データベース・システム 69－8
（１９８９．１．１９）

三次元データベース管理システム
3D - DBMS

石　井　義　興

株式会社　ソフトウェア･エージー

データベースは時間空間の中で変化成長しながら利用されるものである。ここでは時間経過に伴ってどんな成長があるかを例示的に示す。

この例示の中で示された時間要素を含む3種のファイル･タイプすなわちマスタ･ファイル、イベント･ファイル、集計ファイルについてその性質の概要を述べる。特にマスタ･ファイルは時間経過の中でビフォア値を自動的に保持管理する必要性にせまられる。またそうなれば、そのファイル{3次元(3D)マスタ･ファイル}に対する検索は時間要素を含んだものとなる。

3D-DBMSは3Dマスタ･ファイル、イベント･ファイル、集計ファイルを取扱うことができ、3Dマスタ･ファイルに対し時間要素を含んだ検索を実行することができる。この際検索の高速化のため、インバーテッド･インデックスの他に、ここで提案するタイム･インデックス、年齢インデックスを用いる。

ここで提案する3D-DBMSは市販されている西独製リレーショナル型DBMS“ADABAS”をベースにして具体的に日本で開発が予定されている。

Three - Dimensional DBMS
3D - DBMS

Yoshioki Ishii

Software AG of Far East, Inc.

2 - 5 Kanda Surugadai, Chiyoda - ku, Tokyo 101 Japan

Databases grow and vary every moment, and are used in such time space. In this paper, I illustrate how the databases grow and change with the lapse of time by giving samples. It describes the outline of the characteristics of three file types including time factors shown in the samples such as master files, event files and summary files. Especially, the master files indicate that it is necessary to maintain and manage the 'before value' automatically in process of time. At the same time, it illustrates that the access to the file, which is a three-dimensional (3D) master file, should include time factors.

The 3D-DBMS can handle 3D master files, event files and summary files and provide the access including time factors to the 3D master files. At that time, the time index and age index, which are proposed in the paper, are used in addition to the inverted index in order to increase the speed of access.

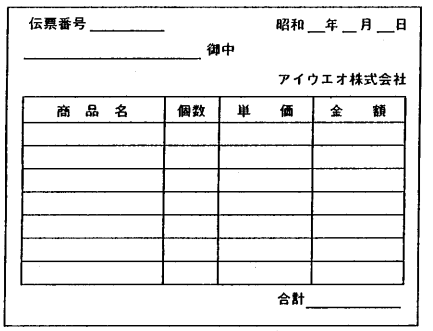
The 3D-DBMS is planned to be developed in Japan based on ADABAS, a relational-type-DBMS software product developed in West Germany.

1. はじめに

まず時間経過に伴ってどんなデータベース･スキーマの成長が起るかを例示的に示し、3次元データベース管理システム3D-DBMSの基本機能、ついで時間要素を含んだ検索の高速化のために提案されたタイム･インデックス、年齢インデックスについて述べる。

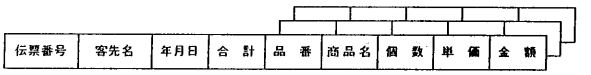
2. 時間経過に伴うデータベース･スキーマの成長

アイウエオ㈱を例にとり時間経過に伴うデータベース･スキーマの成長性について述べる。



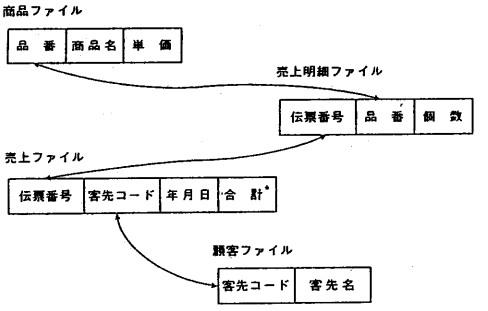
第1図　売上伝票

まず、第1図のような売上伝票が多数発生するので、このデータをデータベース化することを考える(時点A)。伝票情報をそのままファイルにすると第2図のようになる。



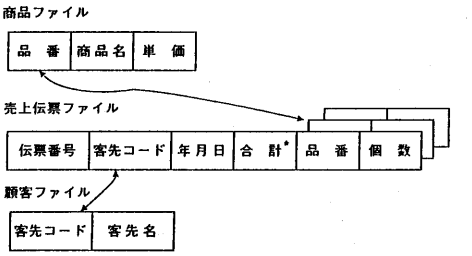
第2図　非正規売上伝票ファイル

しかし、正規化を行った方が良いと考えられ、また客先名は人によって書き方が統一できないことが多いので、客先コードを付け誤りがないようにしたい。その結果設計されたデータベースは第3図のとおりである。第3図で＊を付けた項目はなくとも良いが、検索時点のスピードアップを考慮し、ここでは付加している。これらの項目は更新がないので、これによるマイナス面は少ない。



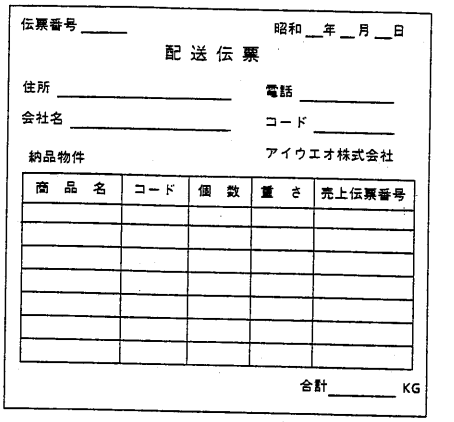
第3図　売上げデータベース(第三次正規形)

ここで正規化しない場合のデータベースの例を第4図に示す。筆者は第4図のデータベース設計が正規化した場合より優れていると考える。なぜなら、第4図中の売上伝票ファイルを第3図中の売上ファイルと売上明細ファイルに分割(非第一次正規形から正規形への分割)すると冗長で、しかももとの伝票に含まれる情報に復元するためには、第3図の場合はJOINを取る回数が多くなる。売上伝票ファイルの各項目には値の変更は起らないので正規化しない方が良いと考える。



第4図　非第一次正規形ファイルを用いた売上データベース

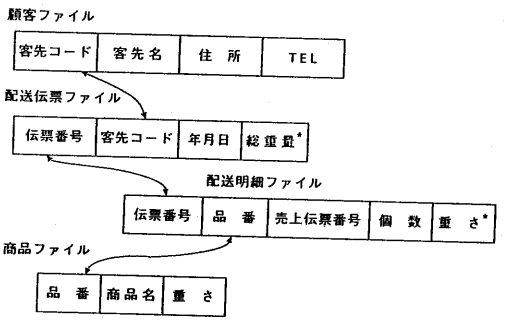
さて、多少日時が経過した時点でアイウエオ㈱では配送に関する情報もデータベース化を計った(時点B)。配送伝票の例を第5図に示す。



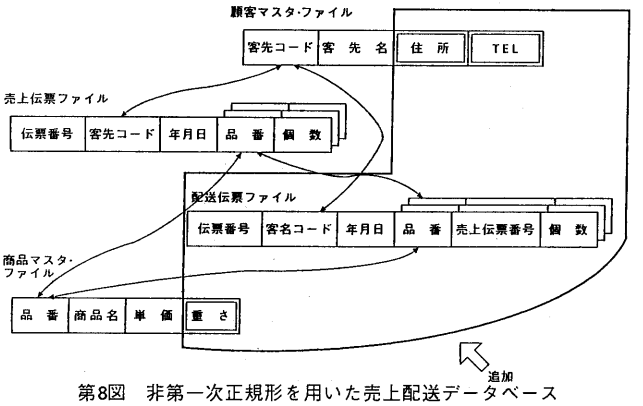
第5図　配送伝票(例)

伝票中の各明細行に売上伝票番号があるのは、売上と配送が同期していないためであり、この配送伝票に示されている配送品がどの注文(売上伝票)に対応するものであるかを明確にさせるためである。第6図に配送データベースのスキーマを示す。正規化のプロセスは第1図から第3図の場合と同様である。

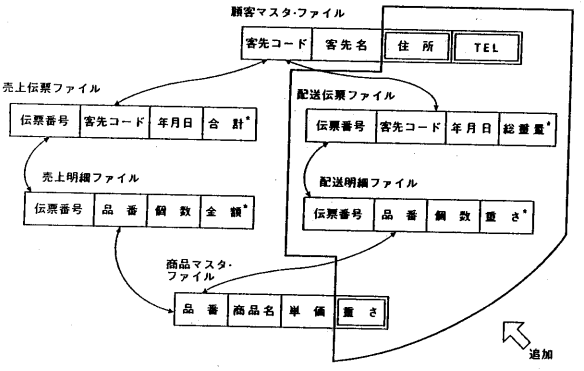
しかし、すでに売上データベースが存在し、商品ファイル、顧客ファイルも存在しており、配送データベースは独立に作成されるのではなく、売上データベースが成長する形になることが望ましい。その結果完成したのが第7図に示す売上配送データベースである。



第6図　正規化された配送データベース



第8図　非第一次正規形を用いた売上配送データベース

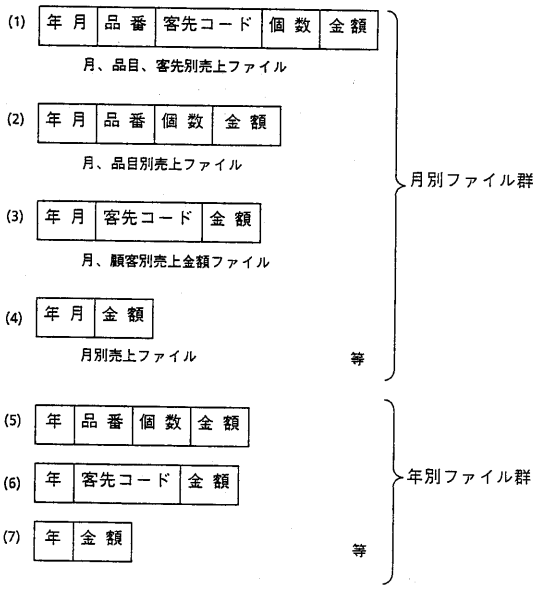


第7図　売上配送データベース

顧客ファイル、商品ファイルは共用されるので、ここではマスタ・ファイル(M)と呼ぶこととする。この例からもわかるように、マスタ・ファイルには時間経過に伴って新項目が追加される。伝票情報だけを考えてもここに示した2種類のみではないため、マスタ・ファイルは伝票情報の正規化に伴って多くの項目が追加されることとなる(マスタ・ファイルの成長)。ここでは伝票(トランザクション)を分解(正規化)するプロセスでマスタ・ファイルが誕生したが、マスタ・ファイルの主キーは人間が認識できるあるエンティティに対応させて作られる。したがってそのエンティティを表現する項目をいくつか書き並べることによりマスタ・ファイルを考えても良い。こうして作られたマスタ・ファイルがすでにあれば、伝票情報を正規化する際はそれを共用することになり、結果は同じである。第8図に第4図と同様に非第一次正規形を用いた例を示す。

第8図の売上伝票ファイル、配送伝票ファイルのようなファイルをトランザクション・ファイル、あるいはある時点で起った事実(イベント)を表しているのでイベント・ファイル(E)と呼ぶ。

アイウエオ㈱は業績が伸び売上げ品目、顧客数、金額が増加してきた。そこで月別に品目、客先別に集計し、その情報も後日自由に検索できるようにすることとなった(時点C)。そのため、例えば第9図に示すような集計結果のファイルを作成した。これらのファイルをここでは集計(サマリ)ファイル(S)と呼ぶ。集計ファイルはアプリケーションの必要に応じて多数作成される可能性があり、適性な判断が要求される。



第9図　集計ファイル例

アイウエオ㈱はさらに業務拡大し、営業所を複数新設することになった(時点D)。営業所が複数存在すると売上伝票番号は会社全体でユニークに番号付けすることは困難になり、営業所ごとに採番することになる。営業所ごとに発行する際の伝票を第10図に示す。

伝票番号 _____　　昭和__年__月__日

_____ 御中

アイウエオ株式会社

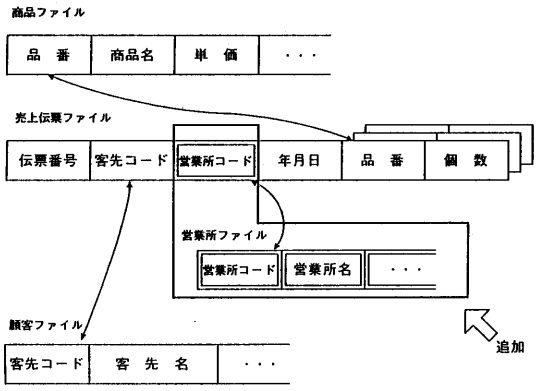
_____ 営業所

商品名	個数	単価	金額

合計_____

第10図　営業所別売上伝票

この場合、既存のデータベース･スキーマにどのような影響が及ぶかが問題となる。売上配送データベースに対するスキーマ変更の例は複雑になるため、ここでは売上データベースに関し、変更を加えたケースを第11図に示す。このケースが示すようにイベント･ファイルにも時間経過に伴って新項目の追加が発生するのである。



第11図　データベースの成長

売上伝票ファイルに新項目営業所コードが付加されると第9図に示した集計ファイルにも影響が現れ、新集計ファイルが付加されるかも知れない (第12図参照)。

年月	営業所コード	品番	個数	金額

年月	営業所コード	客先コード	金額

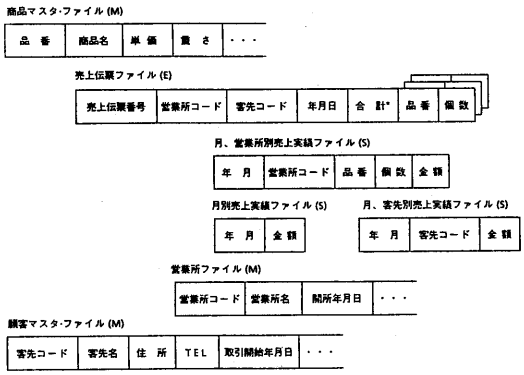
年月	営業所コード	金額

等

第12図　新集計ファイルの追加例

集計ファイルはダイナミックに作成できるので、データベースに常駐させる必要はないが、検索の頻度が多い場合はむしろ常駐させる方が望ましい。なぜなら、当ファイルは内容変更はなく、レコードの追加が時々 (この例では月あるいは年に1度) 発生するだけであり、オンラインによるダイナミックな更新はない。したがってデータベースに常駐させることによるデメリットはほとんど認められない。集計ファイルを持つことにより、検索は非常に高速化される。

第13図に売上伝票のデータベース化が始まり、何回かのスキーマ変更の結果完成した売上情報データベースの図を示す(時点Dにおけるスキーマ)。(ただし、複雑になるためここでは配送関連のファイルは記述していない。)

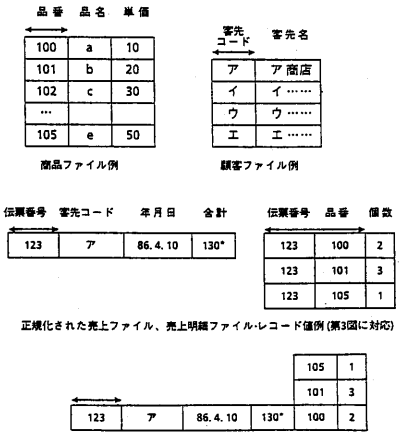


第13図　売上情報データベース

3D-DBMSは例示した時点Aから時点Dへの変化があっても簡単なスキーマ変更だけで、DB全体の再生成が不必要なことと、完全なデータ独立性が要求される。

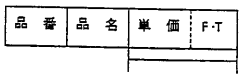
3. 値の変化と非復元性

時点Aから時点Dの間ではスキーマ変更だけが起るわけではなく、値も追加、変更される。第14図に第3、4図で示した各ファイルのレコード値例を示す(時点A)。



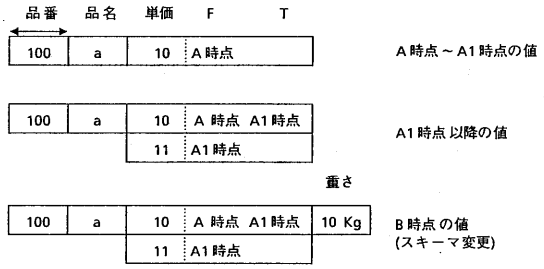
第14図　売上データベースの値例 (時点A)

時間が経過し(時点A1)品番100の単価が10から11に変更されたとする。これに伴って商品ファイルの単価10を11にアップデートすると、伝票番号123の伝票情報は正しく復元できなくなる。なぜなら86.4.10の時点では品番100の単価は10であったからである。正しく復元できるようにするためには単価に関し、何時から何時までその値が有効であったかという情報を持っておかねばならない。また伝票ファイルと商品ファイルのJOINを取る際には時間を加味したものでなければならない。このように項目値のアップデートの際ビフォア値を保存し、かつ何時から何時までその値が有効であったかをも保存しなければならない。そのように考えたファイルを3次元ファイル(3Dファイル)と呼ぶこととする。3D商品ファイルのスキーマを第15図に示す。



第15図　3D商品ファイル

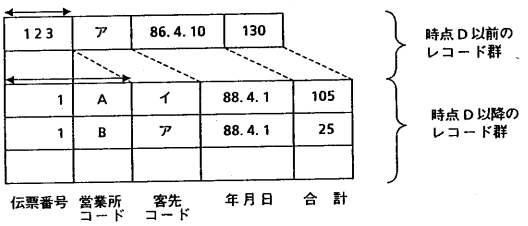
FT は From To を表し、「何時から何時まで」を表す時点A<時点A1<時点Bの時の3D商品ファイル内の品番100のレコードが時間経過に伴ってどう変化するかを第16図に示す。



第16図　3D商品ファイルの品番100レコードの推移

第15、16図は単価をくり返し項目として持つようなイメージであるが、必ずしもくり返し項目として持つという意味ではない。

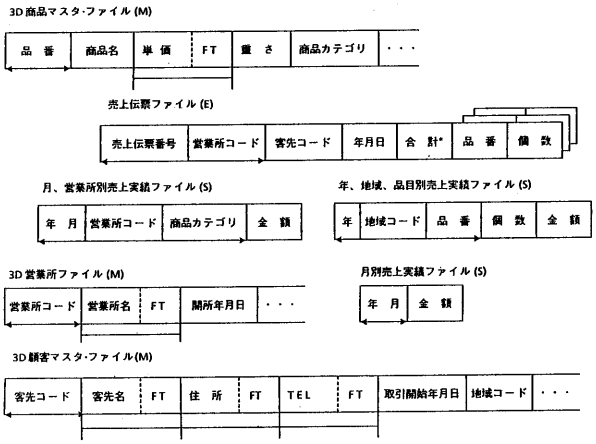
売上ファイルの内容例を第17図に示す。主キーが時点Dを区切として変化する。



第17図　売上ファイルの内容例(時点D以降)

第13図で例示した集計ファイル、すなわち月、営業所別売上実績ファイル、月、客先別売上実績ファイルはアプリケーション・ニーズが変化すると不必要になる場合もある。すなわちファイル自身がある時点で消えたり、ある時点で過去にさかのぼって生成されたりする。

例えば月、営業所別売上実績ファイルにおいては各品目ごとに集計している。が、もし商品の種類が増大するとレコード件数が多くなるので、商品カテゴリを設定し、それごとの集計が必要となる。また月、客先別売上実績ファイルの場合も顧客ごとの集計を取っているが、もし顧客数が増大すると、顧客を地域ごとにまとめて集計する必要が起こる。こうしたニーズの変化は時間経過に伴って自然なかたちで起る(時点E)。つまり第13図で示したデータベースは第18図で例示するデータベース・スキーマに変更される。



第18図　3D売上情報データベース

また、次のような変化にも対応できねばならない。

イ) 品目、客先数の増加によって品番、客先コードの桁数増加

ロ) 営業所の統廃合による変化(東京第1、第2→東京本部)

ハ) 商品のカテゴライズ方法の変化(雑貨→事務用品、家庭用品)

ニ) 地域のとらえ方の変化(東京→首都圏)

このような変化が起っても最小限のスキーマ変更と一部の集計ファイルの再生成にとどめ、データベースの再定義、再生成は不必要でなければならない。

ここで提案する3D-DBMSは、以上例示したような時間経過に伴って発生するスキーマの成長変化にスムーズに対応できなければならない。そして値のアップデートの際ビフォア値を保存したい場合は、3D-DBMSが自動的にビフォア値/アフタ値を保存管理する必要がある。その上問合わせには時間を含んだ質問を受入れ、答も時間を含んだものと

しなければならない。筆者はこの問題について数回論議した。[1][2][4][5]特に[2]では今まで述べてきた問題と同様の問題を提示し、具体的にデータベースのデザインを行ってもらった[3]。

4. 3D-DBMSにおけるファイル·タイプと更新

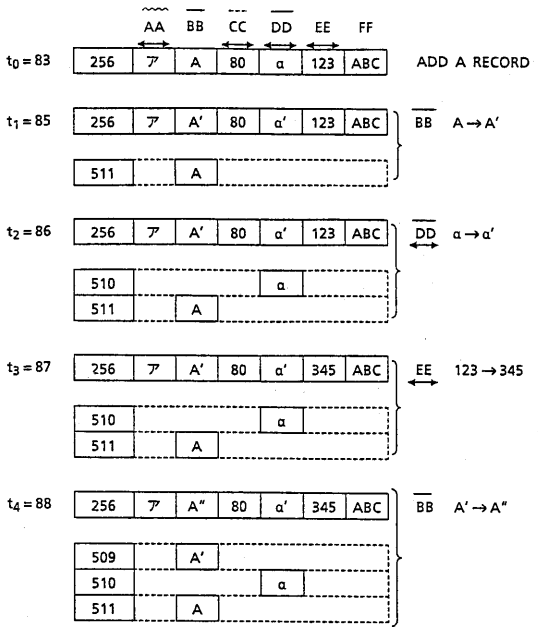
3D-DBMSは3種類のファイル·タイプ、すなわち、マスタ·ファイル(M)、イベント(トランザクション)·ファイル(E)、および集計(サマリ)ファイル(S)を定義可能とし、特にマスタ·ファイルに関しては指定された項目のビフォア値も自動的に保存管理しなければならない。そして、3種類いずれの場合も繰り返し項目を許す必要がある。各々のファイル·タイプは表1で示すような性質を持っている[4]ので、それを十分意識したデータ管理方法が採用されねばならない。

	レコードの追加/削除	項目値の内容変更	新項目の追加	3次元性
マスタ·ファイル	少ない	あり	多い	あり
イベント·ファイル	多い(オンラインで)	なし(誤りの訂正のみ)	少ない	なし
集計·ファイル	多い(バッチ処理で)	同上	少ない	なし

表1　ファイル·タイプとその性質

表1からもわかるようにイベント·ファイル、およびその集計結果である集計ファイルに対しては誤りの訂正以外項目値の変更は起らない。したがって項目値の変更に伴ってビフォア値の履歴を保持する必要のあるファイルのタイプはマスタ·ファイルのみである。3種類のファイル·タイプそれぞれに対応してファイル値の変更には特徴がある。マスタ·ファイル内のレコードは主キーによってアイデンティファイされるあるエンティティを表現している。そのエンティティを表現する項目はいろいろあるが、各項目値はエンティティの時間空間での変化に対応して項目値も変わるが、項目の中には値の変化しないものもある。例えば生年月日、性別等である。また値の変化があったとしても必ずしもビフォア値を持たず、最新の値だけ持っていればアプリケーション上十分である場合もある。マスタ·ファイルは新項目(達)の追加が許されると同時にこれら項目ごとの属性も定義可能としなければならない。すなわち項目値の内容変更に伴ってビフォア値を自動的に保存管理する項目(ここでは世代管理項目と呼ぶ)や値が不変である項目(ここでは不変値項目と呼ぶ)を定義可能とする。ここで、3D-DBMSにより管理される3Dマスタ·ファイルの管理方法を例示的に示そう。

各エンティティに関し256世代まで管理する場合を考える。各エンティティおよびその各世代値は別レコードとし、それぞれレコード番号(システムにより自動的に付けられる)を付ける。すなわち第1のエンティティには256を、第2のエンティティに対しては512を、と256おきに採番する。そして第1エンティティの各世代は256がカレント·レコード、511が最も古い世代レコード、510がその前の世代レコードとする。もし3世代しかない場合はレコード番号256(カレント)、510、511がそれぞれの世代に対応したレコードである。あるエンティティが256世代を超えたときはファイルごとにあらかじめ決められたある時点以前の各世代レコードをオーバーフロー·ファイルへコピーし、それらを削除し、残ったエンティティの各世代レコードのレコード番号を変更する。第19図に3Dマスタ·ファイルの世代レコード例を示す。項目名上の〰〰、——、…… は不変値属性、世代管理属性、年齢管理属性をそれぞれ表す。年齢管理に関しては後述する。項目名上に何もない場合は非世代管理項目を表す。項目名下の↔はその項目に対応しインバーテッド·インデックスを持つことを示す。この例での項目CCは年号値である。時点t_0で1つのエンティティが追加され、$t_1, t_2 \cdots t_4$がアップデートの時点を表す。ここでは年の値すなわちt_0=83年等でアップデートが行われたものとする。



第19図　3Dマスタ·ファイル レコードの値例

カレント·レコード以外の各世代レコード内には世代管理項目のうち変更された項目のビフォア値のみを保持し、他の項目値は持たない。したがってレコードは圧縮されるべきである。

イベント·ファイル内のレコードはある時点における事実を表現したものであるからそのレコードを構成する項目値が変更されることはない。しかしイベント·ファイルのスキーマは世の中の変化やニーズの変化に対応し変更されることがある。(第11図、第17図参照)したがってイベント·ファイ

ルはその時点におけるスキーマに対応したレコードが一般的にはオンラインで追加され、古いレコードがまとめてバッチ処理で削除される性質がある。

集計ファイルは一般的にはイベント·ファイルを時間軸(月や年ごと)とその他の属性で集計したものである。(第9図、第12図参照)

イベント·ファイルを時間軸のみで集計した場合(例えば第9図(4)あるいは(7))はいらなくなればファイルが削除されるが、必要な場合はそのまま保持され、(第9図(4)(7)→第18図)。再定義、再生成は起らない。しかし時間軸と組み合せて他の属性で集計した集計ファイル(例えば第9図(1)(2)等)はその属性(例えば第9図の品番、客先コード)に関する見方の変化によってファイルの削除と他の属性による集計ファイルの再定義、再生成が起る。(第9図→第12図→第13図→第18図) これらはすべてアプリケーション·ニーズの変化に伴って起ることであり、一般的にはこのタイプのファイルはデータベースに保持せず、ダイナミックに作成すれば良いと言われている。しかしもしそうしようとするとイベント·ファイルに常に古いレコードを多く保持し続けなければならず、しかも問い合わせ要求のたびに多くの処理を行い、ダイナミックに答を出す必要がある。どちらにするか、あるいはそのミックスしたものにするかはデザイン上の判断とすべきである。3D-DBMSは以上のファイル·タイプとその性質を十分意識したものとしなければならない。

5. 3D-DBMSにおける3Dマスタ·ファイルに対する検索

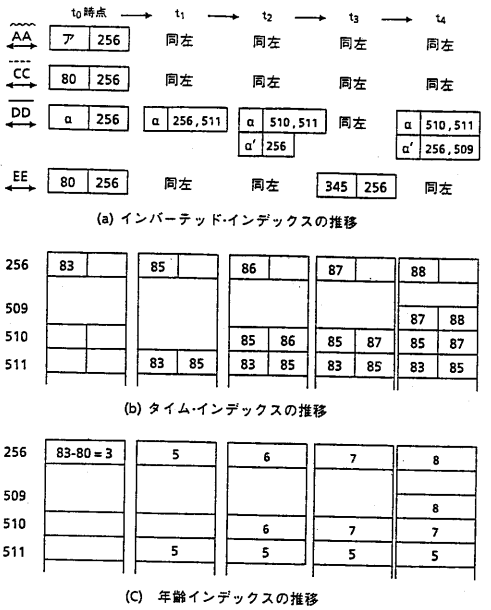
4で述べた3種のファイル·タイプから構成されるデータベース(ここでは3次元データベースと呼ぶ)に対する時間要素を含んだ検索を考える。特にここではインバーテッド·インデックス等を利用し、データ自身を読まずに答のレコード(あるいはエンティティ)を見つける方法を考える。すなわち“時間要素を加味した検索”のスピード·アップのため、どんなインデックスを持てば良いかを提案する。それらはインバーテッド·インデックスとここで提案するタイム·インデックス、および年齢インデックスから構成される。これら3種類のインデックスを用いると次のような時間を含んだ検索をインデックスのみを利用し答を出すことができる。

ア) 84年に～で現在〰の条件をみたす人を見つけ、その人の30才の時の住所と現在の電話番号を出力せよ。

イ) 少なくとも85年から87年まで～条件をみたす人を見つけ、条件～の状態が何時から何時まで続いたかを示せ。

ウ) 3年以上～条件をみたし、35才の時〰条件をみたす人を見つけ、その人の過去から現在までのヒストリーを出力せよ。

以上の時間要素を含んだ検索要求にスピーディに対応するために、簡単な拡張を行ったインバーテッド·インデックスと、各世代レコードが何時から何時までその値を持っていたかを示すタイム·インデックス、および年齢インデックス(生年月日から何時から何時までが×才に対応するかを示すインデックス)を用いる。インバーテッド·インデックス、タイム·インデックス、年齢インデックスを第19図で示した世代アップデート例に対応させて例示したものが第20図である。



第20図　インバーテッド·インデックス、タイム·インデックス、年齢インデックス例

第19、20図で示した例の項目CCの値は年であるがこれは例題をわかりやすくするためのもので、実際は1900年等を0日とする連続日付で持つと良い。項目$\overleftrightarrow{AA}$,$\overleftrightarrow{CC}$は共に不変値項目であるのでインバーテッド·インデックスは不変である。$\overline{BB}$は世代管理項目であるがインバーデッド·インデックス指定がないので、インデックスは作られない。

$\overleftrightarrow{DD}$は世代管理項目でインデックス付であるがt_1の時点では内容が変化していない。したがって世代レコード(511)に値αは保持されないが(第19図参照)、インデックス上は持たねばならない(第20図参照)。

$\overleftrightarrow{EE}$は非世代管理項目と定義されているので、ビフォア値は世代レコードにもなく、インデックス上にもない。したがってこの項目に関しては過去の時間要素を含んだ検索をしてはならない。現在の条件のみである。タイム·インデックス、年齢インデックスは縦に長いテーブルでi番目のエントリーにはレコード番号iのレコードが何時から何時まで有効であったかとそのi番目のレコードが何才のときに対応するものかを示したものである。

タイム·インデックスは3Dマスタ·ファイル1個に1つ作ら

れ、年齢インデックスは年齢インデックス指定のあった項目ごとに作られる。理解し易くするために、ここでは第20図(b)(c)のように書いたが、両者ともより簡単なものにすることができる。

以下 第19、20図を利用し時間を含んだ検索例を示す。

(a) AA＝ア　AND DD＝α AT 86 AND CC≧75

AA＝ア　→{256} *1　→{256, 257, ···, 511} *2 ＝Ⓐ

DD＝α　→{510、511} *1　→{510} *3 ＝Ⓑ

CC≧75　→{256} *1　→{256, 257, ··· 511} *2 ＝Ⓒ

Ⓐ∩Ⓑ∩Ⓒ → 510

*1　INVERTED INDEX を利用して

*2　不変値項目であるから全世代が対象になる。

*3　TIME INDEX を利用して

(b) NOW EE＝345　AND DD＝α' AT AGE 7

EE＝345　→{256} *1　→{256, 257 ---, 511} *4 ＝Ⓐ

DD＝α' →{256, 509} *1　→{509} *5 ＝Ⓑ

Ⓐ∩Ⓑ → 509

*4　現在 EE＝345であるエンティティのすべての世代が対象となる

*5　AGE INDEX を利用して

各エンティティのすべての世代の完全なイメージは容易に作ることができる。第19図に示したエンティティのヒストリーを第21図に示す。

88~	ア	A"	80	α'	345	ABC
87 ~ 88	ア	A'	80	α'	?	?
85 ~ 87	ア	A'	80	α	?	?
83 ~ 85	ア	A	80	α	?	?

第21図　エンティティ256のヒストリー

次に出力の各項目に時間要素を加味した例を示す。

(a) SELECT AA CC EE AT PRESENT

ア	80	345

(b) SELECT BB AT 86, CC DD AT AGE 7

A'	80	α

6. おわりに

ここでは例示的に 3D-DBMS について述べた、筆者らはここで述べたすべての機能を持つ 3D-DBMS を ADABAS 5 をベースにして開発予定である。

参考文献

[1] 石井義興「 情報の単位とその3次元性 」
アドバンスト・データベース・システム・シンポジウム
昭和57年12月(1982)

[2] 穂鷹、石井「 データ・モデリングとデータベース設計 」
アドバンスト・データベース・システム・シンポジウム
昭和58年12月 (1983)

[3] 上林弥彦「 関係データモデルによる設計と比較 」
アドバンスト・データベース・システム・シンポジウム
昭和58年12月 (1983)

[4] 石井義興「 ADABAS 」
bit
昭和59年1月号別冊 (1984)

[5] 石井義興「次世代データベース管理システム 」
アドバンスト・データベース・システム・シンポジウム
昭和62年12月 (1987)