ＩＮＱ（INformation Query）について。

後藤龍男，土居嗣典
（日本電気株式会社）

1. はじめに。

ＩＮＱ（INformation Query）は情報検索に対する効率と柔軟性を目標に開発されたＤＢＭＳである。１９７６年夏に最初のバージョンが市場に提供され，現在幾つかのユーザのもとで稼動している比較的新らしいシステム・ソフトウエアである。

ＩＮＱの機能仕様の決定はある明確な方針にもとずいて行なわれた。すなわち概念の体系化や理論化より利用者にとって何が現実に有用であり不便であったかという様な視点を設計のテーマとしたことである。

元来，ＤＢＭＳを利用する立場からの利害は必ずしも一様ではなく，そこから発生するＤＢＭＳに対する期待や要望を体系化し概念統一することは極めて難かしい。多くの過去のそう云った努力が何かと現実から遊離しがちであったのも無理からぬことと思われる。しかし，ＤＢＭＳが真に実用的でありうるためには常に求められていることの因，何が真に有用であるのかと云った問題認識が中心に置かれなければならない。

2. ＩＮＱの設計目標。

ＩＮＱの設計目標は過去のデータベースシステムに関する実際経験と反省の中から取捨選択し決定された。その基本点は次の様なものである。

①使用にあたっての容易性。
②運用の柔軟性。
③問題に対する即応性。

2.1 容易性について。

過去において従来的なファイル管理レベルからＤＢＭＳへの移行を試みたユーザは当初ある量的の困難さを感じて来た様に思われる。その結果ＤＢＭＳの効用を認識しながら導入にふみきれなかったケースも少なからずあるであろう。その要因は多様であろうが重要な点は共通している。

それはユーザが保有する情報構造をＤＢＭＳが規定するいわゆる「データ構造」へ再編成する作業にある。ほとんどのＤＢＭＳはデータの統合化，一元管理，アクセス手段の効率化のためにデータ格納様式の構造化表現をユーザに強要する。これらは網構造であったり木構造であったりするが，ユーザにとって決して親密な概念とは云えない。その理由はデータ構造と云うもの自体がＤＢＭＳにとっての便宜であって本来的な情報構造の表現様式ではないことにある。例えば階層木構造システムにおける上位セグメントと下位セグメントの階層関係や，ＤＢＴＧ仕様でのレコードタイプ間のセット関係など，構造表現と云う観点からは簡明で強力なものであろうが，ユーザが自分の持つ情報群をその枠組で捉えようとするともはや直観的な理解にもとずく作業ではなくなる。このことがいわゆるデータベース設計を決して容易とは云えない作業にしてしまう原因になる。

データ構造は何通りも考えうるし，ユーザにとってその内どれが最適な選択であるのか判断がつかなくなる。

ユーザにとって最も親密であり直観的把握の容易なデータ格納様式は旧来からの順ファイルであろうと思われる。このことからINQでは基本的なデータ格納様式として順ファイル形式を採用することにより便用容易性を計っている。順ファイル形式そのものはデータ構造表現とはなりえないが，その中にデータの関連を平坦な表現で内包させ応用プログラムに提供することを試みている。

第2はデータベース管理の統合化の問題にある。多くのDBMSはデータ管理の統合化と一元化をデータベース化の主要目的にして来た。冗長なデータの存在はよからぬことであり，管理の統合化こそが進歩の要点であるとされて来た。そのためにデータベース管理者の役割や権限がいっそう強調される様になる。データベース管理者はもはやデータ管理だけでなくそれを通して運用組織の管理にまで発言権を有する様な立場にまでもち上げられぬばかりのいきおいである。

この様な傾向はデータベース導入を企てるユーザにとって負担の大きい要請であることも多いであろう。INQではデータ管理を統合化，一元化と云う観点でとらえず，データを必要とする利用者がそれらを個別に作成し管理すると云う前提に立ち各種の機能を用意する。このことはデータのそれほど厳密な一元化や統合管理を必要としないユーザにとってDBMSの導入を容易なものにする。

2.2 柔軟性について。

DBMSが元来高度で多様な機能により ユーザに便宜と柔軟性を与えることを目的としているにもかかわらず，現実にデータベース化が変化に対し適応しずらいことが多い。その最大原因はデータベースの追加／変更作業が云われる程に容易なものではないことである。その隘路は前項に共通するデータ構造の固定化にある。少し規模の大きなデータベースになると全体の再構成／再編成作業などよほど致命的な状況が起らぬかぎり実施する気にならないと云うのが正直な所であろう。この困難さは作業に対して特別な専門家の存在を要求するし，データベースの維持管理を限られた人々のものにしてしまう結果を生む。

INQはデータベースの中に構造を固定化してしまうことを避け，データベースに対する追加／変更の柔軟性を確保することを試みている。

前項の容易性とあわせて，INQはその便用や運用に際して特別に専門的なトレーニングや経験をさほど必要とせずデータベース化への移行ギャップを低減させる。

2.3 即応性について。

問題（応用プログラム）がDBMSに期待するアクセス形式はある程度のデータ量について連続した処理の効率を重視する場合とある選択基準のもとでデータ検索の応答性に重きを置く場合とがある。この両者に要する機能は端的技術上は2律背反的と云える。

INQはどちらかと云うと検索指向の非定型処理に目標を置いて，その機能と処理性能の最適化を計ることとした。そのためファイルの基本的な物理構造に部分インバーテットインデックス形式を採用している。またエンドユーザ向けの即応性の高い言語インタフェースにも重点を置いている。

3. 機能と構造。

3.1 データベース全体の論理構造。

ＩＮＱデータベースはＩＮＱファイルの集合として構成される。ＩＮＱファイルはデータベース内に最大６３個まで許され，データベースはシステム内に３００個まで登録できる。基本となるＩＮＱファイルはそれ自体で閉じた情報集合体であり，ＩＮＱファイル間には何らの論理的／物理的関連をもたない。強いて関連を求めればファイル内に格納されたデータの値に依存する潜在的なものでしかない。

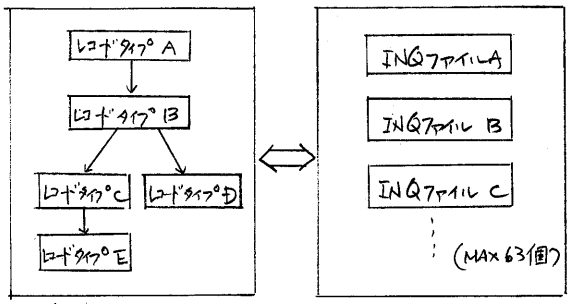

図3.1.1 全体構成の比較

このことは多くの統合型データベースシステムがレコード間（ファイル間）構造として論理的／物理的関連をデータベース内に実体化しているのに較べ最大の特徴であると云える。

このことはデータベースシステムにとって次の様な利点を生む。

①データベース拡張の容易性 ---新たなＩＮＱファイルの追加は他のＩＮＱファイルに影響を与えない。

②データベース保守の柔軟性 ---ＩＮＱファイル単位の破損が他のファイルに影響しない。

統合型データベースの場合新たなレコード（ファイル）の追加は他のレコードとの関連を構造化することを意味するので，その作業は全体的なデータ構造についての見通しが必要となる

3.2 ＩＮＱファイルの論理構造。

1個のＩＮＱファイルは１つの情報主体とその主体に関する情報群に対応して実体化される。すなわちＩＮＱファイルは主体毎に存在し，その情報群がその中に格納される。主体とは情報集合を特徴づける集合体名に相当すると考えても良い。

例えばファイル「地方自治体」は北海道，青森，岩手 --- など全国都道府県集合の集合体名であり，日電，東芝，日立 …… に対する「会社」が集合の情報主体である。そして主体の要素がＩＮＱファイル内の各論理レコードに対応しその各種の情報がデータとしてレコード内に格納されることになる。

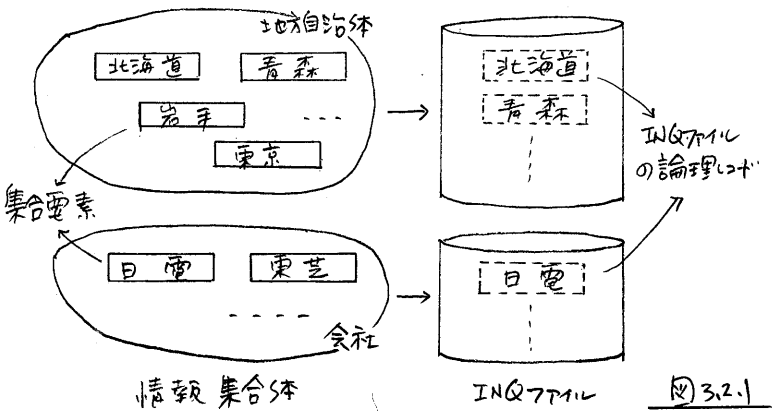

図3.2.1

1個のＩＮＱファイルは複数の主体をあらわすことはない。それ故，ＩＮＱファイルに対する検索は主体で示される集合に対するその全要素中から条件を満足する要素（論理レコード）を選択する操作というパターンをとる。

例えば　条件；“人口８００万人以上の地方自治体は？” ⟶ 結果；論理レコード“東京都”の選択。

多くの統合型データベースが多目的指向であるのに対し，ＩＮＱファイルは上記の意味においてある１つの目的を指向した目的向きファイルであると云える。ＩＮＱデータベース全体として多目的であることを望む場合，各目的向きファイルであるＩＮＱファイル間にデータを重複させる結果となる。統合型システムがデータの一元化を基本的な目標にするのに反し，ＩＮＱはそのことに必ずしもこだわらない。データベースの格納効率や無矛盾性を追求するあまりのデータの冗長性排除はデータ管理者に便宜をもたらしても，エンドユーザにとっては大した利益ではない。

ＩＮＱファイルはそれを必要とするものが作成し維持することを前提に考える。それ故異なるファイル間にデータの重複や相互矛盾がある場合でもデータベース全体の問題とはせず当事者間で解決することになる。

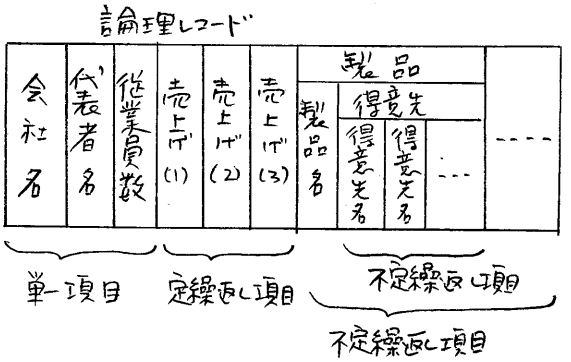

図3.2.1 論理レコードのレコード内構造．

ＩＮＱファイルは要素（レコード）についての順ファイルに近い形式である。レコードには要素に関する種々の情報がデータ項目として格納される。

またレコード内構造として単一項目の他に定繰返し（３次レベルまで）や不定繰返し（２５次レベルまで）項目をもつことができる。

この様にＩＮＱファイルは(図3.2.1)に示される様なレコード内構造を有する平坦な順ファイルとして表現される。レコード内構造としてもつ定／不定繰返しは階層型データベースの階層構造と等価な表現でもある。繰返しのネスティングは階層が多段であることに対応する。

通常の階層型システムにおいてもある応用プログラムが必要とするレコード型の数と階層構造の規模はさほど大きなものでなく，ある程度の規模で大半の問題は充足してしまう。ＩＮＱシステムのもとでは応用プログラムが必要とするデータ実体は基本的に一つのＩＮＱファイルに包含してしまうことを目標にしている。

応用プログラムが２個以上のＩＮＱファイルとその相互関係（ファイル間構造）を必要とするケースに対しＩＮＱは動的なファイル結合機能と仮想ファイルを提供する。

3.3 データ項目とその属性。

データ項目はその属性によって次の様に分類される。

データ項目の属性 ---(a) データ構造に関する属性。
単一項目／可変項目。
基本項目／配列項目／集団項目。
(b) データ操作に関する属性。
検索項目／表示項目。
仮想項目，BY項目。
(c) データ形式に関する属性。
文字型，整数型，実数型，倍精度実数型，数字型，演算符号付数字型，-----

(1) 単一項目；1つのレコードについて1つの値をもつデータ項目である。
(2) 可変項目；1つのレコードについて複数個の値をもつデータ項目ある。
(3) 基本項目：最小単位データに対するデータ項目である。
(4) 配列項目：定繰返しをなすデータ項目である。
(5) 集団項目：複数個のデータ項目の総称としてのデータ項目である。
(6) 検索項目／表示項目：検索項目は検索キーとなりうるデータ項目であり，システムが項目毎にインバーテット・インデックス（逆索引）を設ける。それに対し表示項目はデータの表示だけに用いられる項目であり，逆索引が設けられることはない。

INQシステムでは検索キーとして用いられる可能性がある全てのデータ項目は検索項目として扱われ，その数に制限はない。検索キーとなる可能性のない項目についてだけ表示項目であることを陽に宣言し，逆索引に要する2次記憶スペースを節約する一いわゆる部分インバーテッド・インデックス・システムである。

(7) 仮想項目：その値が他のデータ項目の値や構造情報により決定される項目でありその値自身は実際にデータベース中に格納されない。値の決定方法は，INQファイル定義時にあわせて定義する。その値の管理はシステムが行なう。
決定方法は次の4通りである。
①計算式---データ項目，定数，四則演算子，カッコで表現する計算式にもとずく。
②合計値---可変項目の値の合計値による。（TOTAL）
③合計数---可変項目の繰返し数を値とする。（COUNT）
④平均値---可変項目の平均値による。（MEAN）

(8) BY項目：複合条件検索における限定（Qualification）を指定された項目である。

一般に階層構造に対する条件検索において複合条件の意味にある種のありまいさを含む場合がある。例えば下図に示される様な構造に対する条件"製品としてカラーテレビをもち，売上額が1億円以上の会社"は次の様な2通りの意味にとれる。
①カラーテレビの売上げ額が1億円以上の会社。
②全製品の売上げが1億円以上の会社。

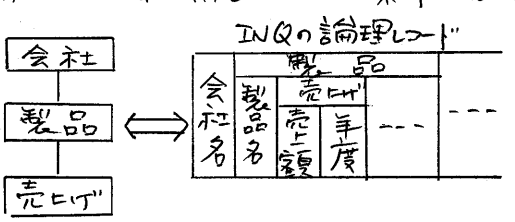

図3.3.1 階層構造

前者の様に階層下位の項目についての条件で上位の条件を限定するためには，検索言語自体に限定機能が必要となる。 INQでは下位の繰返し項目に対する条件により限定されたい場合，これをBY項目として定義しておく。

BY項目と指定した場合，BY項目（製品）と被BY項目（限定している項目すなわち 売上げ額）についてはその各々の逆索引の間に階層関係が作られる。

3.4 INQセクション記述と仮想INQファイル。

INQデータベースは個々の実INQファイルを記述するデータ定義言語（FDL）になって定義される。 これとは別に応用プログラムに対しては別の記述であるINQセクション記述を与えている。 INQセクション記述は実データベースの応用プログラムにとって必要な部分を仮想的なファイルとして表現するものである。

INQセクションは大別して2通りのファイル仮想化を行なう性格をもっている。

① INQファイル内構造のサブセット化。

② 異なった2つ以上のINQファイル間構造の具体化。

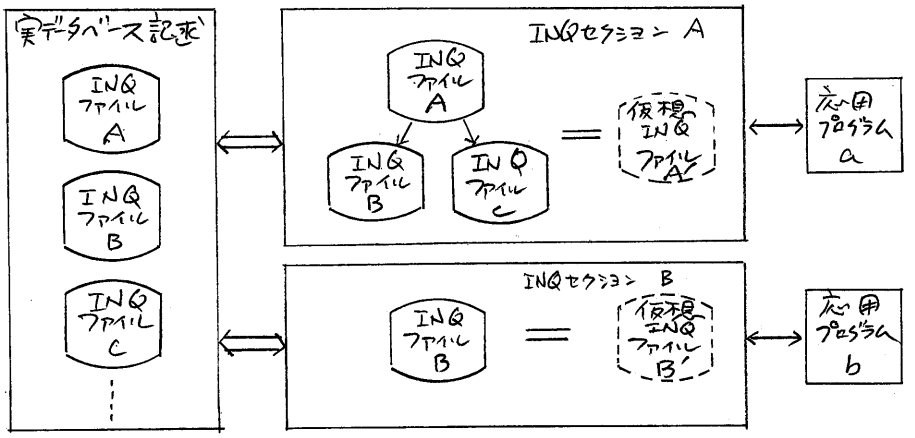

図 3.4.1 INQセクションと仮想ファイル。

FDL記述とINQセクション記述の概念はDBTG仕様におけるスキーマ／サブスキーマの関係に類似しているが，次の点で本質的に異なっている。 すなわち，サブスキーマがデータベース全体記述であるスキーマの単純なサブセット化（部分の切り出し）であるのに対し，INQセクションはINQファイルのサブセット化のみならず複数のINQファイル間に新たなファイル間構造を設定して新たなINQファイル（仮想INQファイル）を生成する機能をもつ。

DBTG仕様を代表とする統合型システムでは，データ構造（特にレコード間構造はあらかじめ固定的に実体化せざるを得ず，問題毎に必要とする構造を最適化することは不可能であって，データ構造の問題適合性にかたよりが生ずることを避けられない。 これは統合型システムでのデータベースの存在の仕方がセットなどのレコード間構造によってしまっていることによる。

INQではINQファイルはそれ自体独立に存在するものであり，ファイル間の関連は応用プログラムが必要とする時点で実体化される。 つまり問題毎に最適化された構造がINQセクション記述で提供される。

INQセクション記述の実データベース（FDL記述）への写像はシステムが管理する。応用プログラムに見えるのは仮想INQファイルA', B'（図3.4.1）であり，両記述の距離によってデータとプログラム間の高い独立性が確保される。

3.5 ファイル結合。

前項に述べたINQセクション記述の内，INQファイル間構造の生成機能を"結合"とよんでいる。2個のINQファイルの結合はDBTG仕様のセット関係などのレコード間構造とほぼ等価であるが，次の2点でそれらとは本質的な相異がある。

①結合関係はあらかじめデータベース内に構造化される必要はない。

②結合はファイル間構造として応用プログラマに見せることなく，一般的なINQファイルと同じレコード内構造をもつ新たなINQファイル（仮想INQファイル）として提示される。プログラマは結合を何ら意識する必要はない。

結合は2個の独立したINQファイル同志で意味的に共通したデータ項目に着目して行なわれる。

図3.5.1は会社ファイルと製品ファイルの間で「製品名」というデータ項目について結合した例を示している。

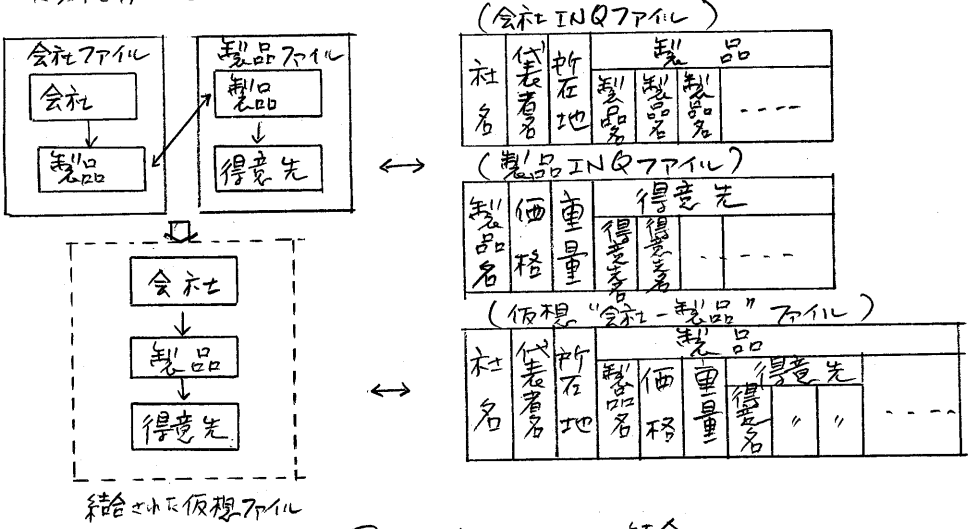

図3.5.1 ファイルの結合。

会社ファイルと製品ファイルはINQファイルとしてデータベース内に実在するが，結合結果の新たな"会社-製品ファイル"は仮想INQファイルである。

結合操作は階層上位のINQファイルの中に下位のファイルを繰返しとして収包させてしまうことに相当する。この様にINQでは応用プログラマに対してあくまでも平坦なファイルを提供する。

結合の対象となる項目（製品）は特定な構造属性を要求されず，応用プログラムにとって意味があれば何であっても良い。データ値のみに関する操作である。

1つのINQセクション内で最高4個までのINQファイルを結合することができる。

一般的に結合（ファイル間構造の実体化）の手法には次の3通りが考えられ，それぞれ長短がある。

① レコード自体に結合のための写像情報（ポインタなど）をもたせる。結合は固定され柔軟性はないが処理効率は高い。
② 写像情報をレコードとは別に保有する。写像情報を変更すれば結合も変えられるが，レコード内容の変更は写像情報の更新も必要とする。
③ 写像情報はもたず結合操作を何らかの写像アルゴリズムとして実現する。結合は固定されず動的に設定しうる。レコード内の値の変更自体が写像情報の変更に相当することになる。

ＩＮＱシステムの結合はこの内第３の方法によっている。この選択の最大の理由は柔軟性とデータベース管理の簡易化にある。反面結合ファイルを参照する場合プログラム手続による写像処理をともなうので処理効率にとって有利な選択ではない。このトレードオフは次の根拠によっている。すなわち，問題（応用プログラム）が必要とする情報は１つのＩＮＱファイルの範囲で完結するべきであり，また大半の問題はそれで充足するはずである。１個のＩＮＱファイルでは満されない比較的まれな問題に対しては結合が適用されるが，その場合処理効率はさほど重要でない。（重要視する様な頻度の高い処理対象であるなら，はじめから拡張したＩＮＱファイルを持つべきである。）

この様なアルゴリズムによるファイル間構造の実現手法は，将来分散型データベース環境を想定した場合，物理的に離れて置かれた２つのファイル間の関係をデータベースプロセスに対し実体化し提供するための手段となる可能性をもつ。

3.6 ＩＮＱファイルの物理構造。

ＩＮＱファイルの物理構造は部分インバーテットインデックス形式によっている。ＩＮＱファイルの格納される物理的な記憶空間は一定の大きさのページにより構成され，ページ単位の入出力制御により入出力効率を高めている。

ＩＮＱファイルは次の３つの部分で構成される。

① ディスクリプタ部 ---- ＩＮＱファイルの論理的構成要素であるデータ項目属性情報やデータ項目間の構造情報が格納される。
② インデックス部 ----- 検索項目毎の逆索引が格納される。（図3.6.1）
③ 実データ部 ------- レコードの各データ項目の値が格納される。レコードは一意のレコード番号（ＲＮ）をもつ。（図3.6.2）

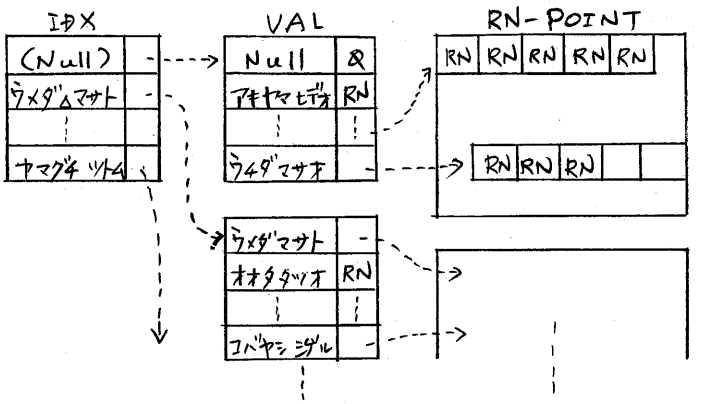

図3.6.1 インデックス部の構造。

3.7 検索とその操作言語。

INQは基本的なデータ操作言語として親言語方式のDMLをもつ。その中の検索機能(RETRIEVE)のレコード選択基準は論理式からなる複合条件にもとずく。条件式は図3.7.1の様な比較オペレータ，論理演算オペレータにより記述される。

次の様な複合条件式にもとずく検索を考える。

"名前"=サトウ AND "年令" > 30才

検索操作は"名前"と"年令"の2つのデータ項目のインデックスを独立に参照し条件を満たす2つのレコードの集合が選択される。

比較オペレータ

機能	オペレータ	
一致	=	EQ
近似的一致	::	NR
不一致	≠	NE
より大	>	GT
より大，一致	≧	GE
より小	<	LT
より小，一致	≦	LE

論理演算オペレータ

機能	オペレータ
論理和	OR
論理積	AND
否定	NOT
演算順	( )

図3.7.1

この集合は実データではなく2進法現によるレコード番号(RN)の集合である。論理演算オペレータで表現されている複合条件の操作はこのレコード番号集合間の集合演算に置きかえられる。

レコード番号群はその値が対応する位置のビットを1としたビットストリングに変換されストリング間の2進演算操作に置き変えられる。この値は図3.7.2の様な3階層から成る論理演算用テーブルの対応ビットにセットされ，このテーブル間で論理積の処理が行なわれる。処理は最上位テーブルから順に演算が行なわれるので，不必要な下位テーブルのアクセスが省略でき高速化が計られる。

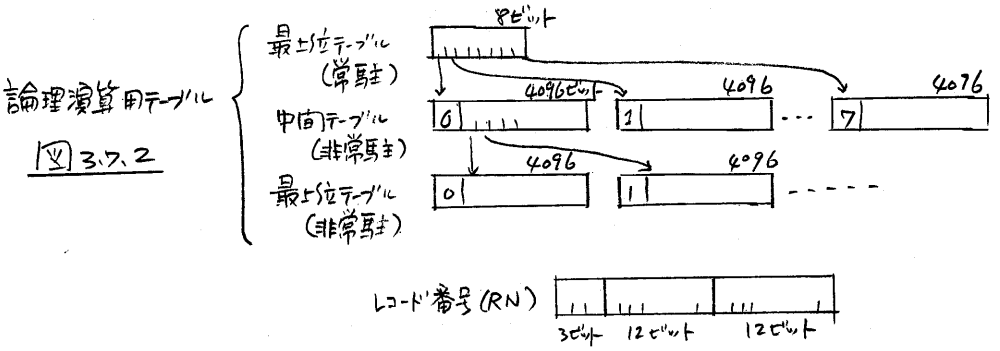

図3.7.2

複雑な条件検索も操作は全てインデックス部と論理演算テーブルのみでなされ実データ部を参照することがなく極めて高速な応答が実現できる。

3.8 データの圧縮と暗号化。

インバーテット方式の難点はデータ格納効率の問題である。INQはデータ格納時の圧縮でこの問題を実質的に解消している。データ値はその属性が文字タイプの場合は指定された文字(ディフォルトはブランク)で，また数字タイプの場合はそれが整数型，実数型の場合を除いて，先行する0がそれぞれサプレスされてパックドデシマルの形式で格納される。

またデータ格納時にユーザの指定によりデータを暗号化し格納することができる。

4. ＩＮＱの構成

ＩＮＱシステム全体は図4.の様な要素から構成されている。この内言語インタフェースは2種のデータ定義言語（ＦＤＬ，ＩＮＱセクション）と3種のデータ操作言語（ＤＭＬ，ＲＰＬ，ＥＵＬ）およびデータ管理用言語（ＤＳＬ）を用意している。ＩＮＱセクション記述とＤＭＬは木スト言語に依存するが他は独立言語である。ＥＵＬは非プログラマ向けの言語インタフェースであり，ＴＳＳ端末からの会話型の検索とデータ加工処理を目的にするものである。

ＩＮＱはＡＣＯＳシリーズの大型機種であるＮＥＡＣシステム６００，７００，８００，９００とそのオペレーティングシステム　ＡＣＯＳ-６のもとで実動する。

5. その他の特徴的機能。

(1) データベースにとって重要な機能である機密保護については前述したデータの暗号化と項目毎に設定できる２０レベルのパスワード機能が用意される。

(2) ＩＮＱの重要な応用システムとして文章情報の処理を想定し，各種の便宜を計っている。文章情報の格納形式には単語単位で不定繰返し項目に収容する方法が有効である。データ圧縮は漢字コードを想定して２文字単位の処理も行なう。

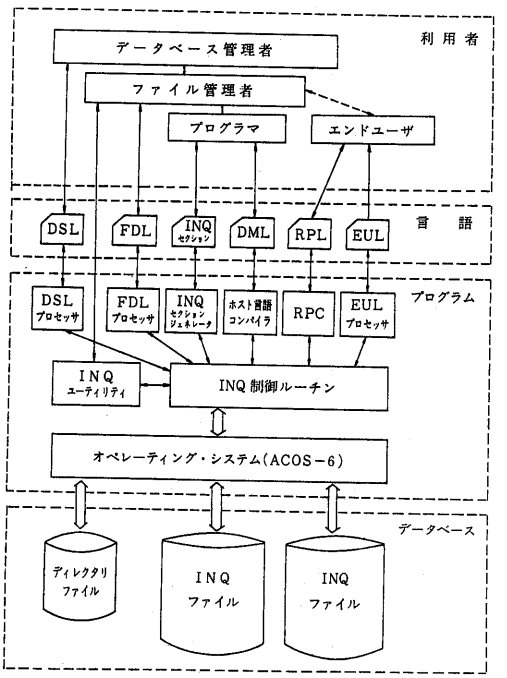

図4. ＩＮＱの全体構成。

6. あとがき。

最近関係モデルに関する議論が盛んである。ＩＮＱの開発を結果として見ると種々の点で関係モデルに似ている様に感じる。ＩＮＱファイルの平坦さは"関係"に当り，結合はＪＯＩＮ操作に似ているし，検索処理を集合演算で表現する点も共通している様に見える。しかし基本思想は全く異質であると云って良い。すなわち関係モデルではデータ集合としての"関係"に完全に論理的な演算操作にとって閉じていることが要請されるため，"関係の正規化"が不可欠であることである。結果として情報集合は沢山の全く平坦な"関係"に分割されてしまう。関係データベースの使用者はその沢山の"関係"を認識してかからねばならないであろう。これはまた別のやっかいさをまねくことにもなりかねないと考える。

〔参考文献〕

ＩＮＱ概説書
〃　文法説明書
〃　運用説明書
〃　エンドユーザ言語説明書