CODASYLデータベース用共通言語の進展'78

植村俊亮（電子技術総合研究所）

1. 背景と概要

本稿はCODASYL（データシステム言語協議会）のデータベース用共通言語体系の最近のおもな仕様改訂をまとめて，データベースシステムにおけるその意義を論じるものである．

CODASYLのデータベース作業班がいわゆるDBTG'71提案[1]を最終的に取りまとめ，さらにCOBOL委員会（当時のPLC；プログラム言語委員会）がその基本線を承認したとき，筆者は幸運にも両委員会に出席して，激しい賛否の討論を直接見聞することができた[2]．この言語仕様に反対する意見は少数派であったが，「必要以上に複雑である」，「データ独立性を欠く」などの批判を言語仕様にこまかく立ち入って展開した．賛成する意見は圧倒的多数であった．しかし賛成論は，批判に技術的に答えるよりはむしろ，「反対意見の主旨はわかるが，提案内容はすでになん年も審議してきたものである．反対意見に具体的な提案が伴ってない以上，それは将来の課題として残すべきで，今回は時間切れである．」といった大義名分論に終始した．本稿の内容の源泉をたどると，この時点までさかのぼりうるものがおおい．たとえば，LOCATION（配置）句の削除（2.1参照）は当時すでに議論の対象になっていた事項である．ただこの改訂のもつ意義，効果，対応策などについてのわれわれの理解は，当時よりはるかに進んでいる．ここで「われわれ」という中には，CODASYLの各委員会関係者をも当然含んでいる．

DBTG'71提案をもとにして，1973年にデータベース記述用言語（DDL；Data Description Language）の仕様が完成して公刊された．[3]はその完訳である．この段階でも，COBOL用発報告（COBOL JOD）と並んでDDL用発報告（DDL JOD）をとにかくまとめることが最優先されて，DBTG'71提案の内容をおおきく変更するような事項はすべて後回しになった．

おなじくDBTG'71提案をもとにして，1975年にCOBOLデータベース村能が完成し，COBOL用発報告1976に組み込まれたが，ここでもDBTG'71提案の基本線をはずれない原則が固守された．[3][4]†

このデータベース用共通言語体系の完成と，E.F. Coddによるデータベースの関係モデルの登場とをきっかけとして，1970年代のデータベースシステム研究はいちじるしい進歩をとげた．

CODASYLでは，DDLはDDL委員会，COBOLデータベース村能はCOBOL委員会がそれぞれ言語仕様の保守改訂を担当して，懸案事項や新たに発生した疑問の解決に取り組んできた．最近になって，アメリカ規格COBOLの再改訂，アメリカ規格DDLを作る方向が固まったことなどの事情が重なって，両委員会は1977年末をひとつの区切りとして，より積極的に言語仕様改訂を行なった．基本線はいぜんとしてDBTG'71提案であるが，新しい研究成果，実際に実動化してみた経験的改良意見などが大胆に取り入れられて，かなりの変貌をきたしており，またそれだけに新たな内題をも提起している．おもな内容は次のとおりである．

† 1977年には，FORTRANデータベース村能も完成したが，本稿ではふれない．

(1) DDLの各記述項の材能を分類し，いくつかの句を削除した．
(2) DSDL (Data Storage Description Language) をDDLから分離した．
(3) 親レコード型と子レコード型とが同一ないわゆる再帰的親子集合を許すことにした．
(4) データベースキーの概念を変更し，またレコードキーの概念を導入した．
(5) 同時実行の制御材能を全面的に改訂した．
(6) ロールバック材能を導入した．
(7) その他

2. データベース記述用言語DDLにおける改訂

2.1 DDL記述項の分類

1973年以後のDDL委員会は，まずDDL記述項を機能的に分類して洗い直す作業を集中的に行なった．ここからその後のいくつかの改訂のきっかけが生まれた．

DDL記述項は次の九つに材能的に分類される．[6]

(1) 固有データ構造 (schema)
(2) 構造 (structure)
(3) 正当性の確認 (validation)
(4) データ操作インタフェース (DML interface)
(5) 呼出し制御 (access control)
(6) 計測 (mesurement)
(7) 性能向上 (tuning)
(8) 資源割当て (resource allocation)
(9) 管理 (administration)

この分類の過程で，まず材能が明瞭でない，個別データ構造記述の側の担当であるなどの理由で，TEMPORARY (一時) 句，ENCODING/DECODING (符号化，復元) 句が削除された．

性能向上材能についても議論が行われて，DDLにおける性能向上材能はデータの表現手法を直接指示すべきものでないこと，それらは記憶構造記述 (storage schema) とでも呼ぶべきものに属すること，DDLでは単にデータの使われ方をデータベース管理システムに知らせるていどに止めることなどの合意に達した．この結果次の四つの句がDDLから削除された．

LOCATION (配置) 句．
PRIOR (逆方向) 句．
LINKED TO OWNER (親接続) 句
SEARCH (探索) 句．

なお追加された記述項はあまりおおくないが，親子集合記述項における
STRUCTURAL (構造制限) 句
は，親子集合における親レコードと子レコードとの間で，指定したデータ項目 (群) の値が同一であることを条件づける句として目新らしい．

2.2 DSDLの分離

アメリカ規格協会/X3/SPARC/データベース研究班による3層スキーマの提唱はデータベースシステム研究の主流をなしつつある．[5][6][7] DDL委員会もこの影響を受けて，これまでの固有データ構造記述をさらに2分する方向に進んでいる．すなわち新たに記憶構造記述 (storage schema) を分離させて，

個別データ構造 (サブスキーマ)
固有データ構造 (スキーマ)
記憶構造 (ストレージスキーマ)

の3層を考える．従来からのDDLは固有データ構造を担当することとし，記憶構造記述のためには，べつにDSDL (Data Storage Description Language) を作る．

イギリス計算機学会でCODASYL方式のデータベースシステムの実動化を研究していたグループが，CODASYL DDL委員会と協力して，DSDL開発に取り組んだ．このグループはDBAWG (Data-Base Administration Working

Group）と呼ばれる．DBAWGの目的はデータベース管理者用の道具を研究開発することだったので，DSDLはうってつけであった．1977年中にDBAWGは一つのDSDL仕様をまとめた[8]．しかしDBAWGはこの言語仕様をDDL開発報告中に組み入れるのでなくて，DDL開発報告の付録として追加する形式にしたいと希望している．次の開発報告中でどう扱われるかは，まだ確定していない．

DBAWGは今回まとめたDSDLの特長を次のように述べている．

（1）DSDLにも高水準のものと，低水準のものとが考えられる．今回のものは低水準のそれである．

（2）DSDLはDDLとははっきり別の言語として考えた．

（3）固有データ構造におけるデータ項目と記憶構造におけるデータ項目との間の写像がなるべく柔軟に行えるように配慮した．

（4）分散型データベースをとくに考えたわけではないが，固有データ構造の一部分だけを記憶構造に写せる機能がある．

（5）このDSDLはCODASYL DDLを念頭に置いている．親子集合の実動化技法としては，ポインタを採用した．

DSDLによる記憶構造記述の意義は図1に明らかであろう．

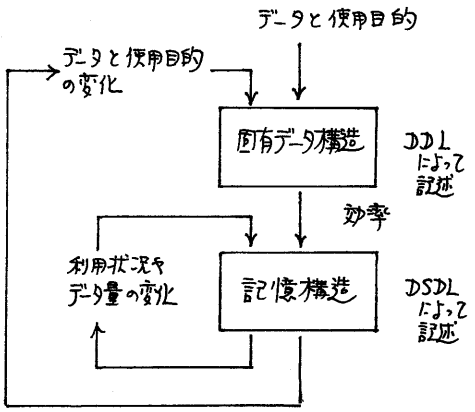

図1　固有データ構造と記憶構造[8]

前述のLOCATION句，LINKED TO OWNER句などの代替物がDSDLに用意されている．ただここでは，POINTERといった用語がもろに表面に現われている．

2.3　一意な識別子

DBTG'71提案では，データベース中のすべてのレコード実現値を一意に識別するデータベースキーの概念があった．データベースキーは，そのレコード実現値がデータベースから削除されてしまうまで，永続的に存在し続けた．実行単位（応用プログラム）はこの値を呼び出すことができ，直接呼出しに使ったり，後続の実行単位が利用したりできた．

しかし実際には，ある実行単位が保存しておいたデータベースキーを使って，後続の実行単位がレコードを参照してみても，さきと同じレコードを呼び出せる保証はどこにもない．その間に別の実行単位が問題のレコードを削除し，データベースキーは別のレコード用に使われてしまっているかもしれない．

DDL委員会では，旧来のデータベースキーの概念はまったく削ってしまって，新たにシステム中心のデータベースキーの概念を導入した．新らしいデータベースキーは次の特長をもつ．

（1）データベース管理システムと実行単位との間で，その実行単位が参照するすべてのレコードを一意に識別するようなデータベースキーを考える．

（2）このデータベースキーは，その実行単位が生きている間だけ有効である．

（3）データベースキーの内部表現は作成者が定める．

（4）実行単位がデータベースキーによってレコードを呼び出したからといって，能率がよいかどうかはわからない（作成者による）．

(5) データベースキー用のデータ項目（TYPE DATA-BASE-KEY）の概念は削除する．

レコードをなんらかのキーを手がかりにして直接呼び出したいときには，次のレコードキーの概念を使えばよい．これはまったく新しい概念である．

レコード中の任意のデータ項目（群）をレコードキーと指定できる．レコードキーは，そのレコード型の範囲内でレコード（一群のこともある）を識別する．一つのレコード型にキーが複数個あってもよい．レコードキーは，そのレコード型が親子集合中でどんな役割りをはたすかには関係がない．

レコードキーをさらにレコード順序キーと指定することもできる．レコード順序キーを指定すると，そのレコード型のすべてのレコードが指定されたキーの順に論理的に並ぶ．

すなわち，従来のデータベースキーによる呼出しはレコードキーによる呼出しに，またデータベースキーによる順序づけはレコード順序キーによるそれに拡張的に置き変えられたと考えられる．データベースキーはデータベース管理システム用の識別子，レコード（順序）キーは，利用者が定義し管理する識別子である．

2.4 再帰的親子集合

親子集合において，親レコード型と子レコード型とが同一であるような構造は，これまでは禁止されていたが，[9] 許すことになった．間接的な輪構造はもともと許していた．今後は直接的な輪構造も許される（図2参照）．DDL

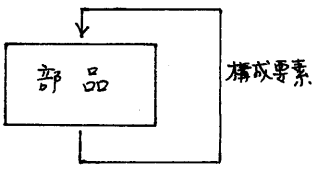

図2　再帰的親子集合

委員会では，これを再帰的親子集合（recursive set）と呼んでいる．再帰的親子集合を構成するレコード型の子属性は手動（MANUAL）でなければならない．またその親子集合選択基準は原則として，現在参照している親子集合によるものとする．

3. COBOL データベース拡能における改訂

3.1 同時実行の制御

これまでのCOBOL データベース機能におけるデータベース同時更新の制御は，従来から問題になっていた．[10] これまでは，個々のレコードをロックするかわりに，監視モードという実行モードが用意されていたが，同時実行の制御にはあまり役に立たなかった．今回これが全面的に改訂された．

同時実行単位がデータベースを更新するときのデータの安全性を確保するために，レコード錠（record locks）を考える．レコードごとに，更新錠と選択錠があるものとする．

実行単位は，自分がSTORE, MODIFY, CONNECT, DISCONNECT命令の対象にした各レコードに更新錠をかけてしまう．このとき同時実行単位は，更新錠のかかったレコードに対しては，FIND, ERASE, MODIFY, CONNECT, DISCONNECTのどの命令も実行できなくなる．

実行単位は，現在指示子が指しているすべてのレコードおよび，保管表（keep list）に登録したすべてのレコードに選択錠をかけてしまう．保管表は実行単位ごとに自由に定義するもので，レコードにKEEP命令を実行すると保管表に登録でき，FREE命令によって保管表のレコード登録を抹消できる．同時実行単位は，選択錠のかかったレコードに対しては，ERASE,

MODIFY, CONNECT, DISCONNECTのどの命令も実行できなくなる.

保管表は実際にはデータベースキーのFIFOスタックであると説明されている.

デッドロックの検出と解決については, (当然ながら) 作成者が定める.

3.2 保管表による再呼出し

COBOLデータベース村能の世界でもデータベースキーが一意に識別するのは, 実行単位の生きている間だけという変更が行われた. そしてデータベースキーの値を利用することは, 場合によっては可能であるが, 値を呼び出すことはできない. 保管表はデータベースキーの値を有効利用する手段に使うことができる.

すなわち, FIND命令のレコード選択式に,

{FIRST / LAST} WITHIN 保管表名

と書くことができるようになった. これは, 一度呼び出したレコードを高速に再呼出しするための手段である.

FIRSTとLASTとをどちらでも書けるのは, FIFOスタックという説明と矛盾している. なおこれだけで十分かどうかという疑問も残る.

3.3 静止点

次の三つの状態を実行単位の静止点 (quiet point) という.

(1) 実行開始の時点.

(2) COMMIT (静止) 命令の実行直後.

(3) ROLLBACK (引戻し) 命令の実行直後.

静止点はいわゆるロールバックのための村能で, 実行が停止するわけでない.

COMMIT命令を実行すると, その実行単位がかけていたすべての更新錠, 属状錠がはずされる. この実行単位の保管表はからになり, 現在指示子は空 (null) になる. そしてここが静止点になる.

ROLLBACK命令を実行すると, いちばん最近の静止点以降に実行したSTORE, ERASE, MODIFY, CONNECT, DISCONNECTの各命令によるデータベース変更をすべてもとにもどす. すなわち, いちばん最近の静止点の状態にもどる. 錠, 保管表, 現在指示子はCOMMIT命令の実行と同じ状態になる.

REMONITOR (再監視) 命令は削除された.

3.4 レコードキー

DDLにおけるレコードキーと同一の概念が登場した. データベースキーの値を利用者が呼び出せなくなり, またLOCATION句も削除されたので, FIND命令のレコード選択式がかなり改訂された.

まず従来の書き方1 (データベースキーによる) にかえて,

{FIRST / NEXT} レコード名-1

[USING {データ名-1} ...]

と書ける. この場合レコード名-1のレコード型には, レコード順序キーがなければならない.

次に書き方2 (CALCキーによる) にかえて,

{ANY / DUPLICATE} レコード名-2

USING キー名-1

と書ける. キー名-1のデータ項目はレコード名-2のレコード型のレコードキーでなければならない.

このほかに, 従来は領域中のデータ

ベースキーの値によって選択する書き方があったが，これらはすべてそのまま残っており，「領域中の次のレコードを選択する」などというあいまいな表現になっている．領域中でレコードがどんな順序に並ぶかは定義しないと明記してあるので，「次の」レコードといってもなにが選択されるかはっきりしない．この種の書き方はなるべくやめよという趣旨に解される．

4. おわりに

これまでの改訂を概観すると，
(1) 古い卵のからの整理．
(2) 3層スキーマへの傾斜．
(3) 関係モデルからの影響．
の3点に大別できる．なおさまざまの問題点が残されているが，全体として好ましい方向への改訂が進んでいるといえよう．親子集合の概念をまったく使わないデータベースでさえも，この言語仕様で扱いうる感触が出てきている．

参考文献

1) Data Base Task Group Report to the CODASYL Programming Language Committee April 1971, ACM (1971)
2) 植村：CODASYLの活動，情報処理才13巻才2号 (1972)
3) CODASYLデータベース用データ記述言語 1973年6月版，情報処理学会 (1977)
4) CODASYL COBOL Journal of Development 1976, Canadian Gouernment 110-GP-1d (1976)
5) 石田：ANSI/SPARC データベースシステム概念，情報処理才17巻才10号 (1976)
6) 植村：CODASYL方式のデータベースシステム，情報処理，才17巻才10号 (1976)
7) T. Toh, S. Kawazu and K. Suzuki: Multi-level Structures of the DBTG Data Model for an Achievement of the Physical Data Independence, Proc. 3rd VLDB (1977)
8) DBAWG CODASYL DDLC Proposal, DBAWG-7701.3 (1977)
9) C.W. Bachman: Why restrict the modelling capability of CODASYL data structure set ?, Proc. NCC 1977 (1977)
10) R.W. Engles: Currency and Concurrency in the COBOL Data Base Facility, in "Modelling in Data Base Management Systems, G.M. Nijssen, ed.", North Holland (1976)
11) CODASYL PLC, DDLC 議事録