上條氏のデータベースについての二，三の注意

西　村　恕　彦†

データベース言語の日本工業規格 (JIS) 制定の過程は，どう見ても 1980 年までかかるであろうが，それをずいぶん先のことと思ってはいけないのであってそれはいま現在の時点からかなり真剣になって作業を開始して，どうにか 1980 年ごろになるだろうということである．

そのような意味において，学会誌にデータベース技術の講座が企画されたことはまことによろこばしい．ただ上條氏の解説を読んでいて気になることがあるので，氏の理解の行きとどいていない点があることと，日本語での文章表現の不適切な点のあることをお節介ながら指摘しておく．

(1-a) 氏はデータベースにおける**オカレンス**という語を，反復あるいは繰返しと解しておられるが，これはまず紛らわしい用語であって，避けるべきである．電子計算機およびプログラム言語においては，データ構造の最も素朴かつ基本的な分類が単純変数と配列 (array, vector) である．そして後者のことを反復あるいは繰返し集団とよんでいる．データ構造におけるレベルも違い，意味も違い，アクセス方法も違うまったく別の対象であるオカレンスに対してこの訳語をあてはめることは，混乱と誤解を招く以外に何の利益もない．

もっともこれは英文でも紛らわしいのであって，たとえば「データ項目のオカレンス回数は，すべてのレコードオカレンスにおいて等しくなければならない」というような表現が DBTG 提案に出てくる．しかし英語の表現が紛らわしいからといって，日本語の表現まで紛らわしくする義理はないのであって，別の概念をさすものである以上，紛らわしくない別の訳語をあてるべきである．

(1-b) そもそも occur という英語は，あるできごとが生起するということ以上の意味は語義的にも派生的にももっていない．したがって，配列 (array) に対してこの語を使うときにも，オカレンス番号は出現番号，オカレンス回数は出現回数と解するのが正確なのであって，ただ後者の場合には意訳として，反復回数・繰返し回数ということばを用いてもまちがいとはいえなかろうということである．

(1-c) ではデータベースにおいてオカレンスという語がなぜ使われるのかというと，それは，生起しうるものを規定するのが「タイプ」であって，実際に値として生起したものをこれとは区別して「オカレンス」とよびたいからである．このことは，データベースにおけるオカレンス (occurrence) の同義語として，インスタンス (instance) あるいはイベント (event) という用語が用いられることでも実証される．後二者の場合にはあきらかに，「反復」という語感は含まれていない（「引用例」，「具体値」というべきか）．

occur あるいは occurrence ということばは確率論のほうでも広く使われていて，「ある起こりうる確率的な事象の組のなかから，ある特定の確率的な事象が実際に具現すること」，あるいは「ある属性のとりうる値の組つまり値域 (range) のなかで，ある一つの特定の値 (value) が観測されること」をさす．このような用法は，データベースにおけるタイプとオカレンスの使い分けによく照応する．

したがってデータベースにおけるオカレンスすなわちインスタンスを，「反復」と解することはまるっきり見当はずれで基本的な誤りであって，「実現値」，「具現値」，「具体値」などと解するべきである．

(2) 氏は**エントリ** (entry) という語を探索の入口と解しておられる．前条のオカレンスとは違って，このエントリはたしかにもともと入口という意味をもっている．このことはたとえば，ある種のプログラム言語にある ENTRY (副入口) 文のことを考えてもあきらかである．

しかしデータベースや表においてエントリというときには，それは（探索の）入口という意味ではなく，「書き入れる」という意味から派生してできた「記載事項」をさす．つまり，表の場合には表を構成する一組の項目群，特に一次元の表であれば見出しと関数からなる一組の値の対をさす．

データベースの場合にも，ある種の関連を有するデータの一組をさす．おおまかにいえば，論理レコード

† 電子技術総合研究所

とかエンティティに近い．つまり，ほぼこれに近いレベルの程度の関連を有するデータの一組をさすことばとして，(よりこまかい点はいちいちのシステムごとに定義しなおされて，)すでに広く使われている．

ただし，データベースにおいて entry という語が絶対に探索の入口をさすことはないとまでは言いきれないが，しかしこれはたしかにより稀な用法であり，一般には前者であるから，一義的に「入口」と割り切るのは誤解と混乱のもとである．

(3) 131ページ左に，「木構造は次の3つの規則によって展開することにより，線型リストとなることでも知られている」とあるが，木をわざわざ線形リストに直すことにはほとんど何の利益もなく，むしろ木は木として残すことがデータを構造化する目的そのものであり，特に検索や構造変更の速度を高める手段をも提供する．

ただ，木を計算機の記憶装置に割り付けるときに，節 (node) を語に，枝 (branch) をアドレスポインタにそのまま単純に対応させると，各セルの有効長が不定になって，プログラム技術上おもわしくない．

その解決法はよく知られていて，名称はいろいろあるがたとえば相続順位リスト (filial-heir list) という呼称は最もよく実体にあっている．論理的な木を記憶装置中の相続順位リストで表現することは，プログラム技術，記憶容量，検索速度の三者をほどよく満足させる．しかし木を線形リストで表現することにはこのような利点はない．

もちろん木のなかのすべての節を順番に一回ずつ走査してゆく必要が生ずることもあるが，これは線形リストとは何も関連しない．さらに，よく知られたデータベースの構造も，必ずしもこのような展開を許すようにはできていない．その点でも，このような展開規則を木の論理構造の一部とみなすことはできない．

(4) 131ページ右に，「**多重リスト**は分岐点を持つリスト構造である．分岐には意味をもたせることにより，データ・ベース・レコードの分類作業をなくすことができる」とある．多重リスト (multi-list) ということばはしかし，別な意味で広く使われている．多数の要素からなる集合があるときに，要素中の各属性ごとに (それぞれ独立に) 線形リストを定義する．各属性ごとに一本ずつ線形リストがあって，それぞれの線形リストはその属性における値の大小順にしたがって各要素をつなぐ．

このようなものが多重リストとよばれ，まさに「分類作業をなくす」という効果を有している．リストの一本一本はそれぞれ独立な線形リストであるから，「分岐点」と氏がいうときにそれが何をさすつもりであるかは，疑わしい．

(5) 氏は**リング**構造を，線形リスト，木，網などと並ぶ論理的な構造の一種として述べておられるが，リングは線形リストや木を記憶装置中で表現するときの作成 (implementation) 上の手法の一つであるにすぎない．単純な線形リストをリングにすることの利益はほとんどないが，木をリングにすることは好ましい．従属セルから上位セルへのポインタの記憶容量を減らし，しかも検索速度はほどほどに保つことができる．ただしこの場合に，各ポインタの性質は線形リストの場合のように均一にはならない．

事務応用上もリングを単に Fig. 3 のように連結してゆくことは要求されず，むしろこの図は一つの木を例示しているらしい．説明文のほうでも「階層関係」を表わすとあって，木と区別されない．

(6) 132ページ右に，「木構造系に対して，2方向性を導入したものに網構造」がある，と述べているが，**二方向**ということばは bidirectional あるいは two way に対応することばとして広く用いられており，つまり，順向ポインタのほかに逆向ポインタをもそなえたリストをさす．

このように紛らわしいことばを用いないで，「一つのセグメントが二つ以上の直属上位セグメントを有するものが網構造である」というべきである．

語義的なせんさくなどは無用のわざのようでもあるが，これらのまちがいは単に訳語のあてはめが不適切だったのではなく，データベースを検討するにあたっての基本的な概念のいくつかに対する誤解のあらわれであるので，ことごとしく述べたのである．

なお上條氏の文に限ったことではないが，学会誌に誤植が多いことは目に余る．学会の機関誌としてはもちろんのこと，公刊物としてお座敷に出すことさえはばかるようなひどい品質のできばえである．紙の量をふやすこともけっこうであるが，質の点での向上にもまだまだ力を注ぐべきである．編集当事者の対策を待ちたい．

(昭和48年3月14日受付)