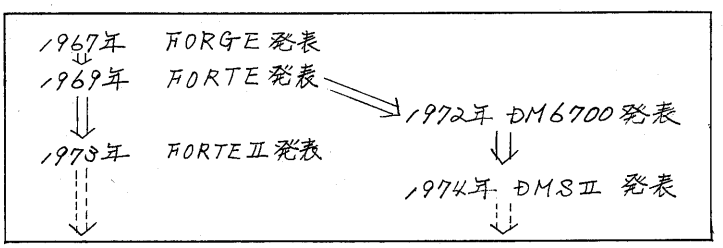
ＤＭＳⅡについて

前田耕一（バロース株式会社）

1．ＤＭＳⅡの位置づけ

ＤＭＳⅡは、バロース社のデータベース管理システムであり、表1.1に示すデータベース・ソフトウェアに関する豊富な経験と実績とを基に6年の歳月をかけて開発し、1974年に発表した。



〔表1.1〕データベース・ソフトウェアの歴史

使用可能機種はB1700，1800，2800，3800，4800，6700，6800，7700，7800シリーズで、バロース社の小型機から超大型機まですべての機種に渡り統一された仕様により提供されている。このＤＭＳⅡの特徴を一言でいうなら、データベース管理ルーチンをオペレーティング・システムに統合するという従来とは異なったアプローチを採用したことである。このため常駐モニターを使用する比較的、伝統的なアプローチを採用していたDM6700に比べて8倍以上の効率向上を実現することができた。このようなアプローチは、オペレーティング・システムを提供しているコンピュータ・メーカーこそが実現できることであり、統一された思想と効率の高いＤＢＭＳを実現するための有力な一方法となるであろう。

ＤＭＳⅡの1977年1月現在の全世界に於ける使用実績は約240社で、米国Data Pro Research Corp. 発行の"Data Pro 70"に於いても表1.2.に見られるような高い評価を得ている。特に処理能力／効率，全体的満足度、導入の容易性，使い易さで高い評点が与えられていることが注目される。

	優	良	可	不可	加重平均＊
全般的満足度	4	1	0	0	3.8
処理能力／効率	5	0	0	0	4.0
導入の容易性	3	2	0	0	3.6
使い易さ	3	2	0	0	3.6
ドキュメンテーション	1	2	2	0	2.8
メーカー・サポート	2	1	2	0	3.0
教育	1	4	0	0	3.2

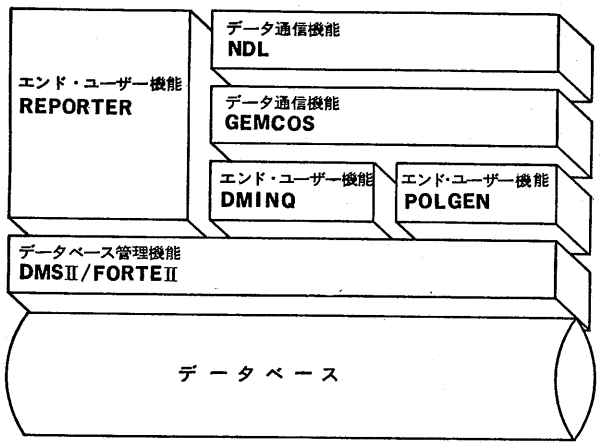
＊加重平均は優(4)，良(3)，可(2)，不可(1)とする。

〔表1.2.〕ＤＭＳⅡの評価

バロース社では、さらにこのＤＭＳⅡを単なるＤＢＭＳとして捉らえるのではなく、情報システム全体をＢＤＭＳ（Burroughs Data Management System）と呼ぶソフトウェア体系として

捉え、その中でのデータベース管理機能としてDMSIIを位置づけた。ここでBDMSの概要について簡単にふれておくことにする。

BDMSでは、整理整頓された情報を、何処からでも、(正当な権利を持った利用者なら)誰れでも、何時でも利用できる情報システムを実現するため、図1.1に示すように、この情報システムに必要な機能を3つに体系化し、それぞれの機能に対応するソフトウェアを開発した。バロース社では、この統一されたBDMSを、小型機から超大型機まですべての機種で提供している。



○データベース管理機能
- FORTEII (File Organization TechniqueII)
  中・小規模のデータベース・システムを実現するのに適したデータベース管理システム。
- DMSII(Data Management System II)
  大・中規模のデータベース・システムを実現するのに適したデータベース管理システム。

○データ通信機能
- NDL(Network Definition Language)
  簡単なNDL言語により，通信制御用プログラムが作成できる。
- GEMCOS (Generaliged Message Control System)
  簡単にトランザクション指向のメッセージ制御システムが作成できる。

○エンド・ユーザー機能
- REPORTER(Report Writer)
  簡単なレポート言語により，バッチ処理で報告書を作成できる。
- DMINQ (Data Management Inquary Facility)
  簡単にデータベースの情報照会ができる。
- POLGEN (Problem Oriented Language Generator)
  エンド・ユーザーの言葉で，データベースを利用するための言語が開発できる。

〔図 1.1〕 BDMSのソフトウェア体系

2. DMSIIの概要

2.1 DMSIIの設計目標

DBMSは、今日ある各種のソフトウェアの中でも、特に高度な設計思想とソフトウェア技術を要求されるものの一つである。バロース社では、このDMSII開発に当り、次のような項目をその設計目標として開発した。

(1) データベース記述とデータ操作については、ユーザー・インタフェースをできる限り単純化し、容易なものとする。

(2) アプリケーション・プログラマが、データ管理の責任を一切負うことなく、かつ、できる限りプログラミング負荷を軽減できるよう、プログラム内でのデータ定義を不用とし、データの利用はその論理構造のみで行なえるようにする。

(3) データとプログラム間に完全な独立性を保持するため、アプリケーション・プログラム内にはデータを構造化し、アクセスするどのようなコードも持込ませない。

(4) データベースの共同利用を促進するため、多くのアプリケーション・プログラムを多重プログラミング／多重プロセスのもとで、バッチ／オンラインの区別なく、かつ、安全に稼動させる。（このことを実現するためには、完全なレコード・レベルでの更新保護機能がなければならない。）

(5) データベースの更新記録と障害回復はシステムの基本機能であり、障害時のユーザー負担を最小にし、データベース利用者へのサービスを停止しないよう 障害回復はできる限り自動化するとともに、障害部分の局所化を行い、他の正常な部分は利用可能にしておく。

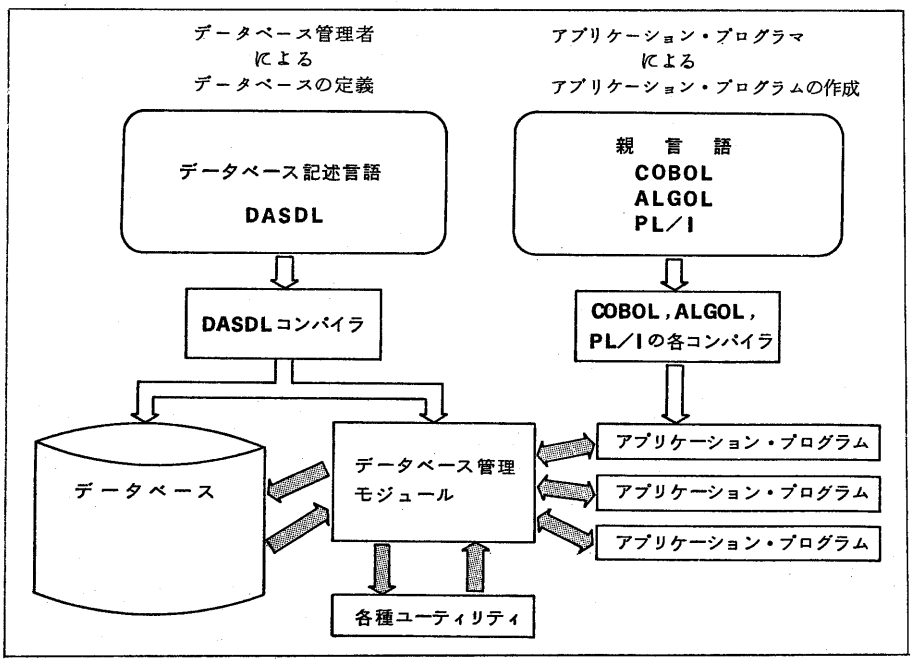
(6) ユーザーの立場を考慮し、効率とコストに重要な影響を与えるスペース（資源）と処理時間に関する変数は、ユーザーにその選択権を与える。

(7) 提供するソフトウェアは、保守性、拡張性を考慮して、すべてハイレベル言語を採用する。

(8) 最高の効率を上げるため、コンピュータ、アーキテクチャとオペレーティング・システムの機能を最大限に利用する。

2.2 DMSⅡの構成要素

DMSⅡは、データベース管理者（DBA）がデータベースを定義するためのDASDL，アプリケーション・プログラマがデータベースを利用するのに使用するCOBOL・ALGOL・PL／Iなどの言語と、データベースを総合的に管理するMCPの中のデータベース管理モジュール，データベースを障害から回復したり、再編成したりするユーティリティのライブラリから構成される。



〔図2.1〕 DMSⅡの処理フロー

3. DMSIIによるデータ構造の表現

DMSIIでは、データ構造を表現するため、データ集合を表現するデータセットと、それらのデータ関連を表現するセット、サブセットより構成される。

3.1 データセット

データセットは、従来のファイルの概念を拡張したもので、論理的に同じ属性を持つデータの集りである。（但し、バリアブル・フォーマット・レコードとして異なる形式のレコードを1つのデータセットに格納することもできる。）このデータセットはプログラムとデータベース管理モジュールの間の情報交換の単位となるレコードより構成される。

3.1.1 レコード記述

レコードは、COBOLのレコード記述と同様にデータ項目の並びから成る。その項目には、基本データ項目、集団項目，制御項目，データセット項目、セット項目、サブセット項目等を記述することができる。データの種類としては、英数字、数値、論理値等がある。

3.1.2. データセットの記憶構造

データセットは、そのデータ特性、ディスク・スペース使用効率とアクセス効率を考慮して次の記憶構造を選択することができる。

データセットの記憶構造	キー値による乱次アクセス	キー値による順次アクセス	物理的配列による順次アクセス
スタンダード (standard)	×	×	○
アンオーダード (unordered)	×	×	○
オーダード (ordered)	○	○	○（論理的配列と同じ）
ランダム (random)	○	×	○
ダイレクト (direct)	○	×	○

〔表 3.1〕データセットの記憶構造

3.2 セット

セットは、データセットのすべてのレコードへの論理的なアクセスの経路である。セットは、あるキーの値に従って特定レコードをアクセスしたり、キーの順番に従ってレコードを昇順あるいは降順にアクセスすることができる。1つのデータセットには、必要に応じて多数のセットを指定することができ、しかも、実行時にはデータセットのレコードの追加、削除，変更に応じて自動的にすべてのセットのメンテナンスが行なわれる。このように、セットはデータの多面的活用を容易に実現することができる。

3.3 サブセット

セットがデータセットのすべてのレコードを対象とする経路であったのに対し、サブセットはデータセットの中のある特定の属性をもったレコードだけ（たとえば、全顧客に対する主要顧客や不良顧客など）を対象とする経路である。このことにより、データセットのレコードを類別化して利用することができ、アクセス効率も向上させることができる。データセットと同様に1つのデータセットに対し必要な数だけサブセットを指定することができる。サブセットには、サブセットへの参入条件を指定し、実行時にレコードの内容により自動的にサブセットのメンテナンスを行なう自動サブセットと、参入条件を指定せず、実行時にデータベースの操作動詞によりメンテナンスを行う手動サブセットがある。

3.4 セット／サブセットの記憶構造

セットやサブセットも、そのデータ特性、ディスク・スペース使用効率とアクセス効率を考慮して次の記憶構造を選択することができる。

•キーを持つセット，サブセットの記憶構造

セット/サブセットの記憶構造	キーによる乱次アクセス	キーによる昇順/降順アクセス	主な用途
索引順次 (index sequential)	○	○	独立セット，独立サブセット
索引乱次 (index random)	○	×	独立セット
オーダード・リスト (ordered list)	○	○	従属セット，従属サブセット

•キーを持たないセット，サブセットの記憶構造

セット/サブセットの記憶構造	アクセスの特徴	主な用途
アンオーダード・リスト (unordered list)	特定の属性を持つレコードをグループ化してアクセス可	従属セット，従属サブセット
ビット・ベクトル (bit vector)	論理演算による複合条件での素早いアクセス可	独立サブセット

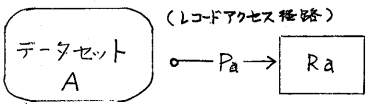
〔表 3.2〕セット／サブセットの記憶構造

3.5 独立／従属構造

データセットのレコード記述では、通常のデータ項目と同じレベルでデータセットやセット、サブセットを記述することができる。このように、レコード記述に含まれるデータセットやセット，サブセットを従属構造といい；それぞれ従属データセット、従属セット／サブセットと呼ぶ。反対にレコードとは独立して記述されているものを独立構造といい、独立データセット、独立セット／サブセットと呼ぶ。この構造により、DMSIIでは、階層構造やネットワーク構造など自由なデータ構造を表現することができる。

3.6 データ構造の表現

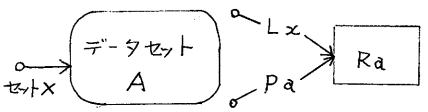
(1) データ関連を持たないデータセット



Pa ; Aデータセットの物理的配列に従うアクセス
Ra ; Aデータセットのレコード
○ ; アクセスの始点

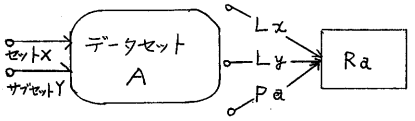
(2) セット／サブセットを持ったデータセット

①



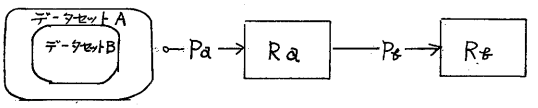
Lx ; Xセットの論理的な経路

②

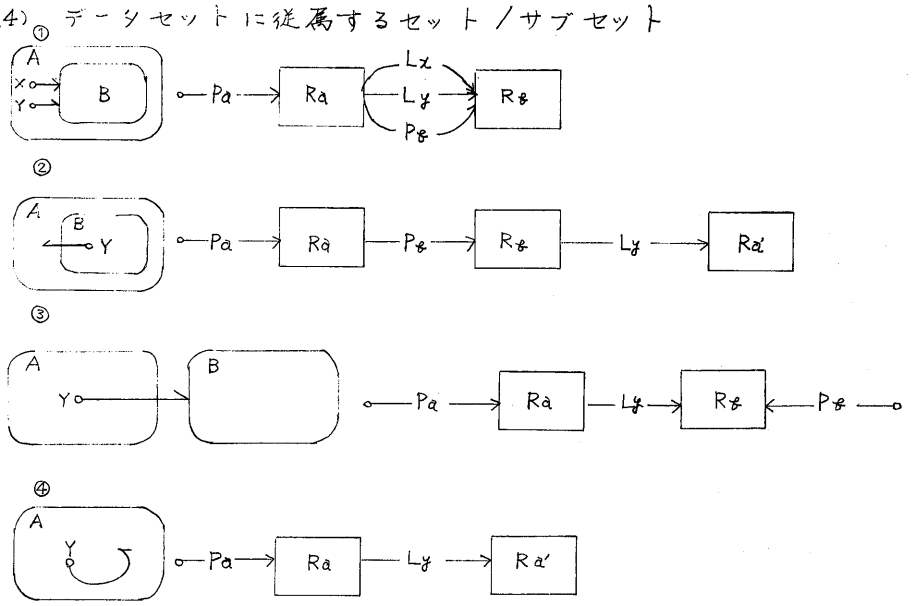


Ly ; Yサブセットの論理的な経路

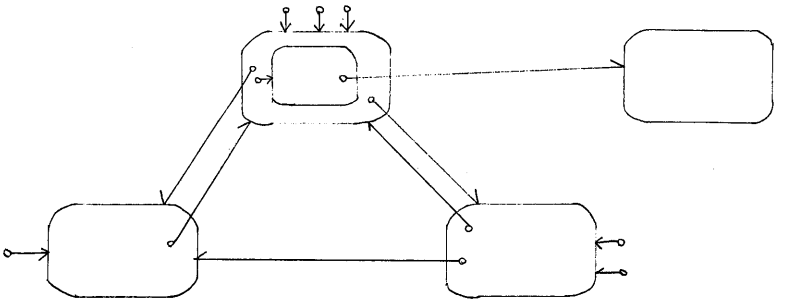
(3) データセットに従属するデータセット



(4) データセットに従属するセット／サブセット



(5) データセット，セット，サブセットの応用



4. DASDL (Data And Structure Definition Language)

データベースを構成するすべてのデータ構造、使用資源の種類と容量等に関する記述はDASDL言語で行う。このDASDL言語は自由書式のハイレベル言語であるため理解しやすく、記述しやすい。

4.1 DASDL記述

(1) データ構造

データ構造の記述は、データセット，セット等の論理記述と、その記憶構造である物理構造の記述に分けて記述できる。しかも、記述のわずらわしい物理構造は記述しなくてもDASDLコンパイラが適切な値を算出，決定してくれるため，データベースの初期設計段階に於けるDBAの負担が軽減され、生産性が向上する。

(2) 使用資源の種類と容量

データセット，セット，サブセット単位にその格納媒体（アクセス・タイム 5mSの高速固定ヘッド・ディスク装置から 42.5mSの大容量ディスク・パック装置まで多くの記憶媒体が選択できる。）と容量を指定できる。また、主記憶装置内でのデータベース・バッファの大きさ等も指定できる。このようにDBAがスペース（資源）と処理時間を考慮して

細かくシステムの最適化を行うことができる。

(3) システムによる自動正当性検査

データベースに格納するデータの正当性検査は、DBAがDASDLでその条件を一度記述しておけば、データ格納時にデータベース管理モジュールにより常にそのデータの正当性検査が行なわれる。このためデータベースの信頼性が向上すると共に、プログラミング効率も向上する。データの正当性検査には次のものが用意されている。

- 新規レコードの格納時に必らずデータ項目が存在しなければならない。—REQUIRED
- データ項目の空値を指定できる。— NULL
- 新規レコードの格納時に、データの初期値を指定できる。— INITIAL VALUE
- 新規レコードの格納時に、データの正当性検査を実行できる。—VERIFY

〔例〕

```
STOCK-ITEMS DATA SET
        (ITEM-NO          ALPHA  (6) REQUIRED;
        LIMIT             NUMBER (5) NULL IS 0, REQUIRED;
        WAREHOUSE         ALPHA (2) INITIAL VALUE IS BLANKS;
        ON-HAND           NUMBER (5);
        REORDERED         BOOLEAN);
        VERIFY  (ON-HAND GTR LIMIT OR REORDERED);
```

(4) データのアクセス権

データベースの利用者単位に利用可能なデータベースの範囲を規定する論理データベースと再写像を記述することができる。このことにより、データベース利用者に最適な情報が提供されると同時に、データの機密保護やデータの独立性が実現される。

- 論理データベース (LOGICAL DATA BASE)

論理データベースは、データベース利用者に最適なデータ構造を提供するため、データセットやセットを再定義して必要なデータ構造のみをとりだすことができる。1つのデータベースに対し、必要な数だけ論理データベースを記述することができる。

〔例〕 DASDL記述

```
A DATA SET
   (A1 ……;
    A2 ……;
    A3 ……;
    ABS SUBSET OF B;
    ACS SUBSET OF C;);
A1S SET OF A KEY IS (A1);
A2S SET OF A KEY IS (A2,A3);

B DATA SET
   (B1 ……;
    B2 ……;);
B1S SET OF B KEY IS (B1);

C DATA SET
   (C1 ……;
    C2 ……;
    CAS SUBSET OF A;);
C1S SET OF C KEY IS (C1);

LDB1 DATABASE  (A (SET A1S), B (ALL));
LDB2 DATABASE  (A (ALL), C (NONE));
```

- 再写像 (REMAP)

独立データセットのレコードは、集団項目、基本データ項目、従属データセット、従属セット、従属サブセット等から構成されるが、再写像によりデータベース利用者にこの中から必要な項目のみをもった固有なデータセットを提供することができる。再写像で

は、このような必要項目の指定のみでなく、項目単位にその項目の更新を禁止したり（READ ONLY）、対象となるデータセットに特定レコードのみ格納を許したり、参照を許したりすることができる。1つのデータセットに対し、必要な数だけ再写像を行なうことができる。

〔例〕DASDL記述

①

```
X DATA SET
   (X1  NUMBER (10);
    X2  ALPHA  (20);
    X3 DATA SET
        (X31  NUMBER    (6);
         X32  ALPHA    (10););
    X4  ALPHA  (30);
    X5 SUBSET OF Y ;);

Y DATA SET
   (Y1  NUMBER  (6);
    Y2  BOOLEAN;
    Y3  ALPHA  (50););

A REMAPS X (X1; X2 READONLY; X4;);

B REMAPS X (B1=X1; B2=X3; B3=X4;);

C REMAPS X (X1; X4; X2; X5;);
```

②

```
D DATA SET
   (D1 … ;
    :
    Dn … ;);

A REMAPS D (D1; …Dn;) SELECT(CLASS > 200 AND CLASS < 300);

B REMAPS D (D1; …Dn;) VERIFY(CLASS > 250 AND CLASS < 400);
```

(5) データベースの障害回復

データベースの障害には、データベースを格納している記憶媒体の障害、データベースを更新中のプログラムの異常終了による障害、同じく更新中のシステムの異常停止による障害等がある。DMSⅡでは、これら各種の障害に応じた回復機能を提供している。DASDLでは、データベースを障害から回復するのに必要な資源と時間に関する変数を記述することができるためシステムに最適な障害回復を行うことができる。

(6) データベース利用状況の報告

データベースの利用状況の報告（STATISTICS）を指定すると、データベース利用状況に関する統計資料が作成される。この資料の中には、データベース管理モジュールの使用資源、データセット，セット，サブセット単位の入出力時間、データ操作動詞の実行回数などが含まれているため、DBAが適切なシステムの運営管理を行なうことができる。

(7) データベースの再編成

データベースの変更、拡張を記述すれば、必要に応じてデータベース再編成プログラムが生成される。このことによりデータベースの柔軟性と拡張性が保証される。

4.2 DASDL記述例

• DASDL記述リスト

```
KOKYAKU DATA SET
  ( KOKYAKU-CODE              NUMBER   (6);
    KOKYAKU-MEI               ALPHA    (20);
    JUUSHO                    ALPHA    (40);
    TORIHIKI-SOOGAKU          NUMBER   (10);
    SAIKEN-SOOGAKU            NUMBER   (S9);
    SAIKEN-GENDO-GAKU         NUMBER   (10);
    GENZAI-TORIHIKI SUBSET OF TORIHIKI
      KEY IS(TORIHIKI-BI, TORIHIKI-CODE);
  ) POPULATION=1000;
CODE-ACCESS SET OF KOKYAKU
    KEY IS(KOKYAKU-CODE);
MEI-ACCESS SET OF KOKYAKU
    KEY IS(KOKYAKU-MEI) DUPLICATES;

TORIHIKI DATA SET
  ( TORIHIKI-CODE             NUMBER   (6);
    TORIHIKI-BI GROUP
      ( TORIHIKI-YY           NUMBER   (2);
        TORIHIKI-MM           NUMBER   (2);
        TORIHIKI-DD           NUMBER   (2););
    TORIHIKI-RYO              NUMBER   (8,2);
    TORIHIKI-TYPE             NUMBER   (3);
  ), POPULATION = 100000;
TORIHIKI-ACCESS SET OF TORIHIKI
    KEY IS(TORIHIKI-CODE);

INITIALIZE;
```

• 記述内容の説明

顧客マスタ・ファイルの記述
- 顧客レコードの記述
  - 顧客コードの記述
  - 顧客名の記述
  - 顧客住所の記述
  - 取引総額の記述
  - 債権総額の記述
  - 債権限度額の記述
  - 現在取引データの記述（取引データへのサブセット）
    - 取引データ検索のためのキーの記述
- 顧客数の記述

顧客コードによる顧客マスタ・ファイルの検索手段の記述
検索キーの記述

顧客名による顧客マスタ・ファイルの検索手段の記述
（DUPLICATES；同姓同名であってもよいとの記述）

取引データ・ファイルの記述
- 取引データ・レコードの記述
  - 取引コードの記述
  - 取引の年月日の記述
    - 取引年の記述
    - 取引月の記述
    - 取引日の記述
  - 取引量の記述
  - 取引タイプの記述
- 取引件数の記述

取引コードによる取引データ・ファイルの検索手段の記述

データベースをディスク上に初期化させる記述

5. ホスト言語

DMSⅡでは、データベース操作機能を拡張したCOBOL、ALGOL、PL/Iをホスト言語として用いることができるため、プログラマの教育期間を大巾に削減できる。また多くのDBMSで採用されているプリコンパイラ方式ではなく、通常のコンパイラを拡張してこのことを実現しているためプログラム開発時、メンテナンス時の負担が軽減される。

5.1 COBOLのデータベース操作動詞

データベース操作動詞として以下の動詞を用いることができる。

• データベース操作動詞一覧とその機能概略

OPEN	：データベースの利用開始と利用形態の宣言	CLOSE	：データベースの利用終了の宣言
FIND	：データセットのレコード検索	MODIFY	：レコードの検索とそのレコードのロック
STORE	：データセットにレコードを格納	DELETE	：データセットからレコードを削除
CREATE	：レコード・エリアの初期化	RECREATE	：レコード・エリアのデータ項目以外を初期化
INSERT	：手動サブセットにエントリを挿入	REMOVE	：手動サブセットからエントリの除去
LOCK	：MODIFYと同じ	FREE	：レコードのロックを解除
SET	：検索すべきレコードの位置決め	GENERATE	：ビット・ベクトル・サブセットの生成
IF	：論理項目の条件判断		

BEGIN-TRANSACTION, END-TRANSACTION ： 障害回復単位の設定

5.2 COBOLプログラム記述例

```
IDENTIFICATION        DIVISION.
ENVIRONMENT           DIVISION.
INPUT-OUTPUT          SECTION.
FILE-CONTROL.
   SELECT INOUT-FILE ASSIGN TO REMOTE.
DATA                  DIVISION.
FILE                  SECTION.
FD INOUT-FILE.
01 IN-RECD.
  03 I-KOKYAKU-MEI    PIC X(20).
01 OUT-RECD.
  03 TORIHIKI-GAKU    PIC 9(10).
  93 TORIHIKI-CODE    PIC 9(6).
  03 TORIHIKI-BI       PIC 99/99/99.
  03 TORIHIKI-RYO      PIC 9(8).
  03 TORIHIKI-TYPE     PIC 9(3).
DATA-BASE             SECTION.
DB KOKYAKUDB.
01 KOKYAKU.
01 TORIHIKI.
WORKING-STORAGE       SECTION.
01 NOTFOUNDMESSAGE    PIC X(10) VA "NOT FOUND".
01 ENDMESSAGE         PIC X(34) VA "NO MORE RECORD,KEY IN NEXT OR END".
PROCEDURE             DIVISION.
START.
   OPEN UPDATE KOKYAKUDB.
   OPEN I-O INOUT-FILE.
LOOP1.
   READ INOUT-FILE INVALID STOP RUN.
   IF I-KOKYAKU-MEI = "END" GO TO FIN.
   FIND KOKYAKU AT KOKYAKU-MEI = I-KOKYAKU-MEI ON EXCEPTION
     IF DMSTATUS (NOTFOUND)
       WRITE OUT-RECD FROM NOTFOUNDMESSAGE GO TO LOOP1.
   MOVE TORIHIKI-SOOGAKU TO TORIHIKI-GAKU.
LOOP2.
   FIND TORIHIKI VIA NEXT GENZAI-TORIHIKI ON EXCEPTION
     IF DMSTATUS (NOTFOUND)
       WRITE OUT-RECD FROM ENDMESSAGE GO TO LOOP1.
   MOVE CORR TORIHIKI TO OUT-RECD.
   WRITE OUT-RECD.
   GO TO LOOP2.
FIN.
   CLOSE INOUT-FILE.
   CLOSE KOKYAKUDB.
   STOP RUN.
END-OF-JOB.
```

参考文献

(1) BDMS概説書

(2) DMSII概説書

(3) B6700/B7700 DMSIIデータベース記述言語解説書

(4) B6700/B7700 DMSII親言語インタフェイス解説書

(5) B1700 DMSIIリファレンス・マニュアル