データベースS

第3回　リレーショナル代数

システム創成情報工学科　尾下 真樹

クリッカー配布

A～Jの箱の中から、自分の学生番号に対応するID番号のラベルが貼られたクリッカーを受け取ること。

A

学生番号	ID
10236076	001
11236005	002
12236016	003
12236031	004
12236032	005
12236036	006
12236038	007
12236039	008
12236043	009
12236044	010

C

学生番号	ID
13236004	021
13236005	022
13236007	023
13236008	024
13236009	025
13236010	026
13236011	027
13236012	028
13236013	029
13236014	030

E

学生番号	ID
13236025	041
13236026	042
13236027	043
13236028	044
13236029	045
13236030	046
13236031	047
13236032	048
13236033	049
13236034	050

G

学生番号	ID
13236046	061
13236047	062
13236048	063
13236049	064
13236050	065
13236051	066
13236052	067
13236053	068
13236054	069
13236055	070

I

学生番号	ID
13236067	081
13236068	082
13236069	083
13236070	084
13236071	085
13236074	086
13236075	087
13236076	088
13236077	089
13236078	090

B

学生番号	ID
12236051	011
12236052	012
12236059	013
12236063	014
12236067	015
12236071	016
12236079	017
12236080	018
13236001	019
13236002	020

D

学生番号	ID
13236015	031
13236016	032
13236017	033
13236018	034
13236019	035
13236020	036
13236021	037
13236022	038
13236023	039
13236024	040

F

学生番号	ID
13236035	051
13236037	052
13236038	053
13236039	054
13236040	055
13236041	056
13236042	057
13236043	058
13236044	059
13236045	060

H

学生番号	ID
13236056	071
13236057	072
13236058	073
13236059	074
13236060	075
13236062	076
13236063	077
13236064	078
13236065	079
13236066	080

J

学生番号	ID
13236079	091
15236201	092
15236203	093
15236204	094
15236205	095
15236206	096
15236207	097
15236208	098
15236210	099
15236211	100

クリッカー配布

- クリッカーを使用する回は、今回同様、授業開始前に配布する
  - 必ず、授業開始前に受け取ること
  - 授業開始後は、一切配布しない
  - 授業終了後に回収するので、必ず返却する
  - 自分のクリッカーID番号をメモしておくこと
- 名簿（履修登録者＋第1･2回の授業に出席した上級年次履修者）に名前がない者は、本日中に申し出ること

クリッカーの使い方

- 指示をされたら、自分が回答した選択肢の番号のボタンを押す
  - 2回以上押したときには、最後に押したボタンが有効になるので、間違った番号を押したときには、正しい番号を押しなおす

今回の内容

- 前回の復習
- 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
  - リレーショナル代数式

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）
  - 3章（3.1～3.5節）
- 「データベースシステム」
  北川 博之 著、昭晃堂 出版（3,200円）
  - 3章 30～54ページ

前回の復習

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
  - 実際にどのようにファイルやメモリにデータが格納されるかといったことは気にせず、概念的なデータ構造を定義できる（詳しくは次回以降説明）
  - 各DBMS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

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

学籍番号	氏名
0123001	尾下真樹
0123002	下戸彩
0123003	本村拓哉

履修

科目番号	学籍番号	成績
01	0123001	60
03	0123002	80
01	0123003	70

リレーションの整合性制約

- リレーションスキーマに、さまざまな制約を設定することで、整合性を保つことができる

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

属性値は各属性のドメイン制約に従う

主キー　外部キー

（超キー、候補キー）

あるリレーションの主キー

参照整合性制約

リレーションスキーマは第一正規形を満たす

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} ※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}

参照整合性制約の例

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

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

リレーションの操作体系

データモデルと操作体系

- すべてのデータモデルは操作体系を持つ
  - データベースにデータを格納するだけでは意味がなく、格納されたデータに検索などの操作を行い、結果をえることが必要となるため

リレーションの操作

- データベースのデータは複数のリレーションにまたがって格納されている
- リレーションに対する操作を行なって、必要なデータを出力する必要がある

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
004	1	宇田 ヒカル	20

部門

部門番号	部門名
1	開発
2	営業

操作

氏名	部門名
尾下 真樹	開発

例：従業員「尾下真樹」の所属する部門名を知りたい

リレーションの操作

- リレーションの操作
  - 主にデータの問い合わせ（検索）に利用
  - 複数のリレーションをもとに、求めるデータを新たなリレーションとして得る
- 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の2種類の操作体系がある

リレーション操作体系

- リレーショナル代数
  - 基本的な演算子を定義
  - 演算子を使った式によって、求めるリレーションが得られるようにリレーションの操作を記述する
- リレーショナル論理式
  - 一階述語理論にもとづき、求めるリレーションの条件を宣言的に記述する
  - リレーショナル論理式には、さらに2種類がある
    - タプルリレーショナル理論
    - ドメインリレーショナル理論

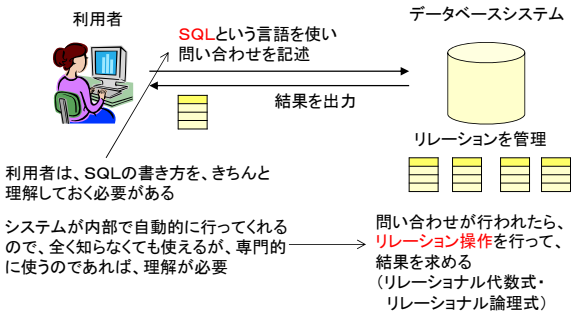
リレーション操作体系の比較

- リレーション操作体系
  - リレーショナル代数
  - リレーショナル論理式
- 両者の比較
  - どちらも同等の記述力を持つ(Ullmanの定理)
  - どちらもあらゆる操作が記述できるというわけではない(例:推移的閉包など、どちらの操作体系でも記述できない操作もある)
- 本講義では、リレーショナル代数を主に説明
  - リレーショナル論理式については教科書を参照

実際のリレーション操作

- 実際のリレーショナルデータベースでは、SQLという言語を使って、よりプログラミング言語に近い形で問い合わせを記述する
- 今回学ぶ代数演算は、リレーショナルモデルがどのような演算をサポートするかという理論的な操作体系
  - 単にリレーショナルデータベースを利用するだけであれば不要
  - 専門的に利用するためには、内部でどのような演算が行われるのかを理解しておく必要がある

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

- リレーション同士の演算
  - リレーショナルデータベースでは、リレーション同士の演算によって複雑な検索などの操作を行う
  - リレーションに対する演算の結果もリレーションになる
  - とりあえず体系的な演算を提供している
    - 数学的に扱えるということが重要
    - 具体的なシステム(プログラム)として実現するときには、また別の工夫が必要になる

リレーショナル代数演算子

- 2つのリレーション同士の演算(2項演算)
  - 和、差、共通部分、直積
- 単一リレーションに対しての演算(単項演算)
  - 射影、選択
- 結合演算(2項演算)
  - 結合、自然結合
- その他
  - 商、改名演算

和

- 和（union）

$$R \cup S = \{ t \mid t \in R \vee t \in S \}$$

  - 2つのリレーションの集合を足し合わせたもの
  - 2つのリレーションスキーマは同一でなければならない（和両立）
    - 次数が同じで、それぞれの属性のドメインも同じ
  - 通常、同一のリレーションスキーマが複数あることはなく、複数の操作結果の演算に使われる

和演算・差演算の例

従業員1

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

従業員2

従業員番号	部門番号	氏名	年齢
002	2	下戸 彩	17
005	1	織口 裕二	35

従業員1＋従業員2（和演算）

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20
005	1	織口 裕二	35

従業員1－従業員2（差演算）

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

差

- 差（difference）

$$R - S = \{ t \mid t \in R \wedge \neg t \in S \}$$

  - 片方のリレーションから、もう一方のリレーションを引いたもの
  - 他は、和演算と同じ
  - 通常、同一のリレーションスキーマが複数あることはなく、複数の操作結果の演算に使われる

共通部分

- 共通部分（intersection）

$$R \cap S = R - (R - S)$$

  - 2つのリレーションの共通部分
  - 2つのリレーションは、和演算・差演算と同様に和成立条件を満たさなければならない
  - 共通部分は、和演算・差演算により表現可能
  - 通常、同一のリレーションスキーマが複数あることはなく、複数の操作結果の演算に使われる

和演算・差演算の例

従業員1

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

従業員2

従業員番号	部門番号	氏名	年齢
002	2	下戸 彩	17
005	1	織口 裕二	35

従業員1＋∩従業員2（共通部分・積演算）

従業員番号	部門番号	氏名	年齢
002	2	下戸 彩	17

直積

- 直積（cartesian product）

$$R \times S = \{ t * u \mid t \in R \wedge u \in S \}$$

（$t * u$はタプルtとuを連結したもの）

  - リレーションの各タプル同士を全ての組み合わせでかけ合わせたリレーションを得る
  - もとのリレーション名をつけて属性名を区別
  - 単体ではあまり意味がなく、別の演算子（選択演算子）と組み合わせて使うことが多い
    - 結合演算子（詳しくは後で説明）

直積演算の例

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
004	1	宇田 ヒカル	20

×

部門

部門番号	部門名
1	開発
2	営業

=

従業員.従業員番号	従業員.部門番号	従業員.氏名	従業員.年齢	部門.部門番号	部門.部門名
001	1	尾下 真樹	27	1	開発
002	2	下戸 彩	17	1	開発
004	1	宇田 ヒカル	20	1	開発
001	1	尾下 真樹	27	2	営業
002	2	下戸 彩	17	2	営業
004	1	宇田 ヒカル	20	2	営業

リレーショナル代数演算子

- 2つのリレーション同士の演算（2項演算）
  - 和、差、共通部分、直積
- 単一リレーションに対しての演算（単項演算）
  - 射影、選択
- 結合演算（2項演算）
  - 結合、自然結合
- その他
  - 商、改名演算

注意：演算子の表記について

- 教科書（参考書）とは表記が異なるので注意
  - 「リレーショナルデータベース入門 」と「データベースシステム」で表記が異なる
    - 前者では角括弧 [ ] を使った表記、後者ではギリシャ文字を使った表記、が使われている
  - 本講義では、演算子の違いが分かりやすいよう、ギリシャ文字を使った表記に合わせている
    - 試験もこちらの表記を用いること
      - 後者の参考書を買っていなくとも、本資料の説明を理解すれば問題ない
      - 演算子の定義はどちらの参考書も同じ

射影

- 射影（projection）
  （パイ）

$$\pi_{A'_1,\cdots,A'_m}(R)=\{t[A'_1,\cdots,A'_m]\mid t\in R\}$$

$$\{A'_1,\cdots,A'_m\}\subseteq\{A'_1,\cdots,A'_n\},\ m\le n$$

  - 指定した属性だけを残して他の属性は排除
    - テーブルから必要な列（属性）のみを取り出す
  - 出力に必要ない属性や、直積で重複した属性を取り除くためなどに使用

射影演算・選択演算の例

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
003	3	本村 拓哉	30
004	1	宇田 ヒカル	20

$\pi_{氏名,\ 年齢}$（従業員）

氏名	年齢
尾下 真樹	27
下戸 彩	17
本村 拓哉	30
宇田 ヒカル	20

射影は必要な属性（列）のみを取り出す

$\sigma_{年齢>20}$（従業員）

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
003	3	本村 拓哉	30

選択は必要なタプル（行）のみを取り出す

選択

- 選択（selection）
  （シグマ）

$$\sigma_F(R)=\{t\mid t\in R\wedge P_F(t)\}$$

  - 選択条件 F で指定した条件を満たすタプルのみを残して、他のタプルは削除
    - テーブルから必要な行（インスタンス、タプル）のみを取り出す
  - 主にデータの検索処理に使われ、使用頻度はかなり高い

選択

- 選択の条件（下記のどれかの方法で記述）
  - 属性 A_i と 定数 c の比較演算 θ による比較
    $A_i \theta c$
  - 属性 A_i と 属性 A_j の比較演算子 θ による比較
    $A_i \theta A_j$
  - 上の2つの論理和（∨）、論理積（∧）、否定（¬）を用いて組み合わせたもの
- 比較演算子
  - θ は $=, <, >, \geq, \leq, \neq$ のどれか
    （シータ）

リレーショナル代数演算子

- 2つのリレーション同士の演算（2項演算）
  - 和、差、共通部分、直積
- 単一リレーションに対しての演算（単項演算）
  - 射影、選択
- 結合演算（2項演算）
  - 結合、自然結合
- その他
  - 商、改名演算

結合

- 結合（join）

$$R \bowtie_F S = \sigma_F (R \times S)$$

  - 実際の応用でよく使われる重要な演算子
    - 非常に重要！
  - 直積と選択を組み合わせた演算子
- 結合の種類
  - 等結合（比較演算子が = の場合、よく使われる）
  - θ-結合（= 以外の比較演算子の場合）

等結合の例

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
004	1	宇田 ヒカル	20

$\bowtie_{\text{部門番号}=\text{部門番号}}$

部門

部門番号	部門名
1	開発
2	営業

＝

従業員.従業員番号	従業員.部門番号	従業員.氏名	従業員.年齢	部門.部門番号	部門.部門名
001	1	尾下 真樹	27	1	開発
004	1	宇田 ヒカル	20	1	開発
002	2	下戸 彩	17	2	営業

例のように複数のリレーションの情報を組み合わせるために、結合が使われる

結合は直積と選択の組み合わせで計算される（次スライド）

等結合の例（1.直積演算）

従業員

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
004	1	宇田 ヒカル	20

×

部門

部門番号	部門名
1	開発
2	営業

＝

従業員.従業員番号	従業員.部門番号	従業員.氏名	従業員.年齢	部門.部門番号	部門.部門名
001	1	尾下 真樹	27	1	開発
002	2	下戸 彩	17	1	開発
004	1	宇田 ヒカル	20	1	開発
001	1	尾下 真樹	27	2	営業
002	2	下戸 彩	17	2	営業
004	1	宇田 ヒカル	20	2	営業

等結合の例（2.選択演算）

$\sigma_{\text{従業員.部門番号}=\text{部門.部門番号}}$

従業員.従業員番号	従業員.部門番号	従業員.氏名	従業員.年齢	部門.部門番号	部門.部門名
001	1	尾下 真樹	27	1	開発
002	2	下戸 彩	17	1	開発
004	1	宇田 ヒカル	20	1	開発
001	1	尾下 真樹	27	2	営業
002	2	下戸 彩	17	2	営業
004	1	宇田 ヒカル	20	2	営業

例のように複数のリレーションの情報を組み合わせるために、結合が使われる

＝

従業員.従業員番号	従業員.部門番号	従業員.氏名	従業員.年齢	部門.部門番号	部門.部門名
001	1	尾下 真樹	27	1	開発
004	1	宇田 ヒカル	20	1	開発
002	2	下戸 彩	17	2	営業

自然結合

- 自然結合（natural join）

$$R \rhd\lhd S = \pi_{A_1,\cdots A_n,B_1,\cdots B_m,C_1,\cdots C_k}\left(\sigma_{R.B_1=S.B_1\wedge\cdots\wedge R.B_m=S.B_m}(R\times S)\right)$$

$$R(A_1,\cdots A_n,B_1,\cdots B_m),\ S(B_1,\cdots B_m,C_1,\cdots C_k)$$

  - これも実際の応用でよく使われる重要な演算子
  - 2つのリレーションを同一の属性同士で等結合し、結合結果から同一の属性を取り除いたもの
    - 通常の等結合の結果には、同一の属性値を持つ属性が重複して存在することになり無駄

自然結合の例

従業員番号	部門番号	氏名	年齢
001	1	尾下 真樹	27
002	2	下戸 彩	17
004	1	宇田 ヒカル	20

⋈

部門番号	部門名
1	開発
2	営業

部門番号の同じタプル同士の組み合わせになる

＝

従業員番号	部門番号	氏名	年齢	部門名
001	1	尾下 真樹	27	開発
002	2	下戸 彩	17	営業
004	1	宇田 ヒカル	20	開発

※ 各属性のもとのリレーション名は省略できる

リレーショナル代数演算子

- 2つのリレーション同士の演算（2項演算）
  - 和、差、共通部分、直積
- 単一リレーションに対しての演算（単項演算）
  - 射影、選択
- 結合演算（2項演算）
  - 結合、自然結合
- その他
  - 商、改名演算子

商

- 商（division, quotient）

$$R\div S=\left\{t\mid t\in R(A_1,\cdots A_m)\wedge\left(\forall u\in S,(t,u)\in R\right)\right\}$$

  - ただし、リレーション R, S について、S の属性は、R の属性の部分集合

$$R(A_1,\cdots A_n),\ S(A_m,\cdots A_n)$$

  - Rから**S**を除いた属性について、全ての**S**との組み合わせがRに存在するものの集合を求める
  - $(R\times S)\div S=R$ が成り立つ
  - 特殊な演算なので、あまり使われない

商の例

プロジェクトメンバ

プロジェクト番号	従業員番号
p 1	1
p 1	2
p 1	3
p 2	2
p 2	3
p 3	2
p 3	3

プロジェクト

プロジェクト番号
p 1
p 2
p 3

プロジェクトメンバ÷プロジェクト

従業員番号
2
3

全てのプロジェクト番号との組が存在する、従業員番号を求まる
（全プロジェクトに参加している従業員が求まる）

R

A	B	C	D
a	b	c	d
a	b	e	f
b	c	e	f
d	e	c	d
d	e	c	f
d	e	e	f

S

C	D
c	d
e	f

R÷S

A	B
a	b
d	e

教科書 図3.9

改名演算

- 改名演算
  （デルタ）

$$\delta_{A\leftarrow B}(R)$$

  - 自然結合や商を適用するために、名前を変える演算
  - 自然結合や商では、属性の名前が一致することが条件になるので、属性の名前が異なっている場合に前もって名前を変更する
  - 使用例は後で紹介

リレーショナル代数演算子のまとめ

- 2つのリレーション同士の演算（2項演算）
  - 和、差、共通部分、直積
- 単一リレーションに対しての演算（単項演算）
  - 射影、選択
- 結合演算（2項演算）
  - 結合、自然結合
- その他
  - 商、改名演算

リレーショナル代数演算子の主な用途

- 選択、射影
  - 必要なデータ（表の行）や属性（表の列）を取り出すために使用
- 結合
  - 複数のリレーションを組み合わせるために使用
- 和、差、共通部分
  - 複数の演算結果同士を組み合わせるときに使用
- 直積
  - 直接は使用しない（結合を定義する上で重要）
- 商（あまり使わない）

リレーショナル代数式

- リレーショナル代数式
  - これまでに出てきたリレーショナル代数演算子を使って、問い合わせを記述したもの
  - 各種演算を組み合わせることでいろいろな問い合わせを記述できる
  - 「リレーショナル代数表現」とも呼ばれる

リレーションの例

学生（学籍番号、氏名）

科目（科目番号, 科目名, 単位数）

履修（科目番号, 学籍番号, 成績）

学生

学籍番号	氏名
0123001	尾下真樹
0123002	下戸彩
0123003	本村拓哉
…	…

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70
…	…	…

科目

科目番号	科目名	単位
001	データベース	2
002	プログラミング	3
…	…	…

リレーショナル代数式の例1

- 科目番号002の履修者の学籍番号と成績の一覧

$\pi_{学籍番号,成績}(\sigma_{科目番号='002'}履修)$

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70

2. 射影（π）により、学籍番号と成績の属性のみを取り出す

学籍番号	成績
00001	90
00003	70

1. 選択（σ）により、科目番号が '002' のデータを取り出す

リレーショナル代数式の例2

- 学籍番号00001の学生が履修した科目名と成績の一覧

$\pi_{科目名,成績}(科目 \bowtie (\sigma_{学籍番号='00001'}履修))$

履修

科目番号	学籍番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70

$\bowtie$

科目

科目番号	科目名	単位
001	データベース	2
002	プログラミング	3

リレーショナル代数式の例2

- 学籍番号00001の学生が履修した科目名と成績の一覧

$\pi_{科目名,成績}(科目 \bowtie (\sigma_{学籍番号='00001'} 履修))$

科目番号	学籍番号	成績	科目名	単位
001	00001	90	データベース	2
002	00001	90	プログラミング	3

科目名	成績
データベース	90
プログラミング	90

リレーショナル代数式の例2

- 学籍番号00001の学生が履修した科目名と成績の一覧

$\pi_{科目名,成績}(科目 \bowtie (\sigma_{学籍番号='00001'} 履修))$

- 別解
  - こちらでも良いが、通常は上の書き方をする
  - 選択は科目には関係ないので

$\pi_{科目名,成績}(\sigma_{学籍番号='00001'}(科目 \bowtie 履修))$

リレーショナル代数式の例3

- 科目番号001の科目について、学籍番号00001の学生よりも成績の良かった学生の学籍番号の一覧
  - 2つの問い合わせの組み合わせ
  - 改名演算子が必要になる

リレーショナル代数式の例3

- 科目番号001の科目について、学籍番号00001の学生よりも成績の良かった学生の学籍番号の一覧
  1. 学籍番号00001、科目番号001の成績 → 履修'
  2. 科目番号001、履修'の成績よりも成績が大きい

履修

科目番号	学籍番号	成績
001	00001	80
001	00002	70
002	00001	80
001	00003	90

履修'

科目番号	学籍番号	成績
001	00001	80

$(\sigma_{科目番号=1 \wedge 学籍番号=00001} 履修)$

リレーショナル代数式の例3

- 1. 学籍番号00001、科目番号001の成績

$(\sigma_{科目番号=001 \wedge 学籍番号=00001} 履修)$

履修

科目番号	学籍番号	成績
001	00001	80
001	00002	70
002	00001	80
001	00003	90

履修'

科目番号	学籍番号	成績
001	00001	80

リレーショナル代数式の例3

- 2. 科目番号001、履修'の成績よりも成績が大きい
  - タプル同士の比較なので選択ではなく結合を使う

$(\sigma_{科目番号=001} 履修) \bowtie_{成績>成績} 履修')$

履修

	科目番号	学籍番号	成績
×	001	00001	80
×	001	00002	70
	002	00001	80
○	001	00003	90

履修'

科目番号	学籍番号	成績
001	00001	80

同じリレーション同士の演算では成績>成績 などと書くと混乱するので、改名演算子を用いて区別

リレーショナル代数式の例3

- 2. 科目番号001、履修‘の成績よりも成績が大きい

$(\sigma_{\text{科目番号=001}}\ \text{履修} \bowtie_{\text{成績>成績}}\ \text{履修'})$

– 改名演算子を使って属性名を読み替える

$(\sigma_{\text{科目番号=1}}\ \text{履修} \bowtie_{\text{成績>成績'}}\ \delta_{\text{成績←成績'}}\ \text{履修'}))$

リレーショナル代数式の例3

- 科目番号001の科目について、学籍番号00001の学生よりも成績の良かった学生の学籍番号の一覧
  1. 学籍番号00001、科目番号001の成績
  2. 科目番号001、履修‘の成績よりも成績が大きい
  – 改名演算子が必要になる

$\pi_{\text{学籍番号}}((\sigma_{\text{科目番号=001}}\ \text{履修}) \bowtie_{\text{成績>成績'}}\ \delta_{\text{成績←成績'}}(\sigma_{\text{科目番号=001}\wedge\text{学籍番号=00001}}\ \text{履修}))$

リレーショナル代数式の例4

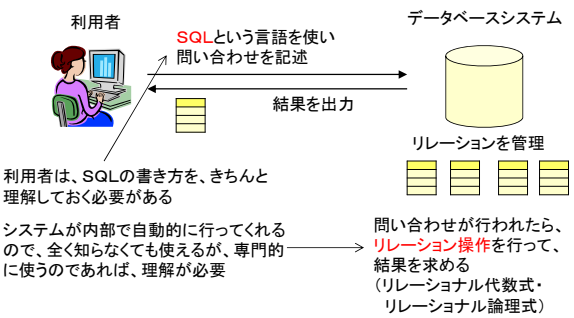
- 履修者が全くいない科目の科目名の一覧
  – 履修者が1人でもいる科目を取り出す
    - 科目と履修の自然結合
    - リレーションは集合なので、同じ科目が複数あっても、射影の時点で重複は取り除かれる
  – 全体の科目から引く
    - 履修者が一人もいない科目が得られる

$(\pi_{\text{科目名}}\ \text{科目}) - (\pi_{\text{科目名}}(\text{科目} \bowtie \text{履修}))$

まとめ

- 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
  – リレーショナル代数式

SQLとリレーション操作の関係



次回予告

- 次回（第4回）
  – データベース言語SQL（1）
- 第5回
  – データベース演習、PostgreSQL入門
  – リレーショナルデータベースを実際に体験する
- 第6回
  – データベース言語SQL（2）
  – SQLのより詳しい書き方を学ぶ
  – 実際のデータベースでSQLを使ってみる