データベース設計論

第5回　関係代数（中級編）

【全ての操作体系に共通な注意事項】

バッグ集合とセット集合

- バッグ集合
  - 重複を許す集合
- セット集合
  - 重複を許さない集合
- 例）$\pi_{grade}(students)$

基本的にSQLではバッグ集合を返す
・理由：重複の削除などは重い処理

基本的に関係代数，関係論理では
セット集合を返す

検索結果のバッグ集合

grade
3
2
3

検索結果のセット集合

grade
3
2

関係代数演算

- 基本演算
  - 和($A \cup \mathrm{B}$), 差($A - B$), 交差($A \cap B$)
  - 直積 ($A \times B$)
  - 射影($\pi_L(R)$), 選択 ($\sigma_C(R)$)
  - 結合($A \bowtie_C B$)
  - 商($A \div B$)
- 使うと便利な演算
  - 自然結合 ($A * B$)
  - ネーミング演算 ($\rho_{S(A_1,\ldots,A_n)}(R)$)

直積 $(A \times B)$

- 全てのタプルの組合せを求める

A(stid, name)

stid	name
g001	chiemi
g002	aya

B(stid, comment)

stid	comment
g001	nice!
g001	I can’t read…

$A \times B$

stid	name	stid	comment
g001	chiemi	g001	nice!
g002	aya	g001	nice!
g001	chiemi	g001	I can’t read…
g002	aya	g001	I can’t read…

θ-結合($A \bowtie_c B$)は直積と選択で表せる

- $A \bowtie_C B = \sigma_c(A \times B)$
- 例）$A \bowtie_{A.stid=B.stid} B = \sigma_{A.stid=B.stid}(A \times B)$

$A \times B$

stid	name	stid	comment
g001	chiemi	g001	nice!
g002	aya	g001	nice!
g001	chiemi	g001	I can't read…
g002	aya	g001	I can't read…

$A \bowtie_{A.stid=B.stid} B$

stid	name	stid	comment
g001	chiemi	g001	nice!
g001	chiemi	g001	I can't read…

自然結合 $A * B$

- 結合する二つのリレーションに含まれる共通の属性で等結合したもの

$$A * B = \pi_{A.stid,A.name,B.comment}(A \bowtie_{A.stid=B.stid} B)$$

A(stid, name)

stid	name
g001	chiemi
g002	aya

B(stid, comment)

stid	comment
g001	nice!
g001	I can’t read…

$A * B$

stid	name	comment
g001	chiemi	nice!
g001	chiemi	I can’t read…

ネーミング演算

- 関係代数演算の結果リレーションの
  リレーション名や属性名を変更する

$$S(A_1, \ldots, A_n) = \rho_{S(A_1,\ldots,A_n)} R$$

- 例）リレーションGameを使って
  Gameの勝ち負けの対応表を求める

↑ 誰が誰に何点勝ってるか

$Game(name, score)$

name	score
chiemi	67
takako	92
aiko	78

$$G_1 = \rho_{G1}(Game(name, score))$$
$$G_2 = \rho_{G2}(Game(name, score))$$

$vs(winner, loser, diff)$

winner	loser	diff
takako	chiemi	25
takako	aiko	14
aiko	chiemi	11

$$\rho_{vs(winner,loser,diff)}(\pi_{G2.name,G1.name,G2.score-G1.score}(G_2 \bowtie_{G2.score>G1.score} G_1))$$

データベース設計論

SQL中級編(集約演算，入れ子問合せ）

本日の内容

SQL中級編（その1）:
関係代数では（単純には）表せない
演算を紹介します

- いろいろな問合せ
  - like演算子，IN, EXISTS演算子
- 集約演算
  - 複数のタプルに対する演算
  - max, min, avgなど
- ~~入れ子問合せ~~
  - ~~問合せ文の中に問合せ文を含める~~

これは次週

SQLに関するいろいろ

スライドにしていなかった項目（SQL編）

- 問合せの結果をセット集合にしたい場合は distinct をつける

```
SELECT distinct grade
FROM students
```

- FROM節に書くリレーションに別名をつけることができる

```
SELECT s2.stid, st.name
FROM students s1, students s2
WHERE s1.stid = ‘g1120511’
  and s1.grade = s2.grade;
```

スライドにしていなかった項目（SQL編）

- SELECT節に書く属性に別名をつけることができる

関係代数のネーミング演算で使った例（再掲）

$Game(name, score)$

name	score
chiemi	67
takako	92
aiko	78

$\rho_{G1}(Game(name, score))$

$\rho_{G2}(Game(name, score))$

$vs(winner, loser, diff)$

winner	loser	diff
takako	chiemi	25
takako	aiko	14
aiko	chiemi	11

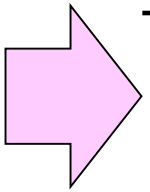
$$\rho_{vs(winner,loser,diff)}(\pi_{G2.name,G1.name,G2.score-G1.score}(G_2 \bowtie_{G2.score>G1.score} G_1))$$

```
SELECT G2.name as winner, G1.name as loser,
       G2.score – G1.score as diff
FROM Game G1, Game G2
WHERE G2.score > G1.store
```

スライドにしていなかった項目(SQL編)

- FROMで複数のリレーションを指定するときは属性名が衝突してはいけない

```
SELECT name, comment
FROM students s, comments c
WHERE s.stid = c.stid
      and s.stid = ‘gxx205xx’
```



エラーで実行できない:
nameがstudentsのnameか
commentsのnameか
判別できない

対策

```
SELECT s.name, c.comment
FROM students s, comments c
WHERE s.stid = c.stid
      and s.stid = ‘gxx205xx’
```

LIKE構文 (Where節)

- LIKE構文を使って部分一致検索ができる

記号	意味
%	0文字以上の任意の文字にマッチ
_	1文字の任意の文字にマッチ

作成しているデータベース名に「映画」が
含まれているチーム名とそのデータベース

```
SELECT t.name, t.database
FROM teams t
WHERE database like '%映画%'
```

集合に対する演算 (Where節）

- IN句を使って属性の値がある集合に含まれているタプルを取り出すことができる

教員かその他のアカウント

```
SELECT s.stid, s.name
FROM students s
WHERE grade IN (10,20)
```

- BETWEEN句を使って指定した範囲の属性値を持つタプルを取り出すことができる

学籍番号がg1120501からg1120510までの学生

```
SELECT s.stid, s.name
FROM students s
WHERE s.stid BETWEEN 'g1120501' AND 'g1120510'
```

文字列の順序関係を使っている

集約演算

集約演算

- WHERE節に該当する全てのタプルの属性にたいする集約演算を行うことができる

sum(*expression*)	合計値を計算する
avg(*expression*)	平均値を計算する
count(*expression*)	nullでない行数を計算する
count(*)	行数を計算する
max(*expression*)	最大値を計算する
min(*expression*)	最小値を計算する

expression：属性，もしくは属性を使った式

集約演算を含む問合せ例

- グループワークを行っている3年生の数

```
SELECT count(*)
FROM students s
WHERE s.grade=3
```

*を指定した場合は
タプル数を数える

- 課題番号2の課題の最新提出日時

```
SELECT max(uploaded)
FROM works w
WHERE w.exid=2
```

GROUP BY句をつかってグループ毎に集約演算

- GROUP BYを使って属性の値ごとにデータをグルーピングできる

例）課題ごとに提出件数を出力する

```
SELECT e.exid, count(*)
FROM exercises e, works w
WHERE e.exid = w.exid
GROUP BY e.exid
```

集約演算count(*)はgroup byで指定した属性の値が同じタプルの集合で演算を行う

課題リスト

No.	課題名	締切	提出件数
1	ER図の設計	2013/10/22	33
2	リレーションスキーマの設計	2013/11/10	27

※ただしこの問合せだと，複数のバージョンを提出した場合別々にカウントされる

GROUP BY句を使うときの注意

- GROUP BY句を使った場合，SELECT節には
  GROUP BYで指定した属性か集約演算しか書けない

ダメな例）課題ごとに提出チーム一覧を求めようとする

```
SELECT e.exid, e.title, w.teamid, count(*)
FROM exercises e, works w
WHERE e.exid = w.exid
GROUP BY e.exid
```

実質は一つの値しかないのだけど，DBMSにはそれがわからないのでNG

結果が集合値をとってしまうのでNG

ただし，DBMSによってはエラーを出さずに，そのグループに該当するタプルの中から適当に一つの値を出力する場合もある

- SQLiteもエラーを出さないので注意

GROUP BYで複数の属性を指定する

- 指定された属性の組合せでグループ化する

kadai

teamid	exid	versionid
1	1	1
1	1	2
1	2	1
2	1	1
2	1	2

```
SELECT teamid,count(*)
FROM kadai
GROUP BY teamid
```

teamid	count(*)
1	3
2	2

```
SELECT teamid,exid,count(*)
FROM kadai
GROUP BY teamid,exid
```

teamid	exid	count(*)
1	1	2
1	2	1
2	1	2

GROUP BY句を使うときの注意

- 2枚前のスライドの例でe.titleも出力したい場合…
  - e.titleもGROUP BYに入れる

```
SELECT e.exid, e.title, count(*)
FROM exercises e, works w
WHERE e.exid = w.exid
GROUP BY e.exid, e.title
```

  - 実質はexidに対してtitleは1つしかないので実際の結果は変わらないのだが、GROUP BYに加えることでDBMSが理解する

異なり数をカウントする

- 集約演算countで重複する値をカウントしないようにするには属性値の前に distinct をつける

kadai

teamid	exid	versionid
1	1	1
1	1	2
1	2	1
2	1	1
2	1	2

```
SELECT teamid,count(exid)
FROM kadai
GROUP BY teamid
```

teamid	count(*)
1	3
2	2

```
SELECT teamid,count(distinct exid)
FROM kadai
GROUP BY teamid
```

teamid	count(*)
1	2
2	1

HAVING節を使って集約演算を含んだ検索条件を書く

- 集約演算を含んだ検索条件はWHERE節には書けない
  - HAVING節に書こう

2つ以上の課題に提出しているチームの名前

```
SELECT t.name
FROM works w, teams t
WHERE w.teamid = t.teamid
GROUP BY w.teamid
HAVING count(distinct w.exid)>1
```

ORDER BY句を使って並べ替え

課題番号2の提出チームを,最終提出日の遅い順に並べる

```
SELECT w.teamid, w.uploaded
FROM works w
WHERE w.exid = 2
ORDER BY w.uploaded desc
```

1つの課題についていくつかのバージョンを提出した場合，同一のチームが何度も登場する

降順にしたい場合は descをつける
昇順にしたい場合は何もつけない

```
SELECT w.teamid, max(w.uploaded)
FROM works w
WHERE w.exid = 2
GROUP BY w.teamid
ORDER BY max(w.uploaded) desc
```

チーム番号でgroup byしているので，同一のチームが何度も登場することはない

SELECT節に入れられるものはORDER BY で並べ替えられる