データベースS

第9回　リレーションスキーマの設計（3）

システム創成情報工学科　尾下 真樹

今日の内容

- 前回の復習
  - 正規形と正規化
    - 第2正規形～第5正規形
  - 正規形のまとめ
- スキーマの設計
  - 概念設計と論理設計
  - 実体関連モデルによるスキーマの設計
  - 正規化によるスキーマの設計

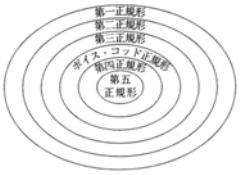
教科書

- 「リレーショナルデータベース入門」
  増永良文 著、サイエンス社（2,600円）
  - 4章
  - 1章（1.5節～1.6節）
- 「データベースシステム」
  北川 博之 著、昭晃堂 出版（3,200円）
  - 4章 55～82ページ

前回の復習

正規形と正規化

- 正規形の種類
  - 第1正規形
  - 第2正規形
  - 第3正規形
  - ボイス・コッド正規形
  - 第4正規形
  - 第5正規形



データベースシステム 図4.7

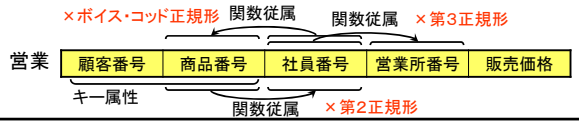
- 関数従属性や多値従属性に注目して、問題のあるレーションを段階的に分解していくことで、より厳しい正規形にすることができる
- 通常は第5正規形まで分解することが望ましい

関数従属性と多値従属性

- 関数従属性　X→Y
  - 属性（の組）X が決まれば、属性（の組）Y が一意に決まる
- 多値従属性　X→→Y
  - ある属性（の組）Xについて、いくつかの属性（の組）Yが存在すれば、必ず全ての XY（RS－XY）の組み合わせが存在する
    - RSはリレーションの全ての属性
  - 関数従属性は多値従属性の特殊なものと言える
    - Yが常に1種類のみ存在するもの

正規形の条件のまとめ(1)

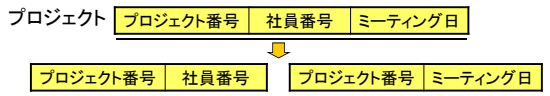
- 第2正規形
  - 候補キー以外の属性は、候補キーの部分属性に関数従属しない(キー属性が複数属性の組であるときのみ満たさない可能性がある)
- 第3正規形
  - 候補キー以外の属性は、候補キー以外に関数従属しない
- ボイス・コッド正規形
  - 候補キーの部分属性は、非候補キーに関数従属しない
    (キー属性が複数属性の組であるときのみ満たさない可能性がある)



正規形の条件のまとめ(2)

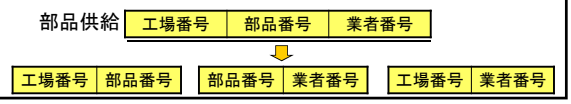
- 第4正規形
  - 多値従属が存在しない(分解後のリレーションが超キー)

多値従属性 プロジェクト番号 →→ 社員番号 | ミーティング日



- 第5正規形
  - 結合従属性が存在しない(分解後のリレーションが超キー)

結合従属性 ＊({工場番号, 部品番号}, {部品番号, 業者番号}, {工場番号, 業者番号})

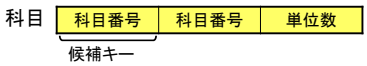


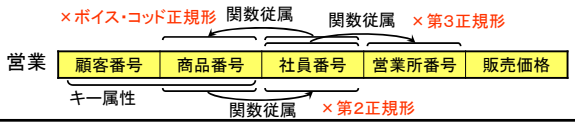
正規形の判定方法(1)

- 関数従属性を正しく書き出す(重要)
- 関数従属性にもとづき、正規形の定義に従って、正規形を判定(基本的にはこれだけ)
- 第2正規形から順番に判定していく
  - 途中の正規形を飛ばさない
- 第4正規形(多値従属性)、第5正規形(結合従属性)は、別に考える
  - ほとんどの場合、これらの正規形は満たすので、あまり心配する必要は無い

正規形の判定方法(2)

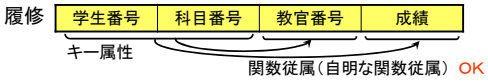
- 1つの属性のみで候補キーとなっていれば、自動的に、第2正規形、ボイスコッド・正規形は満たす
  - 候補キーの部分属性が存在しないため





正規形の判定方法(3)

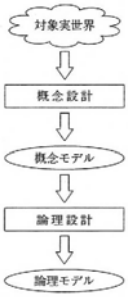
- 自明でない関数従属性(キー属性全体 → 非キー属性 以外の関数従属性)がなければ、第2正規形～ボイス・コッド正規形は満たす
  - 後は、多値従属性(第4正規形)、自明でない結合従属性(第5正規形)がないかを確認



リレーションスキーマの設計

リレーションスキーマの設計

- DBMSを利用するためには、自分の扱いたい自然界のデータを、DBMSの提供するデータモデルを使って記述する必要がある
  - 概念設計
    - 現実のデータの概念を整理
  - 論理設計
    - 具体的なスキーマの記述を決定



教科書 図2.4

リレーションスキーマの設計

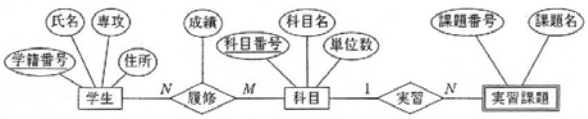
- 概念設計の方法
  - 実体関連モデル
- 論理設計の方法
  - 実体関連モデルからスキーマを決定
  - 仮のスキーマを正規化していくことで、スキーマを決定

実体関連モデル

- 概念設計を行うときのひとつの方法
  - 実体関連モデルは、概念設計の考え方のひとつなので、どのデータモデルにも適用できる
    - 第2回の講義で紹介した各種データモデルとは別なので混乱しないこと
  - 実体と関連
    - 実体
      - ひとつの実体をその属性によって表したもの
    - 関連
      - 複数の実体間の関連を表すもの
      - 関連にも属性を持たせることが可能

実体関連モデルの記述方法(1)

- 実体関連図(ER図)
  - 実体関連モデルを使ってモデル化した概念を図に表したもの
    - 実体は四角、関連はひし形、属性は丸、キー属性はアンダーラインで表されている



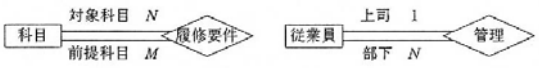
教科書 図2.5

実体関連モデルの記述方法(2)

- 参加制約
  - 関連の整合性を保つための制約
  - 1対1、1対N(1対多)、N対M(多対多)の区別
  - 同一の実体・関連間で複数の関与もありうる
  - 関連が全くなくてもいいか、最低ひとつはいるか
- キー制約
  - その属性値によって実体を一意に特定できるような属性に対する制約
  - キー制約を持つ属性が複数あっても良い
    - 主キーと候補キー

実体関連モデルの記述方法(3)

- 複数の関与を表した実体関連図の例



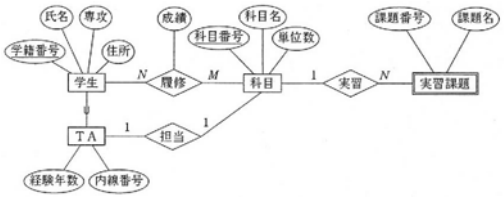
教科書 図2.6

- 参加制約を明確にした実体関連図の例

教科書 図2.7

実体関連モデルの記述方法(4)

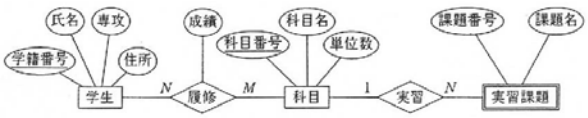
- 汎化階層
  - 抽象的な実体から、具体的な実体に派生
  - オブジェクト指向の考え方



教科書 図2.8

実体関連図の書き方

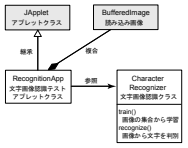
- 実体や関連を書き出していく
  - 実体は四角、関連はひし形で
  - 実体には適宜属性の情報を加える
  - 関連と実体の対応関係(単数or複数など)に注意



教科書 図2.5

実体関連図の応用

- プログラム開発でも、同様の考え方でクラス設計(概念設計・論理設計)を行う
  - オブジェクト指向プログラミングの設計でも、実体や関連に注目してクラスを定義
  - クラス図
    - 実体関連図と同様、各クラスと、クラスが持つ属性やメソッド、クラス間の関連を図に記述
    - Unified Modeling Language (UML)に従って記述



クラス図の例

リレーションスキーマの設計

- 概念設計の方法
  - 実体関連モデル
- 論理設計の方法
  - 実体関連モデルからスキーマを決定
  - 仮のスキーマを正規化していくことで、スキーマを決定

論理設計の方法

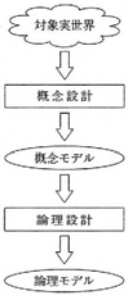
1. 実体関連モデルからスキーマを作る方法
2. スキーマを正規化していく方法
   - 正規形を満たすように、スキーマを分解
   - 一般に、正しく実体関連モデルが設計されていれば、正規形を満たすスキーマが作られるので、正規化の必要はない場合が多い
   - 実体関連モデルを用いず、正規化のみでスキーマの設計を行なうこともできる
- 組み合わせて(あるいは一方のみを)使用
- 両方の方法を使えることが必要

リレーションスキーマの設計

- 概念設計の方法
  - 実体関連モデル
- 論理設計の方法
  - 実体関連モデルからスキーマを決定
  - 仮のスキーマを正規化していくことで、スキーマを決定

実体関連モデルからのスキーマの設計

- スキーマ設計の手順
  - 概念設計
  - 論理設計
- 実体関連モデル
  - 概念設計の方法
  - データベースに格納すべき情報を、実体と関連に分けて整理する
- 実体関連モデルから、スキーマの論理設計を行う



教科書 図2.4

実体からリレーションを作成

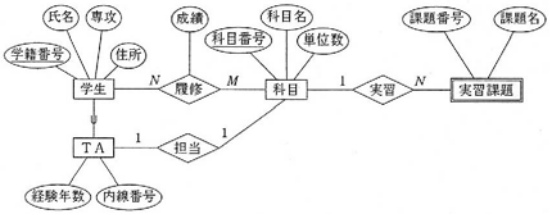
- 実体集合
  - 1つの実体 E からリレーション R を定定義
  - 実体の属性はリレーションの属性に
- 弱実体集合（ある実体 E' に付属する実体）
  - 1つの弱実体 E からリレーション R を定義
  - オーナ実体の主キーを参照として主キーに追加
- 汎化階層（ある実体 E' を派生させた実体）
  - 1つの汎化階層 E からリレーション R を定義
  - 上位実体の主キーを参照として主キーに追加

関連からリレーションを作成

- 2次の関連
  - 1対1の場合
    - いずれか片方のリレーションに、もう一方の主キーを参照として追加（属性を追加）
  - 1対Nの場合
    - Nの側（「1対多」の「多」の側）のリレーションに、もう一方の主キーを参照として追加（属性を追加）
  - N対Mの場合
    - Nの側、Mの側の主キーからなるリレーションを作成
- 3次の関連
  - 2次のN対Mの場合と同様（リレーションを作成）

具体例（1）

- 実体関連図の具体例
  - このモデルからリレーションスキーマを設計してみる



教科書 図2.8

具体例（2）

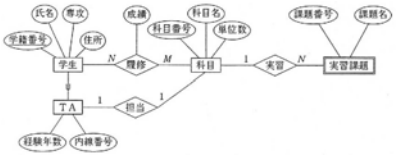
- 実体集合からリレーションを導出
  - 学生（学籍番号, 氏名, 専攻, 住所）
  - 科目（科目番号, 科目名, 単位数）
- 弱実体からリレーションを導出
  - 実習課題（科目番号, 課題番号, 課題名）
- 汎化階層からリレーションを導出
  - TA（学籍番号, 経験年数, 内線番号）

具体例（3）

- 1対1の関連集合からリレーションを修正
  - TA（学籍番号, 経験年数, 内線番号）→ TA（学籍番号, 経験年数, 内線番号, 科目番号）
- 1対Nの関連集合からリレーションを修正
  - 今回は不要
- N対Mの関連集合からリレーションを導出
  - 履修（科目番号, 学籍番号, 成績）

具体例(3)

- 最終的に得られたスキーマ(全ての正規形を満たす)
  - 学生(学籍番号, 氏名, 専攻, 住所)
  - 科目(科目番号, 科目名, 単位数)
  - 実習課題(科目番号, 課題番号, 課題名)
  - TA(学籍番号, 経験年数, 内線番号, 科目番号)
  - 履修(科目番号, 学籍番号, 成績)



リレーションスキーマの設計

- 概念設計の方法
  - 実体関連モデル
- 論理設計の方法
  - 実体関連モデルからスキーマを決定
  - 仮のスキーマを正規化していくことで、スキーマを決定

スキーマを正規化していく方法

- 必要な属性をリストアップしてスキーマを作成
- 関数従属性をリストアップ
- 正規形を満たすように、スキーマを分解していく

具体例(1)

- スキーマを作成
  - 履修(学生番号、科目番号、氏名、所属学部、所属学科、住所、科目名、単位数、成績)
- 関数従属性のリストアップ
  - 学生番号 → 氏名、所属学部、所属学科、住所
  - 科目番号 → 科目名、単位数
  - 所属学科 → 所属学部
  - ※ 自明な関数従属性(候補キー全体→他の属性)はリストアップしても、しなくても構わない
    - 例:学生番号、科目番号 → 成績

具体例(2)

- スキーマと関数従属性
  - 履修(学生番号、科目番号、氏名、所属学部、所属学科、住所、科目名、単位数、成績)
  - 学生番号 →氏名、所属学部、所属学科、住所
  - 科目番号 →科目名、単位数
  - 所属学科 → 所属学部
- 候補キーの一部の属性 → 候補キー以外の属性 への関数従属性があるので、第2正規形を満たさない

具体例(3)

- 正規形を満たすように分解
  - 履修(学生番号、科目番号、氏名、所属学部、所属学科、住所、科目名、単位数、成績)

⇩

  - 履修(学生番号、科目番号、成績)
  - 学生(学生番号、氏名、所属学部、所属学科、住所)
  - 科目(科目番号、科目名、単位数)

具体例(4)

- 分解後のスキーマと関数従属性
  - 履修(学生番号、科目番号、成績)
  - 学生(学生番号、氏名、所属学部、所属学科、住所)
  - 科目(科目番号、科目名、単位数)
  - 所属学科 → 所属学部
- 候補キー以外の属性 → 候補キー以外の属性 への関数従属性があるので、第3正規形を満たさない

具体例(5)

- 正規形を満たすように分解
  - 学生(学生番号、氏名、所属学部、所属学科、住所)

⇩

  - 学生(学生番号、氏名、所属学科、住所)
  - 学科(所属学科、所属学部)

具体例(6)

- 分解後のスキーマ
  - 履修(学生番号、科目番号、成績)
  - 学生(学生番号、氏名、所属学科、住所)
  - 学科(所属学科、所属学部)
  - 科目(科目番号、科目名、単位数)
- このスキーマは他の正規形も全て満たす
  - ボイスコッド正規形(他の関数従属性はない)
  - 第4正規形(多値従属性はない)
  - 第5正規形(結合従属性はない)

具体例(7)

- 最初のスキーマ
  - 履修(学生番号、科目番号、氏名、所属学部、所属学科、住所、科目名、単位数、成績)
- 正規化後のスキーマ
  - 履修(学生番号、科目番号、成績)
  - 学生(学生番号、氏名、所属学科、住所)
  - 学科(所属学科、所属学部)
  - 科目(科目番号、科目名、単位数)

リレーションスキーマの設計のまとめ

- 概念設計の方法
  - 実体関連モデル
- 論理設計の方法
  - 実体関連モデルからスキーマを決定
  - 仮のスキーマを正規化していくことで、スキーマを決定

論理設計の方法のまとめ

1. 実体関連モデルからスキーマを作る方法
2. スキーマを正規化していく方法

- 通常は、1の方法でスキーマを設計し、必要に応じて、2の正規化を行うのが一般的
  - きちんとした実体関連モデルからスキーマを作成していれば、正規化の必要はないことが多い
- それほど実用では使わないが、2の方法をきちんと身につけておくことは重要

まとめ

- 前回の復習
  - 正規形と正規化
    - 第2正規形～第5正規形
  - 正規形のまとめ
- スキーマの設計
  - 概念設計と論理設計
  - 実体関連モデルによるスキーマの設計
  - 正規化によるスキーマの設計

全体のまとめ

- スキーマの正規化
  - 正規化の必要性
  - 正規形と正規化
  - 多値従属性、関数従属性、結合従属性
  - 第1正規形～第5正規形
- スキーマの設計
  - 概念設計と論理設計
  - 実体関連モデルによるスキーマの設計
  - 正規化によるスキーマの設計

次回予告

- PHPによるWebインターフェース
  - Web、PHP のしくみ
  - HTML＋PHP 入門
  - PHPによるインターフェース作成（1）