データベースS

第6回　データベース言語SQL(2)

システム創成情報工学科　尾下 真樹

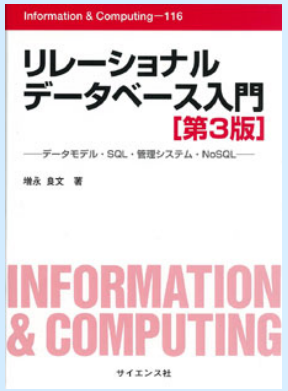
2018年度 Q2

今日の内容

- 前回の復習
- SQLによる問い合わせの記述方法
  - 結合（JOIN）
  - 集合演算
  - 副問い合わせ（入れ子型質問）
  - SQLによる問い合わせの記述の考え方
- データベース演習課題（宿題）

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）
  - 5章（5.5～5.6節）
- 「データベースシステム」
  北川 博之 著、昭晃堂 出版（3,200円）
  - 5章
  - それぞれ、SQLの具体例が載っているので、資料の具体例だけでは分かりにくければ、教科書の例も参考にすると良い



前回の復習

PosgreSQLの使い方

- データベースの作成
- psqlの起動
- テーブルの作成
- データの挿入
- SQLによる問い合わせ
- データの更新と削除
- 複数のテーブルと外部参照整合性制約

演習課題

- 演習1. データベースの作成
- 演習2. データの追加
- 演習3. 問い合わせの実行
- 演習4. 外部参照整合性制約の確認

- 今回は、前回の演習を踏まえて、SQLによる問い合わせの演習を行う

前々回の復習

SQLによる問い合わせの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問い合わせ（入れ子型質問）

SQLの記述方法

```
SELECT    表.属性(値式), …
FROM      表, …
WHERE     条件式  AND ...
```

- SELECT 節
  - 問い合わせの結果として取り出す属性(値式)を指定
- FROM 節
  - どの表(テーブル)から検索するかを指定
- WHERE 節
  - 検索の条件を指定
- ORDER BY節、GROUP BY節、HAVING節(後述)

集約関数

- 検索結果の表に対して集計演算を行い、表の全データを出力する代わりに、集約演算の結果を1行だけ出力する
  - COUNT(行数)、SUM(合計値), AVG(平均点)、MAX(最大値)、MIN(最小値)
  - 出力されるテーブルは1行だけになることに注意

Q. 科目番号 001 の平均点、最小点、最高点

```
SELECT   AVG(成績), MIN(成績), MAX(成績)
FROM     履修
WHERE    科目番号 = '001'
```

グループ表（GROUP BY）

- GROUP BY
  - 指定した属性によりデータをグループ化
    - 属性値が等しいデータ同士をグループにまとめる
  - 集約関数と組み合わせて用いる（重要）
    - 集約関数は、全データではなく、各グループごとの全データに適用される（グループ数分のデータを出力）

Q. 全科目の科目番号と平均点の一覧を出力

```
SELECT      科目番号, AVG（成績）
FROM        履修
GROUP BY    科目番号
```

グループ表の例

- グループ表の例

科目番号	学生番号	成績
001	001001	65
001	001002	75
001	001004	70
002	001002	80
002	001003	60
002	001004	90
002	001005	70
003	001004	70
…	…	…

集約演算は、各グループごとに適用されることに注意

各グループをひとつのデータ（行）として出力

科目番号	成績
001	70
002	75
…	…

グループ表＋グループ選択（HAVING）

- GROUP BY + HAVING
  - HAVING により出力するグループの条件を指定

Q. 履修者が30名以上の科目の
科目番号、履修者数、平均点の一覧

```
SELECT    科目番号, COUNT(＊), AVG(成績)
FROM      履修
GROUP BY  科目番号
HAVING    COUNT(＊) >= 30
```

GROUP BY での処理適用順序

① FROM ……（入力とするテーブル）
② WHERE ……（出力するデータを選択）
③ GROUP BY ……（出力されたデータをグループ化）
④ HAVING ……（出力するグループを選択）
⑤ SELECT ……（出力する属性）

- FROM 節のテーブルの各データの組み合わせから、WHERE 節で書かれている条件を満たす組を選択
- 選択されたデータに対して、GROUP BY 節に書かれている属性の値が同じもの同士でグループ化
- 各グループごとに、HAVING 節に書かれている条件を満たすかどうか判定し、条件を満たすもののみを選択
- 選択されたデータ or グループの属性のうち、SELECT 節に書かれているものを出力

GROUP BY 使用時の注意

- GROUP BY 使用時の注意

```
SELECT      科目番号, COUNT(＊), AVG(成績)
FROM        履修
GROUP BY    科目番号
HAVING      COUNT(＊) >= 30
```

  - GROUP BY 節があるときは、各グループが出力の単位となるので、SELECT 節や HAVING 節にはグループで共通な属性(GROPU BY に使った属性)や集約関数しか書けない (重要!)

WHEREとHAVINGの使い分け

- WHERE
  - データ(インスタンス)に関する条件
    - グループにまとめる前に評価される
    - データに関する条件なので、集約関数は使えない
- HAVING
  - グループに関する条件
    - グループにまとめた後で評価される
    - グループに関する条件なので、集約関数しか使えない

SQLによる問い合わせの記述

表（リレーション）の例

以降の記述例で使う表（リレーション）

学生（学生番号、氏名）

科目（科目番号, 科目名, 単位数）

履修（科目番号, 学生番号, 成績）

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
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
…	…	…

表（リレーション）の例

以降の記述例で使う表（リレーション）

学生（学生番号、氏名）

科目（科目番号, 科目名, 単位数）

履修（科目番号, 学生番号, 成績）

実習課題（科目番号, 課題番号, 課題名）

実習課題

科目番号	課題番号	課題名
001	01	SQL
003	01	プログラム基礎
003	02	プログラム応用

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
…	…	…

SQLによる問い合わせの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問い合わせ（入れ子型質問）

結合質問

- 結合質問（Join Query）
  - 複数の表を組み合わせた質問
    （表同士の結合が起こる質問）
  - 結合を行う表を FROM節に記述する
  - 複数の表のデータを結合するための条件を
    WHERE節に記述する
    - 結合を行うことを明示的に指定しなくとも、自動的に結合が行われる

結合

Q.学生番号 00001 の学生が履修した科目の科目番号、科目名、成績の一覧

履修(科目番号, 学生番号, 成績)

科目(科目番号, 科目名, 単位数)

このような問い合わせを実現するためのSQLを、どのように書けば良いか？

結合

Q.学生番号 00001 の学生が履修した科目の科目番号、科目名、成績の一覧

履修（科目番号, 学生番号, 成績）
科目（科目番号, 科目名, 単位数）

科目番号が同じデータ同士を結合

– 履修の表には、科目番号・成績の情報しかないため、科目名の情報を得るためには、科目の表と結合をする必要がある

結合

Q.学生番号 00001 の学生が履修した科目の科目番号、科目名、成績の一覧

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号  AND
         学生番号 = '00001'
```

– 科目と履修を科目番号で等結合

科目（科目番号, 科目名, 単位数）
履修（科目番号, 学生番号, 成績）

科目番号が同じデータ同士を結合

→（科目.科目番号, 科目名, 単位数,
履修.科目番号, 学生番号, 成績）

結合の例

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号  AND
         学生番号 = '00001'
```

履修

科目番号	学生番号	成績
001	00001	90
001	00002	80
002	00001	90
002	00003	70

×

科目

科目番号	科目名	単位数
001	データベース	2
002	グラフィックス	2
003	プログラミング	2

2つの表の各データ同士の組み合わせのうち、条件を満たすデータ同士の組み合わせが出力される
（この例では、4×3＝12通りのデータのうち、2つが出力される）

結合の例(続き)

```
SELECT   科目.科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号  AND
         学生番号 = '00001'
```

履修.科目番号	学生番号	成績	科目.科目番号	科目名	単位数
001	00001	90	001	データベース	2
002	00002	80	002	グラフィックス	2

⇩ SELECT節で指定された属性のみを出力

科目.科目番号	科目名	成績
001	データベース	90
002	グラフィックス	80

別の結合の例

- 結合を使った問い合わせの例
  - 単に複数のテーブルを結合するだけなく、結合条件を使って、データを選択することもできる

Q.実習課題がある科目の科目名の一覧

```
SELECT   DISTINCT  科目名
FROM     科目, 実習課題
WHERE    科目.科目番号 = 実習課題.科目番号
```

科目

科目番		
001	データベース	2
002	グラフィックス	2
003	プログラミング	2

		題名
001	01	SQL
003	01	プログラム基礎
003	02	プログラム応用

別の結合の例（続き）

- SELECT　DISTINCT　科目名
  FROM　科目, 実習課題
  WHERE　科目.科目番号 = 実習課題.科目番号

科目

科目番号	科目名	単位数
001	データベース	2
002	グラフィックス	2
003	プログラミング	2

×

実習課題

科目番号	課題番号	課題名
001	01	SQL
003	01	プログラム基礎
003	02	プログラム応用

各科目について、結合される実習課題が
存在する科目のみ、出力される
（DISTINCT を指定することで重複を除去）

科目名
データベース
プログラミング

自己結合

- 自己結合
  - 同一の表同士の結合
  - 同一の表を複数使うときに、それぞれの表を区別するために、表に名前をつけることができる
  - 例:
    - FROM　学生 AS x, 学生 AS y

自己結合の例

Q.科目番号 001 の科目に関して、
学生番号 001001 の学生よりも成績が良かった学生の学生番号の一覧

```
SELECT   y.学生番号
FROM     履修 AS x, 履修 AS y
WHERE    x.科目番号 = '001'  AND  x.学生番号 = '001001'
         AND  y.科目番号 = '001'  AND  y.成績 > x.成績
```

- x は科目番号001、学生番号001001の履修データ
- y は x よりも成績の良い履修データ

自己結合の例（続き）

WHERE　x.科目番号 = ‘001’ AND x.学生番号 = ‘001001’
AND y.科目番号 = ‘001’ AND y.成績 > x.成績

履修 AS x

科目番号	学生番号	成績
001	001001	60
001	001002	75
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

履修 AS y

科目番号	学生番号	成績
001	001001	60
001	001002	75
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

SQLによる問い合わせの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問い合わせ（入れ子型質問）

集合演算

- 複数の表同士の集合演算
  - UNION（和集合）
  - EXCEPT（差集合）
  - INTERSECT（共通集合）
- 表の属性は一致している必要がある
- 集合演算では、自動的に重複する行は除去される
  - 重複を除去しない集合演算も存在
  - UNION ALL, EXCEPT ALL, INTERSECT ALL

集合演算の例

- UNION（和集合）の例

Q.実習課題があるか、単位数が5単位以上の科目の科目番号、科目名、単位数の一覧

```
SELECT    科目.*
FROM      科目, 実習課題
WHERE     科目.科目番号 = 実習課題.科目番号

UNION

SELECT    *
FROM      科目
WHERE     単位数 >= 5
```

集合演算の例

- EXCEPT（差集合）の例

Q.実習課題のない科目の科目番号、科目名の一覧

```
SELECT    科目番号, 科目名
FROM      科目

EXCEPT

SELECT    科目.科目番号, 科目名
FROM      科目, 実習課題
WHERE     科目.科目番号 = 実習課題.科目番号
```

（全科目 － 実習課題のある科目）の差集合

SQLによる問い合わせの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問い合わせ（入れ子型質問）

副問い合わせ（入れ子型質問）

- 入れ子型質問（nested query）
  - SQLでは副問い合わせ（subquery）という用語で定義されている
- 問い合わせの結果を使って、さらに問い合わせを行うような問い合わせ
- 入れ子型質問の種類
  - 相関を有しない入れ子型質問
  - 相関を有する入れ子型質問

入れ子型質問

- 入れ子型質問の形
  - 外側の質問の中に、内側の質問がある
  - 内側の質問の結果を、外側の質問で使用
  - 最終的には、外側の質問の結果が出力される

外側

```
SELECT    ・・・・・・
FROM      ・・・・・・
WHERE     ・・・・・・
                内側の問い合わせの結果を使った条件
    （SELECT    ・・・・・・
     FROM       ・・・・・・
     WHERE      ・・・・・・ ）
```

内側

入れ子型質問の形

- 内側の質問の出力はテーブルとなる
  - 内側の質問の出力が1行1列のテーブルの場合
    - 内側の質問の出力を一つの値とみなして、条件式や述語を外側の質問の条件に用いることができる
    - 例：WHERE 属性 =（内側の質問）
  - 内側の質問の出力が一般のテーブルになる場合
    - テーブルに対する述語を外側の質問の条件に用いる
    - 例：WHERE EXITS（内側の質問）
      - テーブルにインスタンスが一つでも存在するかを判定
    - 例：WHERE 属性 IN（内側の質問）
      - 属性値が対象のテーブル中のどこかに存在するかを判定

相関を有さない入れ子型質問

- 相関を有さない入れ子型質問
  - 外側の質問が、内側の質問に影響を与えない
  - 内側の質問を最初に一度だけ実行し、その結果を使って、外側の質問を実行する

外側

```
SELECT    ・・・・・・
FROM      ・・・・・・
WHERE     ・・・・・・
                内側の問い合わせの結果を使った条件
    （SELECT    ・・・・・・
     FROM       ・・・・・・
     WHERE      ・・・・・・ ）
```

内側

相関を有さない入れ子型質問の例

- 相関を有さない入れ子型質問の例

Q.科目番号 001 の成績が平均点よりも高い学生の学生番号と成績の一覧

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = ‘001’ AND
          成績 >
    （SELECT    AVG（成績）
     FROM       履修
     WHERE      科目番号 = ‘001’ ）
```

内側の問い合わせは最初に1度だけ行われる

相関を有さない入れ子型質問の例

- 相関を有さない入れ子型質問の例

Q.科目番号 001 の成
学生の学生番号と成

```
SELECT    学生番号, 成
FROM      履修
WHERE     科目番号 = '00      AND
          成績 >
    (SELECT    AVG(成績)
     FROM      履修
     WHERE     科目番号 = '001' )
```

内側の質問の出力は1行1列のテーブルになるため、出力を一つの値とみなして、条件式に用いることができる

内側の問い合わせは最初に1度だけ行われる

相関を有さない入れ子型質問の例

- この例では、履修が 2回使われるので、厳密に書きたければ名前をつけて区別しても良い
  - 指定しなければ、それぞれ内側・外側で定義したものが自動的に使われるため、省略しても可

```
SELECT    x.学生番号, x.成績
FROM      履修 AS x
WHERE     x.科目番号 = '001'  AND
          x.成績 >
    (SELECT     AVG(y.成績)
     FROM       履修 AS y
     WHERE      y.科目番号 = '001' )
```

相関を有さない入れ子型質問の例

科目番号	学生番号	成績
001	001001	60
001	001002	76
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = '001'
          成績 >
    (SELECT    AVG(成績)
     FROM      履修
     WHERE     科目番号 = '001' )
```

⇨ 64

内側の問い合わせを
最初に一度だけ実行

相関を有さない入れ子型質問の例

科目番号	学生番号	成績
001	001001	60
001	001002	76
001	001004	70
001	001005	50
002	001003	60
002	001004	90
003	001004	70
…	…	…

```
SELECT   学生番号, 成績
FROM     履修
WHERE    科目番号 = '001'
         成績 >
    (SELECT    AVG(成績)
     FROM      履修
     WHERE     科目番号 = '001' )
```

＞ 64

外側の問い
合わせを実行

結果

学生番号	成績
001002	76
001004	70

自己結合 と 相関を有さない 入れ子型質問の関係

- 自己結合 or 相関のない入れ子型質問
  - 検索結果のデータを、さらに条件として使用する場合は、結合 or 相関のない入れ子型質問として書くことになる
    - どちらの書き方を使っても書ける
    - ただし、内側の質問について集約演算が必要な場合は、入れ子型質問として書く必要がある
      - 自己結合だけでは書けない

自己結合 と 相関を有さない 入れ子型質問の関係

例. 科目番号 001 の科目に関して、
学生番号 001001 の学生よりも成績の
良かった学生の学生番号の一覧

- 自己結合 or 相関のない入れ子型質問として記述

例.科目番号 001 の成績が平均点よりも高い
学生の学生番号と成績の一覧

- 相関のない入れ子型質問として記述
- 集約関数を使う必要があるため、自己結合だけでは記述できない

自己結合 と 相関を有さない 入れ子型質問の関係

例. 科目番号 001 の科目に関して、
学生番号 001001 の学生よりも成績が
良かった学生の学生番号の一覧

- 自己結合 or 相関のない入れ子型質問

```
SELECT   y.学生番号
FROM     履修 AS x, 履修 AS y
WHERE    x.科目番号 = '001'  AND  x.学生番号 = '001001'
         AND  y.科目番号 = '001'  AND  y.成績 > x.成績
```

自己結合 と 相関を有さない 入れ子型質問の関係

例：科目番号 001 の科目に関して

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = ‘001’ AND
          成績 >
    （SELECT   成績
      FROM     履修
      WHERE    科目番号 = ‘001’ AND 学生番号 = ‘001001’）
```

```
SELECT    y.学生番号
FROM      履修 AS x, 履修 AS y
WHERE     x.科目番号 = ‘001’ AND  x.学生番号 = ‘001001’
          AND  y.科目番号 = ‘001’ AND  y.成績 > x.成績
```

自己結合 と 相関を有さない 入れ子型質問の関係

例. 科目番号 001 の成績が平均点よりも高い学生の学生番号と成績の一覧

- 相関のない入れ子型質問
  - 自己結合では記述できない（集約関数が必要なため）

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = ‘001’ AND
          成績 >
    （SELECT    AVG（成績）
     FROM       履修
     WHERE      科目番号 = ‘001’ ）
```

相関を有する入れ子型質問

- 相関を有する入れ子型質問
  - 外側の質問が、内側の質問に影響を与える
  - 外側の質問の各データごとに、内側の質問を繰り返し実行する必要がある

外側

```
SELECT    ・・・・・・
FROM      ・・・・・・
WHERE     ・・・・・・
                内側の問い合わせの結果を使った条件
    （SELECT    ・・・・・・
     FROM       ・・・・・・
     WHERE      ・・・・・・ ）
```

内側

相関を有する入れ子型質問の例

- 相関を有する入れ子型質問の例
  - 学生番号001001の学生の履修で、成績が平均点よりも高い科目の科目番号、成績の一覧を出力

```
SELECT    x.科目番号, x.成績
FROM      履修 AS x
WHERE     x.学生番号 = ’001001’ AND
          x.成績 >
    （SELECT    AVG（y.成績）
     FROM       履修 AS y
     WHERE      x.科目番号 = y.科目番号 ）
```

  - 各履修ごとに、その科目の平均点を計算

相関を有する入れ子型質問の例

- 相関を有する入れ子型質問の例
  - 学生番号001001の学生の履修で、成績が平均点よりも高い科目の科目番号、成績の一覧を出力

```
SELECT    x.科目番号, x.成績
FROM      履修 AS x
WHERE     x.学生番号 = ’001001’ AND
          x.成績 >
    （SELECT    AVG（y.成績）
     FROM       履修 AS y
     WHERE      x.科目番号 = y.科目番号 ）
```

外側の質問のデータが内側の質問で条件として使われている＝相関を有する

  - 各履修ごとに、その科目の平均点を計算

```
SELECT    x.科目番号, x.成績
FROM      履修 AS x
WHERE     x.学生番号 = '001001' AND
          x.成績 >
    (SELECT    AVG(y.成績)
     FROM      履修 AS y
     WHERE     x.科目番号 = y.科目番号 )
```

履修 AS x

科目番号	学生番号	成績
001	001001	60
001	001004	70
001	001005	50
002	001001	60
…	…	…

×
×

履修 AS y

科目番号	学生番号	成績
001	001001	60
001	001004	70
001	001005	50
002	001003	60
…	…	…

相関を有する入れ子型質問

Q. 実習課題のない科目の

```
SELECT    科目番号, 科目名
FROM      科目
WHERE     NOT EXISTS
    （SELECT   ＊
      FROM     実習課題
      WHERE    科目.科目番号 = 実習課題.科目番号）
```

内側の質問の出力は一般のテーブルになるため、テーブルに対する述語を条件式に用いる

- NOT EXISTS は、集合が空の時に真になる述語
- 各科目ごとに内側の処理（その科目番号の実習課題があるかを調べる）を繰り返す必要がある

相関を有する入れ子型質問

Q. 実習課題のない科目の科目番号、科目名

```
SELECT    科目番号, 科目名
FROM      科目
WHERE     NOT EXISTS
     （SELECT   ＊
       FROM     実習課題
       WHERE    科目.科目番号 = 実習課題.科目番号）
```

外側の質問のデータが内側の質問で条件として使われている
＝相関を有する

- NOT EXISTS は、集合が空の時に真になる述語
- 各科目ごとに内側の処理（その科目番号の実習課題があるかを調べる）を繰り返す必要がある

相関を有する入れ子型質問の例

```
SELECT    科目番号, 科目名
FROM      科目
WHERE     NOT EXISTS
    （SELECT   ＊
      FROM     実習課題
      WHERE    科目.科目番号 = 実習課題.科目番号）
```

科目

科目番号	科目名	単位数	
001	データベース	2	×
002	グラフィックス	2	○
003	プログラミング	2	×

実習課題

科目番号	課題番号	課題名
001	01	SQL
003	01	プログラム基礎
003	02	プログラム応用

各科目について、結合される実習課題があるかを判定
（内側の問い合わせは3回実行される）

SQLによる問い合わせの記述

- SQLの基本的な書き方
- 条件（WHERE）の書き方
- 出力（SELECT）の書き方
- 順序付け（ORDER BY）
- グループ表（GROUP BY）
- 結合（JOIN）
- 集合演算
- 副問い合わせ（入れ子型質問）

SQLの書き方のヒント(1)

- まずは、問題文の意味(どのようなデータを出力する必要があるのか)を、きちんと理解する
- SQL全体が、どのような枠組みになるか(どの機能を使う必要があるか)を考える
  - 結合、集約関数、GROUP BY、GROUP BY＋HAVING、自己結合、相関のある／ない入れ子型質問
- 必要なテーブルを全て、FROM節に書く
- 問題の条件に従って、WHERE節、GROUP BY節、HAVING節 などを記述
- 出力するべき属性(or集約関数)を、きちんとSELECT節に書く

SQLの書き方のヒント(2)

- 結合
  - 複数のテーブルを使用する場合は、結合が必要

例.学生番号 00100 の学生が履修した科目の科目番号、科目名、成績の一覧

- 出力の科目名は科目の属性、成績は履修の属性なので、両者の結合が必要(科目番号で結合)

```
SELECT   科目番号, 科目名, 成績
FROM     科目, 履修
WHERE    科目.科目番号 = 履修.科目番号  AND
         学生番号 = '00100'
```

SQLの書き方のヒント(3)

- 集約関数
  - データ数・合計値・平均値・最小値・最大値、などを出力する場合は、集約関数が必要

例.科目番号 001 の受講者数、平均点、最小点、最高点

```
SELECT   COUNT(＊), AVG(成績), MIN(成績), MAX(成績)
FROM     履修
WHERE    科目番号 = ‘001’
```

SQLの書き方のヒント(4)

- GROUP BY
  - ・・・ごとに出力、のように、グループごとの値を出力する場合は、GROUP BY ＋ 集約関数が必要
    - SELECT節には、集約に使った属性か集約関数

例. 全科目の科目番号と平均点の一覧

- 問題文から、各科目ごとの値を出力する必要があることが分かる

```
SELECT    科目番号, AVG(成績)
FROM      履修
GROUP BY  科目番号
```

SQLの書き方のヒント(5)

- GROUP BY ＋ HAVING
  - GROUP BY に加えて、出力するグループに条件が指定されていれば、HAVING が必要
    - グループに関する条件は HAVING 節に、データに関する条件は WHERE 節に書く

例. 履修者が30名以上の科目の
科目番号、履修者数、平均点の一覧

```
SELECT    科目番号, COUNT(＊), AVG(成績)
FROM      履修
GROUP BY  科目番号
HAVING    COUNT(＊) >= 30
```

SQLの書き方のヒント(6)

- 自己結合 or 相関のない入れ子型質問
  - 検索結果のデータを、さらに条件として使用する場合は、結合 or 相関のない入れ子型質問として書くことになる
    - どちらの書き方を使っても書ける
    - ただし、内側の質問について集約演算が必要な場合は、入れ子型質問として書く必要がある
      - 自己結合だけでは書けない
  - 先ほどの例のように、どちらの書き方でも書ける

SQLの書き方のヒント(7)

- 相関のある入れ子型質問
  - 各データごとに、別の問い合わせを使った条件が必要であれば、相関のある入れ子型質問になる

例. 実習課題のない科目の科目番号、科目名

- 各科目に対して、実習課題の有無を、別の問い合わせを使って判断する必要がある

```
SELECT    科目番号, 科目名
FROM      科目
WHERE     NOT EXISTS
    (SELECT   *
     FROM     実習課題
     WHERE    科目.科目番号 = 実習課題.科目番号)
```

SQLの書き方のヒント(8)

- よくある間違いに注意
- 複数のテーブルを用いる場合は結合が必須
- GROUP BY を用いる場合は、SELECT節には、GROUP BY に使った属性か、集約関数しか書けない
- WHERE節には、集約関数は書かない
- HAVINGには、集約関数を使った条件しか書けない

データベース演習課題

PostgreSQLによるSQL演習（宿題）

- 前回の演習が終わっていない人は、まず、終わらせる
- Moodleから演習内容が書かれたテキストファイルをダウンロード
- 演習課題のSQLを考えて、実行し、正しいデータが出力されることを確認する
- 演習のテキストファイルに、前回と同様、回答を貼り付けて、Moodleから提出
- 提出締め切り　7/9（月）18:00（厳守）

まとめ

- 前回の復習
- SQLによる問い合わせの記述方法
  - 結合（JOIN）
  - 集合演算
  - 副問い合わせ（入れ子型質問）
  - SQLによる問い合わせの記述の考え方
- データベース演習課題（宿題）

次回予告

- リレーションスキーマの設計
- 正規化の必要性
  - 更新不整合
  - 情報無損失分解と情報損失分解
- 多値従属性と関数従属性

  - 何らかのデータベースを作成するときに、リレーションスキーマを設計するための考え方