COBOL データベース機能

伊藤靖彦・寺尾　実（日本電気）

1. はじめに

CODASYL PLC の DBLTG が提案した COBOL データベース機能提案に関し，その内容の概要およびこの提案に関する各方面からの問題提起（データベース言語作業グループで提起されたものも含む）について述べる。

2. COBOL データベース機能提案の概要

COBOL データベース機能提案（COBOL Data Base Facility Proposal, DBLTG-73001.00, 以下 DBLTG-73001 と略記）は，DBTG 提案のうち，COBOL サブスキーマのためのデータ記述言語と COBOL データ操作言語とを COBOL 文法の書式に合わせ，CODASYL COBOL 開発報告に組み込むための提案であり，1973 年 1 月に DBLTG から PLC へ提案された。DBLTG-73001 はいくつかの提案項目からなる。その構成を COBOL 開発報告に対比させると図 1 のようになる。

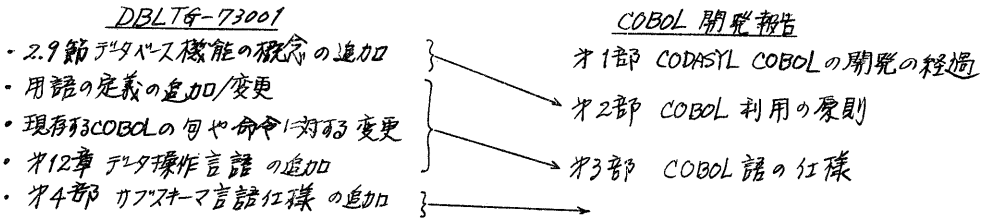

図1　DBLTG-73001 と COBOL 開発報告との対比

DBTG 提案から DBLTG-73001 を作る過程において DBLTG では決定事項を定めながら作業を進めていたが，その決定事項の中にはつぎのような項目がある。

- すべてのデータ操作用動詞は一意にする（すなわち，一般の ファイル処理用動詞とデータベース 操作用動詞とで同じものを使わない）。
- INVOKE 命令はより COBOL らしい表現に改める。
- DBTG の Area という用語には新しい用語（Realm と命名）が必要である（一般の英語，COBOL またはプログラミング言語での使用法と異なるものがよい）。
- その他

以下に COBOL データベース機能の概要を知るために，COBOL サブスキーマ言語と COBOL データ操作言語について述べる。

2.1 COBOL サブスキーマのためのデータ記述言語

サブスキーマは，COBOL プログラムとは独立に翻訳され，ライブラリに登録されているものであり，TITLE，ALIAS，REALM，SET および RECORD の 5 つの部で構成される。

TITLE 部では，サブスキーマ名を宣言する。

ALIAS 部では，スキーマでのレコードやセットの名前をサブスキーマ内で変更して利用できるようにする。

REALM 部では，そのプログラムで使用する領域（realm）を宣言する。

SET 部では，COBOL プログラムで使用するセットの型を指定する。スキーマで指

定するのとは異なったセット選択基準を定義することができる。

RECORD 部では，COBOL プログラムで使用するレコードやデータ項目を定義する。ここでは，スキーマで定義したデータ形式や項目の順序などを変更することができる（これらの変換作業は，データ操作言語を実行するときに，システムにより自動的に行なわれる）。

なお，これらのすべての部において，プライバシロックの指定が可能であり，スキーマにおけるプライバシキーを変更することができる。

2.2 COBOL データ操作言語

COBOL プログラムでデータベースを操作するためには，今までのデータ部にサブスキーマ節を新たに記述し，必要なサブスキーマをライブラリから呼び出しておく。そして，手続き部において データ操作用命令を用いて必要なデータをアクセスする。

主なデータ操作命令には，つぎのものがある。

READY, FINISH — 領域を指定したモード（検索／更新，専有／共有）で利用可能にし，終わると解放する。

STORE — レコードをデータベースに格納する。

FIND — レコード選択式によりレコード実現値を位置づけし，続く命令で利用できるようにする。

GET — 対応するレコード領域の中にデータベースレコードを移す。

MODIFY — レコード中のデータ項目の内容を変更する。

ERASE — レコードをデータベースから削除する。

CONNECT — データベース中のレコードをセットにつなぐ。

DISCONNECT — レコードをセットから切りはなす。

KEEP, FREE — レコードの同時実現の監視要求（要求取り消し）。

REMONITOR — KEEPによる監視情報を一度クリアし，再び監視要求を出す。

ORDER — セット中のメンバレコードの順序を変更する。

3. COBOL データベース機能の問題点

DBLTG-73001 に対して各方面から様々な問題点が提起されているが，ここではそれらのうち代表的と思われるものをいくつか挙げる。

3.1 レコードの修飾

従来の COBOL における修飾は，必要なデータ項目やレコードを一意にするためにのみ使われるものであり，修飾することにより句や命令の意味（semantics）が変わることはない。ところが，DBLTG-73001 のレコード選択式における レコードの修飾では，語 RECORD を使用するのとしないのとではその意味が変わってしまう。他の仕様に変えたほうがよい。

3.2 プライバシ機能

プライバシ機能に関してはつぎのような問題点があろう。

（1）スキーマやサブスキーマで，各種のデータベース情報に関し種々のプライバシロック指定が可能であるが，あまりに指定がありすぎるため，かえってプログラム側でのプライバシキーの指定作業を複雑にするのではなかろうか。

（2）プライバシキーをコンパイル時にプログラムに結合した場合*，スキーマやサブスキーマで対応するプライバシロックが変更されると，プログラムにまで影響を及ぼしてしまう。（* プライバシキーをリテラルで指定する場合）

(3) プログラマがプログラムを作る際に，データ管理者からプライバシキーを教えられるのはよいが，そのためにそのキーが盗まれる危険性がある。

これらの問題点を解決する一つの方法としては，プライバシ制御情報を集中管理し，各データ操作言語命令を実行するときに，プログラム名，プログラマ名またはユーザ名などをもとにして，その実行を許すか否かを判断することが考えられる。

3.3 データの保全機能

DBLTG-73001では，データの保全機能のために，KEEP命令，FREE命令，REMONITOR命令およびREADY命令のUSAGE-MODE指定がある。これらの機能に対してはつぎのような問題点があろう。

(1) KEEP命令とFREE命令を使わずにレコードを更新できるので，同時更新によるエラーが起きる可能性があるにもかかわらず，データベース管理システムはそれを検出できない。

(2) KEEP命令を使用することにより同時更新によるエラーは防げるが，他の実行単位で同じレコードが先きに更新された場合，そのレコードに対し再度処理をやり直す必要がある。したがって，いつでも処理のやり直しができるようなプログラムを組んでおく必要があるので，プログラマの負担が重くなる。

(3) USASE-MODEを変更するためには，一度FINISH命令を実行し，再度異なったUSAGE-MODEでREADY命令を実行しなければならない。しかしそれではシステムのオーバヘッドが大きくなり効率が悪くなることが考えられる。このためデータ保全エラーを防ぐために，プログラマは共用範囲を狭めがちになる。

たとえば，1つの領域に対し最初から専有する必要はないが途中から専有モードで使用するような場合，最初から専有してしまうであろう。こうするとデータベース人の共有度が低下し，システムのスループットが低下してしまう。

(1),(2)の問題点を解決するには，レコードに対するLOCK機能を導入し，レコードを更新するためにはそれにより影響を受けるすべてのレコードをLOCKしなければならないようにしておくことが考えられる。

(3)の問題点を解決するには，READY命令とFINISH命令の実行の間に，USAGE-MODEが変更できる機能を必要とする。方法としては，他のデータ操作言語命令を追加するとか，FINISH命令にTEMPORARYなどのオプションを設けるなどが考えられる。

3.4 データ操作言語のスキーマからの独立性

データのスキーマからの独立性に関してはつぎのような問題点があろう。

(1) データ項目の値をキーとしてレコードを検索する場合，レコードの型がCALCであっても単一セットのメンバであってもよいはずであるのに，現在の仕様では，それぞれに対するレコードの選択基準が異なる。

(2) 数種のデータ操作言語(FIND命令やSTORE命令など)では，領域(realm)を意識してプログラミングする必要がある。

(3) セットのメンバ様式(set membership)の変更とかセットの型の追加や削除があるとプログラムにも影響が出る。たとえば，AUTOMATICメンバに対してはSTORE命令だけでセットへの挿入が行なえたのに，これがMANUALメンバに変更された場合には，STORE命令だけでなくCONNECT命令も実行しなければならない。

(4) セットと繰り返しグループは論理的に同一の概念にもかかわらず，データ操作言語レベルでは区別しなければならない。たとえば，繰り返しグループのデ

ータをアクセスするには一組のFINDとGET命令でよいが，セットの場合には複数組のFINDとGET命令を実行しなければならない。

3.5 その他

(1) 検索機能をFIND命令とGET命令に分離するのはよいが，これらの命令を組にして使用する場合も多いと思われるので，FINDとGET命令の両機能を含むデータ操作言語も必要であろう。

(2) LOCATION MODEがDIRECTであるレコードを，データベースキーを指定してSTOREする場合，そのデータベースキーがすでに他のレコードに割り当てられていると，つぎに大きい値をもつデータベースキーを割り当てる方法をとっているが，この場合exceptionを起こした方がよいと思われる。たとえば，指定したデータベースキーが空いていないなら，別のデータベースキーを指定してもう一度STOREしたい場合もあるだろう。

(3) 1つのプログラムでは，ただ1つのサブスキーマしか使用できないが，複数個のサブスキーマの使用を許してもよいのではないか。

(4) データ操作言語命令を実行している間に何らかのエラーが発生した場合，現仕様では，すべて対応するUSE手続きが実行されて，制御が再びエラーの発生した命令のつぎに移る。そのため，その命令のつぎで再度エラー状態を判定してやらなければ，エラーの場合の行先きを変更できない。

4. おわりに

DBLTG-73001がPLCに提出されてから昨年9月までに，PLCには他の提案を含めて205件の提案がなされているが，そのうち約半数にあたる110件がDBLTG-73001に関するものであった。データベース機能がCOBOL世界に大きな波紋を投げかけていることがわかる。DBLTG-73001に対する提案は，1973年9月にその受付けが締切られ，これらの提案を1つ1つPLCで審議している。そして必要な場合には，DBLTGに仕様の書き直しを命じている。

定かではないが，COBOLデータベース機能は，DBLTG-73001の仕様を生かしつつ(仕様の根本的な変更は行なわずに)今後半年から1年の間にCOBOL開発報告に組み入れられるのではなかろうか。

参考文献

(1) COBOL Data Base Facility Proposal (Proposal DBLTG-73001.00), Jan. 1973
(2) CODASYLのデータベース用共通言語，植村・西村，データベース研究会資料73-4，1973年9月
(3) CODASYL COBOL Journal Of Development 1973, CODASYL PLC, June 1973
(4) PLC Proposals