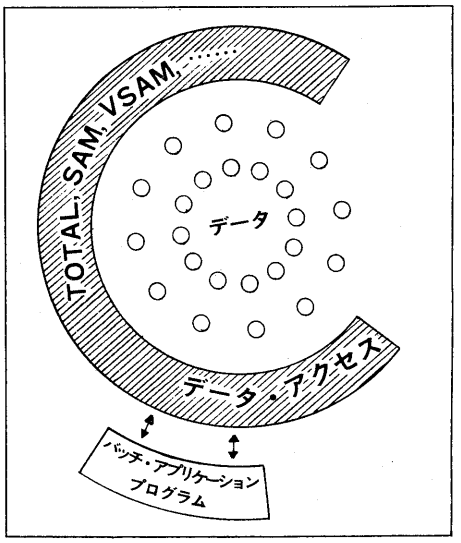
TOTAL

TOTALは全世界で最も多くのユーザーによつて有効的に使用されているデータ・ベース・マネジメント・システム（ＤＢＭＳ）です。TOTALは日本における３０社のユーザーを含め，全世界で1,０００社以上のＩBM 360／370のＤＯＳ，ＯＳ，ＶＳユーザーに有効に使用されています。TOTALはすべての他のDBMSユーザー数よりも多くのユーザーで使われています。例えばＩＢＭのＩＭＳやＤＬ／１ユーザーの２倍，ADABAS，SYSTEM／2000，IDMSといつた特殊目的のDBMSユーザーの１５倍のユーザーをもつています。

セールスマンや学者はTOTALのようなDBMSの効果についてオーバーセールスをよくします。DBMS使用による本当の効果を理解し，それとセールス・トークとを区別するために「すべてのDBMSは単なるアクセス法である」ということを理解することが重要です。



DBMS ……いかなるDBMSでも……現在使用しているシークェンシャル，インデックス，ダイレクトなどの基本的なアクセス法と同様４つの機能しかないのです。つまりDBMSは新規データをファイルに追加し，ファイルからデータを削除し，ファイルのデータを読み・書きします。DBMSを使用する時は，他のアクセス法と同様に，COBOL，PL／I，アッセンブラー，EASYTRIEVEなどの言語を使つてプログラムを作成し，DBMSにどのデータを追加，削除，読み込み，書き込みするかを知らせなければなりません。

それでは今までのアクセス法に代えて，TOTALのようなDBMSを使用することによる効果には，どんなものがあるでしょうか。

これらの効果の詳細について，説明致します。

●同一ファイルをランダムとシークエンシャルで迅速に処理

今まで，バッチとオンラインの処理をする場合に，ユーザーは別々のアクセス法を使うか，１つのアクセス法にして低いパフォーマンスを我慢するかのどちらかを選ばなければなりませんでした。シークェンシャル・アクセス法はバッチ処理アプリケーションに最適ですが，オンライン・アプリケーションには実用できません。

処理＼アクセス法	シークエンシャル (SAM, BSAM)	インデックス シークエンシャル (ISAM, VSAM)	ダイレクト (DAM, BDAM)
バッチ	最良	中途はんぱ	最悪
オン・ライン	最悪	中途はんぱ	最良

ダイレクト・アクセス法を使えばオンライン　アプリケーションに必要な速い応答を得ることができますが，バッチ・

アプリケーションには効率が非常に悪いので使えません。

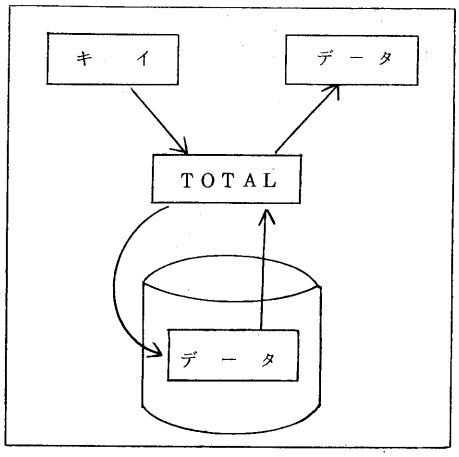
従つて，バッチとオンラインの両方の処理をしているユーザーでは，次のどちらかを強いられることになります。

＊両処理のため２つのファイルを要する。

バッチ処理用のシーケェンシャル・ファイルおよびオンライン処理用のダイレクト・アクセス・ファイルの両方のメインテナンスが必要である。

＊高いパフォーマンスを得ることが不可能。

バッチおよびオンライン処理用にインデックス・シーケェンシャル・ファイルを使うことができる。しかし，両処理のパフォーマンスは共に低く，かつ絶えずファイルの再編成を必要とする。



TOTALはこの問題を解決します。TOTALはバッチとオンラインの両アプリケーションに使われるファイルをアクセスするために設計されており，その両方の使用法において最適のパフォーマンスを発揮します。各々のTOTALファイルをランダムにも，そして，シーケェンシャルにもアクセスすることができるのです。

TOTALファイルやデータ・ベースを直接アクセスするため，ユーザーは求めるデータのキイをTOTALに渡します。TOTALはこのキイからデータが格納されているアドレスを計算し，そのデータを直接アクセスします。TOTALの処理効率は非常に高いといえます。例えば，割り当てたディスク・スペースの余裕率が８～１０％のときでも，データの１項目をアクセスするのに９５％の場合，１回の物理的Ｉ／Ｏ で十分です。この点に関して，ＩSAMとVSAMを比較してみましょう。

1. 求めるレコードが格納されている場所を見つけるためにインデックスを読み込む。この時に１回の物理的Ｉ／Ｏを必要とする。
2. 求めるレコードがその場所にあるかどうかを調べ，そのレコードがある場合は読み込む。この時に１回の物理的Ｉ／Ｏを必要とする。
3. その場所にレコードがない場合には，更に１回またはそれ以上の物理的Ｉ／Ｏ を使つて実際にレコードをアクセスする。

シーケェンシャル処理の場合，TOTALはレコードを物理的順序にアクセスします。つまり，最初のシリンダーの最初のトラックの最初のブロックから，最後のシリンダーの最後のトラックの最後のブロックまでシーケェンシャルに処理します。従つて，TOTALファイル全体を処理するには各シリンダーを１度だけシークし，各ブロックを１度だけ読めばよいのです。この点に関して，ＩSAMやVSAMと比較してみましょう。

1. ＩSAMやVSAMではレコードが格納されている場所を見つけるために常にインデックスをシークする必要がある。
2. レコードが格納されている物理的順序がレコードのキイの論理的順序と異る場合には，同じシリンダーを何回もシークし，同じブロックを何回も読み込む必要がある。

ベンチマーク・テスト

TOTALとISAMの代表的ベンチマーク・テストの結果を示す。このテストではレコード長４００バイトの５０００レコードから成るファイルをシークェンシャルおよびダイレクトに処理した。

	ISAM	TOTAL
全レコードをシークェンシャルに読む		
経過時間	64.34分	7.55分
CPU時間	363秒	18秒
全レコードをダイレクトに読む		
経過時間／レコード	102.2ミリ秒	41.0ミリ秒

●容易な可変データ処理

どのような会社のデータでも，２つの種類に分類することができます。

1. 「物」に関するデータ

これは会社が存在するから，その会社に関するデータが存在するという種類です。例えば，次のような物に関するデータをいいます。

＊ 従業員
＊ 得意先
＊ 納入業者
＊ 原材料および部品
＊ 製品

2. 「活動」に関するデータ

２番目の種類は会社が商売をするから，その営業活動に関するデータが存在するというものです。例えば，次のような活動とトランザクションに関するデータをいいます。

＊ 販売受注
＊ 購買発注
＊ 生産指図
＊ 労働記録票
＊ 請求
＊ 支払
＊ 領収

活動に関するデータは常に物に関するデータと関係しています。またその逆も同様のことがいえます。例えば，各販売受注は１つの得意先と１つまたは複数の製品に関するトランザクションです。１枚の請求書は１つの得意先に関係します。１つの労働記録票は１人の従業員に関係します。

活動に関するデータの変化は一様でなく，その予測もできない場合が多いため，その取り扱いは難しいといえます。

例えば，ある月に得意先Aは多くの注文を出し，得意先Bは注文を全然出さないかもしれません。翌月，得意先Aは少ししか注文を出さず得意先Bは多くの注文を出すかもしれません。このように活動データの変化は多様であり，その予測も困難であるため，格納スペースの効果的使用が難しく，かつアプリケーション・プログラムが複雑になってきます。

TOTALの主な利点はこのような可変データの取り扱いを簡単にすることです。TOTALはデータを２種類のファイルに分けます。

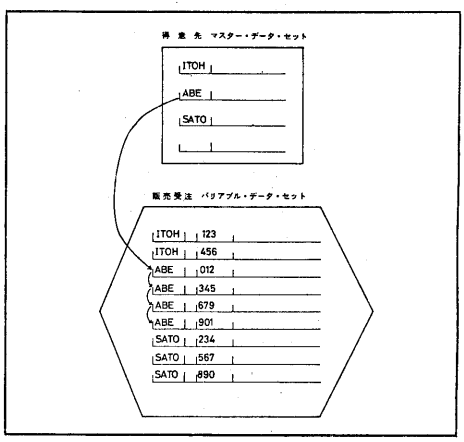
＊　マスター・データ・セットは物に関するデータを格納するために使われる。

＊　バリアブル・データ・セットは活動に関する（可変）データを格納するために使われる。

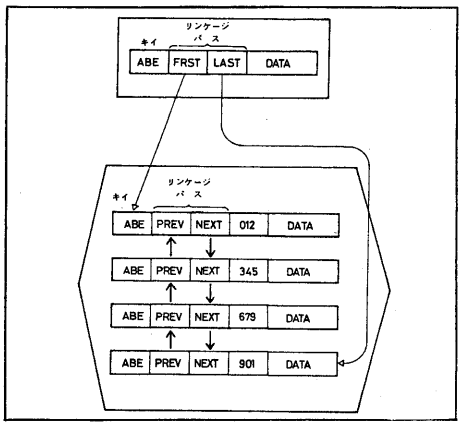
「物」をそれが関係するすべての「活動」と結びつけるために，TOTALはマスター・データ・セットのレコードをバリアブル・データ・セット内のそれが関係するすべてのレコードに関連づけることができます。例えば，得意先ＡＢＥのマスター・レコードはその得意先からの注文を表わすすべてのレコードに関連づけることができます。これにより可変データの取り扱いが次のように容易になります。

＊　ディスク・スペースの利用効率が良くなる。
注文がない得意先には販売受注レコードが作られないのでスペースのムダがない。
1,000や10,000もの注文があるとしても，その得意先には各注文当り１レコードの販売受注レコードが作られるので，それらのすべての情報を格納することができる。

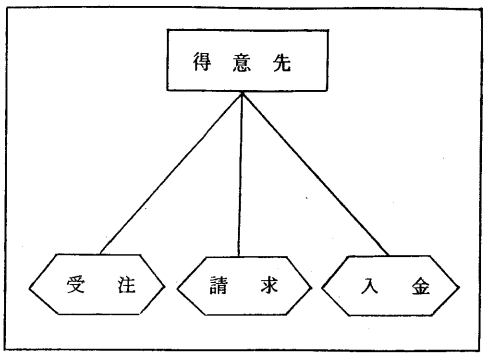
＊　TOTALには簡単で強力なコマンドがあり，これを使って関連する一連のバリアブル・レコードを処理することができるのでプログラミングは容易である。



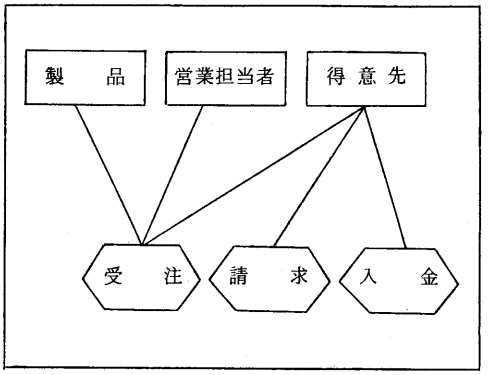
また，１つの「物」を多くの「活動」に関連づけることができます。例えば，得意先レコードをその得意先からの受注，その得意先への請求，そしてその得意先からの入金と関連づけたいとします。このような場合，TOTALでは１つのマスター・データ・セットにいくつものバリアブル・データ・セットを関連づけます。



同様に，１つの「活動」を多くの「物」に関連づけることができます。例えば，受注を，発注した得意先，その受注を受けた営業担当者，そして受注した製品と関連づけたいとします。このような場合，TOTALでは１つのバリアブル・データ・セットにいくつものマスター・データ・セットを関連づけます。



TOTALは自動的に関連づけのためのデータ（リンク情報）を作り，あるファイルの「物」レコードを他のファイルの「活動」レコードに関連づけます。レコードを追加・削除した場合に，TOTALは自動的にこのリンク情報を変更・維持します。ユーザーに求めることは，TOTALに新しいファイルを追加する場合に関係を定義することだけです。



●データの相互関係づけ

TOTALでは最高65,000のファイルを１つのデータ・ベースに組み入れることができます。そして，ファイル間の関係を定義する時に，どのファイルのレコードも最高25,000のファイルのレコードと関係ずけることができます。例えば，次の図は１１のファイルから成る典型的TOTALデータ・ベースです。このデータ・ベースの利点をいくつか挙げてみましょう。

1. データを複数のキイでアクセスできます。例えば，販売受注レコードを４つの方法でアクセスすることができます。
   * 特定の得意先からのすべての販売受注
   * 特定の日に受けたすべての販売受注
   * 特定の日に出荷すべきすべての販売受注
   * 特定の製品に関するすべての販売受注

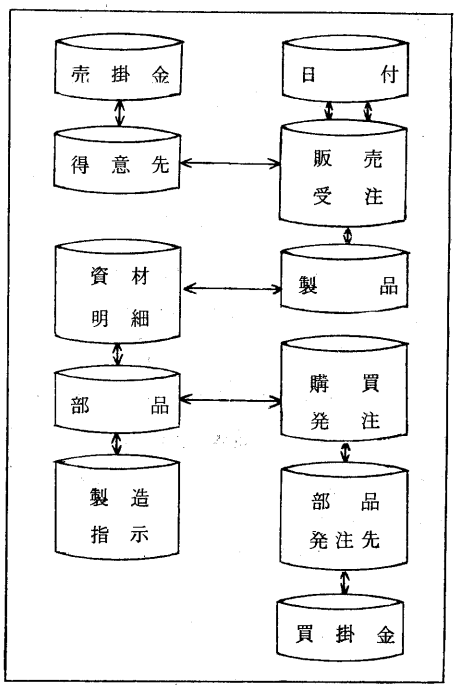
2. データ・ベースのデータをアクセスするときに，プログラマーはそのデータがあたかも１つのファイルにあるかのようにアクセスすることができます。従って，プログラム作成は容易になります。例えば，全製品に関する現在の売掛金レポートを作りたいとします。この場合，先ず製品ファイルを順次に処理し，各製品毎に販売受注ファイル，得意先ファイルそして売掛金ファイルのレコードをたどっていき必要なデ

ータをアクセスすることができます。このためのプログラムを書くのは非常に簡単です。

数種の製品のみに関する同様のレポートを作るのであれば，先ず製品のキイを与えて製品ファイルをランダムに読み，同様にレコードをたどつていき必要なデータにアクセスすることができます。

3. 同じファイルをいくつものアプリケーションに使うことができるので，重複データを減らすことができます。例えば，販売受注ファイルは次のようなアプリケーションに利用できます。

   * 得意先，受注日，出荷日，製品ファイルに関する受注アプリケーション
   * 得意先，売掛金ファイルに関する請求書発行アプリケーション
   * 製品ファイルに関する在庫管理アプリケーション
   * 出荷日，得意先，製品ファイルに関する出荷業務アプリケーション



4. これらの利点は，TOTAL が複数のファイルを論理的に関係づけることにより実現されています。TOTALはいくつものファイルを物理的に１つに結合するわけではありませんので，TOTALデータ・ベース内の各ファイルは物理的に別々のファイルになつています。

   * データ・ベース内の各ファイルをそのデータ・ベース内の他のファイルとは無関係に独立して処理することができる。
   * もしあるファイルが物理的に破壊されたり，論理的に損傷を受けた場合，復旧すべきものはデータ・ベース全体ではなく，そのファイルだけである。
   * もしデータ・ベース内のあるファイルに変更が必要になつた場合（例えば，新しいアプリケーションのために新しい項目を追加するなど），そのデータ・ベース内の他のファイルに影響はない。

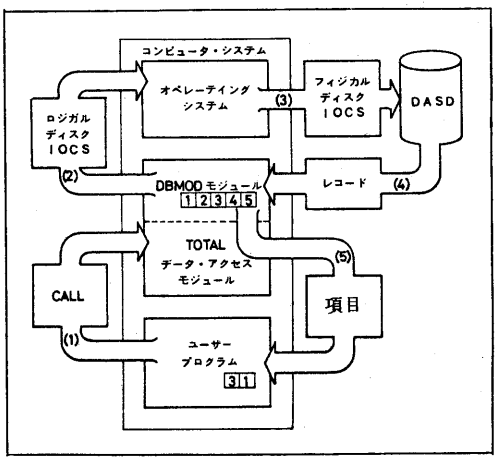
このようにTOTAL はいくつものファイルを論理的に結合し，データの相互関係づけの利点を作り出しています。一方，各ファイルは物理的に別々になつて（すなわち，互いに独立して）いますから，ファイル単位にデータ・ベースを変更・拡張することが容易であり，柔軟性のある構造になつています。

●アクセスする単位はレコードでなく項目

ほとんどのアクセス法を使う場合，ユーザー・プログラムは先ずレコード全体をアクセスし，次に処理に必要な項目を取り出します。このように，処理に必要な項目はレコード内のごく一部であったとしても，プログラムはレコード全体と密着しております。

従って，レコードに少しでも変更があればそのレコードをアクセスしているすべてのプログラムを修正しなければなりません。

TOTALを使う場合，ユーザー・プログラムはレコード全体ではなく個々の項目をアクセスします。次の図に示すように，5つの項目から成り立つレコードがあったとします。ユーザー・プログラムはそのレコードの3番目と1番目の項目のみを要求したとします。



先ず，オペレーティング・システムはTOTAL内にレコード全体を読み込みます。しかし，TOTALがユーザー・プログラムに渡すのは要求された2つの項目だけです。この利点を挙げてみましょう。

1. ユーザー・プログラムはレコード全体でなく，実際に使用する項目にのみ関係しています。このことは新しいアプリケーションのためにレコードやファイルを変更する時に，今まで避けることのできなかった既存アプリケーションの変更を必要としないことを意味します。すなわち，実社会において避けられない事実「成長と変化」をTOTALはこのように解決しているのです。

2. TOTAL を使用してユーザーはいくつものファイルにあるデータを関係づけ，レコード全体でなく個々の項目をアクセスすることができます。従って，ユーザーはデータ・ベースを「自分に必要な項目のみが内部的に関係づけられた形で1つにまとまっているデータ」のように扱えます。これにより新アプリケーション・システムの設計やプログラムの開発が大幅に簡単になります。