データベース設計過程と データベース設計支援ツール

富士通　高橋　浩

1. はじめに

最近，汎用のDBMSの普及とともに多くのデータベースが作成され運用されるようになってきた。しかしデータベースを作成する際のデータベースの設計法は個々の場合に独立に行なわれている状態であり，まだ設計法の確立は見られていない。

データベース設計の仕事はシステム分析と深く係わるため極めて概念的作業がある一方，現実のDBMSを使いこなすには詳細なDBMSの特徴も知った上でないと適切な使用法が見い出せないため，概念レベルからDBMSまでのかなり広い範囲をカバーしなければならない。特に最近は大規模なデータベース・システムの運用も現実化してきているので，データベース設計はますます困難な仕事になってきている。

本稿では上記の問題を解決するため，データベース設計過程を各々作業内容が切れているいくつかの階層に分解し，設計の問題を各階層の問題に振り分けて，各層ごとに解決するための方法を整理する。それとともに，各々の仕事の内容を人間による創造的作業と適当なツールによっても遂行可能な定型的作業に切り分け，後者については，今まで提案されたツールのデータベース設計過程の中での位置づけとその評価も試みることにする。

2. データベース設計過程の考え方

今までにデータベース設計の過程またはデータベース設計の方法論を扱った仕事は少ない。Bubenko[1]等は自分たちの設計支援ツールの位置を示すため，設計過程についての案を示し，その中で概念設計段階での支援ツールの方式を作成している。Merten[2]等は要件と環境から発生するパラメータから論理構造設計，物理構造設計という過程を経てデータベース設計を行うべきことを提案している程度である。

一方，設計対象を実現するための道具を扱った仕事は多い。J. Martin[3]の「Computer Data Base Organization」は，内容を論理構成と物理構成に分け，両者について詳細な技術を紹介している。G. Wiederhold[4]の「Data Base Design」はデータベース構造のレベルを ① 情報を扱う概念レベル ② データを扱う記述レベル ③ データの表現法を扱う構成レベル ④ ハードウェアと対応させる媒体レベル に分け，このうち②はデータベース・システム，③はファイル・システムに対応するとして，この二者に対し詳細な技術と評価法を解説している。

すなわち，現状は貧弱な方法論のもとで，膨大な技術とパラメータを消化しきれなくなっており，ここから種々の問題が発生して来ていると考えられる。そこで，第2章では (2.1) データベース設計の問題点 と (2.2) データベース設計の方法論を扱う。

2.1 データベース設計の問題点

データベース設計過程が十分整理されないで，設計が行なわれている状態では次のような問題が発生する。

a) 論理構造を決める際の条件があいまいである。特に概念レベルの設計手法が欠如しているため，決められた論理構造の基盤が明白でない。
b) 論理構造を物理構造にマッピングする際に考慮すべきパラメータが多すぎて評価しきれない。
c) 設計を順次進めて行くうちに，データの重複をなくすことと，いかに巧みに重複させるかとのトレード・オフ条件が発生し，これを考慮すると論理構造がしだいに理解しにくい，いびつなものになってゆく。
d) プログラムの検索形式が条件検索(注1)なのか，巡航（navigate）検索なのかがうまく切り分けられておらず，しかし処理フローから物理表現の形式（ポインタとして NEXT，PRIOR，OWNER を持つかなど）を決めがちで，プログラムの物理構造独立が実現されない。
e) システム分析，条件設定から論理構造設計，物理構造設計に渡って何度かサイクリックに評価をしなおす場合，終了条件が明確でなく，むしろ途中で新しい条件に気付いて可能な案が発散することが起こりうる。設計の手法としては極めて非効率になりやすい。
f) サイクリックに設計を進めて，設計の前のステップにもどった時，以前の設計のどこまでが妥当なのかの目安が明確でないので，一括して旧案を捨てざるを得ない羽目に落ち入りやすく柔軟性に欠ける。
g) 適当な論理的階層で切れていないので，正当性を検証することが困難である。

これらの問題点は次のような配慮が十分でないことから来ると考えられる。

i) 設計過程の各段階ごとの作業内容が明確に区分されていない。
ii) 作業手順が明確でない。どの段階でどんなドキュメントを作成し，作成されたドキュメントは次にどんな役割をはたすかが整理されていない。
iii) 処理パターンとデータ構造と評価手法には対応関係があるはずなのに，明確な対応づけがなされていないため，ある入力情報にもとづく設計と評価がどんな役割を持つかが明確でない。設計過程の各段階と処理パターン，データ構造，評価手法の対応づけを行う必要がある。
iv) あるトレード・オフ条件が発生した時の調整手段が明確化されていない。
v) 設計途上で実世界の認識に変更があるか，進化（Evolution）のある構造を扱う時の処理方法が明確でない。これに対処するには概念設計段階からの一貫した設計過程が必要である。

次にこれらの問題を解決する設計手法を考えることにする。

注1) 条件検索とは〈属性名〉〈関係演算子〉〈定数〉のような特定の条件にもとずく検索，巡航検索とはある構造上をその構造に沿って GET NEXT のように検索するものを指している。

2.2 データベース設計の方法論

大きく2つの側面がある。一つは設計過程を比較的作業内容が切れているいくつかの階層に分けることであり，他の一つは各階層での処理パターンとデータ構造と評価モデルの対応づけを行うことである。

前者のためにはソフトウェア工学の抽象化の理論とシステム開発技法が大変参考になる。[5]

まずLiskovの抽象化の理論によってDBMSをいくつかのレベルに分割してみる。しかし注意すべきことは，我々はDBMSの設計を目的とするのではなく，DBの設計をするのだということと，抽象化のレベルを設計過程の段階として利用したいということである。[6]

次に，TRW社のSREMに代表されるシステム開発技法を参考にして，最初の要件事項から出発して，順次どのような作業を行ってゆくかの作業工程表を作成する。各階層での処理パターンを記述する方式は各層のRSL（Requirement Specification Language）と見做せるであろう。

この機構内で次のように作業を進展させればよい。

1) 論理的に高度な階層から順次設計を行う。
2) 各階層ごとに抽象化のレベルが切れているので，評価結果と作成ドキュメントによる当該階層の改善，洗練を行う。
3) 各層の設計が終ると，データ構造，評価結果などを記述するドキュメントを作成する。これは次の階層の要件となるものである。
4) 階層内，および階層間にまたがるトレード・オフ条件が発生すると，前者については，当該階層での評価結果を付して，次の段階でのより詳細な設計にまわす。
   後者については，一つ前のステップへのフィードバックを行う。ここでもさらに代替案が発生することになれば，この階層での評価も付して，さらに上の層にもどる。これを影響の及ぶ段階まで繰り返す。上の層での結果は，また下に下りてくる。
5) 4)の作業中，代替案の種類により，データ構造などでかなり変動する部分とあまり変動しない部分に切り分け，構造および機能のモジュラリティを把握しておく。（これは予測される進化に対する対処法として利用できる。）

この他に次の手段を用意しておかねばならない。

i) 各層での評価手法，いずれが良いかを判断する基準
ii) 各層ごとに設計の代替案を創生する手法

1)～5)の過程を繰り返すうち，各層でii)により可能な案が生成され，i)により，評価，判断が行われるので，この判断基準により設計サイクルの終了条件が決められる。

後者，すなわち処理プロセス，データ構造，評価モデルの対応づけについても同様に，抽象化理論による切れ目はかなり切れた対応関係を作っているので，各々明確な入力形式と評価手法を対応させることができる。

以上の事を概念的には図1の垂直関係と水平関係でとらえることができる。

垂直関係により
- 設計対象の切り分け
- ドキュメントの切り分け
- 作業工程の切り分け
- 入力情報の切り分け
- 評価手法の切り分け

が行え，水平関係によって

処理パターン ― データ構造 ― 評価モデル

の無理のない相互関係が維持できるわけである。

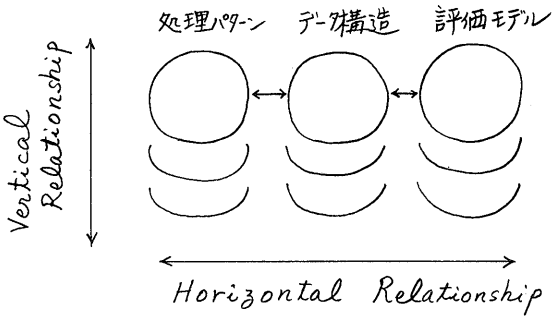


図1：データベース設計過程モデル

以上の考えをもとにデータベース設計過程の提案を図2に示す。

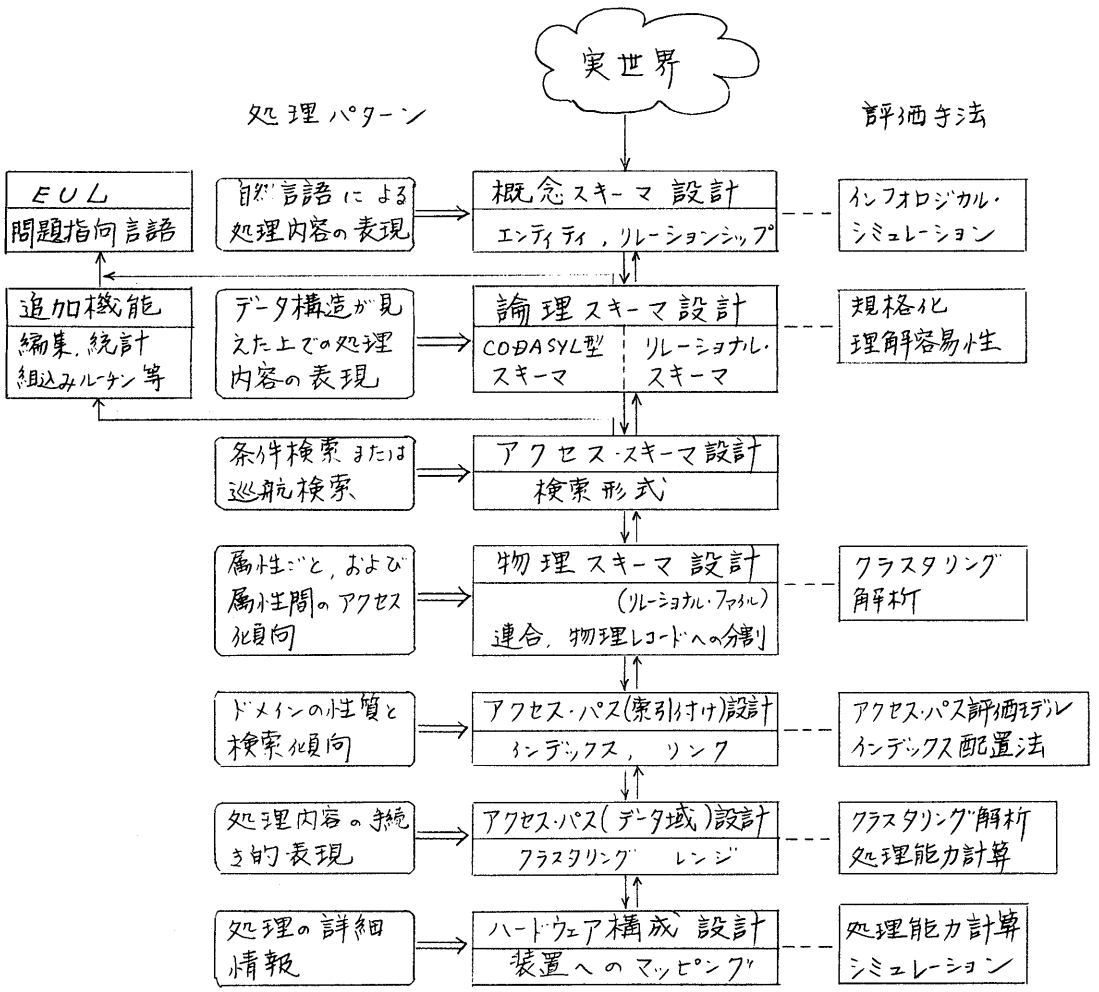


図2：データベース設計過程

この案を決めるには次のような点に配慮した。

i) 概念スキーマから出発することが，システム分析を明確にする上でも進化に対処する上でも必要である。
ii) 全体をスキーマ・レベルの設計とオカレンス・レベルの設計に分け，後者についてもさらに2つの階層を設けた。この程度に分けないと，設計パラメータ数はすでに膨大なものとなっているので処理しきれない。階層を分ける際は処理パターンの無理のないレベル分けに対応づけるようにも配慮した。
iii) データベースは構造設計のみではなく，機能設計（望ましい検索形式，追加機能など）も重要である。これが明確でないと上位ではEOLの設計，下位ではアクセス・パス（索引付け）の設計が十分なものにならない。そのためアクセス・スキーマ設計を導入した。
iv) フィードバックは原則として直前の階層にもどり，それを順次繰り返すものとする。但し次のような特に関係の深い階層はある。アクセス・スキーマ設計 ― アクセス・パス（索引付け）設計，物理スキーマ設計 ― アクセス・パス（データ域）設計。これを入れ子にしてあるのは順次スキーマ設計とオカレンス設計を行ってゆくためである。

図2の案を決めるに際しては R. Yeh[7] と小林[8]の案を参考にした。

3. データベース設計過程と設計支援ツール

図2の設計過程に従って設計を進めるには2つの側面がある。

一つは各設計過程で何を（WHAT）を行うかである。なすべき仕事の内容が明確にされねばならない。また，設計の管理を行うためには作業工程としての区切りが必要で，設計の出力としてのドキュメントが必要とされる。ドキュメントの形式と内容，その利用法，評価手法との関連も決められねばならない。

次は各階層の設計を如何に（HOW）行うかである。今まで提案されたツールによる方法もあるが，それらの使用法は必ずしも明確にされていない。ツールといっても思考ツールで十分効果が上げられる段階もある。ともかく仕事のこなし方のノーハウが蓄積されねばならない。

第3章では前者を（3.1）データベース設計仕様，後者を（3.2）データベースの設計方法および設計支援ツール で扱う。

3.1 データベース設計仕様

各段階ごとに仕事の内容と特徴，ドキュメントの内容，形式，利用法および入力情報，評価内容などを記す。

(1) 概念スキーマ設計

実世界をそのまま計算機の世界へ写像すると次のような問題が発生する。

i) 早くからDBMSの性格に引きずられるので，システム認識の本質を見失う恐れがある。
ii) システム分析の結果が十分整理されないままに計算機上での実現を考えることになるため，要件や環境の変化，実世界の認識の変更に対す

る対策が立てにくい。

iii) システム分析の結果と，実際のDBMSでサポートする論理構造との間に距離がありすぎて，正しいマッピングが行いにくい。

このため，実世界と論理構造の中間に概念スキーマを導入する。これはANSI／SPARCモデルの概念スキーマとはことなって，実世界と外部（論理）スキーマの中間に概念スキーマが存在する。

概念スキーマは，後でふれるいくつかの概念スキーマ記述言語か，または個々の方法で記述される。概念スキーマを記述するための入力情報は，システム分析の結果，対象システムが処理すべき処理内容の自然言語による表現である。これをもとにエンティティ，リレーションシップが抽出される。

概念スキーマ仕様書にはエンティティ，リレーションシップ，エンティティおよびリレーションシップを構成する属性名，各リレーションシップの関係（1：1，1：多，多対多，またはもっと意味的関係），新しい構造にもとずく処理内容のよりフォーマルな表現，システム分析からわかっているある程度の定量的データ（特定の処理の頻度など）が記述される。

(2) 論理スキーマ設計

ここでの目的は概念スキーマ仕様書をもとに実際のDBMSに直結する論理構造を設計することである。論理構造を決める時2つの大きな問題がある。

i) データモデルの選択（EMSモデル（木構造），ネットワーク・モデル，リレーショナル・モデル）

ii) 論理構造はユニークには決まらない．評価基準の設定

前者の問題はすでに環境によってCODASYL型と決まっている場合もある。そうでない場合は可能な案を並列に評価してゆく。但しこの段階では各データモデルの論理的側面だけで判断すべきで，各DBMSにからむ詳細機能は捨象して考えねばならない。

後者の問題はデータモデルに依存する。リレーショナル・モデルの場合は規格化が一つの評価手法である。

論理構造が見えた段階で再度，処理内容の表現を行い整合性をチェックする。その際，処理内容をフォーマルに記述する言語仕様が欲しいが適当なものは存在しない。

論理スキーマ仕様書はインプリメントにつながる最初のドキュメントという意味で重要で，それ以降の設計でよく参照される。

(3) アクセス・スキーマ設計

対象システムの処理パターンで必要とする検索形式をもとに，必要な検索機能を設計する。データモデル，論理構造との対応もとって検索機能が決められねばならない。

アクセス機能仕様書は，関係演算子として＝，＞，…などどの範囲をサポートするか，キーとして採用される属性の定量的性質（重複があるか，キー値の個数など）などが記述される。

(4) 物理スキーマ設計

属性ごと，および属性間のアクセス傾向を知って論理構造を如何に物理構造にマッピングするかを設計する．アクセス傾向の解析にはクラスタリング解析が使用できる．DBMSによっては論理レコードを物理レコードに分割することも行う。

論理スキーマ仕様書，アクセス機能仕様書の情報とバランスをとりながら物理的連合（association）の方法，例えばポインタ方式か，物理的隣接かも決める．それらの結果を物理スキーマ仕様書に書く．

(5) アクセス・パス（索引付け）設計

特に高度の条件検索をサポートするシステムが提供されてくるにつれ，性能を決める階層として重要になる．ここでは主にアクセス機能仕様書と性能に対する要件をもとに，物理スキーマとの対応もとって選択するインデックスの種類，インデックスを作成する範囲を決める。

インデックスの維持はメモリ　アクセス，更新のオーバーヘッドとコストがかかるので，コストとこれから得られる性能，機能上のメリットとのトレード・オフで決めねばならない．このためインデックスを如何に配置すべきかと，インデックスがある場合の性能（応答時間など）と，この時かかるコストとの三者を評価する。

アクセス・パス（索引付け）仕様書には，ある程度処理能力を算定する場合の基礎数値が記述される。

(6) アクセス・パス（データ域）設計

実際にデータ・オカレンスを如何に配置するかを設計する．格納構造を持つDBMSでは格納法も設計する。

この段階で初めて，処理パターンの細かい手続き的表現が考慮される。手続きの形式によって特定のデータ間の近接配置が決定される．これを分析するのにクラスタリング解析が行われる。オーバーフロー域，格納域満杯時のスプリット法なども決められる。

アクセス・パス（データ域）仕様書は選択した処理方式の他に，索引付けの条件とも組み合わせて初期の処理能力算定を記述する。

(7) ハードウェア構成設計

実際の装置へのマッピングを行う。特に索引付け，データ域の配置法の方式と，装置のハードウェア性能，さらに詳細な処理情報（処理多重度，I/O回数，バッファ量 ---）により，より精密な処理能力算定を行う。

処理能力算定を助けるため，また特定のパラメータの感度分析を行うためシミュレーションを行う。性能はデータの再配置により変化するので最適点を発見する。

ハードウェア構成仕様書は処理能力算定書もかねる。

(8) その他

特に必要な追加機能（編集機能，統計機能，特に必要なアクセス機能など）およびユーザーの目的に合わせたエンド・ユーザー言語を設計する。

3.2 データベースの設計方法および設計支援ツール

各段階ごとに設計の方法と参考になる設計支援ツールの例を上げる。

(1) 概念スキーマ設計段階

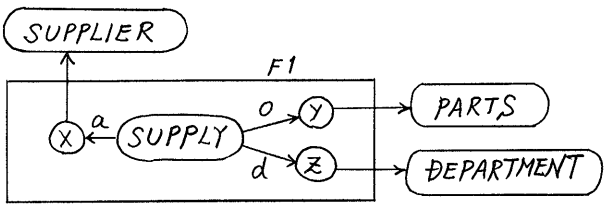
内容： 概念スキーマをフォーマルに記述できる方法で概念の整理を行う。代表的方法としてP. Chen[9]のエンティティ・リレーションシップ・モデルによる方法がある。しかしこの方法は現実のDBMSのデータモデルには近いが概念を整理する時，Smith & Smith[10]によるデータ抽象化の機能（Aggregation, Generalization）が不足していると考えられる。

これを補うためAIのフレーム理論による方法が上げられる。例えば概念スキーマを決める要件である処理パターンが，最初自然言語で書かれているとして，この分析により

SUPPLIERS SUPPLY PARTS TO DEPARTMENT

が必要であるとわかった時フレームは図3のようなものである。

いくつかのcase-labelとフレーム上での演算子を導入すれば，概念レベルでの形式的把握が可能である。



F1[(x∈Supplier)(y∈Parts)(z∈Department);
Supply(agent : x, object : y, destination : z)]

図3 フレームの例

ツール：

- DIAM-II[11]のFORAL (by Senko) ---
  Bubenkoがこれを使って概念レベルを記述している。
- Entity-Relationship Model (by P. Chen)
- フレーム理論による方法 (by R. Yeh)
- IRIA[12]グループの方法 (by P. Moulin et al)

(2) 論理スキーマ設計段階

内容： 概念スキーマ・レベルの記述をもとに論理スキーマを作成する方法はいくつか提案されている。例えばBubenko[13]は概念スキーマの設計とそれを論理スキーマに移行する手続きを提案している。

①対象システムの機能解析と，予測される情報要件，入力トランザクションの把握 ⟶ ②エンティティを概念クラスに分類 ⟶ ③FDを指定 ⟶ ④適当な連合をえらんでAO（Abstract Object）の作成。AOを規格化によりレレベル・オブジェクトとする。各連合に性格づけ（対応関係，メンバーシップ，暗黙のものか導出されるものか）を行う ⟶ ⑤ ③により新AOの導入 ⟶ ⑥暗黙の連合の解析 ⟶ ⑦可能な前例（AO間の関係など）を捜すためマトリックスの作成 ⟶ ⑧誘導（derivation）解析 ⟶ ⑨暗黙の連合に対し，まだ明確でない規則のあるオブジェクトがあれば⑥へ ⟶ ⑩明確で

ない誘導規則を持ったオブジェクトがあれば⑧へ ⟶ ⑪答え
⟶ ⑫ エンティティをユニークに参照する名前，連合の集合にも対応する名前を導入して特定の外部スキーマにマッピング。
CODASYL型とリレーショナル・モデルとではかなり方法論にちがいがある。

ツール：
- EAM (Inferential Abstract Modelling) (by Bubenko)
- Entity-Relationship Model による論理スキーマへの変換 (by P.Chen)
- HI-IQ 言語による検索形式の表現をもとに Assertion を生成し，これから論理スキーマを生成 (by Gerritsen[14]) --- CODASYL型
- 抽象DMLにより処理パターンを記述し，これと物理的実現法とを対応させてページ・フォールト回数最少の条件で解いて論理スキーマを決定 (by Berelian[15]) --- CODASYL型

(3) アクセス・スキーマ設計段階
内容： システム分析の結果，今対象としているシステムを実現するには，どのような人間・機械インタフェースがよいか，どのような検索形式が望ましいかを分析し，それをアクセス機能を表現しうる擬似言語で記述する。

ツール：
- Cardenas[16] が上げた atomic, item, record, query condition の分類. --- その他 query の分類法がいくつかある
- 各種言語を参考にした擬似言語の作成

(4) 物理スキーマ設計段階
内容： 論理レコードをシステム内で実現する物理レコードにマッピングする手法と，それらのレコードの連合を実現する方法が問題になる。
前者はリレーショナル・モデルの場合，次の2つの方法がある。

(A) 水平分解／合成　　STUDENT (S#, SNAME, SEX, BD, PROG)
GR.ST (S#, SNAME, SEX, BD) ⊕ UND.ST (S#, SNAME, SEX, BD)

(B) 垂直分解／合成　　ST (S#, SNAME, SEX, BD, PROG)
ST.1 (S#, SNAME, SEX)　　ST.2 (S#, BD, PROG)

A案はタップル方向でタップルを適当なグループにまとめることで，Generalization と同じだが，データの性質まで調べてグルーピングの方法を決める。
B案は属性間のクロスリファレンスの頻度を表示するマトリックスを作ることによりクラスタ解析で代替案を作成する。その際，一般には複数リレーションに存在する属性にまたがる参照関係が作成されるため，評価が

困難になるが，一つのリレーション内の属性に関するクロスリファレンス・マトリックスからリレーションの分解のみを考える場合は簡単に求められる。

後者はDBMSがどのような連合の能力を持つかに関係し，Berelian, Gerritsenらによって扱われている。

ツール：
- クラスタリング解析 （by Hoffer[17]）
- 可能な物理構造表現によるコスト評価（by Berelian）
- DBMSのモデル化（Gerritsen Model） （by Gerritsen）

(5) アクセス・パス（索引付け）設計段階

内容： インデックスが置かれる時，種々のインデックスごとにどんな性能が得られるかということと，インデックスを如何に配置するかの2つの問題がある。

前者の問題についてS.B. Yao[18]は独自のモデル（ハイアラキカル・アクセス・モデル）を提案し，これにより多様なインデックス構造を表現できることを示した。これによりコストと性能（応答時間）を解析的に求められる。

後者の問題をAnderson等が扱っている。

ツール：
- データベース・アクセス・コスト解析 （by Yao）
- Yaoの方式にもとずくシステム例 （by Teorey, Das[19]）
- 転置ファイルについての解析的仕事 （by Cardenas）
- 2次インデックスの選択法 （by Anderson[20], Bruce Berra）

(6) アクセス・パス（データ域）設計段階

内容： 近接配置を設計するにはクラスタ解析が可能である。例えば，処理パターンから図4のように

$A \longrightarrow B$

$C \longrightarrow D, E$

という検索が多いことがわかったとする。この時，物理的配置からもA，Bおよび C，D，Eを近傍におくためには木構造システムでは

$a_1\ b_1\ b_2\ a_2\ b_3\ b_4\ b_5\ a_3\ \cdots$

$c_1\ d_1\ d_2\ e_1\ c_2\ d_3\ e_2\ e_3\ e_4\ \cdots$

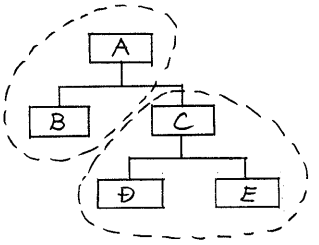


図4

と配置し，ネットワーク・モデルではレコード・タイプBはAの，レコード・タイプD，EはCのNear指定（オーナー・レコード・オカレンスの存在する同一ページになるべく指定メンバ・レコード・オカレンスを置く）を行うようにすればよい。このように如何に配置すべきかを設計する。

また，手続き的関係をもとに，判定ボックスの分岐比率まで考えてシミュレーションを行い，処理能力を評価する。

ツール：
・階層木構造に対するクラスタリング・アルゴリズム（by M. Schkolnick[21]）
・シミュレーション ---種々の方法あり
・キューイング・ネットワーク手法による解析 ---広い範囲を対象にDB DE（by Teorey）

(7) ハードウェア構成設計段階
内容：処理能力計算を，机上計算，評価式による計算，シミュレーションによる評価など多様な方法で行う．同時に処理能力を実現しているメモリ，処理時間，インデックスなどのコストも評価する．
完全な結果は得られないので，擬似データベースを介したシミュレーションやすでに運行しているシステムの統計データの利用も考えられる．

4. おわりに

ここで参考に取り上げた支援ツールの例は，たまたま現在各所で発表されているものである．各ツールはそれぞれに設計の方法論があった上で，支援ツールの方式が展開されるべきだが，必ずしもそうはなっていない．

本稿で提案した方法は，まずどのような過程を経て設計を進めるべきか，各段階では何を評価すべきかを明確にする．その上で各段階に対応する評価モデルと評価を行うためのツールを作成することを述べている．

トップダウン手法は，上位段階の変更が下位段階に大きな影響を与えるが，それでも，どのような変更がどのような影響を与えるかの目安がしだいに立つようになる．影響度の少ない部分のモジュラリティを考えれば，ある程度，拡張可能性やデータベースの進化による影響も予測することができよう．

実際の作業過程に導入する場合の詳細化，適切なツールの選定，また処理パターンの適切な記述機構は今後に残された問題である．

〔謝辞〕本稿を書くにあたり熱心に討論していただき，多くの貴重な助言を下さった小林晏氏（富士通国際研），貴重な示唆をして下さった小林功武氏（産能大）に深く感謝の意を表します．

〔文献〕

1. Bubenko, J. A. : Information Analysis and Database Design, CADIS report 1976
2. Mertan, & Oberlander :
3. Martin, J : Computer Data Base Organization, Prentice Hall 1975
4. Wiederhold, G. : Data Base Design, McGraw-Hill 1977
5. Liskov, B. H. : A design methodology for reliable software systems, FJCC 1972
6. Bell, T. E. & Thayer, T. A. : Software requirements : are they really a problem?, 2nd SE Conf.

7. Yeh, R.T, Roussopoulos, N. & Chang, P. : Data Base Design - An Approach and Some Issues , SDBEG-4 UT-Austin
8. 小林功武 : データベース設計 , 産能短大セミナー
9. Chen, P.P, : The Entity - Relationship Model - Toward a Unified View of Data , TODS Vol 1, No. 1
10 Smith, J.M. & Smith, D.C. : Database Abstraction : Aggregation and Generalization , TODS Vol 2 '77
11 Senko, M.E. : DIAM as a Detailed Example of the ANSI SPARC Architecture , G.M. Nijssen ed. Modelling in Data Base Management Systems pp 73-94 1976
12. Moulin, P., et al : Conceptual Model as a Data Base Design Tool , 同上
13. Bubenko, J.A : IAM : Inferential Abstract Modelling — An Approach to design of Information Models for Large Shared Data Bases , RC 6343 1/4/77 IBM Yorktown
14. Gerritsen, R.: Understanding Data Structures , Working Paper 75-8-01 Univ. of Pennsylvania
15. Berelian, E. : A Methodology for Data Base Design in a Paging Environment , SEL Technical Report No. 109 April 1977 Univ. of Michigan
16. Cárdenas, A.F. : Analysis and Performance of Inverted Data Base Structures , CACM '75. Vol 18, No 5
17. Hoffer, J.A. & Severance, D.G. : The Use of Cluster Analysis in Physical Data Base Design , 1st VLDB
18. Yao, S.B. : An Attribute Based Model for Database Access Cost Analysis ,
19 Teorey, T.J. : The Database Design Evaluator. Part I Overview. Part II User Interface , Part III Design Specifications , Working Paper DE 7.2-1, 2, 3 '77 Univ. of Michigan
20. Anderson, H.D. & Bruce Berra, P. : Minimum Cost Selection of Secondary Indexes for Formatted Files , TODS Vol 2, No. 1
21. Schkolnick, M. : A Clustering Algorithm for Hierarchical Structures , TODS Vol 2, No 1 '77