データベース・システム 39－3
（1984. 1. 26）

データ構造の動的変更が可能なデータベース管理システム －EDBMS

高橋 光吉，小柳 和子
（日立ソフトウェアエンジニアリング）

1、 はじめに

一般に、時間の経過とともに、データベースに蓄積するデータやデータベースを利用するアプリケーションには進化が起る[1]。オーバフローレコードや削除レコードの増大により性能が低化したり、ユーザ組織の変更や規則の改正によってデータ構造の変更が必要になる場合がある。応用プログラムを常に効率良く実行したり、使用環境の変化に歩調を合わせていくためには、データベースの再設計、データのリロード、さらに応用プログラムの変更やリコンパイルが必要になる。このようなデータベース再編成の処理は、煩雑で多大の処理時間を必要とし[2]、データベースの保守、運用上の大きな問題となっている。

日立ソフトウェアエンジニアリングで研究中の実験システム、EDBMS（Evolutionary Data Base Management System）は、上記の問題に着目し、データやアプリケーションの進化に柔軟に対応できるデータベース管理システムをねらったもので、次のような機能の実現をめざしている。

（1）端末からの対話的な操作により、データベース使用環境の変化に応じたデータ構造の動的変更ができる。
（2）通常のデータアクセスと並行して動的な再編成を行う。
（3）再編成を局所化する。

これらの機能により、データベースの設計時に、すべてを正しく決定しなくても、試行錯誤をしながら徐々に――進化的に――ユーザの要求にあった効率の良いデータベースを作り上げることができるようになる。これが本DBMSの名称Evolutionary（＝進化的）の由来である[1],[6]。

本稿では、まずEDBMSの構成を簡単に述べ、ついでデータ構造の動的変更を可能にするダイナミックデータ構造について述べる。

2、 EDBMSの構成

EDBMSは、図1に示すように、四つのユーザインタフェース[6]（DDLコンパイラ、DMLコンパイラ、SMLプロセッサ、Queryプロセッサ）とそれらによって起動され、データベースへの実際のアクセスを実行するランタイムシステム（RTS）[7]とからなっている。

（1）DDLコンパイラ

データベースの構造定義を取り扱うモジュール。データ定義言語（DDL）によって記述されたデータ構造定義プログラムを解析する。解析したデータ定義情報はコントロールデータベース（CDB）と呼ばれるメタデータベースのデータ定義ライブラリ（DLB）に蓄積される。EDBMSのDDLは、ネットワーク型のデータ構造を定義し、データ項目、レコードタイプ，チェイン（CODASYLでいうセット[4]）、およびスキーマを定義する四つのセクションからなる。また、データのアクセスコントロールや正当性制御（integrity constraints）の指定をすることも可能である。

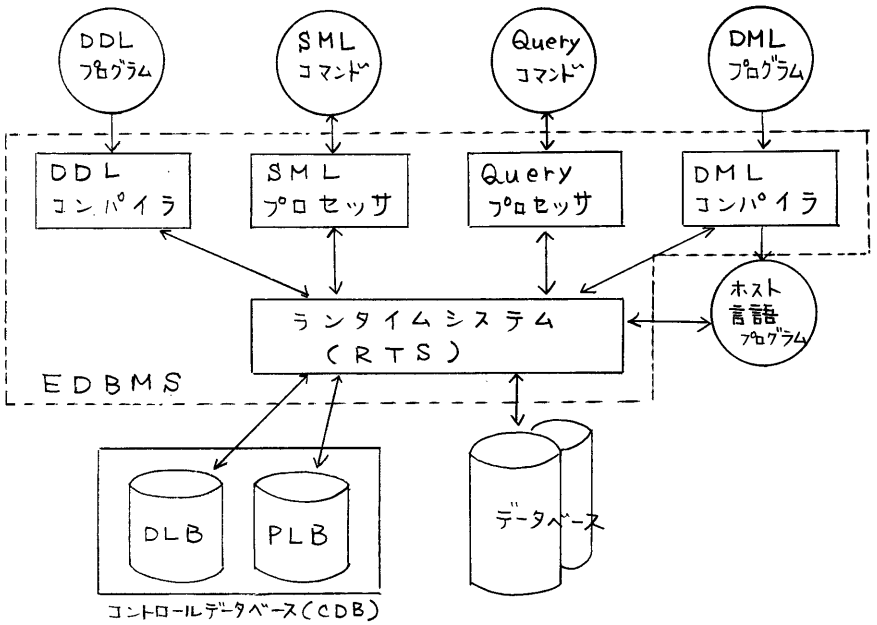

図1　EDBMSの構成

(2) DMLコンパイラ

ホスト言語(Fortran, Cobol など)に組み込まれたデータ操作言語(DML)をホスト言語に変換するプリプロセッサ。チェインの親子関係を利用したデータ検索と、実行時に任意の条件に合うデータ集合を作成する関係モデル的なデータ検索の両方が可能であることが、このDMLの特徴である。

コンパイルされたDMLプログラムは、実行時にランタイムシステムを起動してデータベースにアクセスする。

(3) Queryプロセッサ

ホスト言語を介さずに、端末からの対話的な操作により、RTSを起動し、データベースにアクセスするためのユーザインタフェース。Queryコマンドは、DMLと同様のコマンドを含む。また、コントロールデータベースのプログラムライブラリ(PLB)にあらかじめ登録しておいたプログラムを呼び出して実行するコマンドや、くり返し制御のコマンドがある。

(4) SMLプロセッサ

DDLによって定義されたデータ構造を、端末からの対話的な操作によって変更するためのユーザインタフェース。SMLは、構造操作言語(Structure Manipulation Language)の略称として用いた。SMLの詳細は3章で述べる。

(5) ランタイムシステム(RTS)

RTSは、EDBMSの中核であり、四つのユーザインタフェースからのコ

ントロールデータベースとデータベースへのアクセス要求を実行する。

ＲＴＳには、スキーマテーブル管理、カレンシーテーブル管理、ロック管理、ロジカルアクセス管理、フィジカルアクセス管理などのモジュールがある。外部記憶装置上の物理的なデータ構造の管理や入出力を実行するＲＴＳは、ユーザインタフェースを稼動するための抽象マシンと考えることができる。

（６）コントロールデータベース

データ構造の定義情報（メタデータ）を蓄積するデータ定義ライブラリ（ＤＬＢ）と応用プログラムを格納するプログラムライブラリ（ＰＬＢ）とからなるメタデータベース。

３、ダイナミックデータ構造

ダイナミックデータ構造の概念は、組織や企業の変化、及びアプリケーションの変化に応じてデータベースを徐々に進化発展させていくことができる柔軟なデータベースを実現することである。従来のスタティックなＤＢＭＳでデータ構造の変更を行うには、一般に次のような手順が必要である。

（１）データ構造を定義するＤＤＬプログラムを書き直し、リコンパイルする。
（２）既存のデータをアンロードし、新しいデータ構造に合せてリロードする。
（３）応用プログラムを修正し、リコンパイルする。

ＥＤＢＭＳでは、ＳＭＬコマンドによる対話的な操作によってデータ構造の動的変更を行い、ＤＤＬプログラムの書き直しやリコンパイルの手間を省く。また、実行時の動的な再編成や、再編成の局所化によってデータのアンロード、リロードの省力化を図る。データ構造変更の影響を受ける応用プログラムを選び出し、診断メッセージを発行することにより、応用プログラムの保守を支援する。図２に、ＥＤＢＭＳにおいてデータ構造の変更後、応用プログラムを再実行する場合の処理手順を示す。

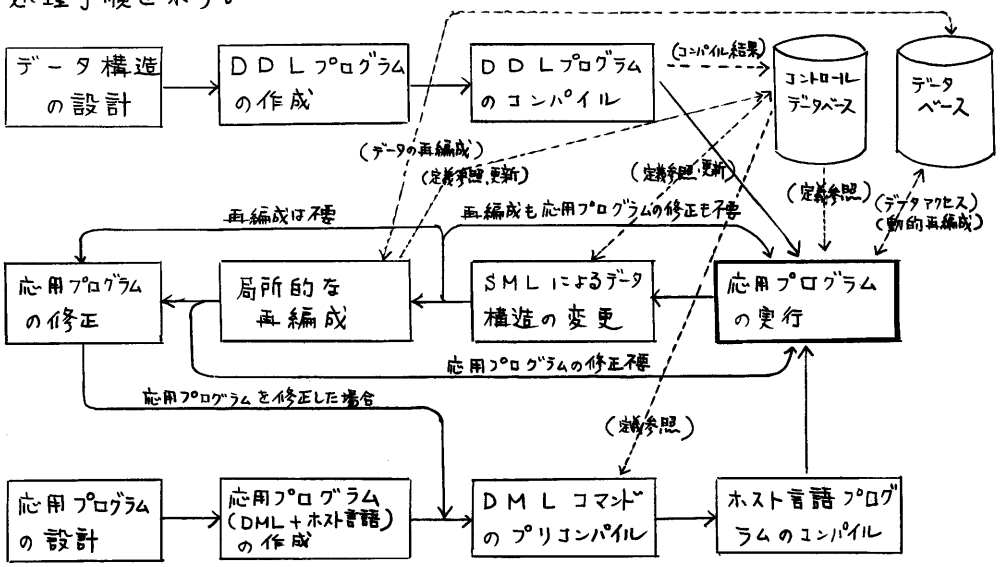

図２、データ構造の変更と応用プログラムの実行

3.1　データ定義のダイナミック宣言

データ構造の定義において変更ができるのは、以下のものである。

（１）データ項目の属性（タイプ，長さ）の変更
（２）レコードタイプへのデータ項目の追加
（３）レコードタイプからのデータ項目の削除
（４）キーの追加、削除
（５）チェイン属性（順序，ソートキー）の変更
（６）スキーマへのレコードタイプ、チェインの追加
（７）スキーマからのレコートタイプ、チェインの削除
（８）データ値の正当性制御の変更

ＤＤＬには、データ構造の定義をダイナミック（可変）な属性とスタティック（固定）な属性とに区別する機能がある。上記の可変なデータ定義属性のうちで使用形態が固定せず将来変更の可能性のあるものを、陽にダイナミックと宣言しておく。ダイナミックと宣言されたデータ定義属性はＳＭＬによって変更することができる。データ定義の属性をダイナミックなものとスタティックなものに区別する理由は、実行時のオーバヘッドの発生をダイナミックな属性をもつデータへのアクセスに限定するためである。ＤＤＬコンパイラがコンパイル結果をコントロールデータベースのＤＬＢに蓄積する時、図３のように、ダイナミックな属性とスタティックな属性に区別する。ダイナミックな属性は、バージョン番号をオーナとするチェインのメンバとして維持される。ＳＭＬによるデータ定義の変更は、そのデータ定義の新バージョンの出現を意味する。ＤＭＬコンパイラが、ＤＭＬコマンドをプリコンパイルする時、スタティックなデータ定義情報のみを応用プログラムと結合する。ダイナミックなデータ定義情報との結合は、応用プログラムの実行時（スキーマのオープン時、またはデータへのアクセス時）に行う。ダイナミックなデータ定義属性は、プリコンパイル後に、変更されることがあるからである。データ構造の動的変更によって実行時のオーバヘッドが生じるが、それはすべてのデータアクセスに対してではなく、ダイナミックと宣言されたデータへのアクセスに対してのみである。

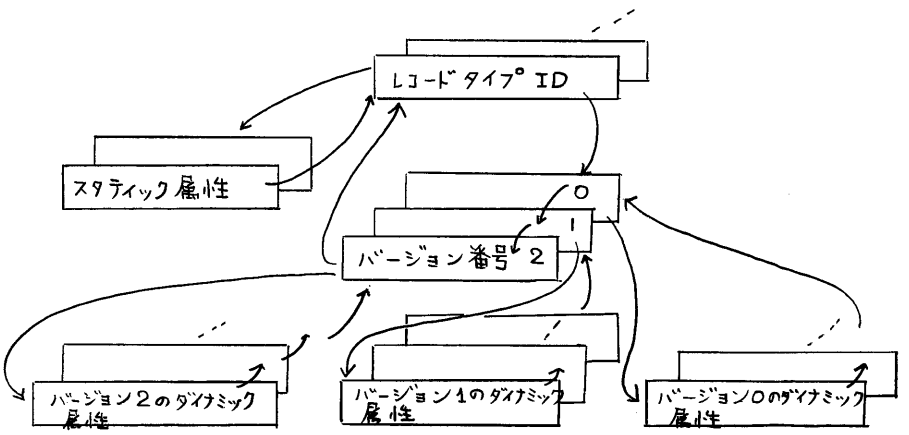

図３、データ定義ライブラリのネットワーク構造
（レコードタイプ定義の例）

図4に、ダイナミックな属性を含むデータ項目とレコードタイプを定義するDDLプログラムの例を示す。大文字はDDLのキーワードであり、小文字はデータ項目、レコードタイプの名称である。データ項目zipcodeのデータタイプはダイナミックと宣言してあるので、データタイプCHARACTERの文字数(長さ)の増減ができる。また、レコードタイプofficeを構成するデータ項目リストがダイナミックと宣言してあることから、新たに別のデータ項目を付け加えたり、既存のデータ項目の削除ができる。

```
DATA  ITEM  DEFINITION
  ITEM: officeid
    TYPE: CHARACTER(4)
  END  officeid
  ITEM: city
    TYPE: CHARACTER(20)
  END  city
  ITEM: zipcode
    TYPE/DYNAMIC: CHARACTER(5)
  END  zipcode
  ITEM: numberofemp
    TYPE: INTEGER
    VALUE  CONSTRAINT: numberofemp > 0
  END  numberofemp
       .
       .
       .
END
```

データ項目定義セクション

```
RECORD  DEFINITION
  RECORD: office
    ITEM  LIST/DYNAMIC: officeid, city,
                zipcode, numberofemp
    KEY: ORDER: PRIMARY  officeid
         RANGE: numberofemp
  END  office
       .
       .
       .
END
```

レコードタイプ定義セクション

図4. ダイナミック属性を持つデータ項目とレコードタイプを定義するDDLプログラム

3.2 SMLによるデータ構造の変更

DDLプログラム中で、ダイナミックと宣言されたデータ定義の属性を変更するためのユーザインタフェースが、SMLである。SMLは、次のような五つの機能をもつ。

(1) DISPLAY ----DLBに蓄積されている最新データ定義を表示する。
(2) CREATE ----データ項目，レコードタイプ，チェイン，スキーマの定義を新たに作成し、DLBに追加する。
(3) MODIFY ----データ定義のダイナミックな属性を変更し、DLB中のデータ定義の新しいバージョンを作成する。
(4) DELETE ----不要となったデータ定義をDLBから削除する。
(5) CONFORM ----データを変更後の最新データ定義に合わせるために、レコードタイプまたはチェインごとに局所的な再編成を行う。

データ定義の変更は、他の関連するデータ定義や応用プログラムに副作用を引き起すことがある。このため、SMLプロセッサはCDBのDLBとPLBを参照することにより、副作用の結果を診断し、データベース管理者やプログラマに報告する機能をもたなければならない。すなわち、

(1) ある定義の変更がもたらす他の定義への影響の診断と、さらに必要となる定義変更の要請。
(2) 定義変更のため、修正、リコンパイルをしなければ実行することができない応用プログラムの抽出。
(3) 定義変更の影響が大きく、局所的な再編成をしなければ応用プログラムの実行ができない場合の警告。レコードタイプ定義のプライマリキーの変更や、チェインの順序がSORTEDである場合のソートキーの変更などの後では、必ずCONFORMによる局所的な再編成が必要である。

現在のプロトタイプEDBMSでは、CREATE，MODIFY機能を実現しているが、他機能については具体的な実現方法を研究中である。図5，6に、MODIFY機能の簡単な実行例を示す。

3.3 局所的なデータベース再編成

データ構造の変更により、DLB中のレコード定義には新しいバージョンが作り出される。データベースにレコードオカーレンスを蓄積する時、常にそのレコードタイプの最新バージョンの定義を使用する。レコード定義とレコードオカーレンスの対応付けをバージョン番号によって行うため、レコードオカーレンスの先頭にはバージョン番号が付加される。したがって、時間の経過とともに、データベース中のレコードオカーレンスとDLBのレコードタイプ定義には、いくつかの異なるバージョンが混在していることがある。(図7の左側)

SMLのCONFORM機能は、指定されたレコードタイプの全レコードオカーレンスを最新バージョンの定義に合うように変換し、再蓄積する。また、不要になる旧バージョンの定義をDLBから削除し、バージョン番号を0に戻す。(図7の右側) チェインに対しては、各チェインオカーレンスへの参加条件(chain constraints)を満たさなくなったメンバの削除や、新しいチェインの順序に従ってメンバの順序のつけ直しなどを行う。これらは、特

```
? RTS ROUTINE NAME
SML/MODIFY  ---------①
WHAT TYPE OF DEFINITION DO YOU WANT TO MODIFY ?
   SCHEMA
   SUBSCHEMA
   CHAIN
   RECORD
   DATA_ITEM
   USER(=USER DEFINED TYPE)
DATA_ITEM  ----------②
WHAT DATA ITEM DO YOU WANT TO MODIFY ?
ENTER ITEM NAME
EVOLNUM2  ----------③
CURRENT TYPE OF THE SPECIFIED DATA ITEM IS
DOUBLE INTEGER ----------
AND ITEM LENGTH IS     4BYTES.--------④
ENTER NEW TYPE :
  INTEGER
  DOUBLE INTEGER
  REAL
  DOUBLE REAL
  COMPLEX
  CHARACTER
REAL  ----------⑤
SML TERMINATED.
```

①SMLのMODIFY機能の実行を指定する

②データ項目の変更を指定する。

③データ項目名を指定する。

④現在のデータタイプと長さが表示される。

⑤新しいデータタイプを指定する。

図5、データタイプ（DOUBLE INTEGERからREALへ）の変更例

```
? RTS ROUTINE NAME
SML/MODIFY  ----------①
WHAT TYPE OF DEFINITION DO YOU WANT TO MODIFY ?
   SCHEMA
   SUBSCHEMA
   CHAIN
   RECORD
   DATA_ITEM
   USER(=USER DEFINED TYPE)
DATA_ITEM  ----------②
WHAT DATA ITEM DO YOU WANT TO MODIFY ?
ENTER ITEM NAME
EVOLCHAR  --------③
CURRENT TYPE OF THE SPECIFIED DATA ITEM IS
CHARACTER ----------
AND ITEM LENGTH IS     7BYTES.--------④
ENTER NEW TYPE :
  INTEGER
  DOUBLE INTEGER
  REAL
  DOUBLE REAL
  COMPLEX
  CHARACTER
CHARACTER  ----------⑤
ENTER NEW ITEM LENGTH
5  ----------⑥
SML TERMINATED.
```

⑥データ項目の新しい長さを指定する。

図6、CHARACTER型データ項目の長さの変更例

定のレコードタイプやチェインに限定した再編成で、データベース全体の再編成でないため、局所的なデータベース再編成ということができる。CONFORMの実行により、応用プログラムの実行時にデータ定義とデータの間の不一致を解決する動的な再編成は不要になり、処理速度は向上する。また、3.2で述べたように、ある種のデータ構造の変更に対しては、CONFORMによる局所的な再編成が必須の場合がある。

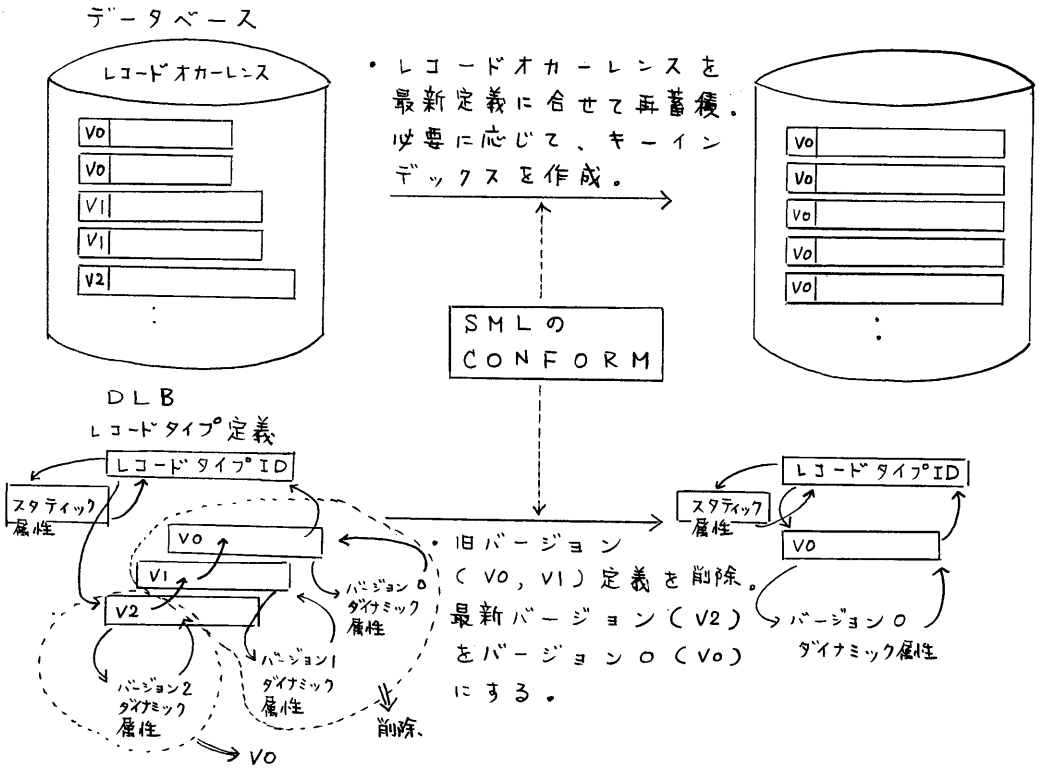

図7. レコードのCONFORM
(V0, V1, V2 はバージョン番号)

3.4 ダイナミックな定義属性の管理と動的再編成

ダイナミックな属性のデータ定義情報が決まるのは、応用プログラムの実行時である。EDBMSのランタイムシステムは応用プログラムの発行するDMLコマンドによって起動されデータベースへのアクセスを実行する。ランタイムシステムの一部であるスキーマテーブル管理(STM)が実行時にデータ定義情報の管理を行う。DMLコンパイラは、応用プログラムをプリコンパイルする時、スキーマ識別番号(SID)とスキーマバージョン番号(SVN)を決定する(図8の①)。応用プログラムの実行は、DMLのOPENコマンドがSTMを起動する(図8の②)ことから始まる。STMは、SIDとSVNをキーとしてDLBを検索し、オープンされたスキーマに関するデータ定義を主記憶上のスキーマ定義テーブル上にロードする(図8の③)。ダイナミックな属性のデータ定義情報の決定はこのスキーマ定義テーブルの参照によって行われる(図8の④)

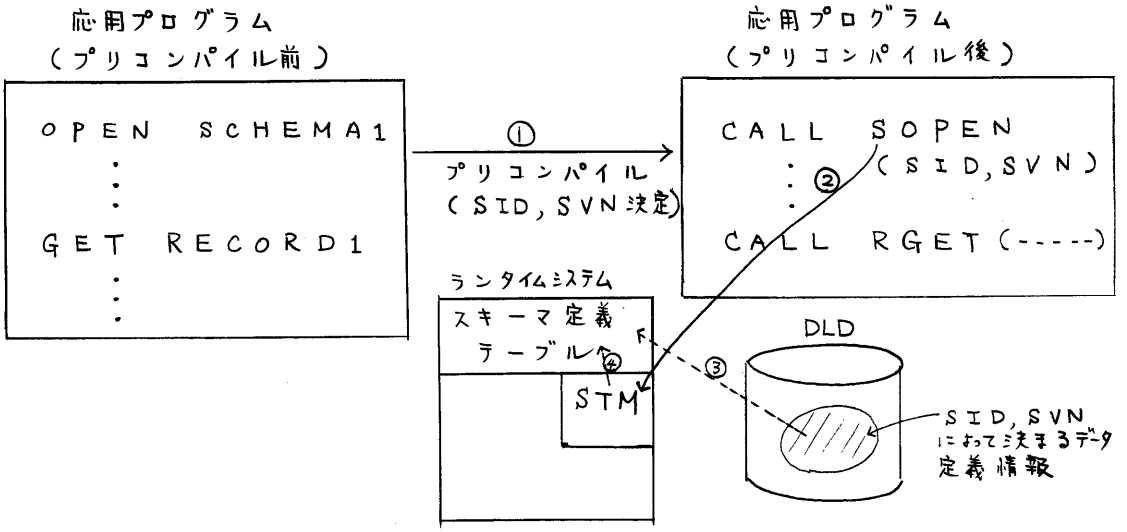

図8、スキーマオープンとスキーマテーブル管理（STM）

スキーマオープン時にロードしたデータ定義のバージョン番号が、検索したデータのバージョン番号と一致しない場合やデータの蓄積と更新後の再蓄積に必要な最新データ定義のバージョン番号と一致しない場合がある。このような場合、ランタイムシステムはSTMを起動し、必要なバージョンのデータ定義をロードする。実行時の動的再編成は、スキーマオープン時にロードしたデータ定義とデータへのアクセス時に必要に応じてロードされるデータ定義との差異を解釈することによって行う。図9は、データ定義に三つのバージョンV0, V1, V2が存在し、V1のデータ定義を利用する応用プログラムがV0のデータを検索、更新後に再蓄積する場合の動的再編成の過程を示すものである。スキーマオープン時には、V1のデータ定義をロードする（図9の①）。検索データがV0である時（図9の②）、V0のデータ定義をロードし（図9の③）、検索データをV0からV1に変換して応用プログラムに渡す（図9の④）。応用プログラムがデータを更新して再蓄積する時（図9の⑤）、最新バージョンであるV2のデータ定義をロードする（図9の⑥）。データは、V1からV2に変換されてデータベースに蓄積される（図9の⑦）。

データの蓄積を常に最新のデータ定義に合せて行うのは、データベース中のデータを応用プログラムの実行中に、徐々に最新のデータ定義に合うようにしていくためである。応用プログラムの実行を停止し、旧定義にもとづいて蓄積されているデータをアンロードした後、最新定義に合せてリロードするというオフライン再編成は不要になる。

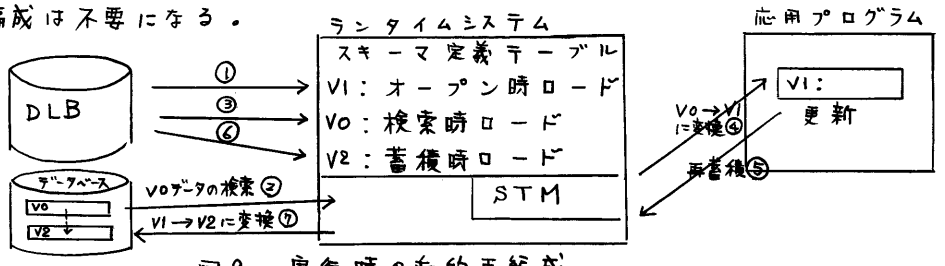

図9、実行時の動的再編成

4、 まとめ

データ定義の属性をダイナミックなものとスタティックなものに分離し、ダイナミックな属性をSMLによって動的に変更することができる柔軟なデータベースの構築ができる。データ構造が柔軟性をもつことに加えて、DMLコンパイラのオプティマイズ処理（ダイナミック定義かスタティック定義かによって生成するコードを変える処理）、CONFORMによる局所的なデータベース再編成、実行時の動的再編成などの機能を実現する。また、データ定義、データ、応用プログラムの一貫性の管理のために、コントロールデータベース（CDB）と呼ぶメタデータベースを導入した。図10に、ダイナミックデータ構造に関する処理をまとめる。

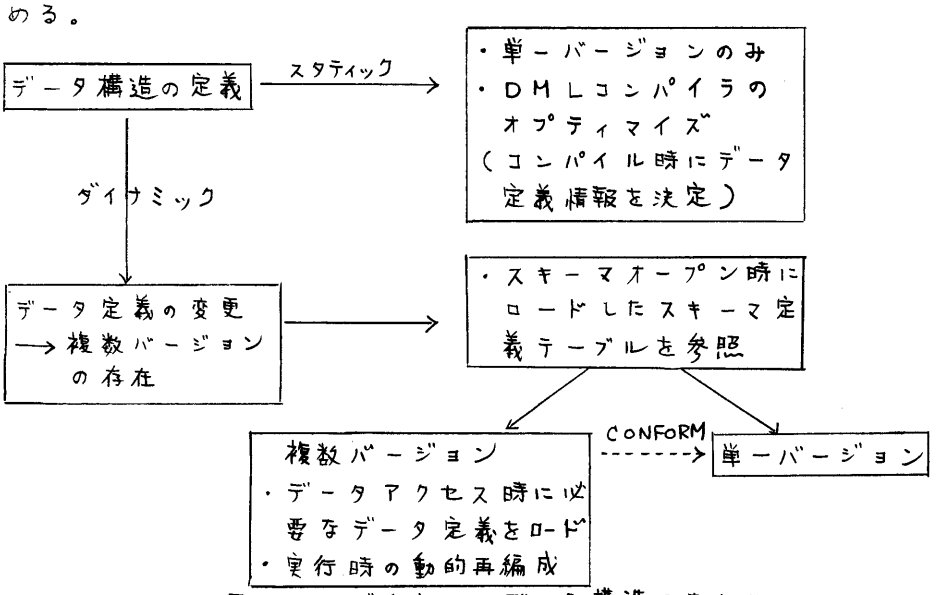

図10、ダイナミックデータ構造のまとめ

謝辞 本研究に対して御指導頂いたJ.C.Browne教授（テキサス大学）、國井利泰教授（東京大学）、國井秀子氏に感謝致します。

参考文献

[1] Kunii, T.L., Browne, J.C., and Kunii, H.S. : An architecture for evolutionary data base system design, Proc. IEEE international conference, Chicago (1978.11)

[2] 村上国男、森道直、中野良平：大規模データベースとその実現技術，情報処理，VOL.23，NO.10，PP.955－961（1982）

[3] 酒井博敬：データベース技術応用の過去・現在・未来，情報処理，VOL.23，NO.10，PP.893－897（1982）

[4] Date，C.J.：An Introduction to Database System，Third Edition，Addison-Wesley，Mass.（1981）

[5] Date，C.J.：An Introduction to Database System II，Addison-Wesley，Mass.（1982）

[6] Browne，J.C.，et al：A Conceptual Design For the Evolutionary Data Base Management System，Austin，Texas（1979.9）

[7] 日立SK，IRA社：Functional and Data Structure Definitions For the Run Time System，Austin，Texas（1981.6）