報　告

パネル討論会

次世代データベース†

パネリスト
石井　義興[1]，　小林　要[2]，　田中　克己[3]
宇田川佳久[4]，　戸沢　義夫[5]，
司会　有澤　博[6]

司会(有澤)　時間になりましたので，パネルディスカッション，「次世代データベース」を始めたいと思います．私はこのパネルの司会を務めさせていただきます横浜国立大学の有澤です．よろしくお願いいたします．この「アドバンスト・データベース・システム」シンポジウムも回を重ねましたが，今回のエントリ数がいま受付で確認しましたところ105だそうです．実はきのうの段階では85でございまして，いよいよデータベースシンポジウムも終わりかなという気もしたわけですが，どうやらまだ大丈夫のようでございます．きょうはホットなディスカッションを展開していければと思っております．

それではいまからデータベースの今後を占うことになるパネルディスカッションを展開していきたいと思います．現在必ずしも「次世代データベース」という言葉が定着しているわけではございませんが，広い意味でのデータベースの今後あるべき姿ということについて議論していただきたいと思います．

まず全体の基調といたしまして，いままで長い間データベースというものを，ユーザサイドのリクアイメントをとおしながらみてこられたソフトウェア・エージーの石井社長にお願いし，あとのパネリストの方々は比較的お若い方ということで，それぞれの次世代データベースの中でトピックとなるような，あるいは要素技術になっていくであろうことを，ご研究になっておられる方々に，言いたいことを言っていただこうという趣旨でパネルを構成しております．

それでは初めに私のほうから簡単に全体のお話をさせていただきます．これは私がよく使うOHPですが

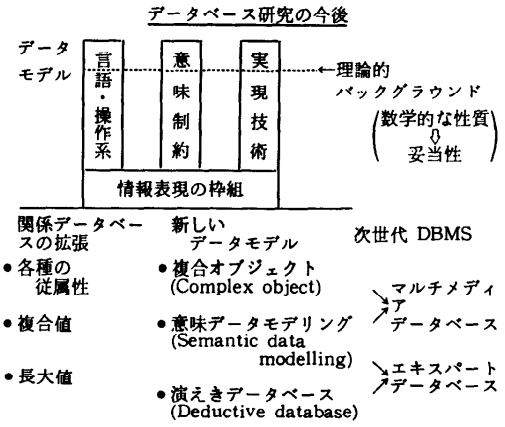


図-1　(有澤)

(図-1)，データベースの今後はどうなるかを書いたものです．もちろんデータモデルが，基本的な概念になるわけですけれども，そのデータモデルにはまず基本的な構造，ここでは情報表現の枠組といっておりますが，そういうものがベースに存在するだろうと思います．その上に何本かの柱があって，ここでは3本書いてあるわけです．たとえば言語であるとか，操作系という幹が一つある．それから意味制約ですが，その表現，あるいはそのチェッキングのための機構，そういうことが一つあるだろうと思います．もう一つはやはり，いくら枠組がきちんとして，制約条件や操作系がきちんと記述できたとしても，実現技術に裏打ちされてないと，机上の空論になるだろうと思います．これは効率や最適化技術の問題も含めて，一つの柱であるといえると思います．さらにそういうものを貫いてそれらの妥当性をいうためには，理論的なバックグラウンド，数学的な枠組というものが，きちんとその背後にないと，単に「私はこれがいいと思う」というだけの議論になってしまいます．数学的な性質が明確になると，なぜそれが妥当であるかということがはっき

† 日時　昭和62年12月4日 (金) 15：00～17：00 の「アドバンスト・データベース・システム シンポジウム」
場所　機械振興会館大ホール (地下2階)
1) ソフトウェア・エージー，2) 富士通・国際研，3) 神戸大，4) 三菱，5) 日本 IBM，6) 横浜国大

りしてくるわけです．データモデル論というのは，数学的な基盤に支えられつつ，こういった柱をきちんと明確にすることによって成り立っているわけですが，考えてみますと関係データモデルというのは，これらに関して非常にきちんと議論されてきてるわけです．たとえば，n 項関係という数学的枠組がベースになって，その上で言語操作系として，コッド自身が提案した関係代数や関係論理という操作系があり，いろいろな概念や操作が明確にされてきている．それから意味制約も，それだけで十分とはいえませんけれども，さまざまな従属性理論が多くの人たちによって，1980年代の初めごろまで活発に議論されてきたということは皆さまご承知のとおりだろうと思います．また実現技術に関してみれば，これは今回もデータベース・マシンのサーベイにもありましたように，あるいはその他の商用システムがすでにたくさん世の中に出ているということからも明らかなように，きわめて高い裏付けができております．そういうわけで関係データベースはきちんとしてるわけですが，では今後どうなるのだろう．関係データベースの拡張ですむのだろうか．たとえば各種の従属性や複合値や長大値といったものを扱った関係データベースの拡張ですむのだろうか．それともやはり新しいデータモデルを提唱すべきなのだろうかということが，ここでの問いかけであります．もし新しいモデルが必要だとすれば，そこではたとえば複合オブジェクトの扱いとか，意味データモデルとか演繹データベースとかいろいろな新しいアプローチがあり得ます．さらに複合オブジェクトと，意味データモデリングから，マルチメディア・データベースというものの姿が浮び上がるのかな，とか，意味データモデリングと演繹データベースから，エキスパート・データベースの枠組ができ上がるのかなというような感じを一応もってるわけです．ここでパネリストの方々への問いかけとしては，こういった考え方が新しいモデルとして，しかもいろいろな要素をみんな取り込んだ一つの姿になるのか，それともそうではなくて，関係データベースのようなものがだんだん拡張されていくのか，あるいは伝統的な関係データベース・システムの回りにインディペンデントに，いろんな技術が積み重なっていって，統合化システムができるのか，こういったこともご議論いただけたらと思ってます．私の話はこれまでにいたしまして，今後のデータベースについて思うところを，お一方ずつお話いただきたいと思います．

ユーザの立場からの展望

石井　私はユーザの立場で話したいと思います，現実にユーザはいろんな業務にデータベースを利用していますが，そこでどういう問題が起こって，どういう問題をまず解決してもらいたいと思っているかを，できるだけ具体的な例で申しあげたいと思います．まず，データベースの成長性ということが1番重要なことになりつつあると考えます．データベースというのは，デザインして作るという段階はだんだん過ぎて，作ったものがゆっくり成長し，変化していく時代になりつつあるわけです．時間の経過にともなって，いろんな問題が起こって参ります．そういう観点に立った議論がどうもないような感じがしますので，その辺の指摘をしたいと思います．前提ですが，まず汎用大型あるいは中型機を使っているということ，しかも事務処理分野の利用で，ここ2～3年で是非実現させてもらいたいとユーザが思っていることを頭に置きました．マルチメディア・データベースや，分散データベースに関しては，時間がございませんので省き，今回はごく普通のリレーションといいますか，ファイル及びその延長上の話にとどめたいと考えております．

まずリレーションといいますか，普通のファイルというのはご存じのように，項目が横に並んでいるわけです．その属性がだんだん時間がたつにともなって変化していくわけです．まず普通のファイルを考えましょう．そのスキーマを R(AA, BB, CC, DD, EE) としましょう．つまりファイル R は AA, BB, CC, DD 及び EE という項目からできていると考えましょう．ファイル R(AA, BB, CC, DD, EE) に値が入ると，1件目のデータを1行目に，2件目のデータを2行目に，3件目のデータを3行目に書くので，これはテーブルになるわけです．

しかしこのテーブルにも時間の経過にともなって変化が起こります．たとえば AA が商品コードだとします．時間の経過にともなって，いままでなかった商品が現れます．そうしますと，それは商品コード体系に影響を及ぼして，一般には末尾に1桁ふやすという現象が起こって参ります．桁数がふえるわけです．そのときにそのファイルを作り直すとか，そのファイルを利用しているプログラムの組み直しやリコンパイルがあると，困るわけです．まずそういった問題を解決

しなければなりません.

またワープロの普及にともなって大量にデータベースの中に文章データが入り込んでくるだろうと思われます. 当然いくつかの項目の中に文章データがなまの状態あるいは多少加工されて入ることになります. CRT を利用し漢字の出力をしたい場合は1行 40 文字しか出ませんから文章を取り扱う場合は40文字, あるいは 40 文字以下で区切って, それの繰り返しにしてファイル中にもたないとハンドリングがめんどうになる. つまり, 繰り返し項目が定義できたほうが良いわけです. BB に文章データを入れた場合, 私はこのようなファイルのスキーマを R(AA, BB^{M}, CC, DD, EE) と書くと理解しやすいと思っております. M はマルチ値という意味です.

次にデータベースのデザインについて考えてみましょう. デザインは, いつもこれから始まると考えがちです. しかし実際はそうではなく, 今の状態にさらに新しいものが少し加わるという形で起こるわけです. つまり新しいアプリケーションがふえることによって, 新しい項目がふえるということが起こって参ります. 項目と項目の途中に新しい項目たとえば FF が入るかもしれませんし, 後のほうに新項目が入るのかもしれません. つまり R(AA, BB, CC, DD, EE) がたとえば R (AA, BB, FF, CC, DD, EE) となるわけです. そういうことがあったときにファイルRの作り直し, あるいは R を利用しているプログラムのリコンパイル, その他のことをしないですむようにしなければ困るわけです.

月日がたつと, アプリケーションの内容が変化し成長してくるということなんです. その裏には必ずこのような問題が起こって参ります. こんなときにもデザインのやり直し, リプログラミングなどということがあってはなりません. 非第1次正規型のリレーションを私はたとえば R(AA, BB^{M}, CC, GG^{M} (DD, EE)) と書くんです. ここで GG^{M} (DD, EE) とは DD と EE という二つの項目からなる組が繰り返すことを表しています. 組をリレーションと考えると, これがネストされてるということでネステッド・リレーションということになるんだろうと思います. こういうようなファイルが現実に表れてきます. いろんな議論があるにしても, 結局現実はこういうようなものになるわけです. そして項目の数が一体どのくらいになるかというと, 現実の世界では, 数百項目に及びます. 私の知っているある会社の人事ファイルは, 項目の数が繰り返しを入れないでなんと1500項目以上ありました. 私の経験では大きな企業の場合, 人事管理のために必要な1人の人間に対する項目の数はこの繰り返しを入れないで500以上あるのが普通です. しかも始めから500項目あるのではなく, 数年のうちにだんだん項目が増加するわけです. 論文の中で議論され良く出てくるのとずいぶん違って参ります. 正規化されたリレーションというのは, 実は R(AA, BB^{M}, CC, GG^{M}, (DD, EE)) のようなもののサブセットつまり特別な場合と考えてもいいわけです. 繰り返しを許さないのが第一次正規型といえるわけです.

次に BB^{M} (DD, EE) の並び方について考えましょう. 値が繰り返す (あるいは多値) としてもその順番に意味をもたせたいのか, もたせたくないのかというようなこともあります. たとえば1月分のデータ, 2月分のデータというように順番に意味をもたせたい場合もあります. また会社が作っている商品名がいくつか並んでいるようなときは順番に意味がないわけですね. その会社の1番重要な商品名を最初に書いて, 2番目をその次に書いてというふうになっていたとしても, 売り上げは年々変わるわけで, その会社のメインの商品はやがてそうじゃなくなることがあると, この順番に意味がなくなるわけです. そしてたとえばナイロンとテトロンを生産してる会社を捜したいという場合は, この値間の AND 条件にマッチするものを検索したくなるわけです.

それから項目 DD の中の何番目に入っていても, たとえば CODD なら CODD という名前が, 何番目に入っていてもいいから見つけ出したいということもありますし, 第1著者として CODD という名前がある場合を検索したいという場合もあります. また第1著者がだれで第2著者がだれという AND 条件の場合もあるわけです. ファイルを正規型にして, ばらばらにすることがいままで提案される一つのやり方なわけですが, ばらばらにしますと, それをひっ付ける (JOIN) ときに非常に時間がかかるわけで, そういう意味で正規化というのは, ある意味で問題があると私は思っているわけです. 正規化はアップデートに強くなるとよくいわれますが, アップデートされない項目というのが, 実はたくさんあるんです. たとえば車を表す情報があるとします, そしてそのファイルを構成する項目にメーカ名というのがあるとします. しかしトヨタの車が途中から日産に変わることはないわけですね. 同様に年式も変わりませんし, エンジンの大

きさも変わりません．したがってアップデートはないわけです．もちろんこれらの項目に対してもアップデートということはあり得ます．しかしそれは間違いの訂正のみです．したがって値の変更はないことはないですけれども，一般にすべての項目の値が，変わるということではないわけです．したがってアップデートに強くなるといっても，アップデートの少ないファイルに関しては，正規化のメリットは少なく，ジョインのオーバヘッドだけが目立つことになります．

正規化し過ぎますとパフォーマンス上の問題が起こります．実際のアプリケーションの中でデータを読む回数と，内容を変更する回数との比較をとってみますと，圧倒的に読む回数のほうが多いわけですね．つまりデータベースの内容というのは，世の中の変化に合わせてアップデートが行われるわけです．つまり住所変更は，世の中の変化にともなって起こるんであって，データベースの利用の頻度に関係ないわけです．そして，利用の頻度というのはデータベースが整備されればされるほど，頻繁になってきますから，結果的に利用の頻度が非常に多くなり，相対的にアップデートの回数は少なくなります．現実に，私どものあるお客様で，1日に1千数百万回も検索要求が出るところがありますが，それに耐えるためには，ジョインをやたらにとるなんてことをやってたら，とても現実に合わなくなるわけです．こういったことができないと困るわけです．つまり，非常に現実的で1番どろどろしたところの話をしてるわけです．非正規型のリレーションを取り扱える．ということはこんなこともできなければなりません．非正規型というと，なにか正規型と違うような感じがするんですけれども，私の感じでは非正規型リレーションは，普通のリレーションを含んでいると考えていいんじゃないかと思っています．

最後に3次元データベースの必要性について述べたいと思います．まずファイル R(AA, BB, …, EE) について，次にファイルにちょっとしたネストが入ったようなファイル R(AA, BB[M], CC, GG[M](DD, EE)) について話しました．しかしもう一つ時間軸というのが非常に重要な役目をもっています．それを絵で書きますと，この図のようになります（図-2）．名前や住所などが入っています．ところが引っ越しますと，普通は内容をオーバライトし，変えてしまいます．しかし履歴を全部とっておきたいことになると，なにか工夫をしなければなりません．この図はこれを分かりやすく絵で書いたわけです．こういうデータベースが

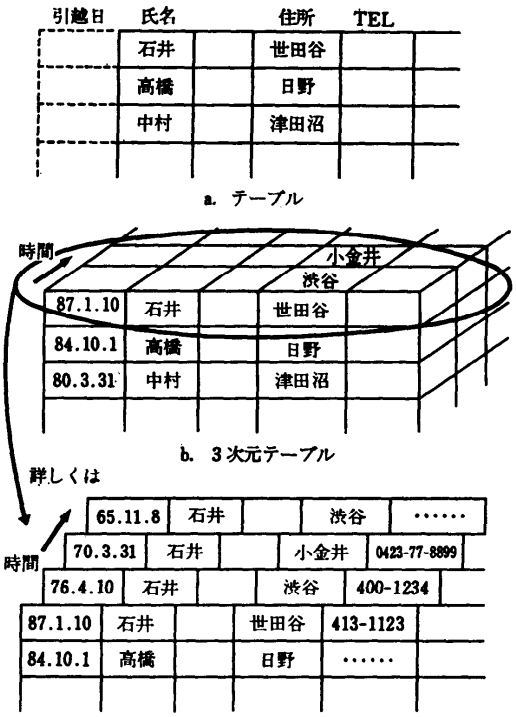


図-2 3次元データベース（石井）

あると，時間に対するハンドリングを，どうしてもプログラム内でハンドリングする方法を取ることになります．リレーショナル・モデルの提案によりデータ・サーチの方法がせっかくノンプロセデュラルになっておきながら時間の要素が入ってくると，時間に関するサーチ条件部分は相変わらずプロセデュラルサーチをやるということになるわけです．時間に関する条件に対応したデータ・サーチを手順的につまり HOW を書かないで，WHAT に切り替える必要があると思うわけです．テーブルは2次元です．それに時間軸を入れて私は3次元データベースといってるわけです．3次元データベースに対しての WHAT の例としては，たとえば昭和何年何月何日の時点で結婚していて，現在は部長職で過去に営業経験が3年以上ある人をさがせというものが一つの例です．営業経験が何年あるかなんてことがデータベースの中に値として入っていれば答を出すのは簡単です．そんなことじゃないんです．いつから営業になって，いつから別の仕事に変わったかということがデータベースに入っているだけです．そしてその合計が3年以上になってるかどうか，ということが分からないと困るわけです．別な例とし

て決算処理を考えてみましょう．決算処理をやりたいときは，すでにその決算の日を過ぎてるわけです．年度が変わりますので，組織変更が行われるとか，いろいろなことでデータベースに対するいろんなアップデートが行われます．アップデートしたあと，決算処理をしたのでは間に合わないですね．これは現実によくあるケースです．現実的には3月31日の段階でダンプをとっておいて，それをリストアして処理すればいいということで軽く逃げられますけれども，3次元データベースという考え方をもって，そのデータベースに対して何時の時点かを指定し決算処理ができるようにしなければならないと思います．また別の例として年度の初めに，営業部門の組織変更を行ったとします．そして売上の前年同月比を取りたいとします．前年の同月は，いまの組織ではなかったわけですから前年同月比を取るとき，前のデータとそのまま比較することはできないわけですね．過去にさかのぼって，過去にはなかった，今の新しい組織に合わせて過去のデータを集計し直したものと比較しなければならないわけです．また30歳の時点で結婚していて，何かの条件を満足する人を捜せという問合せも考えられます．この問合せはさっきの話とちょっと違うんです．また連続して5年以上営業経験があってということも考えなければなりません．また通算5年以上というのは連続しているとは限らないですね．それから医学関係などで，手術後3日目に，こんなことがあったこういう病気の人を捜せということも考えなければなりません．

時間の問題というのは，現実社会の中で，非常に重要なウェエイトを占めているわけですが，いくつか例で述べたような時間の問題を含んだ検索に対応する議論はどうもないような気がしてるわけです．これをノンプロセデュラルにつまり WHAT を指定して答えが得られるようにしたいものです．そのためにはデータベースとして，どういう仕組が必要で，どんなモデルがいいのかということは，やはり重要な問題じゃないかと思います．

司会 どうもありがとうございました．データベースの現場を長くみてこられた方から大変面白い話がうかがえたと思います．それでは続きまして富士通国際研の小林要さんにお願いします．

ソフトウェアモデリングとデータベース

小林 私はソフトウェアモデリングとデータベースという立場でお話したいと思います．主な内容は，まず私自身の，データベースの動向の認識を述べた上で，アプリケーションということに注目したい．それを特にソフトウェアモデリングの立場で考えたい，ということであります．

ソフトウェアモデリングの立場は，ソフトウェアの生産性向上をめざした，いわゆるソフトウェアエンジニアリングの立場なわけですが，それをさらに“モデリング”という視点から攻めようということであります．そういう目でみた場合に，今後のデータベース技術をどのように考えていくべきなのか，ということを述べようと思います．

まずデータベースの現状ですが，第1に RDB がどんどん普及してるという事実があると思います．またその背景にある記憶装置が能力，規模において大変大きなものになってきました．第2に主記憶も大変大きくなっています．10年前は，外部記憶にデータベースを乗せるのが常識だったのですが，現在は数ギガのメインメモリの上にいっぺんに乗ってしまう時代であります．

さらに第3はデータベースが対象にしようとする実際の世界，すなわち対象世界の認識が大変大きな問題になってくることです．記憶装置，主記憶の拡張も含めて RDB の機能拡張の時代が目の前にあります．そういうツールとしての計算機世界はどんどん整っていくわけですが，それに追いつかない問題があります．それが対象世界のモデリングの問題だと思います．

いままでデータベースの問題を考える視点というのは，どちらかというと，データベース管理システムの側と申しましょうか，あるいはアプリケーションを除いた計算機側の立場からアプリケーションとデータの関係を捉えていたのではないかと思います．いわばデータベースシステムの側からアプリケーション世界というものをみているようです．たとえばどういう演算が上から下りてくるかというような視点でしかみてないんではないでしょうか．

ソフトウェア開発というポイントでみますと，実はシステム全体が問題になります．すなわちデータベース側だけの視点でうまくいけば良いのだというわけではございません．ソフトウェア開発の中のデータベース開発部分は大体3分の1ぐらいで，3分の2はデータベースのアプリケーション・ソフトウェア作りで苦

労してるわけです．そうした点に貢献し得る技術がもっとあるべきだろうと考えるわけです．

その場合にアプリケーションとデータベースに共通する問題があります．それは対象世界の認識であります．しかし現在はデータベースからは，データをモデリングするという立場でしか対象世界をみていないように思います．これはいい過ぎなのですが，あえて議論を伯仲させるために申しますと，データベースシステムの側からはデータしかみてないのではないか，すなわちアプリケーションまで含めたデータベースのあり方というものをもっと考えるべきであると思います．

その場合，たとえば言語，あるいはアクセスというようなアプリケーションとデータベースの間のインタフェースの問題点として問題を捉えるかも知れませんが，しかし，先ほど申しましたように，本来の実世界のモデリングに立ち返ってみるべきではなかろうかと思います．

ところでデータの共有という概念がデータベースにはあります．共有という概念には，再利用というポイントもあると考えられます．ソフトウェア開発の立場からいいますと，プログラムの再利用，プログラムの共有は生産性や品質などに非常に効いてきます．その意味ではプログラムも共有再利用の対象であります．

広い意味でのシミュレートされたモデルというものを考えていくためには，データもプログラムも両方合わせた，対象世界というものの認識から始めるべきだと思います．そういう立場でソフトウェアのモデリングというものを考えますと，いくつかの基本的な課題があると思います．

その一つは，データを通じてみえる対象世界の認識のほうがなによりも先に前提にされることであります．データモデルから対象世界がみえるのではなくて，世界の認識が先にないとデータの理解は困難です．つまり対象世界のことを知ってる人ならば，それがどういうスキーマになったかを知るだけでデータベースを使えるわけですが，実世界のことを知らない人，あるいは部分的にしか知らない人がスキーマや制約条件だけをみせられても，それがどういう意味かを把握することは，事実上非常に困難な場合があります．

他方プログラムについても同様で，計算機の動作上の具体的な挙動というものは，そのプログラムのソースコードを追うと分かるわけですが，その計算動作がなんのためになされてるのかという認識が先にないと理解が困難です．プログラム目的の記述がそのままではできないのでだいたいはコメント記述で逃げてしまうのが現状です．

目的や対象世界が分からないと，ソフトウェアの再利用，あるいはデータの共有もないかと思われます．データベースがらみのソフトウェア開発がいまだにニーズに追いつけていないのは，こうした基本課題が解決されてないからだと考えます．

ソフトウェアモデリングの立場では，対象世界と実現世界という二つの世界に分けて考えます．この場合，'実現' というのはソフトウェアの実現（インプリメント）であります．つまり実現世界とは，データベースシステムも含めたソフトウェア全体のインプリメントの世界です．これら二つは世界が基本的に違うと考えます．実現世界は計算機装置の動きによって意味が決まりますが，対象世界は，応用側の意味づけで決まります．対象モデルと実現モデルをそれぞれなんらかの言語で書いた場合に，記述言語の指示対象がそれぞれ違うということになります．

指示対象が違うにもかかわらず，対象世界のモデルと実現世界のモデルとの間をなんとか結びつけてソフトウェアを開発しなければなりません．対象世界と実現世界の間を結びつける基本原理を明らかにして，データやプログラムの共有と再利用の意味基盤をはっきりさせる必要があります．これにヒントを与えてくれてるものがデータベースの世界とプログラム世界にそれぞれあるように思います．

データベースの世界では実は role（役割）という概念があります．プログラムの側にはそれに対応するものがなかったように思います．逆に，プログラムの世界にある inheritance（継承）という概念は，データベース側にはなかったと思います．これらが相互にどう取り込まれていくだろうということが一つのヒントにはなると思います．

役割概念と型概念は，非常に混同されやすい概念であると思います．型というのは relation（関係）を表す表にたとえて考えますと，表の中に現れる値の集合とその上の演算，これが型概念だろうと考えます．それに対して，役割概念は値の集合の直積のつながり，すなわち集合と集合の横のつながりの意味づけ情報であります．述語の引数（argument）の位置関係に込められた情報のようなものですので，個々の述語論理の式をもってきても引数の役割概念は表面からは消えてしまいます．引数の型概念をもってきても示せません．変数概念をもってきても表しにくい．オブジェク

ト概念をもってきても消えるのではないかと思います．要するに引数の位置（ポジション）に込められた概念というものは記述言語の表面からは消えてしまうのです．

実世界側の役割概念がなかなか伝わらないと，データやプログラムの目的がいつまでたっても理解できないことが起こります．値の集合をみても，あるいは論理式を追ったとしても，その式を満足するなにかがあるということまでは分かるのですが，それがなんの「役」(role) に立つのかは分からない．したがって再利用も共有も困難になる．

その意味でモデリングの技術課題の一つに，この役割概念が残されていると思います．データモデリングにおける役割概念をソフトウェアのモデリングにどう役立てるべきか，さらにはプログラミングにおける継承概念を，データベースにどう結び付けていくか，ということが鍵のように思います．

またデータベース技術は，その総体として，データベースのアプリケーション・ソフトウェア技術というところにまで，どんどん踏み込んでいくべきでありましょう．たとえば，ソフトウェアの再利用をデータベースとのかね合いにおいて捉えていくことも必要でしょう．あるいはインタフェースをさらにアプリケーション寄りにするための技術が必要になってくるはずです．広くソフトウェアモデリングという技術体系として，データベースとアプリケーション・ソフトウェアの統合化をはかるべきであろうと思います．

次世代のデータベースを考えてみますと，まずRDB は当然とされる時代になるでしょう．分散，マルチメディア，ネットワークという形態も当然の時代になるでしょう．しかしアプリケーション開発という課題が，やはり残ってしまうだろうと思います．そうした場合に，まじめに対象世界のモデルを作ろうということになるのではないでしょうか．そういう意味ではオブジェクト指向の考え方が一つの大きな方向を与えてくれていると思いますが，それが具体的にどこまで貢献するかということは今後の問題だろうと思います．

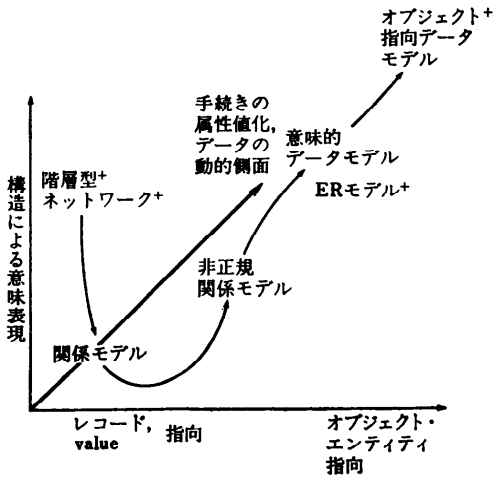
司会 どうもありがとうございました．フロアからもたくさんお話や質問をされたい方もおられると思いますが，どうぞもうしばらくそれをキープしていてください．それではいま出てきましたモデリングの新しい方向性の一つとして，オブジェクト指向の問題につきまして，神戸大学の田中先生にお願いします．

関係 DB からオブジェクト指向 DB へ

田中 有澤先生から「関係データベースからオブジェクト指向データベースへ」という扇動的なタイトルをいただきまして，私はあまり人を扇動するのは好きではないのですが，役割と心得ましてお話をさせていただきます．そこでここでは，どうして関係データベースじゃだめなのか，どこがオブジェクト指向データベースの良い点なのか，それからそもそもその定義すら固まっていないオブジェクト指向データベースというのはなんなのか，まあ，そういうふうなところを中心にお話をしてみたいと思います．ただいまの小林さんのお話にもありましたが，私もデータベースという分野が，これからこの分野だけで閉じたような形で発展していくとは思えません．人工知能・知識ベースという分野，プログラミング言語・ソフトウェアエンジニアリングの分野，そしてデータベースという分野，この三つの分野の融合ということが最近いわれ始めております．私も基本的にこの意見に賛成で，データベース技術のこれからの発展の重要な方向と考えています．当然三つの分野の融合には共通の土俵が必要です．たとえばきょうの午前中の NTT の勝野さんのお話にもありましたが，論理がその共通の土俵の一つと考えられます．たとえば関係データベースやそれをもとにした演繹データベースは論理との親和性が非常に高い．それから知識表現の方法としての論理や，Prolog のようなプログラミング言語としての論理はすでにポピュラになっています．それに対して，オブジェクト指向を一つの共通の土俵と考えますと，もう一つのグループが考えられます．プログラミング言語としてはSmalltalk-80に代表されるオブジェクト指向型プログラミング言語があります．知識表現の分野では，たとえば意味ネットワークやフレームシステムはオブジェクト指向そのものではないかもしれませんけれども，非常に親和性が高いというわけです．それに対応するデータベースの分野はどうなってるかというと，オブジェクト指向的な話はまったくいままでなかったわけではなくて，たとえばセマンティック・データモデルやエンティティリレーションシップモデルというオブジェクト指向にかなり近い話があるわけです．知識ベース，データベース，それからプログラミング言語というものが，なんらかの共通の枠組のもとに一つ

のものに融合していってもらいたいという要請はすでに出てきています．たとえばエキスパート・シェルでも，いま第2世代とかいわれておりますけれども，データベースとのリンクの重要性が指摘されています．また，従来のデータベース言語とプログラミング言語の間のギャップが大きすぎるという問題点も指摘され始めています．ですからそういうときにオブジェクト指向というのがちょうどこの三つをつなぐ一つの大きなキーワードになるだろうと思われるわけです．

次に，オブジェクト指向データベースシステムというものが期待を集めているその背景を少しお話します．まず一つは，既存のデータベース技術の応用分野が広がるにともない，いまのデータベースをさらに高機能化してほしいという要請があります．特にCADの分野のデータベース，それからマルチメディアのデータの扱いを考える上での高機能化が重要なポイントになっています．マルチメディアデータについてはたとえば画像が1枚ここにあるとします．それをデータベース化するときに，固定的なスキーマを先に作るのではなくて画像というインスタンスを先に入れてしまって，それからスキーマを作ったほうがいいのではないかと思うわけです．従来のデータベースの設計ではスキーマをまず設計してデータを入れていくという形ですが，ひょっとしますと，マルチメディアというのは，まずオブジェクトを入れてしまって，そのあとでどんどんインクリメンタルにスキーマを変えていったり，増やしていくといいますか，そういうふうな形のほうがむしろいいのではないかと考えられるわけです．二つ目の現状の問題点としてモデリング能力の弱さがあげられます．ネットワーク型・階層型・関係型のデータベースモデルすべてにいえることですが，このようなレコード指向型のデータモデルや，文字とか数値という具体的な値で表現することを中心においた「バリュ指向型」のデータモデルというのは，一般にデータの意味表現の能力が弱いということが指摘されています．たとえばエンティティとかオブジェクトと呼ばれるものを直接表現することはなかなか難しい．また，複合オブジェクト，これは種々のオブジェクトを入れ子構造などでひとまとめにしたオブジェクトですが，これを一つの処理単位として扱うことが，なかなか難しいというわけです．それから汎化・集約と申しますのは，これは is-a とか part-of と呼ばれる関係ですね．そういうものが素直に表現しにくいという問題がある．それからスキーマとインスタンスというのをいままでのデータベースというのはあまりに厳然と区別し過ぎているんではないか，そこをもう少しゆるめられないかという要請も CAD 分野であるわけです．たとえば Smalltalk-80 のクラスというのは，われわれの感覚ではスキーマに対応し，インスタンスのはまさにデータだと思うわけですが，実はクラスそのものも属性の値として登録できるわけです．そうしますとどこがスキーマでどこがインスタンスかというのをもう少しゆるめたほうがいいのではないか，ということも考えられます．また，現在のワークステーションの高度対話能力を新しいデータベース技術にどう生かすかということも重要です．たとえば概念スキーマをユーザにみせるときに，単なる表であれば通常の CRT ディスプレイの技術でいいわけですが，ワークステーションはもっと優れたユーザインタフェースが提供できるようになっているわけですね．じゃあ，関係モデルはそれを十分活用したようなデータモデルかというとどうもそうではない．それから先ほど申しました，知識ベース・プログラミング言語との整合性の問題があります．このような背景のもとに，最近オブジェクト指向ふうのデータベースの商品が出始めております．たとえばアップルのハイパーカードやデータジェネラルのオブジェクト指向設計データベースなどです．それでは，次に，オブジェクト指向データモデルの私なりの位置づけをやってみたいと思います．いままで出てきたデータモデルを並べますとこのようになります(図-3)．この絵の中で，矢印でだいたいの歴史的な出現順序を表しています．正直に申



+実際的アプローチが先行

図-3 データモデル（田中）

しますと，私が勉強したか，研究したような順序になっております．ここでまず注意していただきたいのは，'+' マークをしてあるところですね．この分野というのは理論が先行していない分野だと思われます．たとえば階層型やネットワーク型，これは実際的なアプローチが先行したモデルです．関係モデル，非正規関係モデルは理論が先行した分野です．ER モデルというのは実際的な立場，意味的データモデルというのは少し中間ぐらいなんですが．そこでここにオブジェクト指向データモデルと書きましたけれども，これは皆さんが納得して，これこそがオブジェクト指向データモデルであるというふうに広く受け入れられたものはまだ存在しないんですが，だいたいのイメージだとお考えください．それは実は理論が先行してる形にはなっていない．先ほどの商品みたいなものですね．それからプロトタイプ開発という形で，それが先行している状況にあります．これに対して，じゃあこれらはいったいどこが違うんだといいますと，いろんなことをいう人が実にたくさんおられます．特に非正規関係モデルと，オブジェクト指向データモデルは一緒だというふうな人もいます．それからウルマンは，オブジェクト指向データモデルはネットワークモデルの子孫であるというようなことをいっております．そこで私なりに勝手な絵を書いてまとめてみたんですが，こんなふうに三つの軸で考えてみたらどうだろうかというわけです．まず縦の軸，これはなにかと申しますと，これはデータの意味をスキーマという形でどのくらい表現しようとしてるかという程度を表しています．関係モデルはもっともフラットで，構造ではなるべく意味は表現しないというわけですから階層型やネットワークモデルよりもより下に位置するだろうということになります．非正規関係モデルは，より構造をもってますので少し上だろうというわけです．それから横の軸，これはなにかと申しますと，これはレコード指向，それからいわゆる数値とか文字とか，そういう計算機の上で表現可能な，いわゆるバリュですね．そういうものを中心に置いてるモデルなのか，そうじゃなくて，よりもっと概念的なレベルに近い，オブジェクトとかエンティティですね．そういうものを中心にしたものなのか，その程度を表した軸です．階層型・ネットワーク型・関係モデル，それから非正規関係モデルはレコード指向であり，バリュ指向です．ER モデルはエンティティという概念が陽に扱えますのでより右に位置します．それをより発展させて，より右に位置づけたものが，意味的データモデル，それからオブジェクト指向データモデルというわけです．私の絵の書き方が下手くそなんですが，この右のほうにいくほど実は奥の軸のほうにもいっているとお考えください．こういうふうなもう一つの軸がある．これはなにかというとデータの動的な側面をどれくらい表現しようとしているかを表してます．いわゆるオペレーショナルなセマンティックであると考えられるわけです．たとえば，オブジェクト指向型ですと，個々のデータ型に可能な操作を一体化するわけですから，そういう手続きを付けることによってデータの意味をやはり表現しようとしてるというわけです．そうしますと階層・ネットワーク・関係モデル，非正規関係モデルというのは前面にありまして，ちっとも奥のほうにはいっていないというわけです．意味的データモデル，この中には1部動的な側面をみようとしたものがありました．オブジェクト指向データモデルというのはより奥のほうにいこうとしてる．それからより上のほうにいこうとしてる．それからより右のほうにいこうとしてる．そういうふうな位置づけができるんではないかと考えるわけです．もう一つ，オブジェクト指向データベースという分野に関して注目をしていただきたいことですが，この分野にはコッド (Codd) がいないんですね．つまり先ほど申しました商品の開発とか，プロトタイプの開発が先行しております．元祖的な広く受け入れられるものがあって，理論的な研究もそれによって活発化されるとか，それからさらに言語の表現能力の議論が広がるとか，そういうふうな神棚的な存在といいますか，そういうふうな人がまだいないわけです．ですからわれわれは，これからどこにいくのかというふうなことがさっぱり分からないわけです．けれども，逆に申しますとコッドの関係モデルというのは出てから15年以上たったわけですけれども，データベースの現在の技術というのは，一つの安定した状態にあるわけですね．ですからいまじっくりと，次世代と申しましてもあまり2年3年じゃなくて，もう少し先をみたじっくりとした研究といいますか，開発といいますか，それをいまやるべきじゃないかと思うわけです．私の予稿には，すでに指摘されているオブジェクト指向データベースの問題点や改良についても触れております．たとえばマルチユーザサポートの機構が Smalltalk-80 にはないから，それをやらないといけないとか，そういう議論が多いんですが，私は，もっと本質的な，オブジェクト指向データベース

の数学的なフレームワークをどうするんだということや，オブジェクト指向データベースのスキーマの設計というのはなんなのか（実はこの場合にはメソッドも含めたようなスキーマ設計になると思いますけれども），そういう議論をするべきだと考えるわけです．さらに，ユーザインタフェースの問題にしましても，じゃあ Smalltalk-80 の MVC モデルのようなものでよいのかどうかという基本的な議論がいま必要じゃないかと考えております．

司会　どうもありがとうございました．次世代データベースのパラダイムについて，大変おもしろいお話をお聞きできたわけですけれども，今度はまたちょっと立場を変えて，三菱電機の宇田川さんのほうから，次世代 DB アプリケーションに向けての拡張関係データモデル，ということでお話をお伺いします．

次世代 DB としての関係モデル

宇田川　私の今日の立場は，CAD データベースを作るうえで，関係データベースが生き残れるかという話をしたいと思います．基本的な立場としましては，関係データモデルの応援演説にやってきたつもりです．なぜ最近 CAD 用のデータベースかという話ですけれども，1番の原因は CAD システムが世の中に普及してきたということだと思います．CAD システムが普及して参りますと，設計オブジェクトがだんだんたまってくるわけですね．そうしますと，作っては捨てではもったいないということで，うまくできたものは検索して再利用したいという要求が出てきます．それから CAD を使い込むに従って，設計ツールもだんだん高度なものがたくさんできてくるわけですね．そうしますと，そういった設計ツール間でのデータの一貫性をとってほしいという要求が出てきます．こういった要求に答えるものが，すなわちエンジニアリング・データベースです．このエンジニアリング・データベースを作るために，関係データベースに基づいたアプローチがとられてきました．これは 1970 年代の後半から，多分 1980 年代の前半ぐらいまでだったと思うんですが，残念ながら機能の面でも，性能の面でもエンジニアリング・データベースの要求を満たすことができなかったというのが正直なところだったと思います．それで近年関係データベース以外のデータモデルが，結構公然と議論されるような時代になってきたわけです．これはいってみれば，関係データベース・クライシスというふうな状況にいま落ち入ってるんじゃないかと思います．そこで私の今日の立場でありますCAD データベースでも，関係データベースは生き残れるかということが問題になるわけです．私の考え方は非常にオプティミスティックでして，今後とも大きな影響力を保ち続けるだろうと思っています．それはプログラミング言語 FORTRAN をみてくれということです（笑）．状況証拠しかありません．将来のことはだれも分かりませんから．プログラミング言語 FORTRAN もいろんなことをいわれてきまして，無茶苦茶なことをいわれたんですが，結局 30 年間しぶとく生き残ってきたわけです．関係データベースも誕生以来 15 年，予稿集には 25 年と書いてありますけれども，これは単純な計算ミスです（笑）．それで 15 年間生きてきたわけですが，これからも生きていくだろう．ただし，先ほど申しましたように性能の面でも機能の面でも要求を満たしていませんから，いくつかの拡張をしていかなくちゃだめだろうというのが基本的な考え方です．それで，どういう拡張をしたらいいのかということですが，3点ほど今回は取りあげてみました．一つは類似検索機能と一貫性管理機能の強化です．それから高速処理のためのなんらかの手法を考えなくちゃいけない．それから Co-Media 化とここでは呼びますが，多種メディアのサポートといったようなことを考えてみようと思います．

1番初めの課題であります類似検索再利用，それから一貫性管理ということですが，どうもエンジニアリング・システムをいろいろと考えてみますと，おもしろい奇妙な現象があるように思います(図-4)．一貫性管理機能と検索再利用という機能がデータベースにあるわけですが，この機能には相反する要求があるようです．つまり一貫性管理機能が要求されるところでは

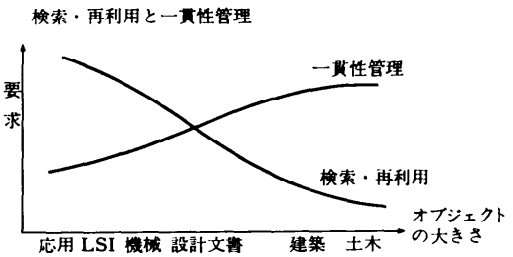


図-4　(宇田川)

わりと検索再利用という要求が少なくなってきます．それからその逆に検索再利用の要求が大きいところでは一貫性管理の要求が低くなってくるように思います．CAD システムというのは，非常に複雑なものですから，一概にはいえないんですけれども，いろいろな設計フェーズで，その要求がどんどん変わるわけなんですが，ごく大ざっぱにいって LSI，機械といったようなところでは，オブジェクトの数が増えてきますから，データの再利用，検索といったような機能が要求されてきます．それから建築ですとか土木といった分野になりますと，一品設計の傾向が強くなりまして，複雑な解析ツールを統合する必要があるようでして，一貫性管理機能のほうがより多く要求されてくるようです．それから，この二つの式ですが，いわゆる検索再利用というのはだいたいオブジェクトの数に比例してふえてくる．それから一貫性管理の機能の要求の強さというのは，設計ツールの数に比例して増えてくると，こういうことはだいたいいえるように思います．ですから CAD システムが発展していくに従ってエンジニアリング・データベースの要求が強くなってくると思うんですが，その要求の内容もちょっと様変りしてきまして，検索というのは，類似検索が多いと思います．すなわち単なるフィールドの値がいくつであるということではなくて，設計オブジェクトが，どのくらい似てるかといった形で検索したいという要求が強くなってきます．それから一貫性管理の手法ですが，これはやはりツールに非常に強く依存してくると思います．ツールの中からデータベースを呼んで一貫性管理をしていく．設計オブジェクトというのは非常に複雑な動作を示しますから簡単な論理演算といったようなものではなかなか書けないというのが現状だと思います．

次に，高速処理に向けて，ということですけれども，これは昔から高速化の手法が研究されてきています．その手法をごく大ざっぱにまとめますとこうなると思います．手法は三つありましてファームウェア化，それから主記憶をふんだんに使っていこうとするもの．それから DB マシンといったようなものを開発しようという，この三つの方法があると思います．このアプローチを比較してみます．まず手法ですね，ファームウェア化というのはどういうことかといいますと，つまり，コードの最適化ということです．それから主記憶データベースの高速化の手法とはメモリの追加ということです．それから DB マシンでは，マシンを開発することだということです．使用するハードウェアを比較してみますと，ファームウェア化は特にありません．主記憶 DB は主記憶だけ．DB マシンはプロセッサ，プラス主記憶，場合によっては2次記憶も入ってくることもあります．これを実現する上でどれくらい労力とコストがかかるかということを考えますと，ファームウェア化は中くらい，主記憶 DB は小，これは主記憶をくっつければいいだけですから簡単です．データベース・マシンになりますとソフトウェアもハードウェアも作らなくちゃいけないので，これはコストが高い，大であると．そして高速化の度合はどれくらいかといいますとファームウェア化ではだいたい中くらい，主記憶 DB も中くらい，DB マシンは大いに期待できるだろうと．これを割り算するわけですね．労力プラス・コスト，割る高速化の度合ということになります．中割中でだいたい中だと．大割大も中ですね．小割中をやりますと大になるわけです(笑)．ということで私としては，やはりこの主記憶データベースというのが有望じゃないかなと思っています．ただし主記憶データベースには，一つの大きな問題があります．リカバリに関する技術的な問題です．すなわち落ちてしまいますと，大量のデータがいっぺんに消えてしまうわけです．ここで，エンジニアリングにおける特有の情報の性質というのを少し強調したいんです．エンジニアリングでは，データを絶対にリカバしなくてはいけないという要求はわりと薄いわけです．事務処理ですと，たとえば銀行のオンライン・システムなんかですと，データが失われたら絶対にだめなんですね．ですから落ちないように細心の注意をしながら，一歩一歩ディスクに書き込みながら先へ少しずつ進んでいくわけです．それに対してエンジニアリングでは，図面の編集中にバーンと落ちてしまって，なくなっても“もう1回編集しなおしてよ”，とこういえばすんでしまうことが多いわけです．ですからリカバリも絶対に戻らなくちゃいけないということじゃないですね．おおむね戻ればいい，ということがあるわけです．主記憶データベースで，リカバリといえばバージョン管理があります．昔作ったバージョンをいくつか管理しておけば，おおむねリカバリできるという考え方から，おそらくエンジニアリング・データベースではバージョン管理が重要視されるのだろうと思います．すなわち，エンジニアリング・データベースでいうバージョンというのは，実は事務用データベースのリカバリに相当するような

機能も含まれていると思います.

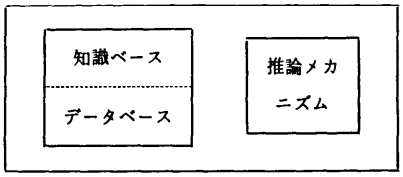
次は3番目の Co-Media ということですが，これはあまり聞きなれない言葉だと思います．いわゆる，マルチ・メディアのことです．マルチ・メディアという言葉はメディアがたくさんあるということでしかなかったんですが，私は Co-Media というのは3種類ぐらいあるだろうと思います．一つは Complex Media というステップがありまして，これは単にフィールドのデータ型として，いろんなメディアをサポートできる段階，こういう段階が第1ステップとしてあると思います．それから2番目は Cooperated Media というステップでして，これはそういった多種のメディアがなんらかの意味的な制約をもっていて，互いに片方が変更されますと，もう片方の違うメディアも変更される．つまり，有機的に結合した Complex Media といったような段階があるだろうということです．その次は Comprehensive Media という段階があると思います．この段階になりますと，いままでのような考え方がもっと大きくなりまして，先ほどの小林さんですとか田中先生の意見にも通じるところがあるんですが，システムですとかオブジェクト，それからプログラムですとかさらに知識，それから Cooperated Media といったようなものが全部有機的に扱えるような，理想的な世界が待ってるんじゃないのかと思います．この Comprehensive Media の段階になりますとコンピュータ・エンジニアというのがいなくなってしまう．いなくなってなににになるかといいますと，みんな Comedian になってしまうわけです (笑).

司会 どうもありがとうございました．最後に落ちもありまして，大変おもしろかったんですが，それではパネリストの最後に日本 IBM 東京基礎研究所の戸沢義夫さんから，知識情報処理の立場から次世代データベースについて，お話を伺いたいと思います．

知識情報処理と次世代 DB

戸沢 知識ベースの立場からデータベースの話をみますと，いろいろとおもしろいところがあるのに気がつきましたので，ちょっとお話をさせていただきます．まずデータベースに，データの代わりに知識を入れたら，知識ベースができるという考え方です．最初この考え方を聞いたときに，なんでこんなばかなことといってるんだろうと思ったわけで，現実にいまの技術を考えますと，いまのデータベースに知識を蓄えても知識ベースにはならないんです．しかしここのところにいろいろとおもしろいアイディアがあります．

知識と推論の枠組によるプログラミング



知識と推論の枠組によるプログラミングではあらかじめ知識を蓄えておくと，推論メカニズムが問題を解いてくれる．知識ベースにも，モデル化，共有，問合せの問題がある．

図-5 (戸沢)

その前に知識ベースを，どう捉えるといいかについて説明させていただきたいと思います．知識ベースというのは，一つの新しいプログラミング・パラダイムだと考えています．つまり，「知識と推論の枠組によるプログラミング」が先にあり，そこから知識ベースという言葉が出てきたと考えています．いままでのプログラミングでは問題を解くための手続きや手順を書いたんです．図を見てください(**図-5**)．知識ベースの新しいプログラミングによりますと，手続きを書きません．問題を解くのに必要な知識だけを，断片的に書き，知識ベースに放り込みます．そうすると推論メカニズムが自動的に問題を解いてくれます．そういうことができる方法があります．といっても，それは理想論でありまして，現実にはそんなふうにはいかないんです．それでも，理想的な「知識と推論の枠組によるプログラミング」というのは手順を書かないで，問題を解くのに必要な知識を蓄えておくと，推論メカニズムが問題を解いてくれると，そうなってます．

知識ベースといってるところは，実はデータベースと込みになってることが多いです．代表的推論メカニズムには，前向き推論でしたら OPS5，OPS83 という言語があります．後向き推論ですと Prolog です．こっち側にある知識ベースとデータベースとの間には点線が引いてあります．これらは理想的には推論メカニズムから離れてるはずなんです．しかし，実際のシステムではこれら全部が一体になっています．

要は図-5 の枠組でアプリケーションを書きたいわけです．アプリケーションを簡単に早く書きたいというのがプログラミングという目的です．この枠組を使うと，いままでより簡単にアプリケーションが開発で

きる，これが知識ベースの捉え方です．知識ベースの人たちの目標がプログラミングにあり，データベースの人たちと違うのでお互いに関心がないというところがあったわけです．

ところで，データベースのほうにはデータ独立という概念があります．これはデータベースでもっとも基本になってます．なぜこの概念が重要であるかというと，要はアプリケーションを開発したいんだけれども，そのアプリケーションの中に，データへアクセスする部分がたくさんあります．一方でデータは世の中が進むにつれてだんだん変わってきます．そういうものに引っ張られて毎回プログラム変えなければいけないのは困るから，ここを切り離しておきたいというのがあるわけです．

こういうふうに切り離しておくと，なかなかいいことがたくさんあります．たとえば同じデータベースを使って別のプログラムをもってくると新しいアプリケーションが簡単に開発できます．また，データの共有もできるようになります．こういういいことがあるから，プログラムとデータを分けておきましょう．それが，データベースの基本になっているデータ独立の考え方です．この観点からみますと，データベースの世界で問題になるところは，後からどんなプログラムがきてもいいように，どのようにデータベースを作っておくかです．

そのためにデータモデルという考え方があります．うまくデータを整理する指針を作りましょう．それがデータモデルの一つ大事な方向です．もう一つは，データが共有されますと，その共有をどう管理すべきかが問題になります．ここからデータベース管理システムが出てきます．データベース管理システムの大きな役割はデータの共有を実現することにあります．こっちでのアップデートがちゃんとあっちに反映されなければなりません．それから，データベースとプログラムが分かれてますから，プログラムからデータベースに対して，どういうふうにアクセスすべきかという問合せの問題があります．データ独立から(1)データモデル，(2)共有，(3)問合せの三つのテーマが出てきます．

いまパネリストの方々のお話をずっと聞いてますとデータモデルが強調されています．ここのところは1番難しいんですが，これがうまくできると非常にいいわけです．あとからプログラムを追加するだけで簡単にアプリケーション開発ができるはずなんです．データベースの世界の人たちは，あるワールドをできるだけ自然にマッピングしてデータベースを作りたいのです．それに対して知識ベースの世界の人たちは，ワールドがありますとそれをマッピングして，データベースを作るんじゃないんですね．それをデータ構造とそのインスタンスと，それからルールで記述することを考えます．それでいかにしてそのワールドそのものがうまく機械の中で表せるかということを一生懸命研究してるわけです．基本的には，ワールドを表したいということでは同じなんです．

ただデータのボリュームが全然違います．知識ベースでは全部メモリ内で処理できる範囲内に納まってるし，データベースでは大容量のディスクを使わなければいけない．そして高速処理や統一性，回復処理，共有化するためのロックやセキュリティなど，いろんな問題が発生する．それらを別にしますと，ワールドを記述する点では知識ベースの分野では非常にいろんな研究がされています．そうすると基本的な発想は同じですから，知識ベース側での研究がデータベースにどんどん使われてくるだろうということは当然考えられます．これからの課題も知識ベースの方向とデータベースの方向は決してずれてない，ということがいえるわけです．

先ほどデータベースであげた三つのテーマを知識ベースに当てはめてみますと，モデル化については，いろいろな研究があります．共有は抜けています．どうやって共有したらいいかという話は，知識ベースの世界では全然されていません．最後は問合せの問題ですが，知識ベースに対する問合せというのはいったいなにかといいますと，こういうことになります．知識をたくさん集めたとします．なにを質問したいかというと，「私はこういう問題を抱えてます．問題を解くのに必要な知識はなんですか」．あるいは「どの知識をどういう順番でどのように適用したらよいですか」となります．ただ現実にはこれができないので，知識ベースと推論メカニズムというのは合体してるわけです．切り離されていないのです．本当は切り離されたら，非常にいいと思います．切り離すときにはデータ独立ではない．別なコンセプトが必要みたいです．それがなにかというのはいまよく分かりません．次世代データベースの一つの方向というのは，こういうところにもあるんじゃないかなと思います．

司会 どうもありがとうございました．皆さんがいろんな立場から議論されているわけですけれども，か

なり違った面から，違った言葉を使いながら，実は同じ議論をしてると感じられるところも多々ありましたし，真っ向から違うというところもあったような気がいたします．これからフロアから活発なご議論をお願いしたいと思います．どうぞ．

討　論

岡本(横浜国大)　最後の話の中で知識ベースの場合は，知識を入れておけば，あと推論機構がやってくれるという話があったんですが，私はそれは嘘じゃないかという話をよく聞きます．知識はその知識を扱う手続きまで含めた形で書いてやらなきゃいけないと考えますが，この考え方はオブジェクト指向に非常に近いものがあると思います．そこら辺のところ，オブジェクト指向の話をなさった田中さんとか，いま知識処理のほうの話をなさった戸沢さんを中心に意見を聞かせていただけませんか．

田中　私は知識ベースや AI の専門家ではありませんけれども，先ほども少し申しましたが最近の第２世代のエキスパートシェルと呼ばれるものですね，これの特徴をみてみますと，単一の知識表現方法をもってるものは少なくて要するに考えられる知識表現方法はもう全部こみで入っていて，適当なものを選びなさいという形ですね．それで，それをまとめるときにフレーム指向，もしくはオブジェクト指向という考え方が有効ではないかと思うわけです．ただし私がよくそういうものをみまして不満なのは，たとえばフレーム・システムとデータベースと結合する機能も必要だからということで，実は個々のフレームと，関係データベースの１レコードを対応づけるとかというふうな形が多いことです．もう少し落ち着いて考えまして，知識表現方法とデータベースとの結合というのは非常に大事だから，その場合に，じゃあ関係データベースでよろしいんですかということをじっくり落ち着いて考えなければいけないと思います．もう一つ不満なのは第２世代型といわれているものはいろんなタイプの知識表現方法があり，たとえば if-then ルールとフレーム型というものが両方使えるものがあります．ところが中味をみてみますと，結局は両方融合したんではなくて，ただ単にこういうアプリケーションならこちらを使いなさい，こういうアプリケーションならこちらを使いなさいということで，ばらばらという感じがするわけです．オブジェクト指向という概念で，じゃあルールもオブジェクトであるという形で本格的に組み込んだものというのは意外に少ないんではないかなという気がしてまして，まだまだ研究の余地があるんではないかな，というふうに考えております．

司会　ひと言だけお伺いしたいんですけど，そのオブジェクト指向という枠組の中で，ルールとか，その他いままで扱ってなかったものも組み込む余地があるとお考えですか．つまりオブジェクト指向という考え方自体の枠組の中でもともとやれるんだけれども，いまはやれてないということですか．本来オブジェクト指向モデルの中ではそんなことはとてもできない，という考え方もあるかとも思うんですけれども．

田中　むしろ，ルールというのを，どういう形のオブジェクトで表現するかという議論自体が意外と少ないんじゃないかと思います．ただ単にフレームで知識を表現しておいて，あとは勝手にそのうえで好きなルールを書いておきなさいというふうな，意外とばらばらのツールが多いような感じをもっております．

戸沢　知識ベースというのは，推論メカニズムが問題を解きます，という話をしたんですけれども，そんなのは嘘じゃないかと質問されましたが，それは現在の知識ベース技術ではそれができないんです．ただし知識ベースシステムの目指してるところは，そういうことを狙っています．そう理解していただきたいんです．データベースでもデータ独立という概念は昔からあったんですが，それは目標であって実現されてなかったですね．知識ベースとオブジェクト指向はどういう関係にあるかということですが，私は，オブジェクト指向というのは，それもアプリケーション開発のパラダイムの一つだと捉えております．知識ベースのプログラミングも一つのパラダイムだし，オブジェクト指向もパラダイムで，それらはどちらがどっちかに含まれるという関係にはありません．知識ベースのパラダイムをとると，オブジェクト指向の考え方を取り入れるのが結構難しいし，オブジェクト指向でやりますと，書きにくい知識もあります．実際にやりたいことはアプリケーション開発を簡単にしたいことです．あるいは，ワールドというものを，できるだけコンピュータの中に素直に記述したいともいえます．ワールドを素直に表現したいといったときに，知識で表現する方法もあれば，たとえばフレームですとか，セマンティックネットですとか，オブジェクト指向とかいう考え方で表現する方法もあります．知識ベースと，世の中を素直に表現することとが必ずしもかみ合ってないというのも正しいと思います．

岡本 データベースがあって，知識ベースがあって，推論機構があってといったときの推論機構と知識ベースとの間のところが確かにはっきりと分かれていないわけですね．手続きみたいなものを合わせもったものが知識だという見方をした場合には，本当の推論機構の核になる部分と，手続き的ではあるんだけれどももっと知識のほうに寄った部分というのを分けることに難しさを感じます．知識処理の技術が進んだ場合に，そこら辺を明確に分ける境界線みたいなものが，はっきりできてくるというふうにはお考えでしょうか．

戸沢 手続きと知識がどこが違うかというのは難しいんです．とにかく分けたいという希望があると，それを目指してるということは間違いありません．ただ将来それができるかどうかについては，現在はなんともいえないです．ただある限られたアプリケーションについていえば，そういうふうに分けて書くことができます．たとえば MYCIN はそれを証明しています．一般的にどんなアプリケーションでも必ず分けられるというふうにはいきません．将来はいろんなアプリケーションにおいても分けられるようになることを期待しています．

司会 いまの点について，小林さんはパラダイムとしての力量としていかがお考えでしょうか．

小林 私の話でも継承（インヘリタンス）ということを申しましたけれど，オブジェクト指向というのを私がどう捉えてるかというのをいいますと，カプセル化(エンカプスレーション：encapsulation)と継承(インヘリタンス：inheritance)であると思います．オブジェクト指向といった場合にいろんな解釈がございます．まずそこで議論が分かれちゃうんですが，カプセル化と継承だと私は思います．その意味では先ほどお話ありましたように，パラダイムの一つでしかないんですが，実はソフトウェアエンジニアリングの歴史からひもといていきますと，カプセル化という技術は，特にオブジェクト指向で始まった技術ではない．古いものでありますが，それなりにモジュラリティを高める技術だったんですね．対象を認識したということは，なんらかのモジュラリティを認識したということで，そのモジュラリティを表す技術としてカプセル化というのは非常に適切だと思います．その意味でなにか認識したものに基づいてモデルを表現するということによって知識を表現していく，という手段として考えますと，オブジェクト指向というのは考えられると思います．かつ継承というメカニズムが実は推論のメカニズムにかかわっておりまして，その意味で知識表現ということと，オブジェクト指向ということがからんでるわけですが，先ほど話がありましたように，手続き的側面があります．つまりカプセル化ということ自体が，実は抽象データ型といいましょうか，データ抽象化 (data abstraction) といいましょうか，数学的な側面からいうと集合とそれに対する演算の集まりとして，つまりメソッドというのがオペレーションでございますから，その意味では代数形として捉えてる基本的な視点があったわけです．ですから代数形として捉えてる側面でのモデリングという本質があるだろうと思いますので，代数的に捉えたほうがいいか，あるいはロジカルに捉えたほうがいいか，あるいはうんぬんといった，そういう違いに結びつく面があると．もう一ついわせていただきますと，継承という概念は，ソフトエンジニアリングのセンスから申しますとモジュラリティに関しては悪化させるのです．つまりモジュラリティ追求の立場からいいますと継承というのは最悪のモジュラリティでございます．つまり性質を受け継いでしまうわけですから相手に依存してしまうのです．その意味で知識の時間的な推移がある場合，状況が変わって知識が書き換えたいというような場合に，この継承のメカニズムというのが知識の更新を阻害する要因になりはしないかという議論があります．

司会 なかなか難しい議論になってまいりましたが，いまのことにかかわっても，違うことでも結構ですが，フロアからどうぞ．

穂鷹(筑波大) 先ほどの田中先生の3次元，ちょっとおもしろかったんですけれども，あの図でリレーショナル・モデルが左側のほうの手前に置かれてたんですが，それで ER モデルが右側のほうに位置してたんですが，そういう理解で普通はいいのかと思いますが，こう考えるともうちょっとリレーショナル・モデルがよくなるんじゃないかなと私は思うんです．つまり関係モデルで，いわゆるバリュとしてとってるものは，実はエンティティをリファしてるのであると．そう考えますとリレーショナル・モデルというのはエンティティ間のリレーションシップを書いてあることで，これはどんなモデルでもリレーションシップというのは出てきますから，関係モデルでバリュといってるものが実はエンティティなのだとして，関係モデルを読み替えますと，それでエンティティを扱ったモデルになると思われます．それからもう一つ，

コンプレックス・オブジェクトを作るオペレーションというのは，いくつか知られてるわけで，アグリゲーションであるとか，あるいはセットを一つのエンティティとみなすという抽象化とか，いくつかあるんですけれども，そのようにコンプレックス・オブジェクトを作るいくつかの簡単なルールで作られたものを，またエンティティとみなすということをやりますと，やはり全部関係モデルの枠内で，いわゆるオブジェクトという概念，いまの小林さんの説明ですとエンカプシュレーションとインヘリタンスと二つの部分の入った段階までのオブジェクト指向あるいは，エンティティ・モデルというのはリレーショナル・モデルの読み替えでいっちゃうのではないかと思います．その点について田中先生，あるいは他の先生方のご意見お聞きしたいんです．

司会 一つ確認させておいていただきたいんですが関係モデル自身は，あくまでシンボル操作の世界であって，いまおっしゃったようにテーブルのアトリビュートとアトリビュートの間に，これとこれは同じオブジェクトを表してるとかいうのは，リレーショナル・モデルが本来もっている枠組の中ではないわけですね．しかしそういう解釈を外から加えるなり，なんらかの機構として埋め込んだならば同じ表現ができるではないか，という主張なのでしょうか．

穂鷹 つまりエンティティをリファしてるものと思って書きくだしますと，実はそれはリレーショナル・モデルではないか．要するにエンティティの名前をバリュとしていってるだけだと思えばですね．

司会 その思うということ自身はリレーショナル・モデルの枠組の中では記述できませんよね．

穂鷹 だからそういうふうに，リレーショナル・モデルのバリュというのは，エンティティを指していたのだとちょっと考えを変えれば，つまりドメインの中にエンティティがつまってたと思えば，それで即オブジェクトをリファしてると考えられるのではないか．あえてオブジェクトだということで，別のモデルにならないのじゃないかということです．

田中 私も実は一部賛成でして，この絵は各モデルの表現能力を厳密に書いた絵ではございません．非常に直感的な絵です．たとえば穂鷹先生がおっしゃった関係モデルのドメインをシンボル，ものの名前とするんであれば，これはレコード指向，バリュ指向です．そういう意味で私のこの絵は関係モデルというよりも，むしろ既存の関係データベースシステムと読み替えたほうがよろしいかと思います．つまり既存の関係データベースシステムでは属性値はたとえば文字型数値型というシンボルでしか表現できません．ですから穂鷹先生のおっしゃった，たとえばドメインにこういうエンティティを入れたような関係モデルというのは，多分拡張関係モデルということで，もっと右のほうに関係モデル・ダッシュという形で位置づけることができるかと思います．それが1点です．それからコンプレックス・オブジェクトの話ですが，これもいい訳するわけじゃないですが，たとえば関係モデルの中にオブジェクトの ID という概念を入れて，それから関係名も入れられるようにしてというふうに，すべてドメインになんでもかんでも入れられるようにしましょう，プログラムも入れましょう，というふうなことをやれば関係モデルであえて表現できないことはないと思うんです．実は議論をややこしくしてますのは，あの1番右上に書いてあります問題のオブジェクト指向データモデルというものは，果たして存在するのかと，実はまだこれが広く受け入れられて，なるほどというものが確立されてないわけですね．ですから方向としてこの三つの軸を示したというふうに理解していただければよろしいかと思います．

司会 いまのことに関しまして宇田川さんもひと言おありになると思いますが．関係モデルというのはどのレベルのモデルであるのかということですね．先ほど拡張関係モデルの中で，実はオブジェクト指向であるような部分がずいぶん出てきたように思われるんですが，いかがでしょうか．

宇田川 エンジニアリングということを考えますと，このオブジェクト指向というのは，どうしても避けられないと思います．特にレコード指向のアクセスが重要だということを考えますと1レコード，2レコードというのはあまり意味がなくて，複数のリレーションの集合が意味のあるオブジェクトということになると思います．そういうものをやはり主記憶にロードできるくらいまで絞り込むのが私はオブジェクト指向ということの意味じゃないかなと思います．それからデータの動的側面ということで石井さんも指摘されてたんですけれども，これも重要だと思います．ただデータの時間としての軸というのが，ディスクリートに起こって，その起こる頻度がどれくらいかということでもって，サポートするレベルが変わってくると思います．人事データなんかですと半年とか1年とか，あるいはその起こったときですね．何か月という単位で

もって起きるんでしょうし，CAD ですと日とか分ぐらいかもしれないですけれども，それぐらいで起こるわけです．そういう頻度ということを考え合わせた上でデータモデルというのを提案していかないと使いものにならないといいますか．まあ使いものになってるものを作ってるわけじゃないですけれども，なかなか応用にまで納得のいくようなデータモデルというのはできてこないんじゃないかというふうに思っています．

司会 関係モデルの拡張でオブジェクトの世界が表せるということに関してイエスかノーかということになれば．

宇田川 オブジェクト指向というのは一つのディスプリンだと思っています．プログラミング言語でも，オブジェクト指向C言語などが出てるわけです．それと同じようにオブジェクト指向リレーショナル・データベースというようなことがあっても，ちっともおかしくないわけで，むしろリレーショナル・データベースの支持をする皆さまはリレーショナル・データベースの美しい理論体系に酔いしれてしまうんではなくて，やはり，まずいところはまずいということを認めて，大いに拡張するということに賛成していただきたいと個人的には思いますけれども．

司会 ではご質問どうぞ．

牧之内(富士通研) 私どもも実はオブジェクト指向データベースの開発をしておりまして，残念ながらまだ田中先生のリストの中には名前が入ってないんですけれども．そこでわれわれが悩んだのも，ちょうどいま穂鷹先生がおっしゃった点なんですが，つまり関係モデルというのは，非常に美しいということで，私も関係モデルにかかわったものとして捨てがたい魅力もある．一方オブジェクト指向データモデルというのも，どうも世界の潮流だから，流れに乗り遅れてはだめだから，開発しなければいけない．それではそこを関係モデルと，オブジェクト指向データモデルと，どう関係づけるか，どこが違うのかということをいろいろ考えたんです．私はそういう意味で穂鷹先生がいうようにオブジェクト指向データモデルは関係モデルの拡張でいいよとは単純にはいえないという気がします．その一つは，これはぼくの誤解もあるかもしれないんですけれども，ぼくが感じてる点は，やはり関係モデルはバリュ主体なんですね．しかしオブジェクトは，やはりオブジェクトであると．それでオブジェクト指向の一つの重要な点は，オブジェクト・アイデンティファイアをもつということで，このオブジェクト・アイデンティファイアというのは，システム全体でユニークなもので，かつオブジェクトを一意にアイデンティファイするものでなければならない．それでリレーショナル・モデルというのは，あれはそういう概念はまったくなくて，たとえば一つの関係にプロジェクションをかけると値が出てくるわけです．それではたとえば，ある従業員の年令と名前をプロジェクションしたときに，そこに出てきた値のペアというのはいったい何なのかということを考えると，それはリレーショナル・モデルだったら，ただ単なるリレーションだよということですむわけですけれども，それじゃオブジェクト指向で，あるオブジェクトの年令と名前をフェッチしてきたとき，それは何なのかというと，やはりあるオブジェクトの年令と名前であって，そのオブジェクトからは離れられないものなわけですね．そういう意味で，なかなかオブジェクト指向データモデルに対して，関係モデルのような，代数演算みたいな感じで，なにか言語体系を構築しようとすると，なかなか難しい点があるなと感じております．

その点が一つと，もう一つはやはり動的な面を表すのに，なかなか難しいんじゃないかと思います．現在最近の TODS に，ストンブレーカが書いてるように，プログラムを入れればいいということなんですけれども，あれは確かに一つ，いろんな面で成功する面はあるでしょうけれども，たとえばメソッドみたいなものを，どういうふうに入れるのかということになると，多分あれには答えが書いてないんじゃないかという気がします．だからその二つの面で，ぼくはやはり関係モデルとオブジェクト指向データモデルというのは違うんだというふうに感じております．われわれの立場として，われわれがいま開発してるもの，ジャスミンという名前なんですけど，それは関係モデルはインナーモデルとして使っております．関係モデルというのは，要するにファイルの物理構造を表すのに非常にいいという考えで使っております．それには使えるんじゃないかという気がします．私は穂鷹先生とか，他の皆さんの議論を聞いてて感じたことです．

穂鷹 いまの牧之内さんの件ですけれども，二つあったうちの動的なところは，私もよく分からなくて，難しいなと思ってます．リレーショナル・モデルでいうとトリガとかなにかその辺で頑張るのかなと思うんですけど．先ほどのオブジェクトの ID のうんぬんというのも，これは単純な問題で，それぞれのリレーショ

ンのところでリファしているアトリビュートのドメインがたまたま全部同じ一つのドメインを指していたと．ですから，ある関係のAというフィールドと，別の関係のBのフィールドがあって，そこのバリュがともにエンティティを指していて同じドメインを指したと考えればよいわけです．オブジェクトというデータのドメインの名前を指したと考えれば全然問題はないというふうに思います．どの範囲内でユニークにするかというのは，ドメインのところでオブジェクトのIDを一貫して考えればよろしい，その辺は別に問題はないと思います．

司会 ですけれども，先ほどの議論の骨子は，むしろシンボル操作系としての関係データベースと，それからオブジェクトという概念，つまりそれは意味に対応すると思うんですけれども，意味の上での操作というものとが，必ずしも1対1に対応できないわけですね．先ほどのプロジェクションの例などはいい例だと思うんですけれども．関係モデルはもともとは tuple is entity のつもりだったかもしれないけれども，ではそのとき，そのプロジェクションをとって，あとに残るものはいったい何の意味を表してるんだということですね．

穂鷹 私リレーショナル・モデルといったときに，その集合演算というのをあまり重視して考えてないものですから．

小島(電子技術総合研) 私のいいたいことは2点あるんですが，一つは，次世代データモデルDBMSができるかということです．新しいことを考える場合も関係モデルも一応意識しておられると思うんですけれども，次世代データベースの環境において，データベースの理論屋ですね．モデル論をやってるとか設計，そういう人たちに未来はあるのかという問題が一つです．

もう一つは関係モデル，いま良きにしろ悪しきにしろ，関係モデルというのを意識せざるを得ない環境にあるんですけれども，たとえば私たちはこれから先，関係モデルにさよなら，といって捨ててしまうことができるかどうか，もし関係モデルは完全になにかほかのものに対して置き替われるんであれば，具体的にどういうところがキーポイントになるのか，もしどなたか考えをおもちであれば教えていただきたいんですけれども．

石井 二つ目のリレーショナル・モデルは捨てられるか，ということですけれども，私は捨てなきゃだめなんじゃないかと思ってます．やはり現実を非常に無視してるところがたくさんあって，私はむしろ捨てたほうがいいと．ただリレーショナル・データモデルの提案の意義とか，それがもたらしたいい点というのは，十二分に意義があったわけです．やはり時代の進歩の1ステップとして意義があったと思います．しかし，現実との間に大き過ぎるギャップがあるので，私は捨てなければだめじゃないかと思っています．最初の質問が聞き取れなかったのですが．

小島 一つ目の質問をちょっと補足させていただきます．関係モデルに対していろいろ能力がたりないということで，より一般的なモデル化の能力であると，たとえば手続き的なものをデータベースに入れる．要するにより汎用の方向に向かって能力を要求されてると思うわけです．とすると最近プログラム言語と同じような，データベースを統合した環境というのがたとえばワークステーションなんかに提供されてますけれども，そうするとたとえばプログラム言語における正当性とか停止性の問題とどう違う，つまりデータベースはデータ管理とかスキーマというのがありましたから，部分的な問題を解くことで問題は簡単に解けたんだという特徴があったと思うんですけれども，より機能が要求されることで，問題はより難しくなって，一般化してしまって結局一般のプログラム言語であるとか，知識表現における問題とほとんど変わらなくなってしまって，じゃあデータベースなにをやっているんだというふうな批判があるんじゃないかという危惧が一瞬したんでお聞きしたかったんです．

石井 私はそういう意味でリレーショナル・モデルの提案は非常に重要だったと思うんです．データを捜すロジックをプログラミング言語の中にどんなふうに書くかということを考え，データ・サーチのためのアルゴリズムを書かないようにしなくちゃいかんのだといい出したわけです．このことをはっきりつかまえていい出したことは非常に意義があることだと思います．そういう意味でプログラミング言語に大きく関連しますが，彼は初めからサブランゲージといっていますから，言語全体を指していないわけです．データを捜す部分の記述についてだけ論じたわけです．私は先ほどその延長上で時間に関する問合せまで広げなければいかんのだということを申しあげたわけです．その意味ではおっしゃるようにプログラム言語や，知識表現の問題と深く関連してくると思っています．先ほど小林さんのおっしゃったプログラミング言語としての改良

のワンステップになったという点でリレーショナル・モデルの意義はかなり重要じゃないかと思いますね.

藤代(東京大) いまそこにみせていただいてる田中先生の図に関してなんですが, データモデルを目的ということをはっきりさせたほうがいいと思いますが, たとえば関係モデルと ER モデルが, あの3次元の指座では同一の土俵で語られてると思うんですけれども. たとえば P. Chen が最初に ER モデルを提唱したときの論文に, そのデータのビューという考え方があって, データベースを設計する際に, 実世界から何段階かあるといってますね. その場合, ER モデルというのは, たとえば1番最初のレベルから2番目ぐらいのつなぎのところをやるんだと, そうするとたとえばオブジェクト指向データモデルというのがもし存在するとすれば, それというのは実世界に1番近いレベルから, 計算機のうえで実働するところまでを全部カバーするモデルなのか, それともきのう増永先生のご講演に関係して質問があったと思いますけれども, エンティティというのにある種の抽象化を施した次の段階のものをオブジェクトと考えて, それをさらに先ほど牧之内先生がおっしゃっていたインナモデルのある種のオブジェクトなのかもしれませんが, そういったものへの中間段階として捉えるのか, どちらなんでしょうか. その辺をちょっと教えていただきたい.

田中 まず, ここでのデータモデルは, 概念レベルや論理レベルなどという区別をしておりません. そもそも次世代データモデルにこのようなレベル分けが必要かどうかも議論のあるところでしょう. 次に, ER モデルというのは, これはエンティティという概念的なものと属性という具体値をいれる入れものがありましたね. だからエンティティとデータ・バリュをちょうどひっつけた中間的な位置づけだと思っておりますので, ER モデルというのは関係モデルよりも, よりオブジェクト・エンティティ指向, より概念モデルのレベルのモデルであるだろうということでちょっと右のほうに書いてるわけです. それからもう一つの質問のオブジェクト指向データモデルというのは, じゃあエンティティだけを扱ってるのかと, エンティティといいますか概念レベルのものだけを扱ってるのかといいますと, 私のイメージはそうではなくて, たとえば有澤先生というオブジェクトがあれば, それは概念的にも, また現実の実世界でも一つのオブジェクトであって, その中をめくっていきますと, どこかにたとえば文字列で住所が書いてあるとか, そういう意味で1番左のいわゆるレコードとかバリュですね, そこをどこか中をめくっていけばどこかにそれにぶつかることもあるという意味で, より実世界のオブジェクトを, とにかく素直に表現しましょうというふうな位置づけですので, エンティティだけを扱ってるという意味ではありません.

周(図書館情報大) 宇田川さんのおっしゃった類似検索という言葉自身がちょっとあいまいですが, ちょっと説明していただきたいんですが, もう一つは, オブジェクト指向データモデルといっても, あるすべてのオブジェクトはあるクラスに属していて, そのクラスは大体分類と同じように分けていて, そうしたらクラスの間のジョインがあるのかどうか. ちょっと聞きたいんですけれども. あとオブジェクト指向モデルでは普通は Smalltalk 言語ではメモリ上で動いているから, クラスのメソッドがなければ, 上にたどっていって使えばいいんですけれども, ここではデータベースですから1次記憶装置に格納する可能性もあるんじゃないかな. そうしたらマルチ・インヘリタンスの場合にはもしすべてのデータ, 情報はメモリの中に置けない場合はマルチ・インヘリタンスのメソッドのアクセスは非常に難しくなるんじゃないかなという質問です.

宇田川 まず類似検索ということですけれども, いままでの類似検索というのはバリュを中心にした, いわゆるレコードの類似性ということを基本にしてるわけですね. これだけじゃなくてリレーション同士の類似性ということも私は考えられるんじゃないかと思います. それで一つのオブジェクトを複数個のリレーションで表現しますね. たとえばAという回路図と, Bの回路図を同じスキーマをもったリレーションで表現するわけですが, そのリレーションどうしの引き算をやりますと距離というのが出てくるはずなんです. なぜかといいますとリレーショナル・データベースのリレーションというのは集合ですから引き算で1意的にその値が決まるわけです. その意味でリレーション同士の類似性ということが, とりも直さずオブジェクトの近さにまで発展してくるという意味での類似検索というのを考えています. 決してレコード単位の類似検索ではないわけです. それからクラスの話ですけれども, クラス同士の間で, いわゆるジョインができるかというのは, これはやはり場合々々によって違うと思うんです. ただし, 一つのクラスのインスタンス同士はスキーマが同じですから, これはいつでもジョイ

ンもできるし，引き算もできるし，たし算もできるというイメージで捉えています．最後の質問はちょっと分からなかったんで．

司会 そろそろ時間になりましたので，まだたくさん議論があるかと思いますが，一応ここで終わらせていただきます．きょうのディスカッションはどうもオブジェクト指向データベースは次世代データベースとなり得るかということにかなり絞られてたような気がいたします．しばらく前からデータベースをやってきた人間の感想といたしましては，一時関係データベースというもののみがデータモデルのようにいわれ，かつ非常に理論的にディスカッションされてた時代があり，それに比べて，現在はかなり柔軟にいろいろな発想が飛びかうという新しい戦国時代を迎えて，これからしばらくの間データベースの世界も非常に楽しく，ホットな議論が続くのではないかと期待しております．ただ先ほど田中先生のほうからありましたようにいまのところ「コッドがいない」．大変象徴的なお言葉なんですが，これに関しては是非われわれ日本人の中から，コッドのようにトータルに新しいモデルを提唱する人間が出て，その人のモデルが 20 年か，25 年ですか，とにかく一定の期間一つの柱になってほしい，というふうに思います．いずれにしてもデータベースの今後の新しい展開に大いに期待したいと思います．熱心なディスカッションをどうもありがとうございました．