データベースS

第4回　データベース言語SQL（1）

システム創成情報工学科　尾下 真樹

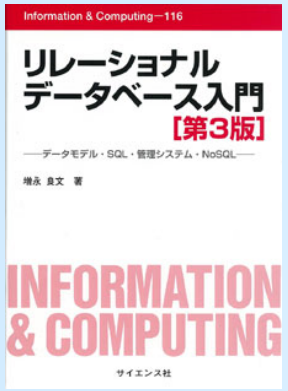
2018年度 Q2

今日の内容

- 前回の復習
- SQLの概要
- SQLによる問い合わせの記述方法
  - 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）

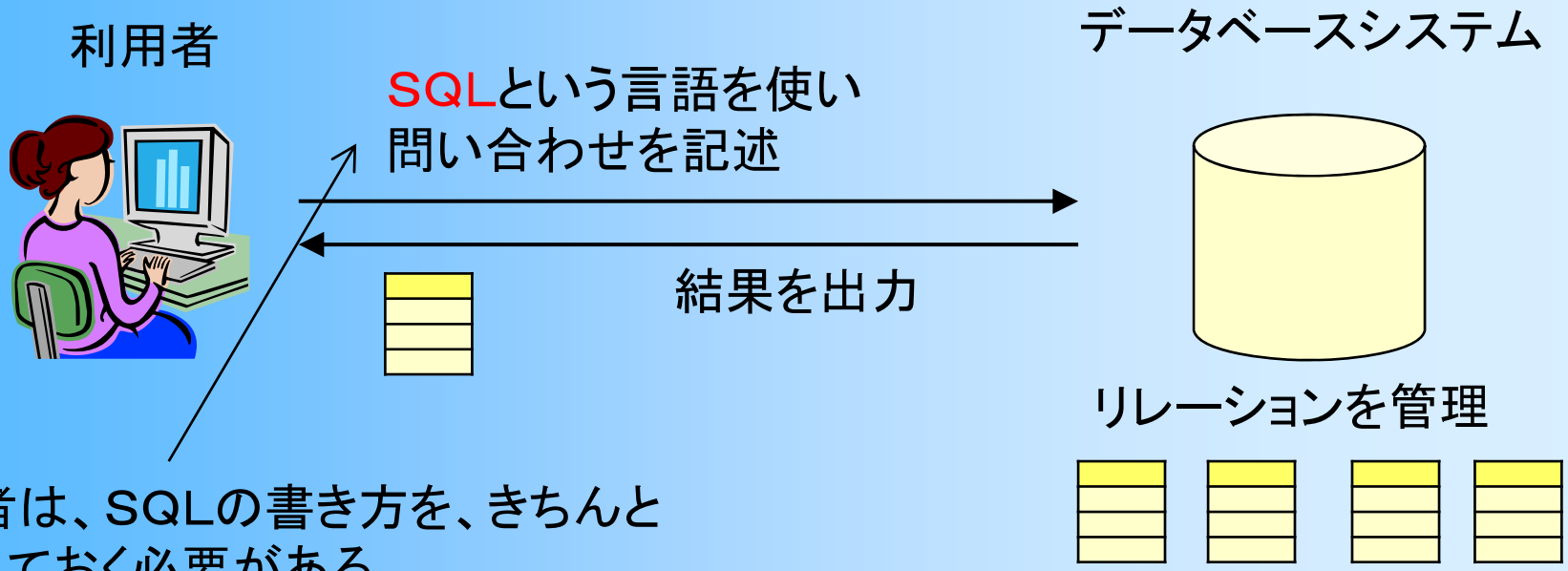
教科書

- 「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（5.1～5.4節、5.7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、教科書も参考にすると良い



前回の復習

SQLとリレーション操作の関係



利用者は、SQLの書き方を、きちんと理解しておく必要がある

システムが内部で自動的に行ってくれるので、全く知らなくても使えるが、専門的に使うのであれば、理解が必要

問い合わせが行われたら、リレーション操作を行って、結果を求める
（リレーショナル代数式・
リレーショナル論理式）

リレーショナル代数

- リレーショナル代数・代数式
  - 基本的なリレーションの演算子を定義
  - 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
- 代表的なリレーショナル代数演算子
  - 選択　σ
    - 必要なデータ（表の行）を取り出すために使用
  - 射影　π
    - 必要な属性（表の列）を取り出すために使用
  - 結合　$\bowtie$
    - 複数のリレーションを組み合わせるために使用

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）
  - 和集合、差集合、共通集合、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式の例

- 科目番号002の履修者の学生番号と成績の一覧

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

- 学生番号00001の学生が履修した科目名と成績の一覧

$$\pi_{科目名,成績}(科目 \bowtie (\sigma_{学生番号='0001'}履修))$$

- 履修者が一人もいない科目の科目番号と科目名の一覧

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

演習問題

- リレーショナル代数式
  - 基本的な問い合わせが書けるように、間違えた人は、きちんと理解・復習しておくこと
  - そもそも演算子がよく理解できなかった人は、理解・復習しておくこと
  - 演算子の定義自体は簡単なので、あとは、きちんと理論的に考えられるかどうかが重要（プログラミングと同じ）

SQLの概要

SQL

- リレーショナルデータベース言語SQL
  - SQL（エスキューエル）
  - データベースの操作、特に問い合わせを行うための言語
  - 使いやすい
  - リレーショナル代数式、論理式などよりも記述が簡単・高機能
    - 代数演算はリレーショナルモデルの操作を規定するもの（利用者が直接使用することはあまりない）
    - SQLはデータベースのインターフェース（具体的な操作ではなく、操作の目的を記述する。）

SQLとリレーション操作の関係（復習）

利用者

SQLという言語を使い
問い合わせを記述

結果を出力

データベースシステム

リレーションを管理

利用者は、SQLの書き方を、きちんと
理解しておく必要がある

システムが内部で自動的に行ってくれる
ので、全く知らなくても使えるが、専門的
に使うのであれば、理解が必要

問い合わせが行われたら、
リレーション操作を行って、
結果を求める
（リレーショナル代数式・
リレーショナル論理式）

リレーショナル代数 と SQL

- リレーショナル代数式
  - どのような処理でデータを出力するか（How）
- SQL
  - どのような条件のデータを出力したいか（What）
- SQLはリレーショナル完備
  - リレーショナル代数式で記述できる問い合わせは全て記述できる
  - SQLでしか記述できないような問い合わせもある

SQLによる問い合わせの例

学生（学生番号, 氏名）
科目（科目番号, 科目名, 担当教官）
履修（学生番号, 科目番号, 成績）

「データベース」を履修している全学生の氏名と成績を出力

```
SELECT   氏名, 成績
FROM     学生, 科目, 履修
WHERE    科目.科目名 = ‘データベース’
          AND 履修.科目番号 = 科目.科目番号
          AND 学生.学生番号 = 履修.学生番号
```

SQLの歴史(1)

- コッドがリレーショナルモデルを提案(1970)
- さまざまなリレーショナルデータベースシステムの開発(70年代)
  - IBM, System R
    - 問い合わせ言語 SEQUEL（Structured English Query Language ）(シークエル)
  - カリフォルニア大学バークレー校, INGRES
    - 問い合わせ言語 QUEL(Query Language)(クエル)
- 商用のシステムが登場(80年代)

SQLの歴史(2)

- データベース言語の標準化の動き(80年代初頭)
- SQL標準規格の制定(1987)
  - 国際規格 ISO 9075、日本工業規格 JIS X 3005
  - IBMのSEQUELがベース
- 規格の改定
  - SQL-92 (SQL2) (1992)が現在普及
  - SQL:1999(SQL3)(1999)はオブジェクト指向拡張
  - SQL:2003、SQL:2008、SQL:2011 SQL:2016

SQLの基本概念

- SQLでは、リレーションを表（テーブル）として扱う
  - 属性→列、インスタンス→行、リレーション→表
  - リレーションと表の主な違い
    - 重複した行の存在を許す
    - 順序が意味を持つ
  - リレーションよりも表の方が便利な時がある
    - 詳しくは後で説明

履修

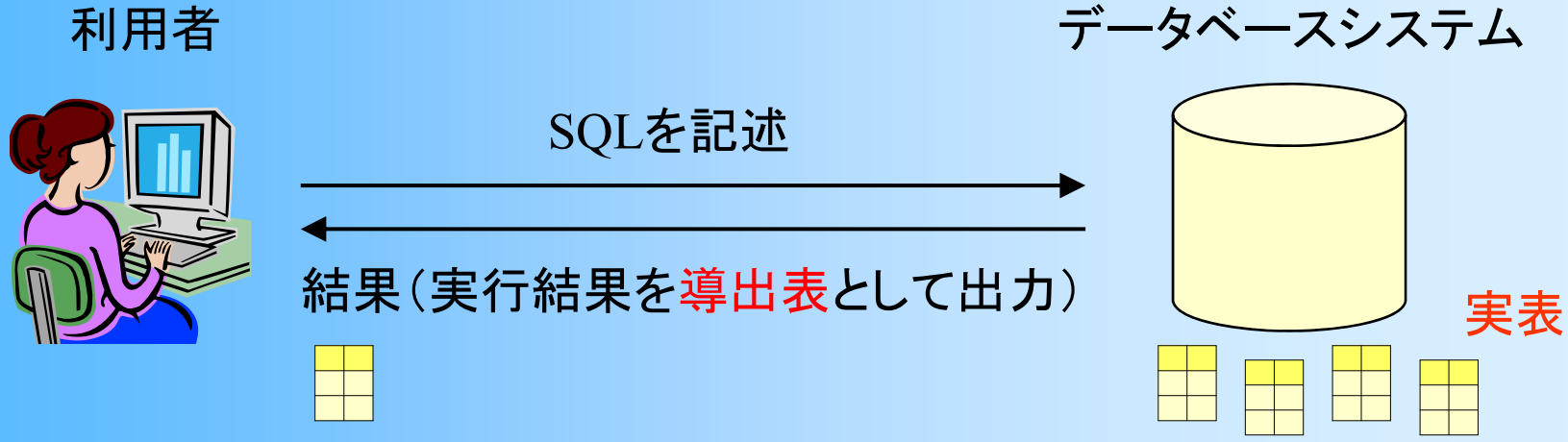
科目番号	学生番号	成績
01	0123001	60
03	0123002	80
01	0123003	70

表の種類

- 実表（base table）
  - データベース内のリレーション
  - 直接更新することができる
- 導出表（derived table）
  - 問い合わせの結果としてできる表
  - 基本的に直接更新することはできない
- ビュー表（viewed table）
  - 実装表を何らかの条件で表を再構成したもの
  - 基本的に直接更新することはできない
  - （本講義・演習では、扱わない）

SQLによる問い合わせ

- データベース内の表（実表）から、どのようなデータを出力するかをSQLとして記述する
- データベースシステムは、SQLで記述された条件に一致するデータを検索し、検索結果を表（導出表）として出力する



SQLを使ってできること

- 問い合わせ（検索）
  - データベースから必要な情報を取り出す
    ＝ 実表をもとに導出表を作成する
- リレーション（表）の生成
- データ（行）の挿入
- データ（行）の削除
- データ（行）の更新
  - これらについては、次回、説明する。

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
…	…	…

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)

単純な質問の例(1)

Q.科目番号001の履修者の学生番号と成績の一覧を出力

```
SELECT    履修.学生番号, 履修.成績
FROM      履修
WHERE     履修.科目番号 = '001'
```

- 表名が一意に決まる時は表名は省略できる

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = '001'
```

単純な質問の例(1)

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = '001'
```

履修

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

出力される表

学生番号	成績
00001	90
00002	80

① FROM節に書かれた表のうち、WHERE節の条件をみたすデータ(行)が出力対象となる

② SELECT節に書かれた属性(列)が出力対象となる

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）

条件（WHERE）の書き方

- 属性に関する条件を記述できる
  - =, <=, >=
  - 複数の条件を AND や OR で組み合わせることもできる
  - NOT で条件を反転することもできる
  - 条件文に文字列を使うときは、クォート’ ’でくくる
    - 成績 = 70
    - 氏名 = ’ODA NOBUNAGA’
    - 氏名 = N’織田 信長’（日本語文字列の場合はNをつける）
    - 学生番号 = ’00100’（学生番号を文字列として扱うとき）

単純な質問の例(2)

- ANDを使う例

Q.学生番号00100の科目番号005の成績

```
SELECT    成績
FROM      履修
WHERE     学生番号='00100' AND 科目番号 = '005'
```

- WHERE節を省略した例

Q. 全科目の科目名と単位の一覧

```
SELECT    科目名, 単位数
FROM      科目
```

述語

- 条件を記述するために使える述語
  - BETWEEN x AND y ・・・ 属性値の範囲を指定
  - IN t ・・・ 表 t に含まれるかどうかを判定
  - LIKE x ・・・ 文字列の部分一致、x を含むか
  - NULL ・・・ 空値かどうかを判定
  - EXISTS t ・・・表 t が空集合でない場合に真

  - EXISTS の例は後述

述語の例

- Q. 科目番号001の科目で80点から90点の成績の学生番号の一覧を出力

```
SELECT    学生番号
FROM      履修
WHERE     科目番号 = ‘001’ AND
          成績 BETWEEN 80 AND 90
```

```
SELECT    学生番号
FROM      履修
WHERE     科目番号 = ‘001’ AND
          成績 >= 80 AND 成績 <= 90
```

※ どちらでも同じ（どちらの書き方でも良い）

述語の例

- Q. 名前に「子」の文字を含む学生の学生番号と氏名の一覧を出力

```
SELECT    学生番号, 氏名
FROM      学生
WHERE     氏名 LIKE '子'
```

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）

出力（SELECT）の書き方

- 基本的には、出力したい属性を列挙

```
SELECT    学生番号, 成績
FROM      履修
WHERE     科目番号 = ‘001’
```

- 特殊な SELECT節の書き方
  - 全属性を出力（＊）
  - 重複の除去（DISTINCT）
  - 集約関数（COUNT, SUM, AVG, MAX, MIN, etc.）

全属性を出力

- 指定した表の全部の列を問い合わせ結果に含めたい場合は、＊で省略できる

Q.単位数が3単位以上の科目の科目番号、科目名、単位数の一覧を出力

- 科目（科目番号、科目名、単位数）

```
SELECT    ＊
FROM      科目
WHERE     単位数 >= 3
```

重複の除去（DISTINCT）

- DISTINCT指定
  - 重複した行を除去するための指定
  - SQLで扱うのは、リレーション（集合）ではなく表なので、重複は自動的には除去されない

Q. 全科目の科目名と単位の一覧を出力

- 同じ科目名で科目番号が異なるデータが存在する場合、通常は同じ科目名のデータが複数出力されるが、DISTINCT指定をすると重複を除去

```
SELECT   DISTINCT 科目名, 単位数
FROM     科目
```

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'
```

集約関数の例（AVG, MIN, MAX）

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’
```

履修

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

WHERE節の条件を満たすデータ

成績	成績	成績
90	90	90
80	80	80

集約演算

AVG（成績）	MIN（成績）	MAX（成績）
85	80	90

集約関数の例（COUNT）

Q. 科目番号 001 の履修者の人数を出力

```
SELECT   COUNT(＊)
FROM     履修
WHERE    科目番号 = ‘001’
```

- COUNTは、出力されるデータの属性ではなく、データの個数を計算する、特殊な集約関数
  - 属性を指定する意味はないので、COUNT関数の引数には、常に、全属性を表す＊を記述
- 上の例では、履修データの個数が履修人数を表すと考えられるため、COUNTを用いる

集約関数の例（COUNT）

- COUNTと他の集約関数の違いに注意

履修

科目番号	学生番号	成績
001	00001	90
001	00002	80
002	00001	90
002	000[illegible]	70

属性値の合計なので「170」

データ数の合計なので「2」

```
SELECT   SUM（成績）
FROM     履修
WHERE    科目番号 = ‘001’
```

```
SELECT   COUNT（＊）
FROM     履修
WHERE    科目番号 = ‘001’
```

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）

順序付け（ORDER BY）

- 指定した属性によりデータの順番を並び替えて出力

Q.科目番号 005 を履修した学生の学生番号と成績の一覧を、成績の良い順番に出力

```
SELECT      学生番号, 成績
FROM        履修
WHERE       科目番号 = '005'
ORDER BY    成績 DESC
```

- ASC（ascendant）・・・昇順、小から大へ
- DESC（descendant）・・・降順、大から小へ

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）

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
```

グループ表 ＋ グループ選択 の例

- グループ表の例

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

WHERE

GROUP BY

条件を満たすグループのデータのみ出力

科目番号	COUNT	AVG(成績)
001	3	70
002	4	75
…		…

HAVING

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

WHEREとHAVINGの使い分け

- HAVINGを使う例

  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
```

  – 「履修者が30名以上の科目」はグループに関する条件なので、HAVING節に記述

WHEREとHAVINGの使い分け

- WHEREを使う例

  Q. 各科目の、科目番号と60点以上の成績をとった履修者の数（※ 0名の科目は出力しなくて良い）

```
SELECT      科目番号, COUNT(＊)
FROM        履修
WHERE       成績 >= 60
GROUP BY    科目番号
```

  – 「60点以上の成績をとった履修者」はデータに関する条件なので、WHERE節に記述

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）

まとめ

- 前回の復習
- SQLの概要
- SQLによる問い合わせの記述方法
  - 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）

次回予告

- 次回（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