データベース講義ノート:7. 表の設計、正規形

国島丈生（t.kunishi@gmail.com）

ここまでいろいろな表の例を示してきたが、どのようにして表を設計すればいいのかについては述べていなかった。今回は、表の設計に関して一つの指針となる関数従属性（functional dependency）という考え方と、それに基づく表の正規形（normal form）について述べる。

第1正規形

ここまで例に出してきた表はすべて第1正規形（1st normal form）と呼ばれる。第1正規形とは、次の条件を満たす表のことである。

- どの属性の値も集合であることはない
- 複数の属性にまたがる値はない

更新不整合

前回まで用いた納品記録の表を1枚にまとめると、次のような表ができる。

商品番号	商品名	定価	客番号	名前	数
G1	テレビ	198000	C1	A店	3
G2	ビデオ	59800	C1	A店	10
G2	ビデオ	59800	C2	Bマート	5
G3	テレビ	98000	C2	Bマート	10
G3	テレビ	98000	C3	C社	2

この表の主キーは {商品番号, 客番号} である。どの属性も単独では主キーにならないことに注意しておくこと。このようにしても、表に含まれる情報は変わらない（実際、ここまで取り上げて来た多くの問い合わせでは、関連する表を全て最初に結合することによって、実質的にこれと同等の表を用いている）。しかし、データを更新しようとすると、次のような不整合が発生することがある（更新不整合）。

- 新しくパソコンを取り扱おうとして、(G4, パソコン, 298000, NULL, NULL, NULL) (NULLは空値を表す) という組を挿入しようとすると、キー制約違反となり、挿入できない。（主キーを構成する属性「客番号」が空値のため)
- C社からG3(テレビ)がすべて返品されたので、(G3, テレビ, 98000, C3, C社, 2)という組を削除した。すると、C社に関する情報が表から消えてしまう。

このような更新不整合は、直観的には、一つの表の中にいろいろな情報を詰め込みすぎているために起こっている。

関数従属性（functional dependency）

ある表の属性集合X, Yについて、常にXの値を決めるとYの値が決まる（表中の組t_1, t_2について、属性集合Xの値がすべて等しいならば、必ず属性集合Yの値もすべて等しい）という関係があるとき、YはXに関数従属するといい、$X \to Y$で表す。前の例では、次のような関数従属性がある。

- 商品番号 → 商品名, 定価
- 客番号 → 名前
- 商品番号, 客番号 → 商品番号, 商品名, 定価, 客番号, 名前, 数（∵商品番号, 客番号 が主キー）

関数従属性には、次のような代数的性質が成り立つ。

- $X \to X$（反射律）
- $X \to Y$ ならば$(X \cup Z) \to (Y \cup Z)$（添加律）
- $X \to Y$, $Y \to Z$ならば$X \to Z$（推移律）

関数従属性は、表がどのような状態であっても（表に含まれている組がどのようになっていても）必ず成り立つ、という性質であることに注意せよ。具体的な表の例を一つ考えて、そのときだけ「Xの値を決めるとYの値が決まる」となっているだけでは不充分である。従って、関数従属性は、スキーマの意味を考察して初めて決定することのできる、定性的な制約である。

完全関数従属性

関数従属性の特殊な場合として、完全関数従属性というものがある。$X \to Y$であり、かつXのどんな真部分集合X'についても$X' \to Y$が成り立たないとき、YはXに完全関数従属するといい、$X \twoheadrightarrow Y$で表す。直観的には「$X \to Y$のうち、Xからこれ以上属性を削れないようなもの」である。

- 商品番号 ↠ 商品名, 定価　は完全関数従属である。
- 客番号 ↠ 名前　は完全関数従属である。
- 商品番号, 客番号 ↠ 商品番号, 商品名, 定価, 客番号, 名前, 数　は完全関数従属である。
- 商品番号, 商品名→商品名, 定価　は完全関数従属でない。

関数従属性とキー

以上の説明からも分かる通り、関数従属性とキーには密接な関連がある。

- 属性集合Xが関係$R(A_1, A_2, \ldots)$の超キーであるとき、$X \to \{A_1, A_2, \ldots\}$
- 属性集合Xが関係$R(A_1, A_2, \ldots)$の候補キーであるとき、$X \twoheadrightarrow \{A_1, A_2, \ldots\}$

逆は成立しないことに注意。（以下に述べるように、キーとは関係しない関数従属性が成立することがある）

第2正規形

$X \twoheadrightarrow Y$が成立するとき、$X \cup Y$を一つの関係として分割することができる。このときの主キーはXとすればよい。前の例で言うと次のようになる。

- 商品番号 ↠ 商品名, 定価… (商品番号, 商品名, 定価)を1つの表にまとめる
- 客番号 ↠ 名前… (客番号, 名前) を1つの表にまとめる
- 元の表から、分割した完全関数従属性の右辺の属性を削除する… (商品番号, 客番号, 数)

こ のようにして得られた表はいずれも、主キー以外の属性が主キーに完全関数従属している。このような表のことを第2正規形（2nd normal form）という。第2正規形の表では、先に挙げた更新不整合は起こらない。しかし、次に示すように、第2正規形でも更新不整合が発生することがある。

第3正規形

更新不整合ふたたび

次の表を考える。

学籍番号	学生名	学科番号	学科名
001	田中	21	通信
002	山田	22	システム
003	森	22	システム

この表の主キーは "学籍番号" であり、完全関数従属性 学籍番号 ↠ 学生名, 学科番号, 学科名 が成立している。したがって、この表は第2正規形である。
しかし、この表にも以下のような更新不整合が発生する。

- 新しい学科「スポーツ」ができたとする（学科番号23）。まだ入学生はいない。このとき、組 (NULL, NULL, 23, スポーツ) を挿入しようとすると、キー制約に違反する。
- 田中君がシステムに転学科したとする。すると、(001, 田中, 21, 通信) → (001, 田中, 22, システム) と組が更新されるが、このとき「通信」学科に関する情報が失われてしまう。

推移的関数従属性

上に挙げた表の設計の問題点は、学科番号→学科名という関数従属性が隠れていたことに原因がある。
実は、学籍番号→学科名 は 学籍番号→学科番号 と 学科番号→学科名 という2つの関数従属性から推移律によって導き出されていた。

このように推移律によって得られる関数従属性を、推移的関数従属性（transitive functional dependency）という。

第3正規形

第2正規形の表のうち、主キー以外の属性がすべて主キーに推移的関数従属しないようなものを第3正規形（3rd normal form）という。つまり、第2正規形の表に推移的関数従属性が含まれている場合、それを用いて再び表の分解を行うと、第3正規形が得られる。

先に挙げた例では、推移的関数従属性 学籍番号→学科名 が含まれているので，その基になった関数従属性 学科番号→学科名 によって表を分解する。

- 学科番号→学科名 から、表 (学科番号, 学科名) を作る。主キーは学科番号である。
- 元の表から、推移的関数従属性の右辺 "学科名" を取り除く。結果として、表 (学籍番号, 学生名, 学科番号) が得られる。

この分解では、属性 "学科番号" について外部キー制約が成立していることに注意せよ。一般に，推移的関数従属性による分解により、外部キー制約が発生する。
第3正規形の表では、先に述べた更新不整合は発生しない。

第3正規形を得るためのアルゴリズム

表Rを分解し、第3正規形とするアルゴリズムをまとめると、次のようになる。

1. Rのスキーマを定性的に分析し、完全関数従属性を列挙しておく。このとき、左辺が同じ完全関数従属性は一つにまとめておく（例えば$X \twoheadrightarrow A$と$X \twoheadrightarrow B$があるとき、これらをまとめて$X \twoheadrightarrow A,\ B$としておく）。
2. 完全関数従属性に着目してRを分解する。
    a. 完全関数従属性$X_1 \twoheadrightarrow Y_1$について、属性集合を$X_1 \cup Y_1$とする新しい表$R'_1$を作る（主キーは$X_1$）。これを、残った属性集合に完全関数従属性がなくなるまで続ける（結果として表$R'_1,\ R'_2,\ \ldots,\ R'_n$が作られる）
    b. Rから$Y_1,\ Y_2,\ \ldots,\ Y_n$を取り除く。結果として、ステップaで作ったどの表にも含まれない属性と、$R'_1,\ R'_2,\ \ldots,\ R'_n$の主キー$X_1,\ X_2,\ \ldots,\ X_n$とからなる表ができる。（これを$R'_0$とする）
    c. $R'_0,\ R'_1,\ R'_2,\ \ldots,\ R'_n$はいずれも第2正規形になる。
3. $R'_0,\ R'_1,\ R'_2,\ \ldots,\ R'_n$のそれぞれについて推移的関数従属性が含まれるかを定性的に分析し、含まれれば、下記に従って分解する。推移的関数従属性が含まれなければ何もしない。
    a. R'_iに推移的関数従属性$X_{i,1} \to X_{i,2} \to X_{i,3} \to \ldots \to X_{i,m}$が含まれれば、属性集合$X_{i,2} \cup X_{i,3},\ X_{i,3} \cup X_{i,4},\ \ldots,\ X_{i,m-1} \cup X_{i,m}$をそれぞれ新しい表$R''_{i,2},\ R''_{i,3},\ \ldots,\ R''_{i,m-1}$とする（主キーはそれぞれ$X_{i,2},\ X_{i,3},\ \ldots,\ X_{i,m-1}$）。
    b. R'_iから$X_{i,2},\ X_{i,3},\ \ldots,\ X_{i,m-1},\ X_{i,m}$を取り除く（$X_{i,1}$は残す）。

c. 得られた関係は全て第3正規形となる。

第3正規形がなぜ重要か

第3正規形でない表Rを、第3正規形の表R_1, R_2, ..., R_nに分解したとしよう。このとき、次の式が成り立つ。

$$R = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$$

つまり、分解した全ての表を自然結合すると、元のRが得られる、という性質がある。この性質を満たす表の分解を情報無損失分解（Information Lossless Decomposition）という。
適当に表を分解しても、無損失な分解とはならない（つまり、元の表を復元することができない）。また、第3正規形以外にも、ボイス・コッド正規形、第4正規形など、いくつかの正規形が知られているが、いずれも無損失な分解になるとは限らない。この意味で、実際にスキーマを設計する場合、第3正規形になるようにするのが、現実的に重要な目安となる。