとしたＡＮＳＩ　Ｘ３Ｊ４　ＣＯＢＯＬ委員会の1980年７月版報告書のXVI"データベースモジュール"をベースドキュメントとして、ＤＭＬの検討を始めた。この結果できあがったのが、ネットワークデータベース言語ＮＤＬの草案(dpＡＮＳ　ＮＤＬ)である。

ＩＳＯでは、ＡＮＳＩからのＮＤＬの提案を受けて、1985年1月に国際規格草案(DP－－Draft Proposal)の登録がなされた。

その後ISO/TC97/SC21/WG3で審議を重ね、1986年11月に国際規格案(DIS－-Draft International Standards)の郵便投票が行われた。この結果賛成の支持が得られ、国際規格(IS－－International Standards)にするための手続きが進められている。

また、ＮＤＬと並行してＳＱＬの国際規格化も

同じ日程で進められている。

２．３　ＪＩＳの活動

ＩＳＯでの動きに合わせて、日本では1984年12月情報処理学会規格委員会の下にデータベース小委員会を設置し、ISO/TC97/SC21/WG3に対応する活動を行ってきた。

日本工業規格としてもＮＤＬがＤＩＳとなったのを受けて、日本規格協会情報技術センターの下にデータベース言語調査委員会を設置し、ＮＤＬ及びＳＱＬの翻訳検討を行い、DISを基にＪＩＳ原案[3)]を作成した。

３．ＮＤＬの概要

ＮＤＬは、ＣＯＤＡＳＹＬ仕様のデータベースの流れをくんでいるので、考え方や用語など、ＣＯＤＡＳＹＬ仕様と類似したものが、多い。しかしＣＯＤＡＳＹＬ仕様の拡張ではなく、全く新しい言語と位置づけている。

このＮＤＬの概要を以下に述べる。

３．１　ＮＤＬの構成

ＮＤＬドキュメントの中では、次の内容について規定している。

（１）概念と用語

ＮＤＬで使用する用語と概念を規定している。

（２）スキーマ定義言語

データベースの論理構造を定義するスキーマ定義言語の仕様を規定している。

（３）サブスキーマ定義言語

データベースのプログラムビューを定義する、サブスキーマ定義言語の仕様を規定している。

（４）モジュール言語

業務プログラムからＤＢＭＳに対して、データベース操作を要求するときのインタフェースとなるモジュール言語の仕様を規定している。

（５）データ操作言語

ＤＭＬの仕様を規定している。

３．２　ＮＤＬの特徴

ＮＤＬの特徴点は、次のようにまとめられよう。

（１）プログラミング言語からの独立

ＣＯＤＡＳＹＬ仕様は、データ操作言語（ＤＭＬ－－Data Manupulation Language）の検討がＣＯＢＯＬの委員会と一体となってが進められてきたため、プログラミング言語は当初ＣＯＢＯＬが前提であった。このためデータベース記述言語（ＤＤＬ－－Data Description Language）も非常にＣＯＢＯＬ言語に近いものであった。

ＮＤＬでは、データベース言語をこのようなプログラミング言語から独立できることが、一つの大きな特徴として根底に流れている。

このプログラミング言語からの独立性を保証するために、新しく「モジュール言語」を取り入れた。これは適用業務プログラムで使用するＤＭＬだけを独立して記述するもので、あり適用業務プログラムからはこのモジュールを呼び出すことにより、データベース操作を実行する。

この考え方から言えば、呼び出しのインターフェースが合っていればどんなプログラミング言語からもデータベース操作ができることになる。しかし、ＮＤＬではＣＯＢＯＬ，ＦＯＲＴＲＡＮ，ＰＬ／Ｉ、Pascal を標準プログラミング言語と規定している。

（２）データ操作の論理性の強化

ＮＤＬでは、レコードを論理的に扱えるようにすることに力が入れられており、これが大きな特徴となっている。

例えば、目的のレコードを捜し出すためには、そのレコードを規定する条件が決まればよいのであって、レコードを規定するキー項目の規定は不要としている。

また、「域」(domain)と言う用語を使って、捜したいレコード群を集合と捉え、その集合の中から順次レコードを取り出していくという考え方でデータ操作方法を規定している。

（３）ＤＭＬとＤＤＬの融合

ＤＭＬとＤＤＬとが一つのＮＤＬの中で規定されており、一つのデータベース言語として融合がはかられている。

（４）仕様の簡明化

ＣＯＤＡＳＹＬ仕様を基本としながらも、ＥＲＡＳＥ文（レコードの消去）での子レコード(member record)の消去規定を簡略化したり、親子集合(set)における親レコード(owner record)と子レコードの関係を、それぞれのレコードの中にあるデータの値によって結び付ける機能など、仕様の簡略化がはかれている。

３．３　データベースの概要

ＮＤＬで規定するデータベースは、レコードをデータ操作の単位とし、レコード間の関係は親子集合で表現する、ネットワークデータベースである。つまり、一つのレコード実現値に対して同じ型の複数のレコード実現値が関連をもつ場合、前者を親レコード、後者を子レコードとする親子集合として実現する。

親子集合の例を図－１に示す。この例では、商品レコードを親レコードとし、その取引を示す伝票を子レコードとしている。

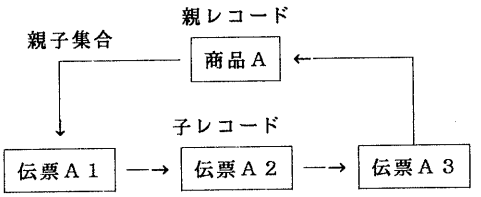

図-1　親子集合によるレコード間の関連

レコードの中には種々のデータが含まれる。このデータ項目(item)を「構成要素」(component)と呼ぶ。レコードは構成要素の集まりであり、データ操作の単位となる。構成要素は単一のデータ項目か、或いは配列である。

図-2にレコードと構成要素との関係を示す。

レコード

構成要素-1（単一項目）	構成要素-2（単一項目）	構成要素-3（配列）			
		#1	#2	#3	#4

図-2　レコードと構成要素の関係

ＮＤＬでは、集団項目や集団配列を規定していない。これは、データの独立性を非常に重視していることによる。

つまり、集団項目や集団配列はデータ項目の物理的な位置関係を意識するもので、データの独立性に反するとの考え方が強い。更にＮＤＬでは、データの内部表現（特に数値項目に問題が多いのが実情）については、一切規定していない。このため、集団項目のデータ属性の扱いが不明確となるのを避けたものである（事実、COBOLとPL/Iとで扱いが異なる）。

またこの考え方は関係データベースと仕様を合わせたものであろう。

４．特徴的な機能

４．１　親子集合

親子集合は親レコードと子レコードとの関係を表すものである。この中で特徴的な機能について以下に述べる。

４．１．１　再帰親子集合

一般に一つの親子集合型では、親レコード型と子レコード型との関係によって、親子関係は明確である。この親レコード型と子レコード型が同じレコード型であるような親子集合を、再帰親子集合という。

再帰親子集合を図-3に示す。

図-3　再帰親子集合型

４．１．２　親子集合型の属性

親子集合型には、親レコード型と子レコード型との関係の強さを表すための属性として、組入れ句(insertion clause)と保留句(retain clause)がある。CODASYLデータベースでは、組入れ句はAUTOMATICとMANUAL、保留句はFIXED,MANDATORY,OPTIONALであったが、これに組入れ句のSTRUCTUALが加わった。

STRUCTUAL属性は、次のような機能をもつ。

子レコードを格納するとき、各親子集合型に関する親レコードを決めなければならない。これを、子レコードに含まれるデータ項目の値に等しい値をもつレコードを親レコードとするものである。

この機能によれば、位置指示子を使わないで親レコードを決めることができる。

STRUCTUALの例を図-4に示す。

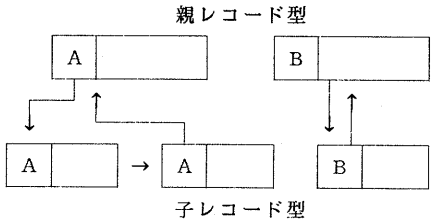

図-4　STRUCTUAL属性の親子集合

４．２　整合性制約

整合性制約(integrity constraints)は、データベースの中のデータの正当性をＤＢＭＳが保証するための機能である。これらの条件は、スキーマ定義で指定する。

特徴的な整合性制約の項目は以下のとおりである。

４．２．１　検査条件

検査条件(check condition)は、あるレコードがデータベース上に存在するために必要な条件をいう。ＤＢＭＳはこの条件を検査することにより、データベース上のデータの正当性あるいは整合性を保証する。

この条件は２種類ある。一つは、そのレコード自身の正当性を規定するものである。二つめは、親子集合の中で親レコードと子レコードとの関係から、整合性を規定するものである。

４．２．２　一意性制約

一意性制約(uniqueness constraint)は、あるレコード型のなかで、ある構成要素の値が一意であることを保証する機能である。

これはいわゆるキー項目に対応するものであるが、特にキー項目と限定はしておらず、どの項目にも適用できるし、また複数の項目に対しても適用できる。

４．３　データ操作言語

ＮＤＬで規定しているＤＭＬは表－１のとおりである。この中から特徴的な点を以下に示す。

表-1　ＮＤＬのＤＭＬ一覧

データ操作文	内　　容
COMMIT	ﾄﾗﾝｻﾞｸｼｮﾝを終了する
CONNECT	親子集合に接続する
DISCONNECT	親子集合から切離す
ERASE	レコードを消去する
FIND	目的のレコードに位置付ける
GET	レコードをプログラムに渡す
MODIFY	レコードの内容を変更する
NULLIFY	位置指示子をクリアする
READY	ﾚｺｰﾄﾞをｱｸｾｽできるように準備する
RECONNECT	別の親子集合に再接続する
ROLLBACK	ﾄﾗﾝｻﾞｸｼｮﾝをﾛｰﾙﾊﾞｯｸで終了する
SRORE	レコードを格納する
TEST	ﾃﾞｰﾀﾍﾞｰｽの状態をテストする

４．３．１　ＦＩＮＤ文

CODASYLではいくつかの形式のあったFIND文もＮＤＬでは一つの形式に単純化されている。

FIND文の形式をよく使われる形式に限定して書くと次のようになる。

```
FIND <探索方向> <域指定> [WHERE <条件>]
        [FOR UPDATE] [RETAIN ...]
  ただし、
   <探索方向>::=
        FIRST | LAST | NEXT | PRIOR
   <域指定>::=
        <ﾚｺｰﾄﾞﾋﾞｭｰ名>
      | [<ﾚｺｰﾄﾞﾋﾞｭｰ名>] IN <親子集合ﾋﾞｭｰ名>
```

ここで注目したいのは、<域(domain)指定>という用語である。この考え方は、条件を満たすレコード群を集合とみなし、その集合から目的のレコードを一件ずつ取り出すことを意味している。

またこの形式からもわかるように、キー項目に関するものは何も表れていない。つまりデータの論理性からいうとキー項目を意識することは不要との考え方を示している。

４．３．２　ＥＲＡＳＥ文

レコードを消去するとき、その子レコードの扱いが問題となる。ＥＲＡＳＥ文にはFULL CASCADEとPARTIAL CASCADEのどちらかが選択できる。

FULL CASCADEを指定すると、全ての子レコードは消去される。PARTIAL CASCADEを指定すると、スキーマ定義での保存句の指定により、子レコードは次のようになる。

固定　(FIXED):　　消去する

準固定(MANDATORY): 例外を発生させる

任意　(OPTIONAL):　親子集合から切離す

このように、消去の機能は非常に単純化されている。

４．３．３　ＲＥＡＤＹ文とＦＩＮＩＳＨ文

READY文では以下の形式のように、レコードビュー名を指定するようになっている。

```
READY  <ﾚｺｰﾄﾞﾋﾞｭｰ名>
         { EXCLUSIVE | PROTECTED | SHARED }
         { RETRIEVE | UPDATE }

       <ﾚｺｰﾄﾞﾋﾞｭｰ名> ....
```

レコードビュー名はサブスキーマで定義するもので、ここでもデータの格納形態を意識しないようにしている。

FINISH文は初期のＮＤＬには含まれていたが、最終的にはCOMMIT文及びROLLBACK文のオプションとしてFINISHの指定ができるようにして、FINISH文は削除された。

４．４　プログラムインタフェース

プログラミング言語からの独立を図るため、新しい概念として「モジュール」を導入している。

このモジュールを含めたプログラムインタフェースについて、特徴的な機能について以下に述べる。

４．４．１　モジュール

モジュールはある適用業務プログラムで発行するＤＭＬをまとめたもので、適用業務プログラムからはそのモジュールの中に書かれている手続(procedure)を呼び出すことにより、データ操作を実行する。

モジュールの書き方は、モジュール言語(module language)として規定されている。

このモジュールと適用業務プログラムとの結合のしかたは、処理系作成者(implementer)に任されている。

モジュールの概念を図-5に示す。

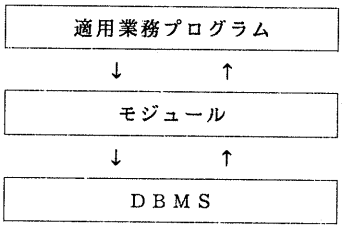

図-5　適用業務プログラムとモジュールの関係

４．４．２　手続

手続は、一つ以上のＤＭＬで構成し、適用業務プログラムからの呼び出しの単位である。つまり、ＤＭＬを基本的なデータ処理単位と考え、手続を論理的にまとまった処理と考えている。

この考え方に沿って、ＤＭＬで例外が発生したときの回復の単位も手続を単位としている。つまり、三つのＤＭＬから成る手続の２番目のＤＭＬで例外が発生すると、既に処理し終わったＤＭＬに対しても、処理前の状態に戻してしまう。そして処理結果を示すステータスコードも、ＤＭＬ単位ではなく手続の処理結果として、適用業務プログラムに返す。

４．４．３　パラメタ

モジュールを通しての適用業務プログラムとのデータの受け渡しにはパラメタを使い、そのパラメタをCALL文の引数とすることによって行う。

ＤＭＬの処理結果を確認するためのステータスコードも、引数によって適用業務プログラムに渡される。このようなＤＢＭＳとの状態確認のための特別なパラメタには、次のものがある。

STATUSﾊﾟﾗﾒﾀ:DMLの処理結果を示すｽﾃｰﾀｽｺｰﾄﾞを返す
TESTﾊﾟﾗﾒﾀ:TEST文の処理結果を返す
RECORDﾊﾟﾗﾒﾀ：処理したﾚｺｰﾄﾞのﾚｺｰﾄﾞ名を返す

データの受け渡しのためのパラメタは、データ項目としての構成要素(component)を単位とする。つまりデータベースと適用業務プログラムとの間では、構成要素ごとにデータの転送を行う。このとき必要に応じてデータの変換を行う。

ＮＤＬではデータの内部表現を規定していないので、ここでプログラミング言語の仕様差を吸収することになる。

５．ＮＤＬに含まれない項目

ＮＤＬの「適用範囲」の中に、ＮＤＬの中で規定していない項目が列挙されている。

その中の主な項目を以下に示す。

①機密保護に関する機能

②スキーマやサブスキーマの一部を作成、修正、削除するための操作言語

③データ辞書(data dictionary)へのインタフェース

④物理記憶構造と物理アクセス方法を定義するためのデータ記憶定義言語

⑤自然言語問合せ機能

6．むすび

ＩＳＯにおけるＮＤＬの審議は、ＩＳ化によって終了した。これからは、同じISO/TC97/SC21/WG3の中で、データベース言語ＳＱＬやリモートデータベースアクセス（ＲＤＡ）の関連で、検討が加えられていくものと予想される。

ＮＤＬはＣＯＤＡＳＹＬデータベース仕様との互換性が保証されていないが、ＡＮＳＩでの規格化に続いてＩＳＯでもＩＳ化が進められており、世の中の注目を集めてきている。

ＮＤＬの中には関係データベースの仕様を意識したものも多く、関係データベースとの共存も実現しやすいと考えられる。

多くの製品に反映されていくことを期待したい。

参考文献

1）ISO/DIS 8907:Infomation processing systems--Database language ＮＤＬ (1986)

2）Olle,T.W.:The Codasyl Approach to Data Base Management (1978)

3）日本規格協会：高度ネットワークのためのプロトコル標準化に関する調査研究（データベース言語調査研究）報告書 (1987)