

令和6年度 高校教諭対象 情報Ⅰ対応プログラミング等研修

情報Ⅰ データベース入門

令和6年8月

KCS 鹿児島情報専門学校

目次

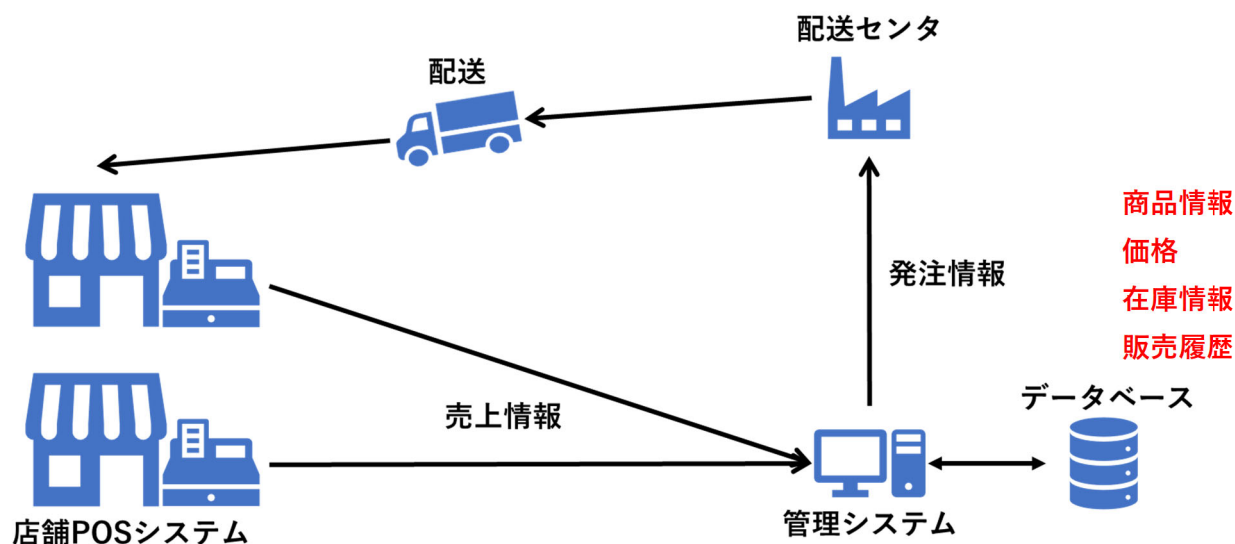
1	データベースの概要	1
(ア)	データベース (DB) とは	1
(イ)	データベースの技術を学ぶ理由	1
2	データベースの基礎	2
(ア)	データベースの種類	2
(イ)	基本用語	2
(ウ)	データ操作の種類	3
(エ)	SQL 文の種類	3
(オ)	データベースの作成	2
(カ)	テーブルの作成	3
3	データベースの利用 1	5
(ア)	レコードの追加	5
(イ)	レコードの更新	7
(ウ)	レコードの削除	7
4	データベースの利用 2	8
(ア)	レコードの選択 (データの検索)	8
(イ)	様々な検索 (WHERE 句での指定)	9
①	範囲指定	9
②	LIKE 検索	10
③	整列	11
④	グループ化	12
⑤	テーブルの結合	13
5	データベースの応用	15
(ア)	データの整理	15
(イ)	テーブルの正規化	15
6	追加実習	17
(ア)	問題	17

1 データベースの概要

(ア) データベース (DB) とは

目的に応じた大量のデータを蓄積し、簡単に効率的に検索など利用できるようにデータの利用価値を高めたものです。データベース (DB) は身近の様々な情報システムやアプリケーションで活用されています。例えば、コンビニでの POS システムや学校の学生情報管理システムがあります。

【例：コンビニでの POS システム】



データをリアルタイムで管理することで店舗は適切な在庫管理をおこない、売上を最大化できる
また、販売履歴の分析により、販売トレンドを把握し、マーケティング戦略を策定可能

【演習：データベースが使われている身近な例をあげてみましょう】

図書管理システム
ホテルやチケットなどの予約システム
銀行の ATM システム
オンラインショッピングの商品・顧客管理システム など

(イ) データベースの技術を学ぶ理由

データベースはあらゆる情報システムの基盤となっており、なくてはならない技術となっています。また、近年では IoT やビッグデータ、AI (人工知能) など扱うデータ量が増大し複雑さも増えています。そして、そのデータをいかに効率良く管理し活用していくかが重要となっています。より良いシステムやアプリケーションを構築するには、高いプログラミング力だけでなく、データを管理するデータベースを理解し活用することも求められるということです。

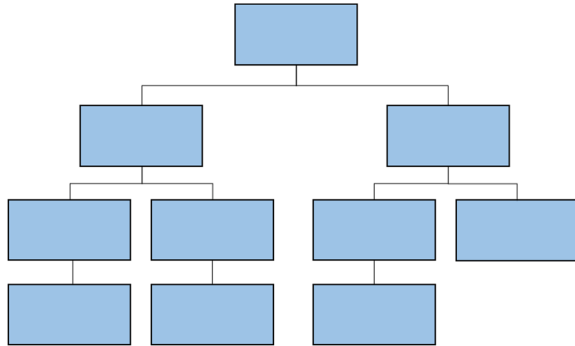
データベースの知識や技術はシステムエンジニアに特化したものではありません。どうすればデータを効率良く管理できるか、データをどう扱えば良いのかなど様々な業種や職種で活用することができます。

2 データベースの基礎

(ア) データベースの種類

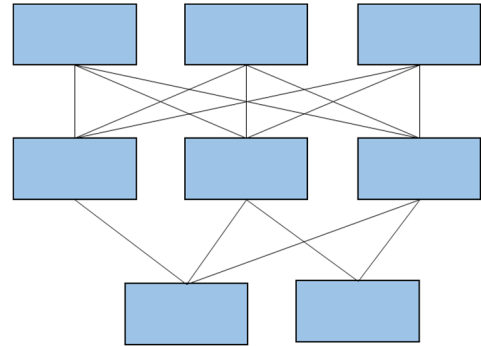
①階層型データベース

個々のデータ同士のつながりが階層構造をなすデータベースです。



②ネットワーク型データベース

網目のように繋がっている構造をもつデータベースです。



③リレーショナル・データベース

データを内容ごとに分類したテーブルで管理するデータベースです。テーブルは2次元の表構造で構成されます。大量のデータを複数の表で効率良く管理します。

		リレーションシップ					
		カラム					
レコード	学生番号	氏名	クラス	国語	数学	英語	クラス 担任
	20240101	荒井 陽一朗	S1A1	78	92	83	S1A1 岩井
	20240102	石山 順子	S1A1	74	94	24	S1A2 吉田
	20240103	浦山 敏哉	S1A2	94	98	61	S1A3 川畑
	20240104	江頭 めぐみ	S1A2	92	71	73	
	20240105	大石 理香	S1A3	13	100	82	
	20240106	大垣 千尋	S1A3	72	22	65	

(イ) 基本用語

- ① **テーブル** … データを内容ごとに分類した表のことです。
- ② **レコード** … 1件のデータのことです。
- ③ **カラム** … 列のことです。
- ④ **主キー** … 1件のレコードを特定することが可能な列です。
- ⑤ **外部キー** … 別の表の主キーを参照する列です。参照先に参照データが存在する必要があります。
- ⑥ **SQL文** … データベースのテーブルを操作するための言語です。

(ウ) データ操作の種類

- ① **選 択** … 条件を満たすレコードのみを抽出して新たなテーブルとして表示する。
- ② **射 影** … テーブルから一部のカラムを取り出し新たなテーブルとして表示する。
- ③ **結 合** … 複数のテーブルを共通するカラムのリレーションシップ（関係）に基づいて結合し新たなテーブルとして表示する。

【元のテーブル】

学生番号	氏名	クラス	国語	数学	英語	クラス	担任
20240101	荒井 陽一朗	S1A1	78	92	83	S1A1	岩井
20240102	石山 順子	S1A1	74	94	24	S1A2	吉田
20240103	浦山 敏哉	S1A2	94	98	61	S1A3	川畑
20240104	江頭 めぐみ	S1A2	92	71	73		
20240105	大石 理香	S1A3	13	100	82		
20240106	大垣 千尋	S1A3	72	22	65		

【 **選択** 】

学生番号	氏名	クラス	国語	数学	英語
20240103	浦山 敏哉	S1A2	94	98	61
20240104	江頭 めぐみ	S1A2	92	71	73

【 **射影** 】

学生番号	クラス
20240101	S1A1
20240102	S1A1
20240103	S1A2
20240104	S1A2
20240105	S1A3
20240106	S1A3

【 **結合** 】

学生番号	氏名	クラス	国語	数学	英語	クラス	担任
20240101	荒井 陽一朗	S1A1	78	92	83	S1A1	岩井
20240102	石山 順子	S1A1	74	94	24	S1A1	岩井
20240103	浦山 敏哉	S1A2	94	98	61	S1A2	吉田
20240104	江頭 めぐみ	S1A2	92	71	73	S1A2	吉田
20240105	大石 理香	S1A3	13	100	82	S1A3	川畑
20240106	大垣 千尋	S1A3	72	22	65	S1A3	川畑

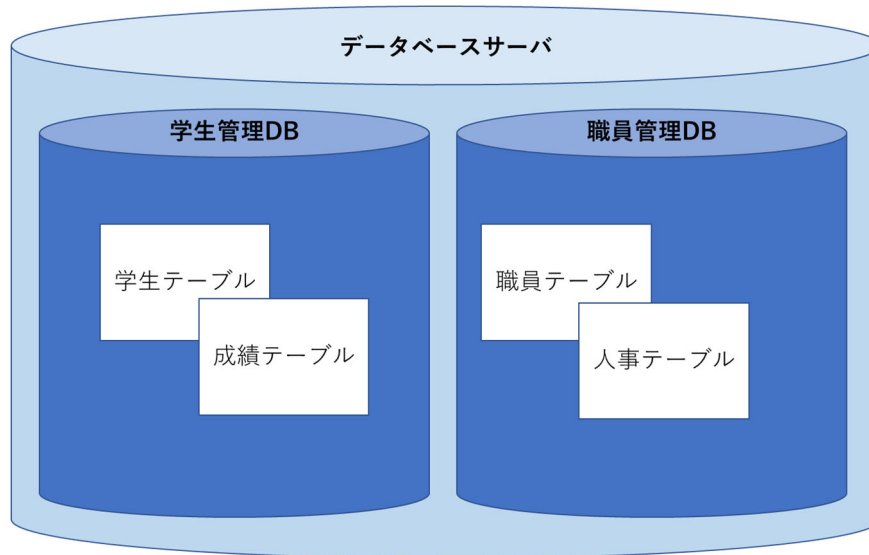
(エ) SQL 文の種類

- ① CREATE DATABASE 文 … データベースを作成します。
- ② CREATE TABLE 文 … テーブルを作成します。
- ③ INSERT 文 … テーブルに1件のレコードを追加します。
- ④ SELECT 文 … テーブルからレコード取得します。
- ⑤ UPDATE 文 … テーブルのレコードの内容を更新します。
- ⑥ DELETE 文 … テーブルのレコードを削除します。

(オ) データベースの作成

データベースのイメージは下記のように、データベースサーバの中に複数の目的別のデータベースが存在しその中に複数のテーブルが格納されます。そのため、1 番最初にデータベースの作成が必要になります。

【イメージ】



【書式】

CREATE DATABASE データベース名 ;

【実習：以下の SQL 文を実行してデータベースを作成と操作するデータベースを選択する】

```
CREATE DATABASE std_db ;  
USE std_db;
```

※2 行目のコマンドは MariaDB や MySQL を使用する場合に必要となるコマンドです。
データベースを操作する場合、最初にどのデータベースを操作するかを指定します。

(カ) テーブルの作成

テーブルを作成する場合は CREATE TABLE 文を使用します。その際、カラム名やそのカラムに格納するデータの型や制約（列制約、表制約）指定する必要があります。

【書式】

```
CREATE TABLE テーブル名 (  
    カラム名 データ型 列制約,  
    カラム名 データ型 列制約, ...  
    表制約  
);
```

【データ型】 ※一部

INT	… 数値
CHAR	… 固定長文字列
VARCHAR	… 可変長文字列
DATE	… 日付

【制約】

PRIMARY KEY	… 主キー
FOREIGN KEY	… 外部キー（表制約で使用）
REFERENCES	… 外部キー（表制約、列制約で使用）
NOT NULL	… NULL 値の格納禁止
UNIQUE	… 一意の値のみ格納可能

【実習：以下の構造を持つテーブルを作成する】

テーブル名：class_tbl

カラム名	データ型	備考
classid	CHAR(4)	主キー
teachername	VARCHAR(50)	NOT NULL

```
CREATE TABLE class_tbl(  
    classid char(4) primary key,  
    teachername varchar(50) not null  
);
```

テーブル名：std_tbl

カラム名	データ型	備考
stdno	INT (8)	主キー
stdname	VARCHAR(50)	NOT NULL
classid	CHAR(4)	外部キー
japanese	INT(3)	
math	INT(3)	
english	INT(3)	

```
CREATE TABLE std_tbl(  
    stdNo int(8) primary key,  
    stdName varchar(50) not null,  
    classid char(4) not null references class_tbl(classid),  
    japanese int(3),  
    math int(3),  
    english int(3)  
);
```

3 データベースの利用 1

(ア) レコードの追加

【書式】

INSERT INTO テーブル名 (カラム名, カラム名…) VALUES (値1, 値2, …);

【例】

INSERT INTO class_tbl(classid,teachername) VALUES('S1A1','岩井');

**INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES (20240101, '荒井 陽一朗', 'S1A1', 78, 92, 83);**

【実習：以下のデータをクラステーブル (class_tbl) に追加する】

クラス ID	担任名
S1A1	岩井
S1A2	吉田
S1A3	家永
J1A1	岩井
J1A2	吉田
J1A3	家永

INSERT INTO class_tbl(classid,teachername) VALUES('S1A1','岩井');
INSERT INTO class_tbl(classid,teachername) VALUES('S1A2','吉田');
INSERT INTO class_tbl(classid,teachername) VALUES('S1A3','家永');
INSERT INTO class_tbl(classid,teachername) VALUES('J1A1','岩井');
INSERT INTO class_tbl(classid,teachername) VALUES('J1A2','吉田');
INSERT INTO class_tbl(classid,teachername) VALUES('J1A3','家永');

【実習：以下のデータを学生テーブル（std_tbl）に追加する】

学生番号	氏名	クラス	国語	数学	英語
20240102	石山 順子	S1A1	74	94	24
20240103	浦山 敏哉	S1A2	94	98	61
20240104	江頭 めぐみ	S1A2	92	71	73
20240105	大石 理香	S1A3	13	100	82
20240106	大垣 千尋	S1A3	72	22	65

※氏名はダミーデータ生成ツールで出力したデータを使用しています。

```
INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES(20240102,'石山 順子','S1A1',74,94,24);
```

```
INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES(20240103,'浦山 敏哉','S1A2',94,98,61);
```

```
INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES(20240104,'江頭 めぐみ','S1A2',92,71,73);
```

```
INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES(20240105,'大石 理香','S1A3',13,100,82);
```

```
INSERT INTO std_tbl (stdno, stdname, classid, japanese, math, english)
VALUES(20240106,'大垣 千尋','S1A3',72,22,65);
```

(イ) レコードの更新

【書式】

UPDATE テーブル名 SET カラム名 = 値 WHERE 条件 ;

【実習：以下の赤字太字の部分のデータを「60」に更新する】

学生番号	氏名	クラス	国語	数学	英語
20240102	石山 順子	S1A1	74	94	24
20240103	浦山 敏哉	S1A2	94	98	61
20240104	江頭 めぐみ	S1A2	92	71	73
20240105	大石 理香	S1A3	13	100	82
20240106	大垣 千尋	S1A3	72	22	65

UPDATE std_tbl SET english = 60 WHERE stdno = 20240102;

UPDATE std_tbl SET japanese = 60 WHERE stdno = 20240105;

UPDATE std_tbl SET math = 60 WHERE stdno = 20240106;

(ウ) レコードの削除

【書式】

DELETE FROM テーブル名 WHERE 条件 ;

※WHERE 句で条件を指定しなければ全行を削除することになるので注意

【実習：以下のデータを学生テーブル（std_tbl）から削除する】

学生番号	氏名	クラス	国語	数学	英語
20240104	江頭 めぐみ	S1A2	92	71	73

DELETE FROM std_tbl WHERE stdno = 20240104;

4 データベースの利用 2

(ア) レコードの選択 (データの検索)

【書式】

```
SELECT   カラム名  |   *  
FROM     テーブル名  
WHERE    行の制限の条件  
GROUP BY グループ化するカラム名  
HAVING   グループの制限の条件      (※今回は記載なし)  
ORDER BY カラム名  ASC | DESC      (ASC:昇順、DESC:降順)  
;
```

【実習：std_tbl から全行全列のデータを取得する】

```
SELECT * FROM std_tbl;
```

【実習：std_tbl から学生番号が「20240103」の学生番号と氏名を取得する】

```
SELECT stdno,stdname FROM std_tbl WHERE stdno = 20240103;
```

【実習：国語 (japanese) が 60 点の学生番号と氏名、国語の点数を取得する】

```
SELECT stdno,stdname,japanese FROM std_tbl WHERE japanese = 60;
```

【実習：英語 (english) が 80 点以上の学生番号と氏名、英語の点数を取得する】

```
SELECT stdno,stdname,english FROM std_tbl WHERE english >=80;
```

【実習：数学（math）が 50 点未満の学生番号と氏名、数学の点数を取得する】

```
SELECT stdno,stdname,math FROM std_tbl WHERE math < 50;
```

【実習：クラスが「S1A1」の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE classid = 'S1A1';
```

【実習：クラスが「S1A1」または「J1A1」クラスの全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE classid = 'S1A1' OR classid = 'J1A1';
```

【実習：クラスが「S1A2」で、数学（math）が 70 点以上の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE classid = 'S1A2' AND math >= 70;
```

（イ）様々な検索（WHERE 句での指定）

① 範囲指定

指定した範囲にカラムの値が含まれているデータを取得します。

範囲は、「値 1 以上、値 2 以下」となります。

【書式】

```
カラム名 BETWEEN 値 1 AND 値 2
```

【実習：国語の点数が 70 点以上 89 点以下の学生番号と氏名、国語の点数を取得する】

```
SELECT stdno,stdname,japanese  
FROM std_tbl  
WHERE japanese BETWEEN 70 AND 89;
```

【実習：数学の点数が 60 点以上 70 点未満の学生番号と氏名、数学の点数を取得する】

```
SELECT stdno,stdname,math FROM std_tbl  
WHERE math BETWEEN 60 AND 69;  
  
SELECT stdno,stdname,math FROM std_tbl  
WHERE math >= 60 AND math < 70;
```

② LIKE 検索

LIKE 句を使用することで、あいまい検索を行うことができます。あいまい検索とは、指定した文字列が含まれるかどうかの検索です。ワイルドカードを使用して任意の文字を表します。

・ワイルドカード

% … 0 文字以上の任意の文字 (パーセント)

— … 1 文字の任意の文字 (アンダースコア)

【書式】

カラム名 LIKE ‘文字列パターン’

【実習：氏名が「田」で始まる学生の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE stdname LIKE '田%';
```

【実習：氏名が「子」で終わる学生の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE stdname LIKE '%子';
```


【実習：氏名に「田」が含まれる学生の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE stdname LIKE '%田%';
```

【実習：氏名のうち3文字目が「木」の学生の全列のデータを取得する】 難問

```
SELECT * FROM std_tbl WHERE stdname LIKE '__木%';
```

※「_」は2つ並んでいます。

③ 整列

ORDER BY 句を使用することで、指定したカラムにおいて昇順（ASC）または降順（DESC）でデータを整列（ソート）することができます。整列順を省略すると自動的に昇順となります。

【書式】

```
ORDER BY カラム名 [ASC | DESC]
```

【実習：学生番号の昇順で全行全列のデータを取得する】

```
SELECT * FROM std_tbl ORDER BY stdno ASC;
```

※ASCのみ省略可能

```
SELECT * FROM std_tbl ORDER BY stdno;
```

【実習：数学の点数の降順で全行全列のデータを取得する】

```
SELECT * FROM std_tbl ORDER BY math DESC;
```

【実習：クラスが「S1A1」のデータを、英語の点数の降順で全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE classid = 'S1A1' ORDER BY english DESC;
```

【実習：国語の点数が 80 点以上の学生を国語の点数が高い順で全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE japanese > 80 ORDER BY japanese DESC;
```

④ グループ化

GROUP BY 句を使用することで、指定したカラムの値でグループを作りそのグループごとに 1 件のデータを取得します。また、同時にグループ関数を使用することでグループごとに集計処理を行うこともできます。

・グループ関数

COUNT()	… 指定したカラムの行数を集計します (NULL 値は除外)
AVG()	… 指定したカラムの平均を計算します
SUM()	… 指定したカラムの合計を計算します
MAX()	… 指定したカラムの最高値を計算します
MIN()	… 指定したカラムの最低値を計算します

【書式】

```
SELECT      カラム名  
FROM        テーブル名  
GROUP BY    グループ化するカラム名
```

【実習：全学生の数学の平均点のデータを取得する (グループ関数のみ)】

```
SELECT avg(math) FROM std_tbl;
```

【実習：全学生の英語の最高点のデータを取得する（グループ関数のみ）】

```
SELECT max(english) FROM std_tbl;
```

【実習：クラスごとの国語の平均点のデータ（クラス、平均点）を取得する】

```
SELECT classid,avg(japanese) FROM std_tbl GROUP BY classid;
```

【実習：クラスごとの数学の平均点を求め、平均点の降順に整列したデータを取得する】

```
SELECT classid,avg(math) FROM std_tbl GROUP BY classid  
ORDER BY avg(math) DESC;
```

⑤ テーブルの結合

2つ以上のテーブルを1つのテーブルにまとめることを結合と言います。リレーショナル・データベースでは複数のテーブルでデータを管理しその複数のテーブルを必要に応じて組み合わせてデータの参照や更新などデータ操作を行っています。

【書式】

```
SELECT   カラム名  
FROM     テーブル名1, テーブル名2  …  
WHERE    テーブル名1. カラム名  =   テーブル名2. カラム名  … ;
```

【結合の例】

学生テーブルとクラステーブルでは「クラス」というカラムにおいて共通する値を持っています（リレーションシップがある）。その共通する値を組み合わせることで2つの表を結合します。WHERE句で指定している条件は「結合条件」と呼びます。また、SQL文の実行結果として1つのテーブルになるだけで、学生テーブルとクラステーブルは別々に管理されたままです。

リレーションシップ

学生番号	氏名	クラス	国語	数学	英語	クラス	担任
20240101	荒井 陽一朗	S1A1	78	92	83	S1A1	岩井
20240102	石山 順子	S1A1	74	94	24	S1A2	吉田
20240103	浦山 敏哉	S1A2	94	98	61	S1A3	川畑
20240104	江頭 めぐみ	S1A2	92	71	73		
20240105	大石 理香	S1A3	13	100	82		
20240106	大垣 千尋	S1A3	72	22	65		

結合の SQL 文を実行すると下記のような結果を得られます。

学生番号	氏名	クラス	国語	数学	英語	クラス	担任
20240101	荒井 陽一朗	S1A1	78	92	83	S1A1	岩井
20240102	石山 順子	S1A1	74	94	24	S1A1	岩井
20240103	浦山 敏哉	S1A2	94	98	61	S1A2	吉田
20240104	江頭 めぐみ	S1A2	92	71	73	S1A2	吉田
20240105	大石 理香	S1A3	13	100	82	S1A3	川畑
20240106	大垣 千尋	S1A3	72	22	65	S1A3	川畑

【実習：以下の SQL の文を実行してテーブルの結合を行います】

SELECT * FROM std_tbl,class_tbl WHERE std_tbl.classid = class_tbl.classid;

5 データベースの応用

(ア) データの整理

データベースに蓄積されるデータは膨大であり、データ追加されるだけでなく既存のデータの更新や削除など様々なデータ操作が行われています。安定した情報システムやアプリケーションの運用のために、日々変化する膨大なデータを不整合（データの重複や余分なデータなど）なくデータを整理、管理する必要があります。そのためにデータの正規化を行います。

(イ) テーブルの正規化

正規化は複数の段階に分けて実施しますが、最初に行う正規化は「別の表に分ける」という作業を行います。別の表に分けるのは同じデータが複数回出現する部分に対して行います。その際、分けた表とのリレーションシップを保持するために外部キーを設定する必要があります。

参考までに、正規化は第1～第3正規化と続きます。下記の例では第3正規化を行っています。

【例：正規化前】

学生番号	氏名	クラス	担任	国語	数学	英語
20240101	荒井 陽一朗	S1A1	岩井	78	92	83
20240102	石山 順子	S1A1	岩井	74	94	24
20240103	浦山 敏哉	S1A2	吉田	84	88	61
20240104	江頭 めぐみ	S1A2	吉田	79	85	58
20240105	大石 理香	S1A3	川畑	13	100	82
20240106	大垣 千尋	S1A3	川畑	72	22	65

重複しているデータ

「担任」というカラムにおいて同じデータが複数回出現しています。その状態の問題点は、担任名に変更があった場合、データの件数分だけ SQL 文を実行する必要があるということです。100 件重複していれば 100 件の SQL 文が必要となります。その際、変更忘れがあるとデータの不整合に繋がります。

【例：正規化後】

リレーションシップ



学生番号	氏名	クラス	国語	数学	英語
20240101	荒井 陽一郎	S1A1	78	92	83
20240102	石山 順子	S1A1	74	94	24
20240103	浦山 敏哉	S1A2	94	98	61
20240104	江頭 めぐみ	S1A2	92	71	73
20240105	大石 理香	S1A3	13	100	82
20240106	大垣 千尋	S1A3	72	22	65

クラス	担任
S1A1	岩井
S1A2	吉田
S1A3	川畑

データの不整合を防ぐために正規化することでデータの重複をなくすることができました。担任名に変更があったとしても、右側の表の担任名を1件変更するSQL文だけですみます。さらに安定したデータベースを目指すのであれば「クラス」においてもコード化して別の表に分けることが望ましいです。

6 追加実習

(ア) 問題

追加で基礎的な実習を提示しています。データベースの操作は他のプログラミングなどと同じ様に数をこなすことが重要です。まずは基礎を反復練習してデータ操作に慣れることが重要です。一部の問題では今回の講義内容を超えた問題もあります。難易度の高い問題になっていますが是非挑戦してみてください。追加実習で使用するテーブルは以下のような構成になっています。

【テーブル構成】

クラステーブル

classid	teachername
J1A1	岩井
J1A2	吉田
J1A3	川畑
S1A1	岩井
S1A2	吉田
S1A3	川畑

学生テーブル

stdno	stdname	classid	japanese	math	english
20240101	荒井 陽一朗	S1A1	78	92	83
20240102	石山 順子	S1A1	74	94	24
20240103	浦山 敏哉	S1A2	94	98	61
20240104	江頭 めぐみ	S1A2	92	71	73
20240105	大石 理香	S1A3	13	100	82
20240106	大垣 千尋	S1A3	72	22	65
20240107	大野 敦史	J1A3	100	36	37
省略					
20240149	山田 直紀	S1A3	48	30	45
20240150	吉岡 恵	J1A3	29	63	39

※全50件

資格テーブル

certid	certname
1	基本情報技術者試験
2	ITパスポート試験
3	セキュリティマネジメント試験
4	応用情報技術者試験
5	情報処理安全確保支援士
6	データベーススペシャリスト試験
7	ネットワークスペシャリスト試験

資格管理テーブル

stdno	certid	passDate
20240102	7	2023/2/14
20240103	1	2021/12/4
20240105	2	2023/1/18
20240107	4	2022/4/26
20240108	5	2021/5/10
20240109	3	2020/6/11
20240112	3	2020/6/21
省略		
20240145	3	2023/4/19
20240145	5	2023/7/13

全20件

【課題 1 : class_tbl から全行全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM class_tbl;
```

【課題 2 : cert_tbl から全行全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM cert_tbl;
```

【課題 3 : stdcert_tbl から全行全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM stdcert_tbl;
```

【課題 4 : std_tbl から全行全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM std_tbl;
```

【課題 5 : stdcert_tbl から stdno と certid の列のみを取得する SQL 文を実行しなさい】

```
SELECT stdno,certid FROM stdcert_tbl;
```

【課題 6 : cert_tbl から certid が「1」のデータの全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM cert_tbl WHERE certid =1;
```

【課題 7 : std_tbl から classid の値が「S1A1」のデータの全列を取得する SQL 文を実行しなさい】

```
SELECT * FROM std_tbl WHERE classid = 'S1A1';
```


【課題 8：cert_tbl から certid の値が「1」または「2」の行を取得する SQL 文を実行しなさい】

```
SELECT * FROM cert_tbl WHERE certid = 1 OR certid = 2;
```

【課題 9：英語の点数が 0 点以上 29 点以下の学生番号と氏名、英語の点数を取得する】

```
SELECT stdno,stdname,english FROM std_tbl  
WHERE english BETWEEN 0 AND 29;
```

【課題 10：数学の点数が 90 点以上 100 点以下の学生番号と氏名、数学の点数を取得する】

```
SELECT stdno,stdname,math FROM std_tbl  
WHERE math BETWEEN 90 AND 100;
```

【課題 11：クラスが「S1A2」または「J1A1」または「J1A2」クラスの全列のデータを取得する】

```
SELECT * FROM std_tbl  
WHERE classid = 'S1A2' OR classid = 'J1A1' OR classid = 'J1A2';
```

【課題 12：クラス名に「S1」が含まれる学生の全列のデータを取得する】

```
SELECT * FROM std_tbl WHERE classid LIKE '%S1%';
```

【課題 13：std_tbl と class_tbl の 2 つの表を結合して全行取得する SQL 文を実行しなさい】

```
SELECT * FROM std_tbl,class_tbl WHERE std_tbl.classid = class_tbl.classid;
```

【課題 14：std_tbl と stdcert_tbl の 2 つの表を結合して全行取得する SQL 文を実行しなさい。】

```
SELECT * FROM std_tbl,stdcert_tbl WHERE std_tbl.stdno = stdcert_tbl.stdno;
```

【課題 15:std_tbl と stdcert_tbl と cert_tbl の 3 つの表を結合して全行取得する SQL 文を実行しなさい】

```
SELECT * FROM std_tbl,stdcert_tbl,cert_tbl
WHERE std_tbl.stdno = stdcert_tbl.stdno
AND stdcert_tbl.certid = cert_tbl.certid;
```

【課題 16:std_tbl において、classid でグループ化し行数をカウントしてその昇順で表示する SQL 文を作成しなさい】

```
SELECT classid,count(stdno) FROM std_tbl GROUP BY classid
ORDER BY count(stdno) ASC;
```

【課題 17:std_tbl と stdcert_tbl を結合する。そして、stdNo、classid、stdname でグループ化し行数をカウントして昇順で表示する SQL 文を作成しなさい】 **高難易度**

```
SELECT std_tbl.stdno,classid,stdname,count(std_tbl.stdno)
FROM std_tbl,stdcert_tbl
WHERE std_tbl.stdno = stdcert_tbl.stdno
GROUP BY std_tbl.stdno,classid,stdname
ORDER BY count(std_tbl.stdno) ASC;
```

【課題 18:std_tbl と stdcert_tbl を結合する。そして、stdno、classid、stdname でグループ化し行数をカウントする。そして、その行数が 2 以上のデータを降順で表示する SQL 文を作成しなさい】

高難易度 ※HAVING 句を使用した例です。HAVING 句はグループに対して条件を指定します。

```
SELECT std_tbl.stdno,classid,stdname,count(std_tbl.stdno)
FROM std_tbl,stdcert_tbl
WHERE std_tbl.stdno = stdcert_tbl.stdno
GROUP BY std_tbl.stdno,classid,stdname
HAVING count(std_tbl.stdno) >= 2
ORDER BY count(std_tbl.stdno) DESC;
```

