データベースS

第2回　データモデル

システム創成情報工学科　尾下 真樹

2018年度 Q2

今日の内容

- 前回の復習
- データモデル
  - 各種データモデル
- リレーショナルデータモデル
  - リレーションスキーマ
  - リレーションの整合性制約

教科書

- 「リレーショナルデータベース入門［第3版］」
  増永良文 著、サイエンス社（3,200円）
  - 1章(1.4)、2章(2.1～2.9)
- 「データベースシステム」
  北川 博之 著、昭晃堂 出版（3,200円）
  - 1章・2章

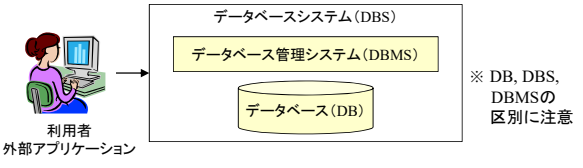
前回の復習

データベースの概要

- データベースって何だろう？
  - 大量のデータを効率良く管理するためのシステム
  - データのモデリング、検索インターフェース、大量データ処理などの機能を提供
  - 他のプログラムからデータを利用できる
- データベースの応用
  - 企業の顧客・売上データなど
  - 最近では、地理データベース、マルチメディア、DNAデータ、科学技術データ なども扱われる
  - SEなど情報系に就職する人の多くは、何らかの形でデータベースシステムに関わる可能性が高い

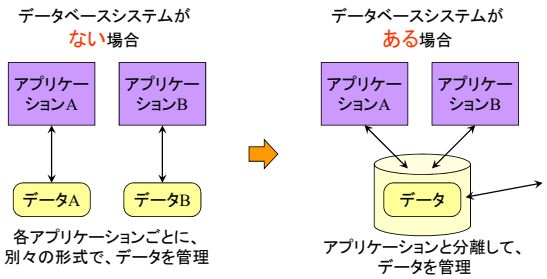
データベースシステム

- データベースシステム（Database System）
  - データベース（Database）
    - データを組織的・永続的に記録するための装置
  - データベース管理システム（Database Management System）
    - データベースを管理・利用するためのソフトウェア



データベースシステムの利点

- データをアプリケーションと分離して管理



データベースシステムの特徴

- 大量のデータを効率的に管理・処理するためのしくみを提供
  - データを体系的・組織的に定義・管理するためのデータ記述方法（データモデル）、管理方法
  - 大量のデータを効率良く処理するための方法
- データを常に正しく保つためのしくみを提供
  - 実際の運用では、とにかく正しいデータを保つことが必須とされる
  - 整合性の維持、機密保護、障害回復

データベースシステムの機能

- 基盤となるデータ記述・操作系
- 効率の良いデータアクセス機構
- 整合性の維持
- 機密保護
- 同時実行制御
- 障害回復

データモデル

- データモデル
  - データベースに格納するデータ構造（スキーマ）を記述するための枠組み
  - どのようにファイルやメモリにデータが格納されるかは気にせず、概念的なデータ構造を定義
  - 各データベースシステムはある特定のデータモデルをサポート
    - リレーショナルモデルが代表的
    - これまでにさまざまなデータモデルが開発されてきた
  - データモデルに基づいた操作言語が存在

リレーショナルデータモデル

- リレーショナルデータモデル
  - 表形式のデータ構造（リレーション）によりデータを格納するデータモデル
  - リレーション同士の演算によって、さまざまな処理を実現できる
  - 他のデータモデルと比べて、単純、データ独立性が高い、といった利点がある
    - ただし、可変長のデータや、データ構造が複雑なデータには不向き

リレーショナルデータモデルの例

学生

学生番号	氏名
0123001	織田 信長
0123002	豊臣 秀吉
0123003	徳川 家康
…	…

科目

科目番号	科目名
01	データベース
03	コンピュータグラフィックス
…	…

履修

科目番号	学生番号	成績
01	0123001	60
03	0123002	80
01	0123003	70
…	…	

スキーマとインスタンス

- スキーマ（scheme）
  - データベースに格納されるデータのデータ構造、データの型、データ同士の関連、各種制約を記述したもの
  - メタデータ（データについてのデータ）
- インスタンス（instance）
  - スキーマにもとづいて格納されたデータ
  - Javaのクラスとオブジェクトの関係と同じ（クラス→スキーマ、オブジェクト→インスタンス）

スキーマとインスタンス

- リレーショナルデータベースの例
  - リレーション
    - 複数の属性の組み合わせによりデータを表現
  - スキーマ
    - リレーションの項目の型、属性制約、キー制約など
  - インスタンス
    - それぞれのデータ、表の各行に相当

学生

学生番号	氏名
0123001	織田 信長
0123002	豊臣 秀吉
0123003	徳川 家康

履修

科目番号	学生番号	成績
01	0123001	60
03	0123002	80
01	0123003	70

データモデル

データモデル

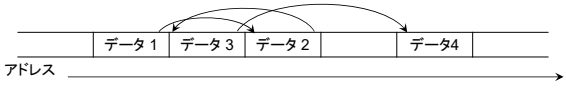
- データベースに格納するデータの構造を記述するための枠組み
  - データ構造、データの整合性を保つための制約を記述するための規約
- データモデルの役割
  - データベースシステムが提供するインターフェースとしての役割
  - 現実世界の情報をデータベースとしてモデル化するためのツールとしての役割

データモデルの歴史

- 昔は、COBOLなどの手続き型プログラミング言語を使ってファイルを直接操作していた
- より効率的に処理を行うためのシステムが研究される
  - ネットワークデータモデル、階層データモデル
- データモデルと物理構造の分離の考え方が進む
  - リレーショナルデータモデル
  - オブジェクト指向データモデル

データベースの物理構造

- データベース内のデータ（インスタンス）は、ファイル（あるいはメモリ）上に格納される
  - それぞれのデータ（インスタンス）をレコードとして、ファイルやメモリ上に並べて格納
  - データの繋がりや集まりは、前のレコード内に、次のレコードのアドレスを格納することで表す
    - ファイルやメモリ上では離れた場所に格納されているレコード間にも、繋がりの情報を記録できる



データモデルの種類

- リレーショナルデータモデル
- ネットワークデータモデル
- 階層データモデル
- オブジェクト指向モデル

リレーショナルデータモデル

- リレーショナルデータモデル
  - 1970年にCoddによって提案される
  - 従来のデータモデルと比べて、単純、データ独立性が高い
    - 複数の属性の組み合わせによってそれぞれのリレーションを定義
    - リレーション同士の演算によりさまざまな処理を実現
  - 可変長のデータやデータ構造が複雑なデータには不向き
    - 可変長のデータの例：住所録のデータベースで、ひとりの人物が複数の電話番号、住所を持つ、など

リレーショナルデータモデルの例

科目

履修

学生

科目

科目番号	科目名
01	データベース
03	コンピュータグラフィックス
…	…

履修

科目番号	学生番号	成績
01	0123001	60
03	0123002	80
01	0123003	70
…	…	

学生

学生番号	氏名
0123001	織田 信長
0123002	豊臣 秀吉
0123003	徳川 家康
…	…

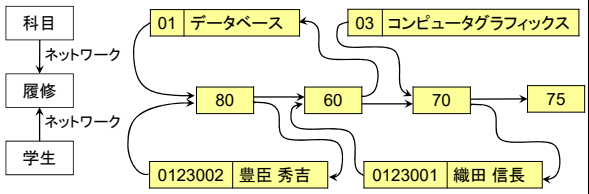
データモデルの種類

- リレーショナルデータモデル
- ネットワークデータモデル
- 階層データモデル
- オブジェクト指向モデル

ネットワークデータモデル

- ネットワークデータモデル
  - CODASYL（The Conference on Data Systems Language）仕様[1973] が起源
  - 各インスタンスはひとつのレコードで表される
  - 関連するレコードの間を一連のリンク（ネットワーク）で繋ぐことによってデータを表現する
    - レコード間に親子関係を定義し、親→全ての子を順番にリンク→親に戻る、というリンクにより関係を表現
  - リンクを順にたどっていくことで各種処理を実行

ネットワークデータモデルの例



親となる「科目」のレコードから、子となる複数の「履修」のレコードに、順番にリンクを繋ぐ

親となる「学生」のレコードから、子となる複数の「履修」のレコードに、順番にリンクを繋ぐ

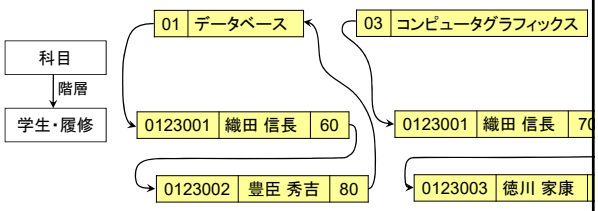
データモデルの種類

- リレーショナルデータモデル
- ネットワークデータモデル
- 階層データモデル
- オブジェクト指向モデル

階層データモデル

- 階層（ハイアラキカル）データモデル
  - IBMのIMS[1977] や MRIのSYSTEM2000[1985]
  - レコードのツリー構造によってデータを表現
  - ひとつの親レコードから複数の子レコードに接続
    - ツリー構造になり分かりやすい
    - １対多の関係は表現しやすいが、多対多の関係は表現しにくい
  - 論理関連
    - 複数のツリーでレコード間にリンクを張ることも可能
  - 木をたどっていくことで各種処理を実行

階層データモデルの例



親となる「科目」のレコードから、子となる複数の「履修」のレコードにリンクを繋ぐ
（実際には、子のレコードを順番にリンクでつなぐことで、複数の子へのリンクを実現）

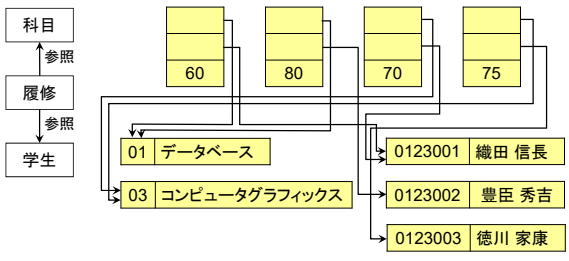
データモデルの種類

- リレーショナルデータモデル
- ネットワークデータモデル
- 階層データモデル
- オブジェクト指向モデル

オブジェクト指向データモデル

- オブジェクト指向データモデル
  - オブジェクト指向によりデータをモデル化
  - スキーマとしてクラスを定義
    - 可変長の属性やオブジェクト間の参照も定義できる
    - メソッドが定義できる
    - 継承・多態などのオブジェクト指向の概念が使える
  - Java や C++ などのオブジェクトに永続性を持たせるようなイメージ
    - プログラム終了後も、生成されたオブジェクトが残る
  - プログラミング言語から直接操作可能
  - 現在はまだあまり利用されていない

オブジェクト指向データモデルの例



注意：実体関連モデル

- 教科書で説明されている「実体関連モデル」は、データモデルの一種ではなく、全く別の概念モデルのひとつ
  - 詳細は、後日の講義で説明

リレーショナルデータモデル

リレーショナルデータモデル（復習）

- リレーショナルデータモデル
  - 1970年にCoddによって提案される
  - 従来のデータモデルと比べて、単純、データ独立性が高い
    - 複数の属性の組み合わせによってそれぞれのリレーションを定義
    - リレーション同士の演算によりさまざまな処理を実現
  - 可変長のデータやデータ構造が複雑なデータには不向き

リレーショナルデータモデルの例（復習）

学生

学生番号	氏名
0123001	織田 信長
0123002	豊臣 秀吉
0123003	徳川 家康
…	…

科目

科目番号	科目名
01	データベース
03	コンピュータグラフィックス
…	…

履修

科目番号	学生番号	成績
01	0123001	60
03	0123002	80
01	0123003	70
…	…	

スキーマとインスタンス（復習）

- リレーショナルモデルのスキーマ・インスタンス
  - スキーマ
    - リレーション名、リレーションの各属性名、ドメイン
    - キー制約、参照整合性制約、属性制約などの制約
  - インスタンス
    - 属性値の組によって表される各データの集合

従業員

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39

スキーマ
※ 実際には各種制約を含む

インスタンス

スキーマとインスタンス

- スキーマ
  - データベースに格納されるデータのデータ構造、データの型、データ同士の関連、各種制約を記述したもの
  - 最初に一度定義したら基本的に変更されない
- インスタンス
  - スキーマにもとづいて格納された実際のデータ
  - データベースの利用に応じて追加・削除・変更される

リレーションスキーマ

- リレーションスキーマの表現

$$R(A_1,\cdots,A_i,\cdots,A_n) \quad A_i \in D_i$$

  - R リレーション名
  - A_i 属性名
  - D_i ドメイン（属性値のとりうる範囲）
  - 例：科目（科目番号、科目名、単位数）
    $D_{1(科目番号)} = \{001, 002, \ldots\}$,
    $D_{2(科目名)}$ = 文字列, $D_{3(単位数)} = \{1, 1.5, 2, \ldots\}$

リレーション

- リレーション（relation）
  - リレーションスキーマに基づいて格納される実際のデータ（インスタンス）の集まり
    - 数学的には全てのDの組み合わせの部分集合となる
      - 全部で $D_1 \times \cdots \times D_n$ 通りのデータがありうる
      - インスタンスはその部分集合になる
  - インスタンスのことをタプル（tuple）ともいう
    - タプルは値の組を表す一般的な用語
  - リレーションの次数
    - 属性の数＝リレーションの次数＝リレーションの項数
      - 単項リレーション、二項リレーション、・・・、n項リレーション

リレーションの表現

- リレーションは表形式で表現できる
  - あくまで集合を表として表しているだけなので、厳密には、リレーションと表は異なる
    - タプルの順番は意味を持たない
    - 全く同じ値を持つタプルが複数あることは許されない
    - 属性の順番も本来は意味はない

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39
004	1	柴田 勝家	60

第一正規形

- 第一正規形（first normal form）制約
  - ドメイン（属性の取りうる値）は分解不可能な単純な値でなければならないという制約
  - 各属性は、単一の値でなければいけない
    - 第二、第三正規形なども今後の講義で出てくる

第一正規形を満たさないスキーマの例（ひとつの属性に複数の値がある）

科目番号	科目名	学生番号	履修者	成績
01	データベース	0123001	織田 信長	60
		0123002	豊臣 秀吉	70
03	コンピュータグラフィックス	0123001	織田 信長	60
		0123003	徳川 家康	75

第一正規形（続き）

- 第一正規形を満たすように2つのリレーションスキーマに分割するなどする必要がある

学生

学生番号	氏名
0123001	織田 信長
0123002	豊臣 秀吉
0123003	徳川 家康

科目

科目番号	科目名
01	データベース
03	コンピュータグラフィックス

履修

科目番号	学生番号	成績
01	0123001	60
03	0123002	80

空値

- 空値
  - 属性値が存在しないことを示す特殊な属性値
  - 「NULL」と表す
- ゼロや空白と空値の区別に注意
  - ゼロや空白の場合は、その属性値がゼロあるいは空白と分かっているもの
  - NULLの場合は、値が存在しないか、値がまだ分からないということ

履修

科目番号	学生番号	成績
001	00001	90
001	00002	0
001	00003	NULL

※ NULLは、値がないことを表す（例えば成績が未評価など）

リレーションの整合性制約

- リレーションの整合性を保つために満たされていなければならない制約
  - ドメイン制約
  - キー制約
  - 参照整合性制約
  - 一貫性制約
  - 従属性制約
- データベースを設計する時は、リレーションスキーマに加えて、これらの制約も指定する

リレーションの整合性制約

- ドメイン制約
- キー制約
- 参照整合性制約
- 一貫性制約
- 従属性制約

ドメイン制約

- 属性値はドメインの要素でなければいけない

$$A_i \in D_i$$

  - 基本的には、整数、実数、文字列などの型によって規定される
  - 数値型の場合は、値の範囲も規定されることもある（例えば、成績は0～100の値、など）
  - 文字列型のドメインなどは、特定の値の集合に規定されることもある（例えば、都道府県など）

キー制約

- キー
  - 複数のインスタンスが同一の属性値をもつことがないような属性、あるいは、属性の集合
- キー制約
  - キー制約が指定された属性（属性の集合）は、複数のインスタンスが同一の属性値を持つことが許されない
  - キー制約を持つ属性（属性の集合）の属性値が指定されれば、リレーション中のひとつのインスタンスを特定できる

キーの種類

- 超キー
  - 複数のインスタンスが同一の属性値をもつことがないような属性、あるいは、属性の集合
  - キーの条件を満たす、全ての属性、属性の集合
- 候補キー（単にキーと呼ぶこともある）
  - 最小の超キー（候補キーの部分集合が超キーとならない）
- 主キー
  - 候補キーのうち、属性値が空値にならず管理上適当なもの
  - 例：学生（学生番号、氏名、住所、学科）

超キー
主キー
候補キー

※ どの属性が超キーになるかはデータに依存
※ 超キーは他にも存在

キー制約の役割

- キー制約は、スキーマの正規化を行なう上で、重要になる
  - 詳しくは後日の講義で説明

キー制約の定義方法

- 候補キーは、リレーションにどのようなデータを格納するかによって決まる
  - どの属性 or 属性の組の値が、リレーション中で一意になるか？
- 超キーは、候補キーに任意の属性を加えたもの
  - 超キーは、候補キーを定義するための概念
  - 通常は、超キーを直接意識する必要はない
- 主キーは、候補キーから、属性値が空値にならない任意のものを選択
  - 通常は、ひとつの整数値の属性からなる候補キーを主キーとする

キー制約の例

従業員（従業員番号, 氏名, 部門番号）

- 従業員番号は、全ての従業員に異なる番号が振られているとする
- 同じ部門には、同じ氏名の従業員は存在しないものとする

主キー {従業員番号} ※ 2つの候補キーのどちらでも可
候補キー {従業員番号}, {氏名, 部門番号}
超キー {従業員番号}, {氏名, 部門番号},
{従業員番号, 氏名},
{従業員番号, 部門番号},
{従業員番号, 氏名, 部門番号}

参照整合性制約

- 参照整合性制約（外部キー制約）
  - 他のインスタンスを参照する属性に対する制約
    - リレーションのある属性値が、他のリレーションのあるインスタンスの主キーの値でなければならない
    - 例: 科目（科目番号, …）、履修（…, 科目番号, …）
    - 同一リレーションのインスタンスを参照することもある
- 外部キー
  - インスタンスを参照する属性を外部キーと呼ぶ
    - 上の例では、履修リレーションの科目番号が外部キー
  - 外部キー（FK）と主キー（PK）のドメインは同じである必要がある

参照整合性制約の例

従業員（従業員番号, 部門番号, 氏名, 年齢）

部門（部門番号, 部門名）

※ 下線のある属性が主キー（候補キー）とする

- このとき、従業員の部門番号には、参照整合制制約が存在することになる
  - 従業員の部門番号は、ある部門を指すためものものなので、必ず、どれかの部門の部門番号の値でなければならない（存在しない部門番号は不可）
  - このとき、従業員の部門番号 ＝ 外部キー となる

参照整合性制約の例

従業員

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39
004	1	柴田 勝家	60

主キー　外部キー

部門

部門番号	部門名
1	開発
2	営業
3	総務

主キー

この場合、従業員の部門番号は、必ず部門の部門番号（部門の主キー）に存在する必要がある
→ 参照整合性制約

他の参照整合性制約の例

- 主キーと外部キーの属性名が異なる場合
  - 従業員（従業員番号, 氏名, 所属部門番号）
  - 部門（部門番号, 部門名）
- 同一リレーションのデータを参照する場合
  - 従業員（従業員番号, 氏名, 上司の従業員番号）
- 主キーの一部に参照整合性制約がある場合
  - 業務（従業員番号, 業務名, 業務期間）
  - 従業員（従業員番号, 氏名, ・・・）

※ 参照の有無により、判断する必要がある

一貫性制約

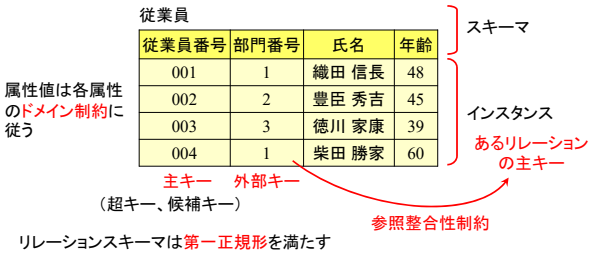
- 一貫性制約
  - リレーションが満たさなければならない制約
  - 個々の属性のドメイン制約だけでなく、複数の属性値の組が満たさなければいけない条件を含む
    - 1つのタップル内での条件
    - 複数のタップルでの条件
    - 1つのリレーション内での条件
    - 複数のリレーションでの条件
  - スキーマ定義時に条件を定義しておくことで、制約に違反するような変更を防ぐことができる

従属性制約

- 従属性
  - 複数の属性値の間で満たさなければいけない制約
- 従属性の種類
  - 関数従属性
    - R(…, X, …, Y, …)で、複数のタプルのX同士が等しければ、Y同士も等しいこと
    - 例:科目(科目名, 担当教官, 単位数) X:科目名、Y:単位数
  - 多値従属性、結合従属性など
- 従属性はデータベース設計において重要
  - 従属性がきちんと記述されていれば、データ更新時などに自動的にそれを満たすよう処理できる
  - 詳しくは後日の講義で説明

まとめ

- リレーショナルスキーマとリレーション
  - 表形式(リレーション)によるデータ表現



従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39
004	1	柴田 勝家	60

まとめ

- データベースシステム(復習)
- データモデル
  - 各種データモデル
- リレーショナルデータモデル
  - リレーションスキーマ
  - リレーションの整合性制約

次回予告

- リレーションの操作体系
  - リレーショナル代数とリレーショナル論理式
  - SQLとリレーション操作の関係
- リレーショナル代数
  - リレーショナル代数演算子
  - リレーショナル代数式