講　　座

データ・ベースの管理技術 (II)

上　條　史　彦†

2.3 データ独立性

データ・ベース・システムに望まれる最も重要な機能はデータ独立性を完成させることであろう．GUIDE-SHARE 合同のデータ・ベース所要機能調査グループの報告書††によれば，“わずかのように見えるソフトウェアの改造作業に巨額の費用がかかり，データ処理の能力に与える影響が無視できなくなってきている．フィジカル・データの形式，構造がアプリケーション・ロジックに与える影響をなくすれば，費用の増加傾向を逆転するのに大きな役割を果たすであろう．”それゆえにデータ独立性は重要だとして，達成目標の第一にこれをかかげている．

2.3.1 記述言語と操作言語

データ独立性を具体化する方法は，ファイルおよびデータの記述をアプリケーション・プログラムから外し，そこだけ独立した形でプログラミングおよび実行ができるようにすることである．CODASYL DBTG のことばを借りればデータ記述言語†††とデータ操作言語††††を分離することであろう．事実，ほとんどのデータ・ベース・システムが両者を別体系としている．前出の IMS/360 では DBD (data base description) と称するファイル目録を有し，これの保守・追加・新製などはデータ検索とは全く別の手立て (utility program) で行なうようにしている．

データ記述言語には2種の使い方がある．データ・ベースを登録する場合と，アプリケーション・プログラムの中で論理構造を指定する場合である．

データ・ベースの登録にあたってはプログラム言語の形にしなくとも，ユーティリティ・プログラムで十分であるが，後者においてはデータ操作言語と同体系にした方が，1回の翻訳作業で足りるので有利である．登録すべき内容には次の項目がある．

データ・ベース名
フィジカル・ファイル (データ・セット) 名
ページまたはエリア名
セグメント名と上位セグメント (セット関係)
アドレス・ポインタ名
フィールド名と長さ，属性，セグメント内での位置
アクセス・メソッドと入出力関係の属性

データ・ベースの登録には専用のファイルを用意する場合が多い．DBMS† は登録ファイルをもとにアクセス管理情報をまとめたテーブルを作る．このテーブルには機密保護，排他制御，使用統計などが含まれる．データ・ベースは使用される部分だけが指定に従って OPEN されるので，テーブルはそれ自身かなり複雑なデータ構造となる．部分的な OPEN に加えて，大きなフィジカル・ファイルの場合には部分的なマウントが必要となる．たとえば磁気ディスク5台にまたがるファイルで2台だけの内容を必要とする時，必要な部分を含むものだけがオンラインとなっていればよいようにするのである．部分 OPEN，部分マウントは共に DML，DDL の分離を進める重要な前提技術といえる．

アプリケーション・プログラムの中でのデータ記述はコンパイラの翻訳作業をやさしくすることが大きな目的である．COBOL における DATA DIVISION，PL/I における DCL の機能そのままといえる．もちろん記述内容は論理構造であるが，必ずしも1個のデータ・ベース記述（すなわち1個の DDL 記述）の範囲におさめる必要はない．いくつものデータ・ベース

† 日本アイ・ビーエム(株)

†† Data Base Management System Requinements, Joint GUIDE-SHARE Data Base Requirements Group, November 1970.

††† データ記述言語 (data description language) データ・ベースの論理・記憶構造，各部の名称・属性を詳細に規定・登録する言語．略称 DDL．必ずしもコンパイラ言語である必要はなく，パラメトリックな方法も用いられる．

†††† データ操作言語 (data manipulation language) データ・ベースに対し追加・削除・変更・探索などの作業を行なう時，その手順を指定する言語．略称 DML．

† DBMS (data base management system) データ・ベース管理システム．

の部分的な使用が許されるべきであるし，ロジカル・データ・ベースの例のように，いくつものデータ・ベースにまたがった論理構造も使用できるに越したことはない†.

DBMS に要求される一つの大きな機能は多種プログラム言語とのすり合わせ，すなわちランゲージ・インターフェース††で，DDL はその重要な一翼をになうものである.

データ操作言語，すなわち DML はアプリケーションプログラムの手順記述に用いられるもので，2種に大別される．親言語方式と独立言語方式である.

親言語としてはアセンブリ言語，COBOL, PL/I, FORTRAN のいずれもが使用されている．アセンブリ言語にマクロ命令を追加してチェイン・ファイルを処理した BOMP†††，輪構造を論理構造として採用し，COBOL を拡張して磁気テープ・ファイルに階層データ・ベースを可能にした IDS††††，階層木構造を扱うアクセス方式としての DL/I††††† などが先駆者であろう．BOMP はアセンブリ言語，後2者は COBOL を親言語としたわけだが，考え方に言語仕様の拡張という方式 (IDS) と連絡文 (CALL 文など) を用いて親言語の文法のわくの中で処理する方式との2種類がある．CALL 方式は同じアクセス・メソッドで多種言語と連結できる利点を有する．COBOL と PL/I はデータ構造の取り扱いができる言語なので，共通のアクセス・メソッドを用いて共用データ・ベースを使用できる．一方，文法を拡張する方式は，すでに高水準言語に慣れたプログラマにとっては使いやすく，言語体系も統一がとれる．しかしながらコンパイラは全く異なった能力を追加せねばならず，全面的作り換えが必要である．主要な高水準言語が標準文法 (ISO standards) をもちつつある現在，文法の拡張には慎重を要する．CODASYL DBTG の作業はこうした事実をふまえたものであって，COBOL の分野でのデータ・ベースの将来を暗示するものといえよう．もちろんプログラム言語の標準にこだわる限り，異言語の間でのインターフェースを作るのは容易ではないので，各種言語に共通するデータ・ベース・システムができるかどうかは拡張方式では不確かといわざるを得ない.

独立言語方式ではフォーマット系† と高水準言語系が市場にだされている.

フォーマット系のシステムは結果としてどのような表が欲しいのか，個々の項目はどんな手順で作りだすのかを指定することによってプログラミングに替えるものである．現在は原始的なていさいのシステムが多いが，問題向き言語への指向をもつ．入力もし易く，プログラミングもやさしい．欠点は当初システム設計時に考えられなかった報告内容や様式に対する拡張性を欠くことだろう．それにしても非定型，非定例の処理要求に応えるためには一つの解決策である.

高水準言語を用いると，処理手順が明確にされるので，データ・ベースの安全のためにはよい．また一般にコンパイラによる目的モジュールの効率は，インタプリータを中心とするフォーマット系より高い．さらにフォーマット系にみられる拡張性不足という弱点はみられない．これらの長所とひきかえに，高水準言語を用いることによる短所もある．原始プログラムの作成・検査工程ははるかに複雑となるし，全く新しいプログラム言語を，COBOL, FORTRAN, PL/I 等に加えて導入しなければならない．プログラムの製作工数としては COBOL 等の汎用言語よりははるかに少ないが，そうかといって非 DP 要員に全てをまかせるには難しすぎる．しかし GIS が1960年初頭にフォーマット形式のシステム†† として製作され，10 回近く改訂されたのちに，現在の手順記述式の高水準言語におちついた歴史を考えると，フォーマット形式のシステムの限界がきびしいものであることが理解できる.

データ・ベースに対する処理内容が，ファイルの更新・維持の方向に傾けば傾くほど親言語系が有利になり，単なる問合せ処理で単発的なものになれば独立系が活きてくる．というのが言語方式に関する一般的な結論であろう.

2.3.2 入出力の管理

本項ではファイルに対する入出力の方法とデータ独立性の関係をとりあげる.

† このようにアプリケーション・プログラムの中で記述される論理構造を CODASYL DBTG Report では SUB-SCHEMA と呼んでいる．前出の個々のデータ・ベースに関する定義は SCHEMA で，CODASYL の場合には SUB-SCHEMA は SCHEMA のサブセットである.

†† ランゲージ・インターフェース (language interface).

††† BOMP (Bill of Material Processor) 部品情報をたくわえ，組立工業における所要量計算，製造工程管理などに用いられたチェイン・ファイルの処理プログラム．IBM 社.

†††† IDS (Integrated Data Store) COBOL のスーパーセットとしてデータベース機能を追加したシステム．GE 社.

††††† DL/I (Data Language One) 複雑・大量の木構造データをオンラインで扱うために開発されたもの．ノースアメリカン航空機会社.

† フォーマット系 (format driven system) 出力表の様式と項目を指定して，入力プログラムとする.

†† GIS の前身は FFS (Formated File System) である.

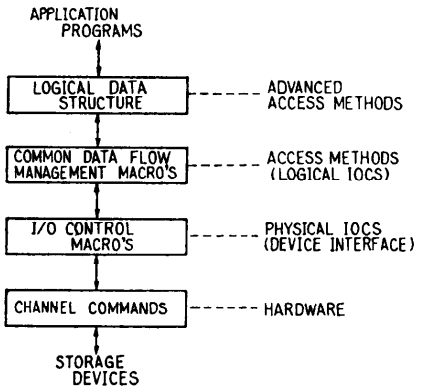

Fig. 12 入出力管理の水準

入出力の管理はいくつかの水準にわけて設計される．簡単に説明すると，命令コードのみを用いる水準，ハードウェア・インターフェースの標準化（装置からの独立性），フィジカル・ファイルの標準化（アクセス・メソッド）の水準などとなる．

（a） ハードウェアによる水準

一部の小型計算機をのぞくと，現在では入出力制御チャネル装置が一般化している．この装置に対して与える命令をチャネル・コマンド†もしくはコマンドと呼んでいる．コマンドを組合せて極く簡単なプログラムを作るわけであるが，この水準での入出力動作は単純な作業パタンの組合せなので，人手でコード化する必要はない．必要なコマンド・プログラムを与えられた目標に対して組合せるようなマクロ命令をアセンブリ言語に組込んだのがフィジカル IOCS†† である．

（b） フィジカル IOCS

別名デバイス・インターフェース・モジュールとも呼ばれる通り，入出力装置の制御形式ごとに一つずつ制御ルーチンを用意する．コマンド・プログラムの合成ばかりでなく，装置の動作順序，時間依存性なども解決しなければならない．なかでも誤り制御と再始動の手順は欠くことのできない機能である．

（c） アクセス・メソッド

アクセス・メソッドの目的はプログラムを装置特性から独立させる点にある．一般に行なわれる方法は標準的なデータ・セットを編成することで，OS/360 では順次編成，索引つき順次編成，直接編成，区分順次編成の4種の標準を設定し，入出力装置の異同にかかわらずプログラムの交換が行なえるようにしたことは記憶に新しい．機能からみると，

アドレス変換
索引の取扱い
ブロッキング・デブロッキング
入出力バッファの制御

などが主要なものということができる．

（d） 論理構造の処理

論理構造が与えられる場合には，単なるブロッキング・デブロッキングやアドレス連鎖の自動解読ではアプリケーション・プログラムとのインターフェースが不完全なものとなる．アプリケーション・プログラムが要求しているような形のレコードはそのままの形で存在しないのが普通だからである．一般に必要とされるのは

階層情報の解読
フィジカル・ファイルの探索
アクセス・メソッドとの結合
セグメント探索条件の解読と処理
排他制御
機密保護機能

などとなろう．

いろいろなアクセス・メソッドにもとづくフィジカル・ファイルを1回の指令にもとづいて処理するので，アクセス・メソッドからの独立性は最低条件となる．

将来の方向として，セグメント内でのデータの位置，データの構造，種類などが変動しても，動的に追従できる管理体系が話題になるが，不幸にしてそれが実現された話は聞いていない．

2.3.3 バインディング

バインディングに起因するシステム・オーバーヘッド†がデータ独立性の水準を高める上で最も大きな問題である．データ・ベース側の動的変動からプログラムを切り離すためにはバインディングの時期を，入出力実行時に近づけてやればよいからである．

入出力時バインディングは最も極端な方法であろう．すぐ想像できるように，データ目録の検索から始まる多段階の作業を毎回行なうのは非現実的な発想である．しかしながら今後の外部記憶装置の高速化にあわせて，できるだけバインディングを入出力時に近づけること，換言すれば経済的に実現可能な線を見出だしてゆくことが，データ・ベース・システム設計者の課題である．

コンパイル時バインディングは入出力時とは逆の方

† チャネル・コマンド (channel command).
†† フィジカル IOCS (physical input output control system).

† バインディング作業を行うために要する演算・入出力時間をさす．

法で，原始プログラム作成の時に一緒にバインディングも処理してしまおうとするものである．普通使われているコンパイラはこの原則にもとづいている．実行時間や所要主記憶を最適化するのには都合がよいが，同一データ・ベースに複数のプログラムからの入出力を行なわせるためには，逐次スケジュールとする必要があり不都合である．変動対応性は全くない．

DML をインタープリータまたはコンパイル・アンド・ゴー型に設計すると上記の欠点を多少取り除くことができる．しかしながら実行に際してコンパイル作業やインタープリート作業を伴うため，くり返し作業の多いデータ処理には適していない．後述の問合せ作業，などでも偶発的な問合せでくり返す回数が非常に少ない場合にはこうしたシステムが採用されている．

現在実用性が高いとして採用されている方法としてはロード時バインディングがある．プログラムはロード・モジュールとして準備できるので，オペレーショナルな作業に対して十分対応できる．原始プログラムは論理構造のみを定義するようにし，それを前提としてコンパイルを行なう．現存するコンパイラがそのまま使える利点がある．シンボリック・レファレンス†の利点を導入するために，プログラム中で参照するデータ名をまとめて表を作ることがある．PSB†† などと呼ばれるこの表はロード・モジュールとは別に登録しておき，バインディングを行なう際，ロード・モジュールと一緒にロードして使用する．トランスファ・ベクトルの拡張とも考えられるが，論理構造は EXTERNAL ではないので，コンパイラに自動的に作らせるわけにはゆかない．

PSB から DBD††† をいくつか参照することによって，データ・アドレスが解読されることになる．この方法によると同一データ名に対して複数のプログラムを

READ, READ
READ, WRITE
WRITE, READ

の組合せの場合には併行してスケジュールする可能性が残される．データ・ベースを多数のプログラムに対して共有化できるようになる．もちろん排他制御など保全性の技術がともなうことが，共有化への大前提である．

† シンボリック・レファレンス (symbolic reference) データ名を原始記号のままでロード・モジュール中に残し，使用時にデータ目録を参照することによって，プログラムと装置上のアドレスを対応させる技術．

†† PSB (program specification block) IMS/360 の制御ブロック．

††† DBD (data base description).

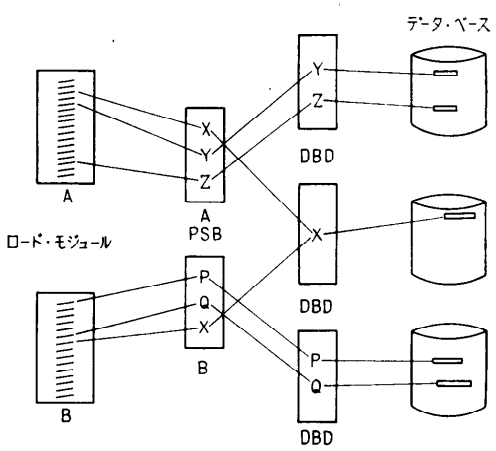

Fig. 13　ロード時バインディング

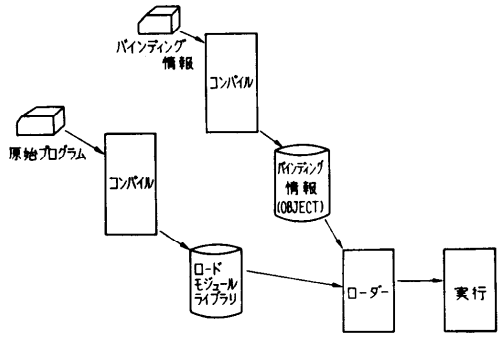

Fig. 14　ロード時バインディング

ある．

コンパイラを拡張して同じことを行なったものが，DBTG 案の SCHEMA, SUBSCHEMA で，前者には DBD，後者には PSB の機能が含まれている．よく似ているが，コンパイラによってバインディングの制御情報が供給される関係上，コンパイル時バインディングの性質が影響としてあらわれたり，COBOL 以外の言語との連絡が考えられていないなどの相違点がある．

2.4　データの検索

2.4.1　索引の設計

データ検索の主要な手段は索引である．索引，なかでも2次索引は非常に有効な方法なのでよく研究されている．索引はキーと装置アドレスを用いるので装置に対する独立性がない．場合によっては同種の装置でも可搬性†が失なわれてしまう．これを避けるために

† 可搬性 (portabilty) データ・ベースを一つの装置から他の装置へ移動できること．

はデータの主キーによる索引すなわち主索引†を用意し，2次索引はアドレスの代りに主キーを用いる方法がある．索引へのアクセス回数が1回増すので，高速性が重んじられる場合には装置アドレスが用いられているが，データ検索が複雑になるにつれて主索引を経由する方法が有利となってくる．検索の複雑さとともに問題となるのはデータ更新の頻度であろう．なかでも2次索引のキーとして採用したフィールドの更新のためには

主データ・セットの更新
索引の再構成
アドレス・ポインタの更新
逆アドレス・ポインタの更新

の4種の作業を行なわねばならず，大きなシステム・オーバーヘッドをともなう．したがって2次索引の設計にたあっては単に索引時の効率（これだけでも装置特性を考慮すると複雑な問題である）のみならず，検索そのものの複雑度，データ更新の頻度，装置からの独立性など多くの要因を一つ一つの利用体系について評価しなければならない．

索引の構成法としては多重リスト，セラ・マルチリストなどが代表例となろう．

多重リストはデータ属性の一つの値に対して，多数の対応レコードをアドレス・ポインタを用いてリスト化したもので，索引に要する記憶域を圧縮したり，索引とデータ・ベースを分離するのに有効な方法である．さらにアクセスをセル単位で層化したセラ・マルチリストは順次編成のフィジカル・ファイルを部分的に逐次探索することができるので，速度をあげることができる．セラ・マルチリストを主索引に応用し，かつ索引をフィジカル・ファイルの内部に分布させた例がいわゆる索引つき順次編成である．

2.4.2 マッピング

キーとデータ・アドレスを結びつける時には，両者の間の関係が重要な設計因子となる．

いま人事ファイルで社員番号をキーとし，個人レコードをデータとすると，両者の間に 1：1 の関係が成立する．これを単純マッピング††と呼ぶことにしよう．個人レコードは1人1個と限っているので，個人レコードの所在がわかれば，対応する社員番号を捜すことができる．すなわち逆マッピング†††も単純である．

同じ対応関係を社員番号と姓名の間で考えてみよう．姓名には同姓同名の人がいる可能性がある．すなわち姓名から社員番号へのマッピングは $1:n$ となる，一方社員番号の側からは 1：1 である．逆マッピングはこの場合“複雑”†であると呼ぼう．

正順のマッピングと逆マッピングそれぞれに対して単純な場合と複雑な場合があるので組合せは4通りあることになる．2次索引は逆マッピングの性質によって複雑さが左右される．マッピング，逆マッピングともに複雑な例としては，

部品番号　と　上位部品番号，
倉庫番号　と　在庫品名，
姓名　　と　特技記録

など日常よく利用されるデータがあげられる．複雑なマッピングを同一データ・ベースの索引に多重に使用する場合には前出の処理効率，更新頻度について十分検討すべきである．

2.4.3 データ検索とプログラム言語

データ・ベースを管理・維持するためにはデータの目録の作成・保守にあたるデータ記述言語（DDL），データ・ベース利用上の手続きを記述するデータ操作言語（DML）があることについてはすでにのべた．データ操作言語にはデータ・ベースに対する問合せ††機能は含まれるものの，その主目的はデータの追加・削除・更新にある．データの変更にあたって手続き上の誤りは絶対に避けなければならないので，データ操作プログラムは仕様・製作工程などに起因する誤りを検査して取り除いておかねばならない．換言すればプログラムは厳重な試験を通るべく入念に作られることを意味するので，製作には時日を要する．

データ・ベースに対する問合せの中には，簡単な処理だけでよいから，即座に結果が欲しいという種類のものも多い．むしろ近年のようにコンピュータに情報の管理をまかせる場合には，単純な問合せの即時処理は情報システムにおける必要条件と目される．こうした要求を満たそうとするものが問合せ言語である．

問合せ言語はその性質上プログラミングがやさしく，コンピュータ・システムの知識を前提としないことを条件とするので，独立言語ないし亜自然言語の形をとるか，あるいはフォーマット言語の形をとることが多い．その機能としては未だはっきりした守備範囲

† 主索引（primary index）装置上のデータの地理的分布に従う索引．
†† 単純マッピング（simple mapping）．
††† 逆マッピング（inverse mapping）．

† 複雑なマッピング（complex mapping）．
†† 問合せ（query）単に特定のレコードの内容を問合せるだけではなく，集計，分類，作表などの処理を含む．

は定められてはいないが，IBM 社の IQF† の例では

QUERY:	問合せ始め,
LIST:	編集・出力,
SORT:	分類,
COUNT:	件数の集計,
TOTAL:	合計,
LIMIT:	出力行数の制限,
WHEN:	条件文,
END:	問合せ終了

の8個の句が中心となり，これに利用者の指定句を追加するための DEFINE, AS, ENDEF, DELETE の4句をもって構成されている．この他に補助的な ASC, DES の2句は SORT の順を指定し，ON は TOTAL, COUNT に対する制御を行なう．これにナルワード†† を加えて非定型問合せ文††† が作れるようにしている．

問合せ文には定型・非定型の2種がある．定型的なシステムはコマンドとパラメータの組合せに代表され，入力時に間違いが許されない．これに対して非定型問合せ文は IQF のナルワードのように無意味な単語が許されるので，擬英語文のような形の入力ができ，冗長性を加えられる．

2.5 データの保護

データは共有という環境の下では2種類の危険にさらされている．意図をもってデータを盗み出したり，他人の秘密を発こうとするのがその一つであり，データ・プライバシの侵害という形で考えられている．他は意図の有無にかかわらずデータ・レコードが破壊されたり，間違った変更をうけてしまうもので，データの保全性の良否という範ちゅうに入れられる．

いずれにしてもデータがデータ・ベース作成時から間違っていたり，更新の手続きが正しくなかったり，あるいは更新入力のデータがあやまっているような場合については，データ処理システムが自動的に判断を下すことが難しいのでのぞかれている．

2.5.1 データ・プライバシの保護

データの盗用・破壊を防ぐことは，意図的に行なわれる場合には難しい問題である．暗号のような技術的な方法は，相手方にもれてしまえば，逆の対抗手段を開発されることによって効果が減殺されてしまう．

データ・アクセスの承認体制† を確立することが最も基本となる保護手段であろう．

データを使用しようとする人が，有資格者かどうかを検定する方法は種々とある．

（1） パスワード††

使用権をデータ・ベース管理システムにあらかじめ合言葉の形で登録し，使用時に入力して確認する方式．パスワードは個人ごと，あるいは使用目的ごとに作っておく．同一パスワードに対して，アクセスできる範囲や，データの変更を許す範囲をデータ・ベースごとに変えておく．さらにそのパスワードが入力された場所・時間について調べ，有効性を判定する必要がある．

データ・ベース管理システムはパスワードの使用記録・使用統計をとることによって，パスワードの盗用などの事態の発見に資する．この種の記録の検査はデータ・ベース管理者に課せられた重要な仕事である．

（2） 入/出力場所の確認

とくにオンライン・システムの場合には，入出力が行なわれる端末装置の位置によって，資格の有無をかなり推定することができる．たとえば倉庫での入出庫データ収集用端末から行なわれる人事異動に関する問合せは，まず受付けないことにした方がよかろう．

（3） 機械的な資格検査

入出力装置やファイル庫に錠をつけたり，オペレータ卓に鍵による入力装置を用意したり，あるいは守衛を巡回させたりする類いの方法．

（4） 利用者の同定†††

音声のように個人特有なデータを，システムの側で認識しようとする方法である．声を聞いてだれであるかを推定する方法は現在の技術では実用化されていない††††．しかし特定の単語を発声させ，その中からいくつかの特徴をとらえて，入力された発声が登録されたものと同じかどうかを判定させること††††† はある程度可能である．資格検査の諸方法の中ではもっとも CPU オーバーヘッドが大きい．いずれにしても CPU オーバーヘッドは，プライバシ保護のための泣きどころである．

いかなる保護手段を用いても安全性は確実とはいえないので，現今，法的手段で保障しようという動きが

† IQF (Interactive Query Facility).
†† ナルワード (null word) プログラムを読みやすくするための句で，文法上は意味がない．
††† 非定型問合せ文 (unstructured query).

† アクセスの承認 (access authorization),
†† パスワード (password) 合言葉.
††† 利用者の同定 (user identification).
†††† 音声認識 (voice recognition).
††††† 音声同定 (voice identification).

あるのは知られている通りである．それにしても火災，地震などには万全ではあり得ない．人間のモラルの向上と，データ・ベースの予備をそなえるのが最善の途である．

2.5.2 データの保全

データの保全についてはとりあげるべき項目が二つある．データ・ベース・システムの保全性と，データそのものの保全性である．

データ・ベース・システムの作動環境としてはハードウェア，システム・コントロール・プログラム，データ・ベース用ソフトウェアおよび同一ハードウェアの下に働く他のソフトウェアが考えられる．一つ一つのMTBF† を長くすることが基本的解決策であるが，それでもなお故障は避けられないし，ソフトウェア間の相互干渉もなくしてしまうことはファイル・シェアリングという目的からしてできない相談である．対策としては次の項目があげられている．

（1） ハードウェアに対する再試行機能の拡充，

（2） ソフトウェアにおける異常回複機能のくみこみ（ダンプ条件の回避），

（3） チェック・ポイント/リスタート

データそのものの保全については，ソフトウェア間の干渉の排除が重要なテーマである．いくつものプログラムが並行してデータ・ベースを共有する場合には更新の順序が処理後の内容に大きな影響を及ぼすことは周知の事実である．これを無くすことは並行処理を否定することとなり，処理効率が著しく低下する．現在とられている対策は排他制御による，処理過程の再現性の確保である．事後異常が発見されても，ファイルは更新前の形に戻すことが可能となる．

（1） 排他制御

一つのトランズアクションによる更新処理が完了するまでは，他の更新作業を拒否する方法．セグメントごとに行なうのが理想的だが処理効率を考慮して，セグメント・タイプごとに行なう場合，ファイルごとに行なう場合などがある．

（2） ジャーナリング††

データ・ベース処理の記録をとっておくことにより，事故時に任意の時点までデータ・ベースの内容を戻すことができる．ファイル変更を逆に辿って事故直前まで回復させることもできる．

（3） チェック・ポイント/リスタート

特定時点の状態にデータ・ベースを戻し，処理をそこから再開できるようにする．

などの手続きがすでに開発されている．

† MTBF(mean time between failure) 平均故障間隔．

†† ジャーナリング (journaling).

参考文献

1) A Survey of Generalized Data Base Management Systems, CODASYL Systems Committee, May 1969.
2) ACM Professional Development Seminar Student Handout, Wiley Systems, Inc. April 1971.
3) CODASYL COBOL Journal of Development 1969, No. 110-GP-a, April 1970.
4) CODASYL Data Base Task Group Report, May 1971.
5) Data Base Management System Requirements, Joint Guide-Share Data Base Requirements Group, Nov. 11, 1970.
6) Dodd, G. G.: Elements of Data Management Systems, *Computing Surveys* 1, 2 (June 1968), 117-133.
7) Engles, R. W.: A Tutorial on Data Base Organization, Presented in # 29 Guide Meeting, November 1969.
8) Henry, W. R.: Hierarchical Structure for Data Management, *IBM Sys.* J. No. 1, 1969, 2-15.
9) Hoffman, L. J.: Computers and Privacy: A Survey, *Comp. Surveys* 1, 2 (June 1969), 85-103.
10) IMS/360 General Information Manual, GH 20-0765, IBM, 1971.
11) IMS/360 System/Application Design Guide, SH 20-0910, IBM, 1971.
12) Interactive Query Facility (IQF) for IMS/360, General Information, GH 20-1074, IBM 1971.
13) Introduction to "Feature Analysis of Generalized Data Base Management Systems," CODASYL Systems Committee, *CACM* 14, 5 (May 1971), 308-318.
14) Steig, D. B.: File Management Systems Revisited, *Datamation*, October 1972, 48-51.
15) Tani, P. Y. Comparison of DBMS Reports, Presented in Guide Meeting, August 2, 1971.
16) Welke, L.: A Review of File Management Systems, *Datamation*, October 1972, 52-54.

（昭和47年1月5日受付）