表とデータベース†

Tables and Databases − An Allegory

ネットワーク情報学部　石鎚　英也
School of Network and Information　Hideya ISHIZUCHI

Keywords : table, relation, functional dependency, database, normal form, SQL

1．プロローグ

「久しぶりやなぁ、プータン。元気かいな〜。」
「あっ、ぺぺさん。こんにちは。ご無沙汰してます。」
「そういやぁ、こないだの試験どうやった？」
「･･･」
「ま、ええわ。ところでなぁ、ももちゃんクラブの仕事、ぎょうさんたまってんねん。手伝ってくれへんか。」
「でも今からバイトが…。」
「そんなん、かまへん。電話しときぃ。」
「･･･」
というわけで、プータンは仕事を手伝うことになった。

☆☆☆☆☆

「ところで、仕事って何ですか？」
「色々あるが、まず表を作ることからやな。」
「表って？」
「何言うとんねん。『会員が星の数ほど増えたよって、今までみたいにチンタラしてたらあかん。活動の役割分担ぐらいちゃんとしときなはれ』いうて、去年の“ももちゃんクラブ”の総会でつっこまれたやんけ。せやから、次の総会で、とりあえず幹部だけでもデータ集めて“情報公開”すんねん。会員の“知る権利”に応え、“アカウンタビリティ”の向上を図るっちゅうわけや。じゃ、頼んだでぇ。」
「『頼んだでぇ』って？…いっしょにやるんじゃ…行っちゃったし。」

☆☆☆☆☆

「できました。」
「ずいぶん早いな…ン？何やこれは。」
「表です。」

ももちゃんクラブ（オリジナル）		
会員	ぬいぐるみ経験	
ぺぺ	もも	4
	スヌ	15
プー	もも	2
	うさこ	1
かな	もも	3
	ぷりん	5
けい	スヌ	10
じい	チャッピ	20

「んなこたぁ見りゃ分かるが、なんでこんなに複雑なんや？」
「複雑ですかぁ。会員名とぬいぐるみ（以下「ぬい」と略記）の種類と収集歴を書いただけですけど…。」
「パソコンにデータ入れて、データベースにする言うてるやろ。人間には分かりやすうても、機械はそうはいかんのやでぇ。“会員名”と“ぬいぐるみ経験”は属性や。わいの“ぬいぐるみ経験”の（属性）値は（もも, 4）やら（スヌ, 15）やら、ややこしいやろ。“ぬいぐるみ経験”やめて、“ぬいぐるみ”と“経験”の２つに属性を分けるべきやな。『関係のどの属性の値もそれ以上分解できない値となっているとき、その関係は第１正規形である』んやな、これが。エッヘン。」（「関係」は表に相当する。正規形については【付録1】参照）
「（なんかぁ、エラそうだしぃ）。データベースにするなんて聞いてないけど…でも、よく知ってますね。」
「いや、なに。昔、“ご隠居はん”がそんなこと言ってたような…ハハ。ともかく、行の途中で分割したりせんと、レコード[1]は１行にきっちり書いた方がええで。“Simple is beautiful. Beautiful is powerful.”やよってにな。じゃ、頼んだでぇ。」
「『頼んだでぇ』って…また行っちゃった。」

† 本解説記事（文献 [1] の続編）は、ネットワーク情報学部１年次配当科目「情報処理概論」の授業で配布した読み物の一部に加筆・修正を行ったものです。話に登場する人物・団体等は架空のものです。

1 record：値の並びとして構成されるデータ (文献 [3])。表の各行に相当する。以下用語については主に [3] に拠った。

2. それから数日後

ガラガラガラ。
「どなたかと思うたら、プータンか。まあお上がり。」
「こんにちは、ご隠居さま。実は、これこれしかじかで…“神様の弟子”と異名を取るご隠居さまに、表の作り方を教えてもらいたいのですが。」
「『これこれしかじか』って、誰に話しとるの？ま、大体のところは分かったがな。先ず、表を見せてごらん。おお、これか。最初よりは良くなっておるな。だが、クラブ活動の役割分担などは入れなくても良いのかな？」

ももちゃんクラブ（Ver.2）		
会員	ぬいぐるみ	経験
ペペ	もも	4
ペペ	スヌ	15
プー	もも	2
プー	うさこ	1
かな	もも	3
かな	ぷりん	5
けい	スヌ	10
じい	チャッピ	20

「あっ、いっけなーい。そうでした。“ぬい”のことしか頭になくて（ポリポリ）。では、早速っ。」

☆☆☆☆☆

「できました。ふぅ。」
「仕事が早いのう。」
「よく言われます。早いのが取り柄だって。」
「…ン？コリャなんじゃ。」
「表ですが…（またなのォ）。」
「なんか言ったか…まあよい、ボチボチいこう。ちょっと説明しておくれ。」
「＜ハイ＞。ぬいぐるみの収集と研究を“ももちゃんクラブ”ではしています。“会員”は会員名、“住所”は会員の住所で、“ぬい”は担当するぬいぐるみを示しています。“経験”は、ぬいに対する経験年数です。また、2大“ぬいぐるみどころ”のハム・シティとペンギン村では特色が違うんで、2地区に分けて、住所によってどちらか1つを担当してもらってます。＜／ハイ＞。」
「XMLじゃぁないんだから、変なタグなんか使わんでよろし。ところで、“ももちゃんクラブ（決定版かな？）”を作っておって、何か気付かなかったかな？」
「ハァ、“経験”確認するの結構大変でした。」
「うーむ、それだけか…。…そもそもこの表は、関数従属性が考慮されておらんし、第3正規形にもなっておらんではないかッ。無駄じゃ、無駄じゃ。分解せいっ。分解っ！」

ももちゃんクラブ（決定版かな？）				
会員	住所	ぬい	経験	地区
プー	東京	もも	2	ペンギン村
プー	東京	うさこ	1	ペンギン村
ペペ	大阪	もも	4	ハム・シティ
ペペ	大阪	スヌ	15	ハム・シティ
かな	横浜	もも	3	ペンギン村
かな	横浜	ぷりん	5	ペンギン村
けい	神戸	スヌ	10	ハム・シティ
じい	東京	チャッピ	20	ペンギン村

「ひえー。こ、こわ～い。」
「ごめんごめん。年甲斐もなく、少しばかり興奮してしもうた。今のは独り言じゃ。閑話休題。先ず、関数従属性じゃ。“会員”名が1つ決まると“住所”が1つだけ決まるな。同様に“住所”が1つ決まるとその担当“地区”が1つ決まる…。」
「そして、“経験”（年数）が1つ決まると、“会員”名が1つ決まる。ちょっとひねってみました(^O^)。」
「ばっかも～んっ。それはたまたまじゃよ。幹部以外の会員で、経験年数が同じ人がいるかも知れんじゃろ。意味を考えなさい、意味を。いかん、いかん。また独り言を言うてしもうたわい。プータンや、お利口だから泣くんじゃないわよん。」
「こわくて、気持ちわりぃぃぃ。」
「で、だ。このように『ある属性（集合）Xの値が1つ定まると、ある属性Y（集合）の値がただ1つ定まるとき、YはXに関数従属するといって、X→Yと書く』んじゃな、これが。そしてぇ、
『ある属性Xの値が決まると、その他の全ての属性の値が決まる』…①
とき、Xをキー（key：候補キー）という。キーXは属性の集合でもかまわんが、そのときには、
『無駄があってはならーん！』…②
つまりじゃ、Xが集合のとき、Xから何かの属性aを除いても（これをX-{a}と書くんじゃが）、『属性集合X-{a}の値が決まると、(やはり)その他の全ての属性の値が決まる』ということであれば、Xには無駄があったというわけじゃ。そして、条件①を満足する無駄のない属性（集合）Xをキーと呼ぶわけじゃよ[2]。ここで問題じゃが…例えば、{日付，客先番号，客氏名，品目番号，品名，販売数量}をフィールド[3]とする販売表があったとして、何がキーになるかの？」

2　この場合、X→Yなる関係は、全関数従属性（full functional dependency）とも呼ばれる。
3　field：レコードの構成要素。欄とも言う。属性あるいは表の各列に相当する。

「なんとか番号というのはIDのことみたいだから、X=｛客先番号，品目番号｝がキーですか。」
「これこれ、フィーリングだけで答えるものではないぞよ。確かに、“客先番号→客氏名”とか“品目番号→品名”は成り立つ（つまり、｛客先番号，品目番号｝→｛客氏名，品名｝）。じゃが、Xから日付や販売数量は決まらんぞ。」
「じゃあ、決まらないものぜーんぶ入れて、Y=｛客先番号，品目番号，日付，販売数量｝にすれば、キーになりませんか？」
「うーむ。大胆じゃが、確かに、Yの値が決まれば、それ以外の属性（つまり客氏名と品名）は決まるから、レコードは識別されて1つに決まるナ。じゃがな、これは『②無駄があってはならーん！』に違反しとるぞよ。“販売数量”が無駄じゃ。販売数量は、客氏名と品名と日付が決まれば決まる[4]から、はずしてよいじゃろ。つまり、Z=｛客先番号，品目番号，日付｝がキーというわけじゃよ。」
「なーるほどぉ。ちょっとだけ分かったような気が…（錯覚かな？）。でもご隠居さん、この函数従属性って、何かの役に立つんですか？」
「エッ、“函数”と言うたのか。若いに似合わず、随分とオールドファッションじゃのぉ。それはともかく函数、もとい、関数従属性に何の御利益があるかというと、これを使ってな、表を分解できるのじゃよ。そして、分解により各表が簡単になって、データの重複も少なく、データの挿入・削除・更新が楽になるのじゃな、むにゃむにゃ…。」
「・・・」
「ま、ボチボチ行こう。取り敢えず“ももちゃんクラブ（決定版かな？）”を2つの表に分けるとすると、それぞれどのようなフィールドにすればよいかな？」
「｛会員，住所｝と、｛ぬい，経験，地区｝の2つでどうでしょう。」
「ちょっ、ちょっと待ちんしゃい。それだと2つの表にリレーション[5]が付けにくいんでないかい。フィールドに重なりがあってもいいんじゃよ。」
「なーんだ。それならやっぱり主役の“ぬい”を共通にして、｛会員，住所，ぬい｝と、｛ぬい，経験，地区｝に分けるとよいのでは。」
「うーむ。ほとんどなーんも考えとらんようじゃなぁ、ふわァ。じゃあ、実際に表を作ってごらん。」

☆☆☆☆☆

「できました。ふーぅ。」
「仕事が早いのう。ゆっくり寝とるひまもないわい。どれどれ。」
「“其の一”と“其の二”は“ももちゃんクラブ（決定版かな？）”を属性集合｛会員，住所，ぬい｝と、｛ぬい，経験，地区｝に関して射影[6]した表です（ちょっと勉強してきちゃった～♪）。」

其の一		
会員	住所	ぬい
プー	東京	もも
プー	東京	うさこ
ぺぺ	大阪	もも
ぺぺ	大阪	スヌ
かな	横浜	もも
かな	横浜	ぷりん
けい	神戸	スヌ
じい	東京	チャッピ

其の二		
ぬい	経験	地区
もも	2	ペンギン村
うさこ	1	ペンギン村
もも	4	ハム・シティ
スヌ	15	ハム・シティ
もも	3	ペンギン村
ぷりん	5	ペンギン村
スヌ	10	ハム・シティ
チャッピ	20	ペンギン村

「おぉ、なんというすばらしい答。ワンダフル、ツーダフル！…では、“（自然）結合[7]”も知っておろうな。この2つの表を結合してごらん。」
「エー？。せっかく分解したのにぃ…元に戻るだけじゃないのかなぁ。」

☆☆☆☆☆

「できました。ふ～～ぅ。」
「仕事が早いのは感心ぢゃ。やってみての感想はどうじゃね。」
「今度はよく分かりました。元には戻りません。“其の一＆其の二”は、大きさ2倍の表になっちゃいました。網掛けしてあるレコードが、元の表にはないデータです。」
「その通りじゃ。実はな、表を分解してからくっつけ直すと、一般に“元のレコード＋α”となるんじゃよ。だが、多くの場合、うまく分解すると、α＝φ（空。つまり元の表そのまま）にできる[8]。」

---

4 「特定の日に特定の客が買った特定の品物の数量」は一通りしかないからね。
5 relation：ここでは、表と表との関係。
6 【付録2】を参照。
7 【付録2】を参照。
8 “無損失（lossless）分解”と言う。

其の一＆其の二				
会員	住所	ぬい	経験	地区
プー	東京	もも	2	ペンギン村
プー	東京	もも	4	ハム・シティ
プー	東京	もも	3	ペンギン村
プー	東京	うさこ	1	ペンギン村
ぺぺ	大阪	もも	2	ペンギン村
ぺぺ	大阪	もも	4	ハム・シティ
ぺぺ	大阪	もも	3	ペンギン村
ぺぺ	大阪	スヌ	15	ハム・シティ
ぺぺ	大阪	スヌ	10	ハム・シティ
かな	横浜	もも	2	ペンギン村
かな	横浜	もも	4	ハム・シティ
かな	横浜	もも	3	ペンギン村
かな	横浜	ぷりん	5	ペンギン村
けい	神戸	スヌ	15	ハム・シティ
けい	神戸	スヌ	10	ハム・シティ
じい	東京	チャッピ	20	ペンギン村

「どうすればできるんですか？」
「一つは意味を考えることじゃ。“其の一”、“其の二”は意味不明な表じゃ。」
「そういえば、適当な名前が見つからなかったので、“其の一、二”なんて名前つけちゃいました。」
「もう一つは、前言ったように関数従属性を考えて分けることじゃな。属性｛会員，住所，ぬい，経験，地区｝の間にどんな関数従属性があるか挙げてごらん。“会員→会員”とか、“｛会員，ぬい｝→ぬい”などは関数従属性じゃが、当たり前なのでそれ以外じゃよ。」

「うーん、そうですねぇ。“会員→｛住所，地区｝”とか、“｛会員，ぬい｝→経験”とかですかぁ。」
「うーむ、すばらしい（なでなで）。考えればできるではないか。」
「えへッ、また誉められちゃった。」
「では、この２つの従属性を使って、表を作ってごらん。」

☆☆☆☆☆

クラブ１号		
会員	住所	地区
プー	東京	ペンギン村
ぺぺ	大阪	ハム・シティ
かな	横浜	ペンギン村
けい	神戸	ハム・シティ
じい	東京	ペンギン村

クラブ２号		
会員	ぬい	経験
プー	もも	2
プー	うさこ	1
ぺぺ	もも	4
ぺぺ	スヌ	15
かな	もも	3
かな	ぷりん	5
けい	スヌ	10
じい	チャッピ	20

「ほれっ！」
「わっ、もうできたのか。ビックリするではないか。
「かなりコンパクトで、いー感じになりました。今度は結合しても大丈夫ですぅ。今度こそ決定版かな？」
「うむ。ずいぶんと良くなった。それにしてもなんちゅうネーミングかねぇ。飲み屋のチェーン店かと思ったぞヨ。でも、まだ改善の余地ありなんだな、これが。まず、関数従属性“会員→｛住所，地区｝”じゃが、これは２つの関数従属性、“会員→住所”と“住所→地区”に分けられるので、“クラブ１号”の表もさらに分けられるハズじゃ。ちょいとやってごらん。」

☆☆☆☆☆

「ハーイ。ついでに、クラブ２号もリネームして、新装開店で～す。」
「よかろう。とりあえず完成じゃ。」
「(●＾ｏ＾●)」
「この３つを結合しても、最初の表（も、疲れたんで“完成版かな？”と短縮形で呼ぶぞ）が復元できるハズじゃ。そして、これらの表は第３正規形になっておる。そもそも第３正規形[9]とは…いやいや、それは後回しじゃ。ところで、こうやって表を小分けにすると、どういうメリットがあるかのう。」
「zzz…」
「これこれ、安心して眠ってはいかんぞ。まず、データの挿入の場合じゃ。氏名と住所は分かっているが、ぬい情報が分かっていない者が新規会員になる場合、“完成版かな？”では、空白が生じて好ましくない（他人が「記入漏

9　（ボイス・コッドの）第３正規形…第１正規形の関係Ｒの任意の属性集合Ｘに対して、以下の条件が成立する場合。正規形には他にも色々ある（【付録1】を参照）。
「Ｘに含まれないいずれかの属性ａがＸに関数従属であるとき、Ｒの任意の属性ｂはＸに関数従属」

れでは？」と思うではないか)。改訂版では、“会員住所”表にだけ登録しておけばよい。削除についても、担当を一時的に外れる会員がおった場合、“完成版かな？”では、住所も含めて全て抹消されてしまうぞ。これでは休会か退会か分からんではないか。さらにじゃ、住所変更などの更新のケースでも、“完成版かな？”では、対応するデータを見つけて全て変更せにゃならん。例えば、プータンが大阪に行ったとすると、“完成版かな？”では、2ヶ所の変更が必要じゃが、この改訂版では1ヶ所だけ変更すれば済むし、担当地区は変更不要じゃ。」

会員住所	
会員	住所
プー	東京
ぺぺ	大阪
かな	横浜
けい	神戸
じい	東京

住所－地区対応	
住所	地区
東京	ペンギン村
大阪	ハム・シティ
横浜	ペンギン村
神戸	ハム・シティ

ぬいの経験		
会員	ぬい	経験
プー	もも	2
プー	うさこ	1
ぺぺ	もも	4
ぺぺ	スヌ	15
かな	もも	3
かな	ぷりん	5
けい	スヌ	10
じい	チャッピ	20

「なーるほどぉ。」
「ところで、本来ならもっとIDを使ったほうが良いな。ここになが～い名前のぬいがあったとしてごらん。例えばじゃ、“モナリザ白黒ビットマップ風ぬいぐるみ[10]”とか、会員名にしても、“竜宮の乙姫の元結の切り外し[11]”とか、“じゅげむじゅげむ、ごこうのすりきれ、かいじゃりすいぎょのすいぎょうまつ…”。」
「わっ、分かりました。資源の無駄ですからもうやめて下さい。要するに、長い名前があったりすると問題だと。」
「うーむ。久々に一席やろうと思ったのじゃが。ま、そういうことじゃ。入力の手間もミスも避けたいんで、普通はIDを用いるナ。」
「そうすると、表の数が更にめっちゃ増えませんか？」
「勿論増えるが、それでものちのち便利でナ。例えば、“プータン”の名前を間違えて、“ぶーたん”と入力してしまったとしてごらん。改訂版でも3ヶ所の修正がいるが、IDを使っていると1箇所で済むぞよ。大したことないと思うかも知れんが、10万レコードの表を考えてみるとよい。それにじゃ、もし名前が“じゅげむじゅげむ、ごこうのすりきれ、…”。」
「わっ、分かりました。」

…かくして、ようやくのことで表が完成したのだった（フ～ゥ)。　(完)

【付録1】正規形のいろいろ[12]

ここでは、正規形に関する定義を多少フォーマルに記してみましょう。

第1正規形

関係のどの属性の値もそれ以上分解できない値となっていること。

例：性別

次の表（上）は第1正規形ではない。

・問題点：「氏名→性別」という関係が分からないし、更新も分かりにくい（例えば、Pepeをfemaleに変更する場合、性別の方をmaleからfemaleに変更してしまうかも知れません)。
・下表は第1正規形に変更したもので、変更時のあいまい性や手間が減少します。

氏名	性別
{Poo, Pepe}	male
{Bunbun}	female

氏名	性別
Poo	male
Pepe	male
Bunbun	female

10　解像度の高い静止画や動画は「重い」のでウェブページに掲載する時などデータを圧縮したり減色したりするが、当然ながら画質とのトレードオフがある。図は、授業のレポート課題の1つ（ルーブル美術館オフィシャルサイト（文献[5])のモナリザの画像を白黒ビットマップに変換したもの)。これじゃダリだ。

11　日本で一番長い植物の名前（らしい)。

12　用語については文献[3]を、定義や例については文献[6], [7]を参照した。

第2正規形

第1正規形であって、全てのキー無縁な属性が全てのキーに全関数従属すること。ここで、「キー無縁」(nonprime) な属性とは、どのキーにも属さない属性のこと（いずれかのキーに属する属性は prime と呼ばれます)。

例：フライトスケジュール

下表で、「フライト」と「日付」の両方が決まれば、残りの「パイロット」と「ゲート」は一意に決まります（またいずれかが欠ければ「パイロット」は決まりません)。つまり、{フライト, 日付} はキーであり、「フライト」と「日付」は prime です。しかし、他にキーはなく、「パイロット」と「ゲート」はキー無縁です。

フライト	日付	パイロット	ゲート
123	1/1	Poo	1
123	1/2	Pepe	1
321	1/5	Bunbun	2

・ゲートはフライトのみに依存して決まるとするならば、(キーである {フライト, 日付} に全関数従属していない[13] ので）この表は第2正規形ではありません（明らかに第1正規形ではありますが)。
・問題点：最初のレコードで、フライト123のゲートを3に変更したとすると、2番目のレコードも変更が必要となります (つまり、他のレコードのスキャンが要ります)。
・下表のように2つに分ければ、それぞれの表は第2正規形となり、変更は1箇所で済みます。

フライト	日付	パイロット
123	1/1	Poo
123	1/2	Pepe
321	1/5	Bunbun

フライト	ゲート
123	1
321	2

第3正規形

第1正規形であり、キー無縁な任意の属性は、全てのキーについて（狭義）推移従属にならないこと。ここで、属性 a が属性集合 X に（狭義）推移従属している（transitively dependent）とは、「X → Y」かつ「Y → X でない」かつ「Y → a」なる属性集合 Y が存在することです。

例：

次の表において、X={フライト, 日付} はキーです（それ以外のキーはない)。従って、キー無縁な属性は、「ID」と「パイロット」です。X がキーなので「X → ID」ですが、「ID → パイロット」も成立します。そして、明らかに「ID → X」ではありません。従って、「パイロット」は X に推移従属しており、この表は第3正規形ではありません。ただし、キー無縁な属性「ID」と「パイロット」は、いずれも X に全関数従属しているので第2正規形ではあります。

フライト	日付	ID	パイロット
123	1/1	1234	Poo
123	1/2	2345	Pepe
321	1/5	1234	Poo

・問題点：最初のレコードで、ID1234のパイロットをBunbun に変更したとすると、3番目のレコードも変更が必要となります（つまり、他のレコードのスキャンが要ります)。
・下のように2つの表に分けていれば、それぞれの表は第3正規形となり、変更は1箇所で済みます。

フライト	日付	ID
123	1/1	1234
123	1/2	2345
321	1/5	1234

ID	パイロット
1234	Poo
2345	Pepe

第4正規形他

第4正規形は、関数従属性を拡張した多値従属性（multi-valued dependency)という（関数従属性以外の）関係によって定義される正規形（高次正規形）です。第5正規形など他にも様々な正規形がこれまで定義されてきましたが、ここでは触れません。関心のあるみなさんは、専門書を参照してください。

【付録2】Access によるデータベース操作

本文の最後にある3つの表（テーブル）を例として、Microsoft Access 2003（以下アクセスと呼ぶ）によるデータベースの簡単な操作を説明します。

表の作成

表（テーブル）の作成手順は以下の通りです。

• アクセスを起動し、「新規作成」+「空のデータベース」メニューを選択します。そして、適当なフォルダに適当な名前 (既定の拡張子は “mdb”) でファイルを保存します[14]。

13　②が満たされないこのような関数従属性は、部分従属である (partially dependent) と呼ばれる。

14　データを入力する前に保存操作が（システムから）要求されます (他の操作の途中段階でも)。データが「命」のデータベースソフトらしい配慮です。

• 図 1のような画面が開きます。図に示されているようにテーブルは3通りの方法（デザインビュー・ウィザード・データ入力）で作成することができます。ここでは、デザインビューを使って、本文にある「ぬいの経験」表を作成してみましょう。

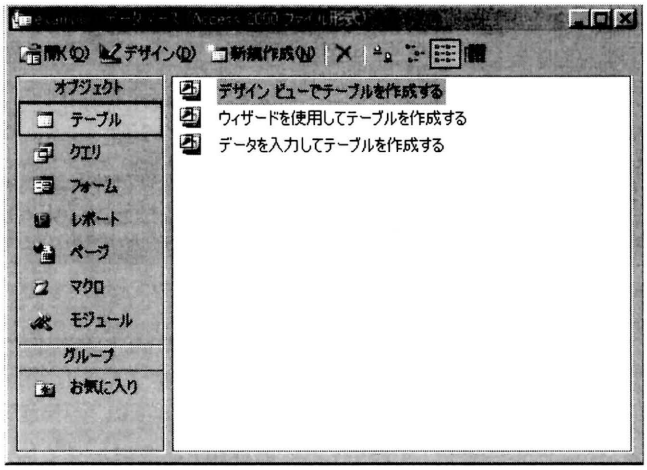

図 1

• 「デザインビューでテーブルを作成する」をダブルクリック（あるいは、ツールバーの「開く」をクリック）すると、図 2のような画面が現れます。

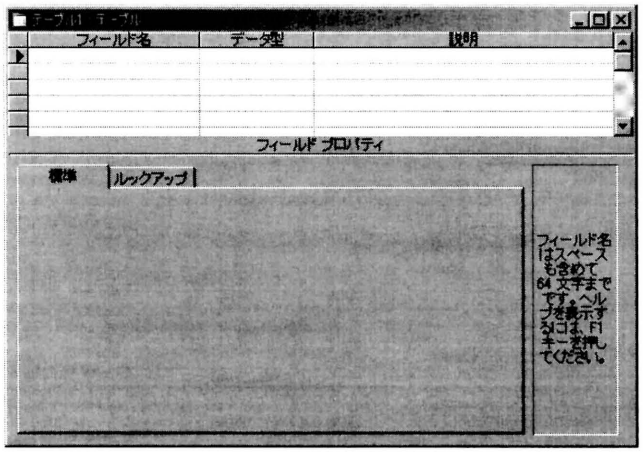

図 2

• これは表を構成するフィールドの定義を行うものです[15]。フィールド名はその名の通り、フィールドの名前を示します。「ぬいの経験」表を作成する場合だと、フィールドは3つで、フィールド名は「会員」、「ぬい」、「経験」となります。また「データ型」は、フィールドのデータの型を示すもので、テキスト型・数値型・日付／時刻型・通貨型・オートナンバー型などがあります。「ぬいの経験」表では、「会員」、「ぬい」フィールドがテキスト型、「経験」フィールドが数値型になります。

[15] フィールドの定義の並びはレコード型と呼ばれます。従って、図2はレコード型を定義するものです。

図 3

• 図 3のようにデータ型を設定したら、次にレコードのデータを入力しましょう。「表示」＋「データシートビュー」かメニューバーのビューアイコンで「データシートビュー」を選択します。

• 「まずテーブルを保存する必要があります。保存してもよろしいですか？」というメッセージが表示されますので、「はい」を選択し、適当なテーブル名（「ぬいの経験」）をつけて保存します。しかし、ここで「主キーが設定されていません。」と警告が出ます。主キーはレコードを一意に識別するフィールドです。「ぬいの経験」表だと、会員・ぬい・経験のいずれのフィールドの値が決まっても他のフィールドの値は一意に定まりませんので、主キーは別に設定が必要ですが、ここでは設定しないことにしましょう[16]。

• 画面が図 4のように変わります。この画面から各レコードのデータを入力していきます。

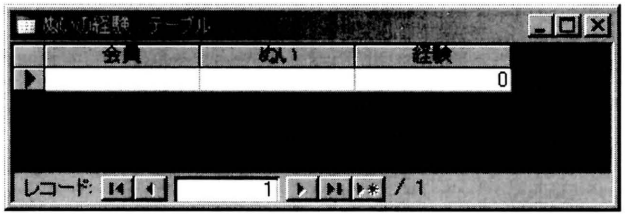

図 4

• 図 5のようにデータ入力が終了したら、「ファイル」＋「上書き保存」（あるいはメニューバーのディスケットアイコン）で保存しておきましょう。

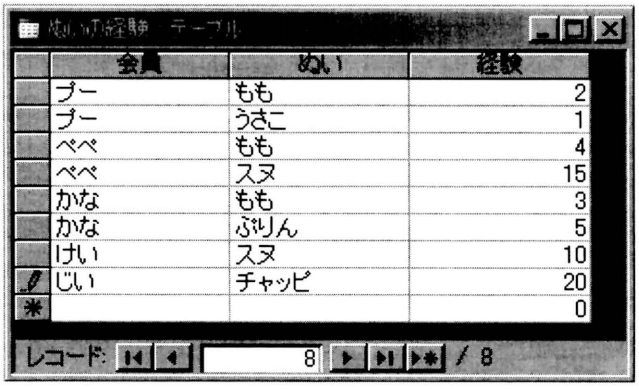

図 5

[16] この段階で、主キーを設定すると、「ID」というフィールドが自動的に作成されます。主キーの設定は必須ではありませんが、表と表との関係（リレーション）をつける場合（後述）などに必要となることがあります。通常は、どの表でも付けておく方が良いでしょう。

• 同じようにして、「会員住所」と「住所－地区対応」表を作成し、保存して終了しましょう（図 6）。

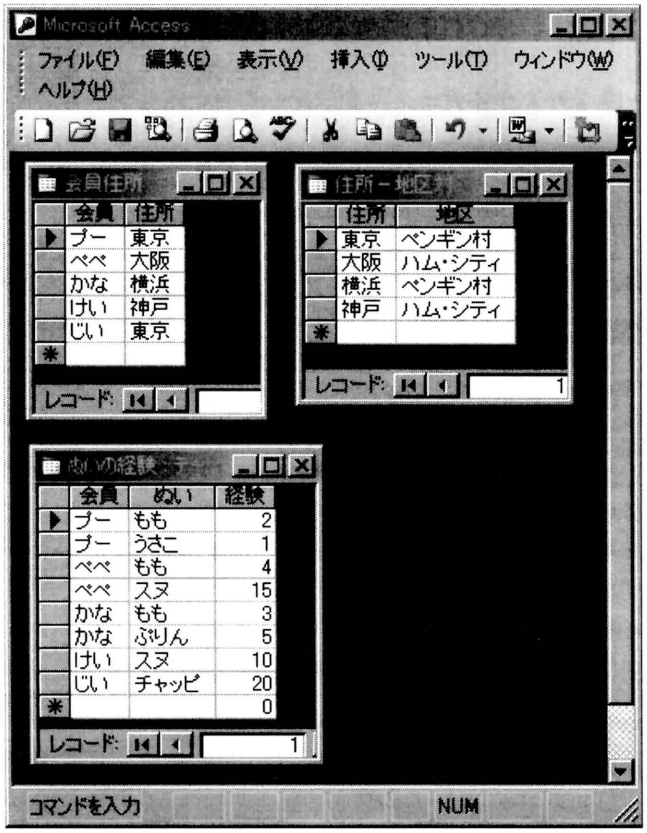

図 6

基本操作 1

数に加減乗除という演算があるように、データの型に応じて色々な演算（操作）を考えることができます。

例えば文字列だと、2つの文字列を結び付けて1つの文字列を作る連接（concatenation）や文字列の一部を取り出す操作がありますし、論理値（真・偽）については、その否定をとる NOT 演算、「かつ」を示す論理積 AND（conjunction, 連言）や「または」を示す論理和 OR（disjunction, 選言）といった演算があります。

構造を持った表のようなデータについても演算（操作）を考えることができます。選択(selection)・射影(projection)・結合（join）といった操作が基本的です。以下アクセスでの操作について Q&A 形式で述べてみましょう。

1. 選択（selection）

選択は、特定の条件を満たすレコードを表から取り出す操作です。

Q1.「ぬいの経験」表で、経験 10 年以上のレコードは？

A1.

1. 先ほど作成したデータベースを開きます。
2. 初期画面（データベースウィンドウ）で、「オブジェクト」の「クエリ [17]」をクリックします（図 7）。

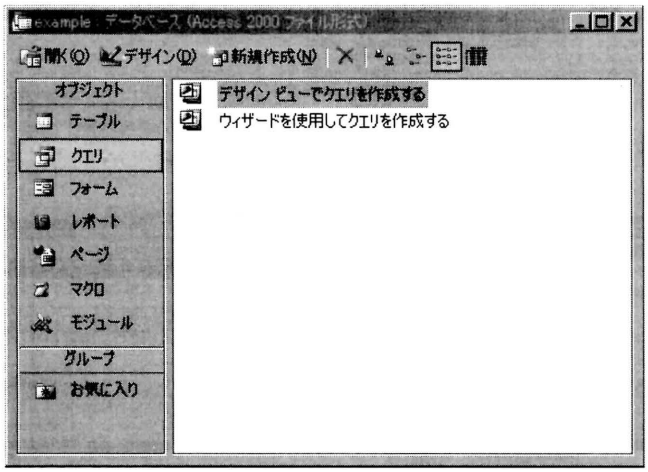

図 7

3. クエリの作成画面が現れますが、ここでは「デザインビューでクエリを作成する」を選択してダブルクリック（あるいは、「開く」をクリック）してください。
4.「テーブルの表示」画面で「ぬいの経験」を選択し、「追加」ボタンをクリックすると、「クエリ 1：選択クエリ」の中に「ぬいの経験」表が表示されます（図 8 の上の部分）。「テーブルの表示」を閉じます。「クエリ 1：選択クエリ」の画面の下方に検索の条件を設定します。Q1 の場合には、表の経験フィールドの値が 10 以上ですから、図 8 のように設定します。「経験」フィールドの抽出条件が「>=10」となっていることに注意してください。

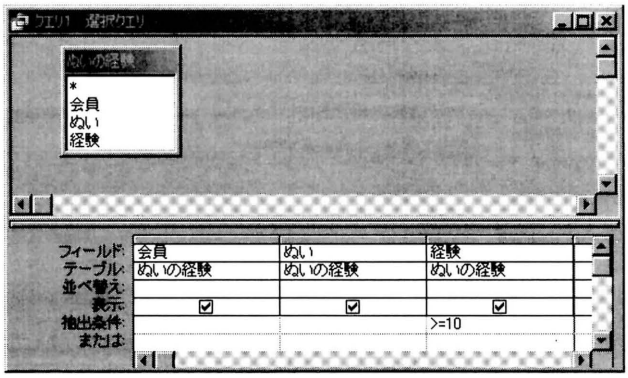

図 8

なお、この例のように、全てのフィールドを表示する場合には、全フィールドの指定「ぬいの経験.*」を用いて、図 9 のようにより簡単に設定することもできます。図では、「経験」フィールドが重複しないよう 2 列目の「表示」のチェックを外しています。

5.「ファイル」メニューの「名前を付けて保存」を実行し、名前をつけて表を保存します（ここでは、「ベテラン」という名前を用いました）。貼り付ける形式は「クエリ」です。

17 query: データベースに対する問い合わせや更新の操作。

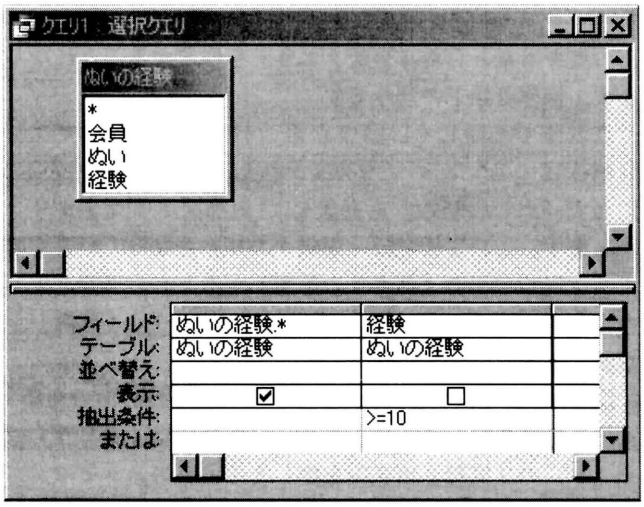

図 9

6.「表示」メニューの「データシートビュー」(あるいはツールバーの「ビュー」ボタン）をクリックし、経験 10 年以上のベテラン（？）が図 10 のように正しく表示されているか確認してください。

図 10

2. 射影（projection）

射影は、特定のフィールドを表から取り出す操作です。

Q2.「会員」と「ぬい」との対応は？

A2.「ぬいの経験」表を対象に、「会員」と「ぬい」のフィールドだけを取り出せば良いわけですから、図 11 のようなクエリを作成することになります(ここでは、「好きなぬい」という名前で保存しました)。データシートビューに変えると、図 12 のように表示されるはずです。

なお、これまでの手順から分かる通り、アクセスでは選択と射影を同時に実行する（特定の条件を満たすレコードの特定のフィールドを 1 回の操作で取り出す）こともできます。

3. 結合（join）

結合は、表と表とをつなげた表を作る操作です。

Q3.「会員住所」表と「会員－地区対応」表を結合すると？

A3. これまでと同様にクエリを作成していきます。「テーブルの表示」画面で「会員住所」表と「会員－地区対応」表を追加します。両表に住所というフィールドがありますが、どちらかの住所フィールドをクリックし、他方の住所フィールドにドラッグすると、両フィールドが線で結ばれ、関係付けられます（リレーションシップ)。選択クエリの画面で図のように設定します（ここでは「会員担当地区」という名前を付けています)。

デザインビューからデータシートビューに切り替えて、正しく結合されているか確認してください（図 14)。

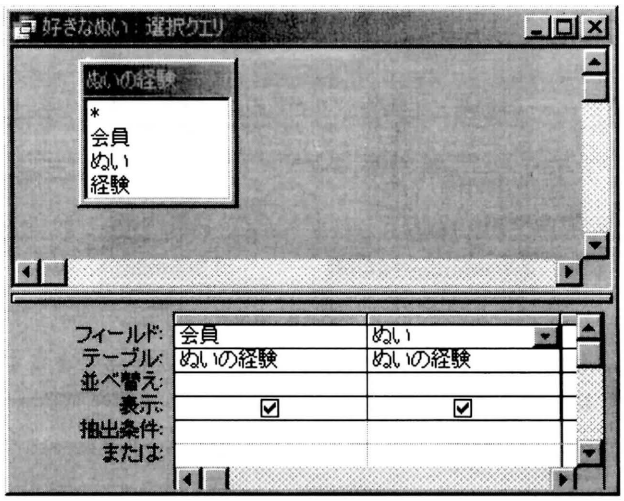

図 11

図 12

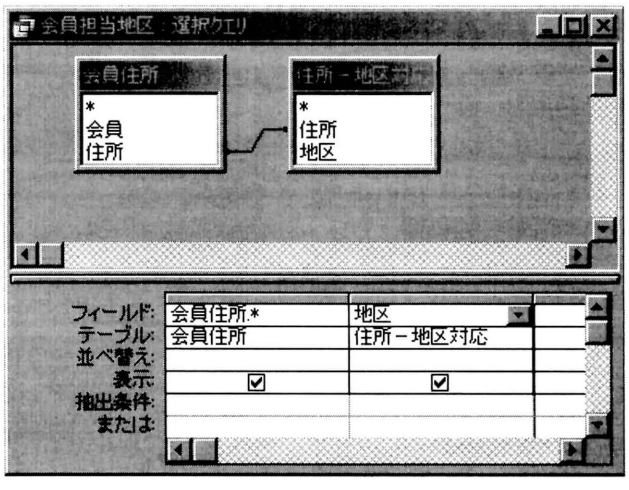

図 13

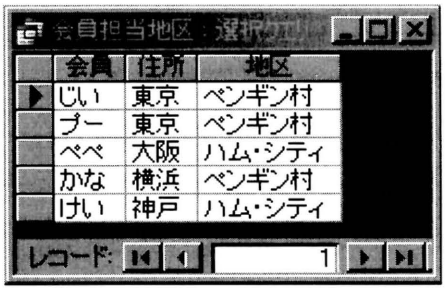

図 14

同様にして、3つの表を同時に結合することもできます（図 15・図 16）。

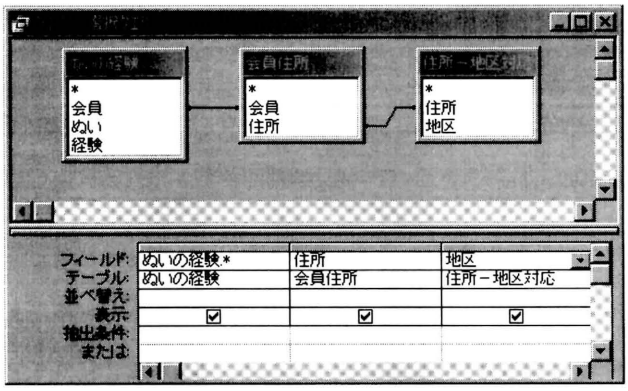

図 15

図 16

なお、クエリのデザインビューでは様々な抽出条件が指定できます。例えば、日付型のフィールドでも「>=2007/01/01」 と か「between 2007/01/01 and 2007/12/31」のように値の範囲を限定できますし、「Like "* 京都 *"」という条件で、フィールド値に京都府や東京都を含むレコードが取り出せます。また、「NOT」演算子で除外値を指定したり、AND や OR で組み合わせた条件を指定することもできます。アクセスのヘルプ（[7]）を参照してください。

基本操作 2

これまでは、主に検索のための問い合わせでしたが、データの更新にもクエリを使います。例えば「ぬいの経験」表（図 5）で「チャッピ」を「チャッピー」に変更したいとすると、この表を対象とした「更新クエリ」を作成することになります。操作は以下の通りです：

これまでと同じく、図 7 から「デザインビューでクエリを作成する」をダブルクリックし、「テーブルの表示」画面で「ぬいの経験」を選択し、「追加」ボタンをクリックします。このままでは「選択クエリ」となっていますので、ツールバーの「クエリの種類」ボタン右の下向き矢印を押して（あるいは「クエリ」コマンドから）「更新」を選択します。

図 17 のようにフィールド・テーブル・レコードの更新・抽出条件を指定して、ツールバー（あるいは「クエリ」コマンド）の「実行」ボタンを押します。警告メッセージが出ますが、「はい」を押します。「ぬいの経験」表（図 5）を表示して、最後のレコードが変更されていることを確認してください。

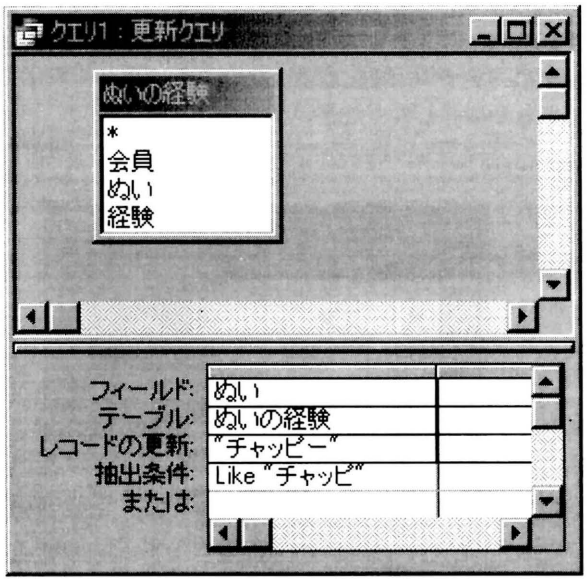

図 17

なお、変更するレコードが少なければ、直接テーブルを編集した方が早いかも知れません。しかし、例えば、大きな商品表で、全製品の単価を一斉に 5%値下げするような場合だと、更新クエリが必要となるでしょう（この場合、「レコードの更新」欄は「[単価]*0.95」のようになります）。

SQL

SQL（Structured Query Language）は、関係データベース言語の 1 つで、これにより、データベースに問い合わせを行ったり、更新作業を行ったりすることができます。例えば、学部サーバーの MySQL でデータベースを作り、PHP 等のスクリプトでそれを操作する時など SQL の文を書く必要があります[18]。

18 例えば、文献 [2], [4] を参照。

アクセスによる前述の「基本操作」では表立って現れていませんが、実は、クエリ画面の操作に対応してSQL文が生成されています。選択のクエリ（図10）では、以下のような文が生成されています：

```
SELECT ぬいの経験.*
FROM ぬいの経験
WHERE (((ぬいの経験.経験)>=10));
```

これは、ツールバーのビューボタン（あるいは「表示」コマンド）を使って、図9（デザインビュー）や図10の画面（データシートビュー）からSQLビューに変更すれば確認できます。

SELECT ～ FROM ～ WHEREというのはSQL文で検索指定する場合の基本的な書式です：

- SELECTの後には属性名の並びが書かれます。この例では、「ぬいの経験」表の全属性を示します。
- FROMの後には関係名（表の名）の並びが書れます。この例では、「ぬいの経験」表です。
- WHEREの後には条件が書かれます。この例では、「ぬいの経験」表の「経験」属性の値が10以上という条件です。

結合操作（図15・図16）だと、下記のようにもう少し複雑になります。これは、内部結合（INNER JOIN）という結合方法[19]ですが、他に交差結合・左結合・右結合といったものもあります。

```
SELECT ぬいの経験.*, 会員住所.住所, 住所
－地区対応.地区
FROM (ぬいの経験 INNER JOIN 会員住
所 ON ぬいの経験.会員 = 会員住所.会員)
INNER JOIN 住所－地区対応 ON 会員住所.住
所 = 住所－地区対応.住所;
```

おわりに

多少とも実用的なウェブ上のアプリケーションを作るならデータベースについての基礎的な理解は不可欠だと思われます。表計算ソフトほど馴染みがないと思いますが、個人ユースで、あるいは、演習授業・プロジェクト等でのアンケートデータの集約等でデータベースソフトも試してみてはいかがでしょうか。

[19] 自然結合（natural join）に同じ。

参考文献

[1] 石鎚英也、「情報と意思決定」、専修ネットワーク＆インフォメーション、No.1, 2002.
[2] 田中ナルミ、岡部忠光、「MySQLコマンドブック」、ソフトバンク、2005.
[3] 長尾真他、「岩波情報科学辞典」、岩波、1990.
[4] 星野努、「PHP+MySQLで作る最速Webシステム」、技術評論社、2004.
[5] ルーブル美術館オフィシャルサイト
http://www.louvre.or.jp/
[6] D. Maier, The Theory of Realtional Databases, Computer Science Press, 1983.
[7] Microsoft Office Accessヘルプ.