

# 2-4 データベース

東京大学 数理・情報教育研究センター

2021年5月10日

# 概要

- データベースの初歩を学びます
- データベースから必要なデータを抽出し、データ分析のためのデータセットを作成できるようになることを目指します

# 本教材の目次

1. データベースとRDB	4
2. データベース言語とSQL	6
3. 主キー	12
4. ERモデル	15
5. その他のデータベース	18
6. まとめ	19
7. 付録	20

# データベース (database, DB) とは

- 組織化されたデータの集まりを意味します
  - この「組織化」とは特定の操作のために準備されていることを意味します
  - 操作の典型例：検索，追加，更新，削除
- データベースの例
  - ネットショッピング
  - 銀行口座
  - ルート検索
  - 学務管理
  - デジタルライブラリ
- オンラインシステムは基本的に何らかのデータベースを伴います

# リレーショナルデータベース (RDB)

- 関係 (リレーション) データに基づくデータベースです
  - 1970年にEdgar F. Coddによって理論的基盤が提唱されました
- 関係データ = 表
  - 行 (レコード, タプル) と列 (属性) の構造を持ちます
  - 行が一単位のデータを表し, 列が1種類のデータを表します
- 表の例

氏名	所属	Email
東大太郎	理学部	taro@s.u-tokyo.ac.jp
東大花子	薬学部	hanako@f.u-tokyo.ac.jp
東大史郎	法学部	shiro@j.u-tokyo.ac.jp

- 横方向のひとまとまり (赤) がレコードです
- 縦方向のひとまとまり (青) が属性です
  - 氏名・所属・Emailはデータの種類を表す属性名です

# データベース言語とは

- データベースの機能を提供するプログラミング言語です
  - つまり、データベースはデータベース言語によって組織化されます
- データベース言語の3つの側面
  - データ操作言語：操作（検索、追加、更新、削除等）を記述する言語
  - データ定義言語：データの構成を記述する言語
  - データ制御言語：データのアクセス制御を記述する言語
- RDB向けのデータベース言語の標準としてSQLがあります
  - Structured Query Languageに由来します
  - 「エスキューエル」もしくは「シークェル」と発音します
  - SQLは前述の3つの側面を全て併せ持つ言語です
- 以降、RDBとSQLを通してデータベースの基本を説明します

# SQLによるデータ操作：SELECT

- 表から一部のデータを取り出す操作です
  - 条件を満たすレコードから指定された属性だけを返す
- 例：「名簿」表

氏名	所属	Email
東大太郎	理学部	taro@s.u-tokyo.ac.jp
東大花子	薬学部	hanako@f.u-tokyo.ac.jp
東大史郎	法学部	shiro@j.u-tokyo.ac.jp

```
SELECT 氏名,Email  
FROM 名簿  
WHERE 所属 == '理学部';
```



氏名	Email
東大太郎	taro@s.u-tokyo.ac.jp

# SQLによるデータ操作：INSERT

- 表にデータを追加する操作です
- 例：「所在地」表

キャンパス	市区町村
本郷	文京

```
INSERT INTO 所在地  
VALUES ('駒場', '目黒');
```



キャンパス	市区町村
本郷	文京
駒場	目黒



# SQLによるデータ操作：UPDATE

- 表の一部データを更新する操作です
- 例：「預金」表

預金者	預金額
東大太郎	100,000
東大史郎	1,000,000

```
UPDATE 預金  
SET 預金額 = 0  
WHERE 預金者 == '東大史郎';
```



預金者	預金額
東大太郎	100,000
東大史郎	0

# SQLによるデータ操作：DELETE

- 条件を満たすレコードを削除する操作です
- 例：「預金」表

預金者	預金額
東大太郎	100,000
東大史郎	1,000,000

```
DELETE 預金  
WHERE 預金額 < 200000;
```

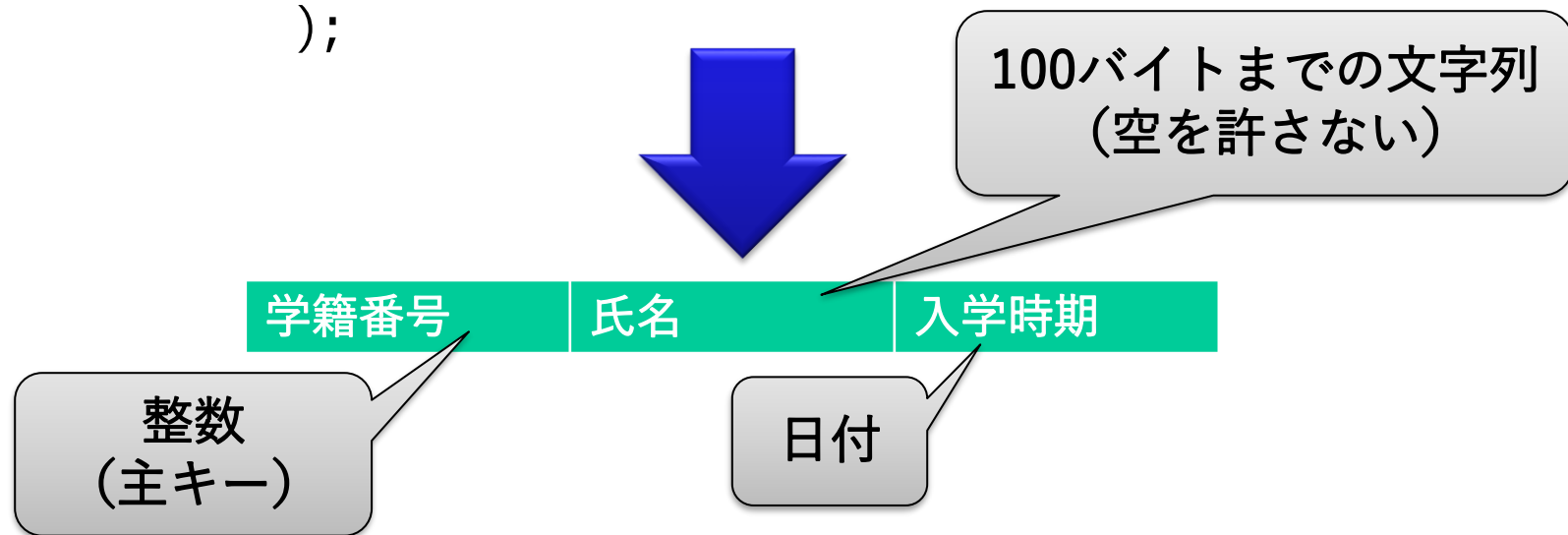


預金者	預金額
東大史郎	1,000,000

# SQLによるデータ定義：CREATE

- 表を作成する操作です
- 例：空の「入学」表の作成

```
CREATE TABLE 入学 (  
    学籍番号 INTEGER PRIMARY KEY,  
    氏名 CHAR(100) not null,  
    入学時期 DATE  
);
```



# 主キー (primary key)

- 主キーとは表中のレコードを一意に特定できる属性を意味します
  - 例えば、前頁の「入学」表における「学籍番号」が主キーです
  - 主キー以外で一意に特定可能な属性（の集合）を代理キーと呼びます
- 主キーは表に必須ではないですが、ないと不便なことが多いです
  - 「入学」表に氏名と入学時期が同じ学生を入れられなくなります
- しばしば機械的に割り振られたID番号が用いられます
  - 例：マイナンバー
- ある表の主キーの値はその表との関連を表現できます
  - 主キーの値さえあれば他の属性の値も参照可能です
- 別の表の主キーを参照する属性を外部キー (foreign key) と呼びます
  - 外部キーの値には参照先の主キーの値しか許容されません

# 複数の表からなるデータベース

- 複雑な関係データも複数の表に分解すると見通しが良くなります
- 例：「開講」表

開講コード	科目	開講先
0301	1001	03
0302	1002	03
0501	1001	05

複数の科目を複数の部局で開講する多対多の関係

外部キー

科目番号	科目名
1001	統計学
1002	データベース

外部キー

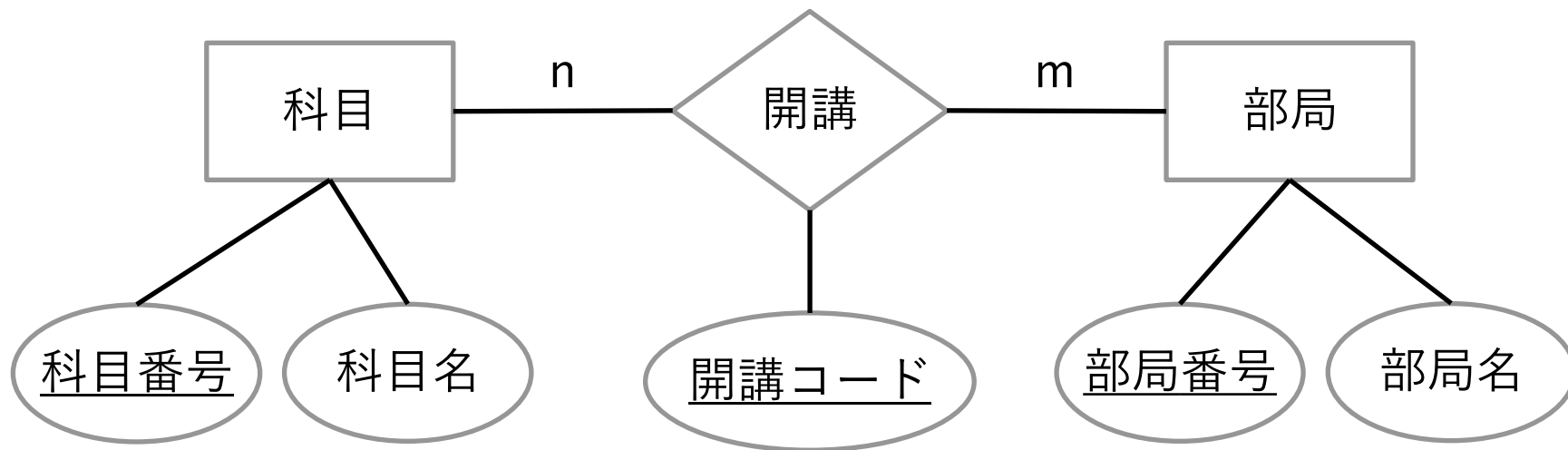
部局番号	部局名
03	理学部
05	薬学部
07	法学部

# どうやってデータベースを設計するのか

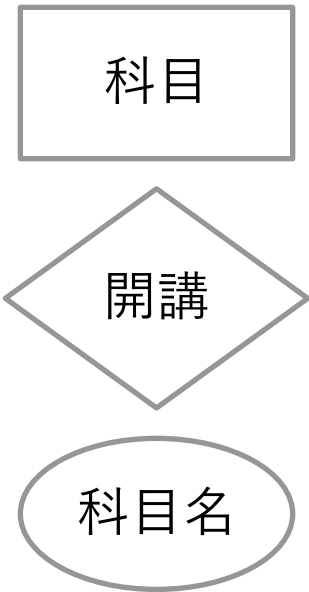
- 既存のデータベースを操作することと新しくデータベースを作成することは根本的に別の行為です
  - 既存のデータベースの構造を知っていれば、それへの検索・追加・更新・削除のために、どのようなSQLクエリを書けばよいかは分かります
  - 新しくデータベースを作る際には、どのような構造にするかを事前に決める必要があります
- 対象（何らかの概念や状況）をデータベースの構造で表現することをデータモデリングと呼びます
  - データモデリング自体はデータベースに限定されません
  - プログラミングで所望の計算を関数で表現することと似ています
- RDB向けのデータモデリングとしてERモデリングがあります
  - ERモデルの作成を通してRDBを設計することです

# ERモデル（実体関連モデル）

- 実体（entity）と実体間の関連（relationship）に基づくモデルです
- ERモデルはダイアグラムの形で記述されます
  - ここで紹介する記法は、ERモデルを提案したChenのスタイルに準じます
- 例：「開講」のERモデル
  - 「開講」を「科目」と「部局」という実体間の関連としてモデル化



# ERモデルの構成要素と構造

- 実体：

科目
- 関連：

開講
- 属性：

科目名
- 各要素は、複数のインスタンス（実例）から成る集合です
- 要素を線で繋ぐことでモデルの構造を記述します
- 実体と関連を繋ぐ線には濃度制約を記述します
  - 1を書くとき実体のインスタンスがただ1つしか関連しないことを表します
  - nなどの変数を書くとき、不特定のインスタンスが関連することを表します
  - 2重線で繋ぐとき実体のインスタンス全体が関連することを表します
- 属性は、線で繋がった対象のインスタンス全てが持ちます
  - 主キーには下線を引きます



# ERモデリングの考え方と使い方

- ERモデルは対象を説明する道具です
- 同一対象に対するERモデルは様々考えられます
  - 最も良く説明していると思えるERモデルを作ることが肝要です
- ERモデルに対応するRDBも1つに定まる訳ではありません
  - ERモデルはRDBに繋げるための土台でしかありません
- 例えば、次のようにすれば、ERモデルからRDBを構成できます
  - それぞれの実体を1つの表にする
  - 一対一の関連で繋がる実体は、表を外部キーで参照する
  - 一対多や多対多の関連は、外部キーで双方を参照する表にする
    - インスタンス（外部キーの組）を番号付けして主キーとする
- 実際には、SQLによるデータ操作が効率良くなるように、RDBの構造を工夫することが良くあります
  - 例1：一対一関連で繋がる複数実体を1つの表にまとめる
  - 例2：多対多関連を一対多関連に分解して複数表にする

# その他のデータベース

- RDBとは異なるデータモデルを採用するデータベースもあります
  - ドキュメント指向DB：ドキュメントの集合を保持するDB
  - 列指向DB：RDBと同様に表データだが、列を丸ごと操作するDB
  - XML DB：XMLの木構造を保持するDB
  - グラフDB：一般のグラフ構造を保持するDB
- RDBでないデータベースにはSQLでない言語を使うのが一般的です
  - データベースシステム毎に固有の言語であることも多いです
- 「RDB以外」を指す言葉としてNoSQLというものがあります
  - 字面からはSQLに対応する言語を指しているように見えますが、言語を指している訳ではありません
  - データベースシステムの分類として用いられます
- 複数の異なるデータベースを統合したものをデータウェアハウスと呼ぶことがあります
  - データウェアハウスもデータベースの一種と見做せます
  - 企業データを管理蓄積する文脈で使われる言葉です

# まとめ

- データベースの基本としてRDBとSQLを紹介しました
- ERモデリングによってRDBを設計する方法を紹介しました
  
- RDB以外のデータベースもSQL以外のデータベース言語もありますが、RDBとSQLは基本であり伝統なので、押さえておきましょう

# 付録：RDBの正規形

- RDBには幾つか正規形があり，その形にすることを正規化と呼びます
- 第1～第3正規形が一般的で，正規化の際によく使われます
  - Edgar F. Coddによって，RDBの提唱と同時期に提案されました
- ◆ 第1正規形：セル中の値はアトム（分割不可能な値）しかない
  - リストのような複数要素からなる値はダメ
- ◆ 第2正規形：第1正規形且つ，複数の属性の組を主キーとしたとき，他の属性は，その構成要素の属性の（一部ではなく）全てに依存する
  - 主キーが単一属性ならば定義上問題にならない
- ◆ 第3正規形：第2正規形且つ，他の属性が主キーの属性に直接依存する
  - 推移関係による間接的な依存が存在しない
- 普通に設計したRDBでは自然と満たされていることが多いです
- 正規形があることを知っておくことには価値があります