

データベース 演習課題レポート

第1回提出

学生番号: 222C1029

氏名: 江藤 洸陽

2023 年 11 月 17 日

第1章 データベーススキーマの設計

1.1 初期スキーマの作成

歴代の横綱の情報を管理するためのデータベースを作成する。

横綱の情報を表すために必要な属性を書き出して、以下の初期スキーマを作成した。

横綱 (横綱代位, 出身, 四股名, 部屋番号, 部屋名, 横綱昇進年)

1.1.1 属性の説明

初期スキーマにおける各属性の役割とドメインは、以下の通りである。

横綱代位 横綱の代数を表す。2桁の数字による文字列となる。

出身 横綱の出身地域を表す。最大4文字の文字列となる。

四股名 横綱の四股名を表す。最大7文字の文字列となる。

部屋番号 力士が所属する相撲部屋の番号を表す。2桁の数字による文字列となる。

部屋名 力士が所属する相撲部屋を表す。最大4文字の文字列となる。

横綱昇進年 横綱に昇進した年を表す。4桁の数字による文字列となる。

1.2 リレーションに格納されるデータ

横綱リレーションに格納されるデータは、以下の条件を満たす。

1. 横綱には固有の横綱代位が割り当てられており、横綱代位が同じである横綱が複数存在することはない。
2. 四股名と横綱昇進年の両方が同じ横綱が複数存在することはない。
3. 各部屋には固有の部屋番号を割り当てており、部屋番号が同じである横綱は部屋名も同じとなる。

1.2.1 候補キー・主キー

条件1より、{ 横綱代位 } は横綱リレーションの候補キーとなる。

条件2より、{ 四股名, 横綱昇進年 } も従業員リレーションの候補キーとなる。

ここでは、2つの候補キーのうち、横綱代位 を主キーとする。

主キー属性に下線を引いた初期スキーマは、以下の通りである。

横綱 (横綱代位, 出身, 四股名, 部屋番号, 部屋名, 横綱昇進年)

1.2.2 関数従属性・多値従属性

条件 3 より、部屋番号 部屋名 の関数従属性が存在する。

1.3 リレーションスキーマの正規化

横綱リレーションが全ての正規形を満たすように、正規化を行う。

1.3.1 第 1 正規形

横綱リレーションは、全ての属性が単一の値を持つため、第 1 正規形を満たす。

1.3.2 第 2 正規形

横綱リレーションは、候補キーの一部の属性から候補キー以外の属性への関数従属性は存在しないため、第 2 正規形を満たす。

1.3.3 第 3 正規形

従業員リレーションは、部屋番号 部屋名 の関数従属性により、候補キー以外の属性から候補キー以外の属性への関数従属性が存在するため、第 3 正規形を満たさない。第 3 正規形を満たすようにするために、従業員リレーションを、以下のように分解する。

横綱 (横綱代位, 出身, 四股名, 部屋番号, 横綱昇進年)

部屋 (部屋番号, 部屋名)

1.3.4 ボイス・コッド正規形

横綱リレーションは、キー属性の一部が非キー属性に関数従属しないのでボイスコッド正規形を満たす。

部屋リレーションは、キー属性の一部が非キー属性に関数従属しないのでボイスコッド正規形を満たす。

1.3.5 第 4 正規形

横綱リレーションは、すべての非キー属性が他の非キー属性に対して完全関数従属しているので第 4 正規系を満たす。

部屋リレーションは、すべての非キー属性が他の非キー属性に対して完全関数従属しているので第 4 正規系を満たす。

1.3.6 第5正規形

横綱リレーションは、結合従属性が存在していないので第5正規系を満たす。
部屋リレーションは、結合従属性が存在していないので第5正規系を満たす。

1.4 正規化後のリレーションスキーマ

最終的に、以下のリレーションスキーマが得られた。

横綱 (横綱代位, 出身, 四股名, 部屋番号, 横綱昇進年)

部屋 (部屋番号, 部屋名)

最終的に得られたリレーションスキーマには、下記の参照整合性制約（外部キー制約）が存在する。

横綱の部屋番号（部屋の部屋番号を参照）

第2章 データベースの作成

2.1 テーブルの定義

前章で設計した以下のリレーションスキーマにもとづいてデータベースを作成する。

横綱 (横綱代位, 出身, 四股名, 部屋番号, 横綱昇進年)

部屋 (部屋番号, 部屋名)

参照整合性制約 (外部キー制約)

横綱の部屋番号 (部屋の部屋番号を参照)

テーブル名・属性名をアルファベットに置き換えて、以下のテーブルを作成する。

yokozuna(id, fro, name, room_no, promo_year)

room(room_no, name)

2.2 テーブルの作成

以下のコマンドを使用して、横綱 (yokozuna) テーブルを作成した。

ソースコード 2.1: 横綱テーブルの作成のコマンド

```
1 create table yokozuna(  
2     id varchar(2) not null unique,  
3     fro varchar(4),  
4     name varchar(7) not null,  
5     room_no varchar(2),  
6     room varchar(4),  
7     promo_year varchar(4),  
8     primary key( id )  
9 );
```

以下のコマンドを使用して、部屋 (room) テーブルを作成した。

ソースコード 2.2: 部屋テーブルの作成のコマンド

```
1 create table room(  
2     room_no varchar(2) not null unique,  
3     name varchar(9) not null,  
4     primary key( room_no )  
5 );
```

以下のコマンドを使用して、横綱 (yokozuna) テーブルに参照整合性制約を追加した。

ソースコード 2.3: 横綱テーブルへの参照整合性制約の設定のコマンド

```
1 alter table yokozuna add constraint yokozuna_room_key foreign key (room_no) references
   room (room_no);
```

以下のコマンドを使用して、全てのテーブルに対して、ウェブサーバへの利用権限を設定した。

ソースコード 2.4: 横綱・部屋テーブルへの利用権限の設定のコマンド

```
1 grant all on yokozuna to apache;
2 grant all on room to apache;
```

2.3 初期データの挿入

以下のテキストファイル・コマンドを使用して、横綱 (yokozuna) テーブルに初期データを挿入した。

ソースコード 2.5: 横綱テーブルの初期データ (yokozuna_data.txt_list.php)

```
1 63, 青森, 旭富士正也, 02, 大島, 1990
2 64, アメリカ, 曙太郎, 09, 東関, 1993
3 65, 東京, 貴乃花光司, 05, 二子山, 1995
4 66, 東京, 若乃花勝, 05, 二子山, 1998
5 67, アメリカ, 武蔵丸光洋, 07, 武蔵川, 1999
6 68, モンゴル, 朝青龍明德, 03, 高砂, 2003
7 69, モンゴル, 白鵬翔, 06, 宮城野, 2007
8 70, モンゴル, 日馬富士公平, 01, 伊勢ヶ濱, 2012
9 71, モンゴル, 鶴竜力三郎, 08, 井筒, 2014
10 72, 茨城, 稀勢の里寛, 04, 田子ノ浦, 2017
```

ソースコード 2.6: 横綱テーブルへの初期データの挿入

```
1 COPY yokozuna from yokozuna_data.txt
```

以下のテキストファイルとコマンドを使用して、部屋 (room) テーブルに初期データを挿入した。

ソースコード 2.7: 部屋テーブルへの初期データの挿入 (room_data.txt)

```
1 insert into room values( '01', '伊勢ヶ濱' );
2 insert into room values( '02', '大島' );
3 insert into room values( '03', '高砂' );
4 insert into room values( '04', '田子ノ浦' );
5 insert into room values( '05', '二子山' );
6 insert into room values( '06', '宮城野' );
7 insert into room values( '07', '武蔵川' );
8 insert into room values( '08', '井筒' );
9 insert into room values( '09', '東関' );
```

ソースコード 2.8: 部屋テーブルへの初期データの挿入

```
1 \i room_data.txt
```

2.4 テーブルの格納データの確認

SQL を使って横綱 (yokozuna) テーブルに格納されている全てのデータを表示すると、以下のようになる。

```
agdo2465=> select * from yokozuna;
id | fro | name | room_no | room | promo_year
----+-----+-----+-----+-----+-----
```

63		02		旭富士正也		1990
64		09		曙太郎		1993
65		05		貴乃花光司		1995
66		05		若乃花勝		1998
67		07		武蔵丸光洋		1999
68		03		朝青龍明德		2003
69		06		白鵬翔		2007
70		06		日馬富士公平		2012
71		06		鶴竜力三郎		2014
72		06		稀勢の里寛		2017

(10 rows)

SQL を使って部屋 (room) テーブルに格納されている全てのデータを表示すると、以下のようになる。

```
agdo2465=> select * from room;
```

room_no		name
-----+-----		
01		伊勢ヶ濱
02		大島
03		高砂
04		田子ノ浦
05		二子山
06		宮城野
07		武蔵川
08		井筒
09		東関

(9 rows)

第3章 データベースへの問い合わせ

作成したデータベースに対して、データベースを利用する際に使われる問い合わせの具体例を考えて、その問い合わせに対する SQL と実行結果を最低 3 つ示す。問い合わせの具体例は何でも構わないが、出力結果が想定した問い合わせと一致しているか（正しい結果が出力されているか）を確認すること。なるべく GROUP BY (+HAVING) や入れ子型問い合わせ（相関あり・なし）などの、授業で学習したやや複雑な構文を用いた SQL を含めることが望ましい。

2 章で作成したデータベースに対して、以下のような SQL を使った問い合わせのテストを行った。

3.1 問い合わせ 1

問い合わせ、その問い合わせを実行するための SQL、データベースに対して実行した結果を順番に記述する。実行結果は、2.4 節で示した、データベースの格納データに対して SQL を実行したときの結果を示すこと。なお、verbatim 環境を使うと、文字数が多い行はページの横幅を超えてしまうため、手作業で適当に改行を挿入して、ページ内に収まるように調整する。整形が難しい場合は、verbatim 環境の代わりに、lstlisting 環境を用いても構わない。

問い合わせ：

全ての従業員の従業員番号、氏名、部門、年齢の一覧を出力する。

SQL：

```
select id, employee.name, department.name, employee.age from employee, department
where employee.dept_no = department.dept_no
```

実行結果：

```
abcd1234=> select id, employee.name, department.name, employee.age from employee, department
where employee.dept_no = department.dept_no;
```

id	name	name	age
0001	織田 信長	開発	48
0002	豊臣 秀吉	営業	45
0003	徳川 家康	総務	39
0004	柴田 勝家	開発	60
0005	伊達 政宗	営業	15
0006	上杉 景勝	総務	26
0007	島津 家久	開発	35

(7 rows)

3.2 問い合わせ 2

同様に、2 つ目の問い合わせについても、問い合わせ、SQL、実行結果を記述する。

問い合わせ：

従業員番号が'0003' の従業員と同じ部門に所属する従業員の、従業員番号と氏名の一覧を出力する。

SQL：

???????

実行結果：

???????

3.3 問い合わせ 3

同様に、3 つ目の問い合わせについても、問い合わせ、SQL、実行結果を記述する。

問い合わせ：

各部門で最も年上の人間を検索して、各部門の部門名、部門内の最年長者の氏名、年齢の一覧を出力する。

SQL：

???????

実行結果：

???????

4 つ以上の問い合わせを作成する場合は、節を追加して、同様に記述する。

第4章 Web インターフェースの作成

4.1 作成したインターフェースのメニューページ

作成したインターフェースのメニューページの URL を示す。

2章で作成したデータベースを利用するためのインターフェースを開発した。
作成したインターフェースのメニューページの URL は、下記の通りである。

`http://db.tom.ai.kyutech.ac.jp/~????/?????.html`

4.2 作成したインターフェースの構成

作成したインターフェースの概要を説明し、インターフェースを構成する全てのファイルと、それぞれの役割や階層構造が分かる説明を示す。

従業員と部門を結合して一覧表示する機能を作成した。

従業員テーブルへのデータの挿入・削除・更新のための機能を作成した。

部門テーブルについては、更新の頻度が少ないため、データの挿入・削除・更新の機能は作成していない。

また、従業員を部門と年齢の条件を組み合わせで検索できる機能を作成した。

インターフェースを構成するファイルの役割や階層構造は、下記の通りである。

- メニューページ (menu.html)
 - － 従業員・部門の一覧表示 (employee_list.php)
 - － 従業員のデータ追加の入力フォーム (employee_add_form.php)
 - * 従業員のデータ追加の処理実行 (employee_add.php)
 - － 従業員のデータ削除の選択 (employee_delete_form.php)
 - * 従業員のデータ削除の処理実行 (employee_delete.php)
 - － 従業員のデータ更新の選択 (employee_update_form1.php)
 - * 従業員のデータ更新の入力フォーム (employee_update_form2.php)
 - ・ 従業員のデータ更新の処理実行 (employee_update.php)
 - － 従業員の検索の入力フォーム (employee_search_form.php)
 - * 従業員の検索結果の表示 (employee_search_result.php)
 - － ...
(全てのファイルの構成を説明する。)

4.3 メニューページ

メニューページのソースファイル全体を引用して、メニューの項目を説明する。また、メニューページをウェブブラウザで表示したときのスクリーンショットを示す。メニューが複数のページにより構成される場合は、その全てについて説明する。

メニューページは、上記の階層構造においてメニューページの下位のページである、従業員・部門の一覧表示 (employee_list.php)、従業員のデータ追加 (employee_add_form.php)、従業員のデータ削除 (employee_delete_form.php)、従業員のデータ更新 (employee_update_form1.php)、従業員の検索 (employee_search_form.php) の各ページへのリンクを含む。

ソースコード 4.1: menu.html

```
1 <HTML>
2 <HEAD>
3   <TITLE>データ操作メニュー</TITLE>
4   <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 </HEAD>
6 <BODY>
7
8 操作メニュー<BR>
9
10 <UL>
11   <LI><A HREF="employee_list.php">従業員の一覧表示</A>
12   <LI><A HREF="employee_add_form.php">従業員のデータ追加</A>
13   <LI><A HREF="employee_delete_form.php">従業員のデータ削除</A>
14   <LI><A HREF="employee_update_form1.php">従業員のデータ更新</A>
15   <LI><A HREF="employee_search_form.php">従業員の検索 (部門名での検索)</A>
16 </UL>
17
18 </BODY>
19 </HTML>
```

図 4.1 は、ウェブブラウザでメニューページを表示したときのスクリーンショットを示したものである。

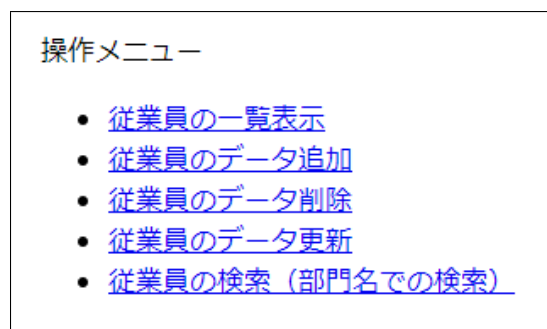


図 4.1: メニューページの表示結果

4.4 従業員の一覧表示

以降、各機能ごとに節 (4.x) を分けて、各可能の実現方法を説明する。一つの機能が複数のウェブページから構成される場合は、さらに各ウェブページごとに小節 (4.x.x) を分けて説明する。また、各機能について、節の最後に小節を追加して、実行結果の例を示すこと。

4.4.1 従業員の一覧表示 (employee_list.php)

各ウェブページの説明では、小節の末尾に作成したソースファイル全体を省略せずに引用し、その重要な箇所について、行番号で参照しながら説明を加える。ソースファイルの全ての行について説明を行う必要はない。前のページから渡される入力や、そのページで使用する SQL とその作成方法、結果の表示方法等の重要な点について説明する。

全従業員の一覧を表示する PHP プログラムを含むページである。従業員と部門のテーブルを結合して、従業員番号にもとづいて昇順で並べて表示する。また、全体で何件の従業員のデータが存在しているかの情報を、従業員一覧の後に表示する。

プログラムの 27 行目で従業員の一覧を作成する SQL 文を作成して、30 行目でその SQL 文を実行している。

ソースコード 4.2: 従業員の一覧表示のための SQL

```
1 select id , department.name, employee.name, age from employee , department where employee
   .dept_no = department.dept_no order by id
```

この問い合わせは、前ページでの利用者の入力によって変化するようなことはなく、毎回同じ問い合わせを実行するため、ソースファイル中で固定の SQL 文を文字列として設定している。

ソースコード 4.3: employee_list.php

```
1 <HTML>
2 <HEAD>
3   <TITLE>従業員リスト</TITLE>
4   <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 </HEAD>
6 <BODY>
7
8 <CENTER>
9
10 検索結果を表示します。<BR><BR>
11
12 <!-- ここから PHP のスクリプト始まり -->
13 <?php
14
15 // データベースに接続
16 //   your_db_name のところは自分のデータベース名に書き換える
17 $conn = pg_connect( "dbname=your_db_name" );
18
19 // 接続が成功したかどうか確認
20 if ( $conn == null )
21 {
22     print( "データベース接続処理でエラーが発生しました。<BR>" );
23     exit;
24 }
25
26 // SQL を作成
27 $sql = "select id , department.name, employee.name, age from employee , department where
        employee.dept_no = department.dept_no order by id";
28
29 // Query を実行して検索結果を result に格納
30 $result = pg_query( $conn , $sql );
31 if ( $result == null )
32 {
33     print( "クエリー実行処理でエラーが発生しました。<BR>" );
34     exit;
35 }
36
```

```

37 // 検索結果の行数・列数を取得
38 $rows = pg_num_rows( $result );
39 $cols = pg_num_fields( $result );
40
41
42 // 検索結果をテーブルとして表示
43 print( "<TABLE BORDER=1>\n" );
44
45 // 各列の名前を表示
46 print( "<TR>" );
47 print( "<TH>従業員番号</TH>" );
48 print( "<TH>部門</TH>" );
49 print( "<TH>氏名</TH>" );
50 print( "<TH>年齢</TH>" );
51 print( "</TR>\n" );
52
53 // 各行のデータを表示
54 for ( $j=0; $j<$rows; $j++ )
55 {
56     print( "<TR>" );
57     for ( $i=0; $i<$cols; $i++ )
58     {
59         // j行i列のデータを取得
60         $data = pg_fetch_result( $result, $j, $i );
61
62         // セルに列の名前を表示
63         print( "<TD> $data </TD>" );
64     }
65     print( "</TR>\n" );
66 }
67
68 // ここまででテーブル終了
69 print( "</TABLE>" );
70 print( "<BR>\n" );
71
72
73 // 検索性数を表示
74 print( "以上、$rows 件のデータを表示しました。<BR>\n" );
75
76
77 // 検索結果の開放
78 pg_free_result( $result );
79
80 // データベースへの接続を解除
81 pg_close( $conn );
82
83 ?>
84 <!-- ここまででPHPのスクリプト終わり -->
85
86 <BR>
87 <A HREF="menu.html">操作メニューに戻る</A>
88
89 </CENTER>
90
91 </BODY>
92 </HTML>

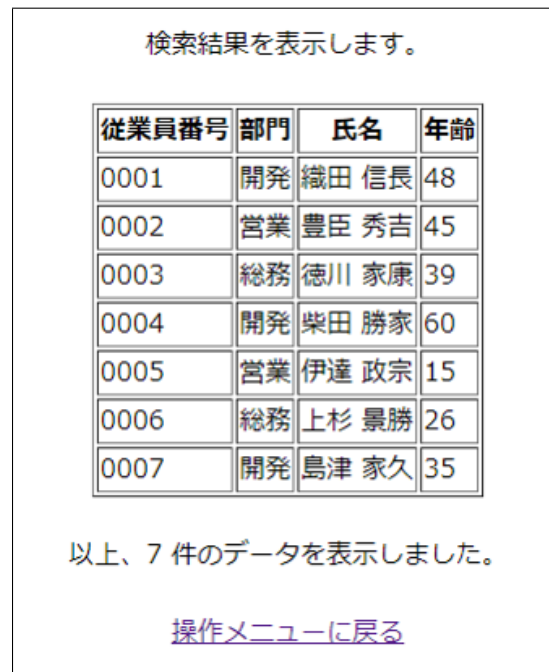
```

4.4.2 従業員の一覧表示の実行例

実際にインターフェースを操作したときのスクリーンショットを使って、作成した機能が正しく動作していることが分かる結果を示す。複数のページや操作からなる機能については、複数のスクリーンショットを使って、操

作の過程や結果が分かるようにする。(後の例を参照。) LaTeX では、図は自動的に配置されるため、本文中で正しい図番号を使って参照されていれば、対応する説明から少し離れた位置に図が配置されていても構わない。スクリーンショットを示すときには、画面に表示・入力されている文字が読めるような、適切な解像度・大きさの画像を用いること。

図 4.2 は、一覧表示の操作を行ったときのスクリーンショットを示したものである。



従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15
0006	総務	上杉 景勝	26
0007	開発	島津 家久	35

図 4.2: 従業員の一覧表示の実行結果

4.5 従業員の追加

...

4.5.1 従業員のデータ追加の入力フォーム (employee_add_form.php)

...

4.5.2 従業員のデータ追加の処理実行 (employee_add.php)

前のページから渡される入力がある場合は、その説明を行う。

前のページのフォームに対して入力された、以下の情報を受け取る。プログラムの??~??行目で、これらのデータを取得して、各変数に代入する。

- 従業員番号 (name) \$name
- 部門番号 (dept_no) \$dept_no
- 氏名 (name) \$name

- 年齢 (age) \$age

プログラムの??行目で、従業員テーブルにインスタンスを追加するための SQL 文を作成する。SQL 文の %1, %2, %3, %4 には、変数 \$name, \$dept_no, \$name, \$age の値が挿入される。??行目で、作成した SQL 文を実行する。

ソースコード 4.4: 従業員の追加のための SQL

```
1 insert into employee values ( %1, %2, %3, %4 )
```

...

4.5.3 従業員の追加の実行例

...

4.6 従業員の削除

...

4.6.1 従業員のデータ削除の選択 (employee_delete_form.php)

...

4.6.2 従業員のデータ削除の処理実行 (employee_delete.php)

...

4.6.3 従業員の削除の実行例

...

4.7 従業員の更新

...

4.7.1 従業員のデータ更新の選択 (employee_update_form1.php)

...

4.7.2 従業員のデータ更新の入力フォーム (employee_update_form2.php)

...

4.7.3 従業員のデータ更新の処理実行 (employee_update.php)

...

4.7.4 従業員のデータ更新の実行例

複数のページによって実現される機能については、複数のスクリーンショットを用いて、正しく機能が実行されている例を示す。

図 4.3～4.7 は、従業員「伊達 政宗」の部門を「総務」から「営業」に更新する操作を行ったときの、一連のスクリーンショットを示したものである。

従業員データ 更新フォーム

更新したい従業員を選択して送信ボタンを押してください。

従業員番号	部門	氏名	年齢
<input type="radio"/> 0001	開発	織田 信長	48
<input type="radio"/> 0002	営業	豊臣 秀吉	45
<input type="radio"/> 0003	総務	徳川 家康	39
<input type="radio"/> 0004	開発	柴田 勝家	60
<input checked="" type="radio"/> 0005	総務	伊達 政宗	15

以上、5 人の従業員が登録されています。

[操作メニューに戻る](#)

図 4.3: 従業員の更新の実行結果 (1) : 選択フォームから更新する従業員を選択

従業員データ 更新フォーム

部門 : ☐ 開発 ☐ 営業 ☒ 総務

氏名 : 年齢 :

性別 : ☒ 男 ☐ 女

図 4.4: 従業員の更新の実行結果 (2) : 入力フォームに移り、選択した従業員の現在の情報が表示される

従業員データ 更新フォーム

部門： ☐ 開発 ☒ 営業 ☐ 総務

氏名： 年齢：

性別： ☒ 男 ☐ 女

図 4.5: 従業員の更新の実行結果 (3)：入力フォームで、部門の情報を変更して、送信ボタンを押す

クエリー「update employee set dept_no='02', name='伊達 政宗', age=15 where id='0005';」を実行します。
データの更新処理が完了しました。

[操作メニューに戻る](#)

図 4.6: 従業員の更新の実行結果 (4)：更新処理の実行結果が表示される

検索結果を表示します。

従業員番号	部門	氏名	年齢
0001	開発	織田 信長	48
0002	営業	豊臣 秀吉	45
0003	総務	徳川 家康	39
0004	開発	柴田 勝家	60
0005	営業	伊達 政宗	15

以上、5 件のデータを表示しました。

[操作メニューに戻る](#)

図 4.7: 従業員の更新の実行結果 (5)：一覧表示を行い、従業員の情報が更新されていることを確認する

4.8 従業員の検索

...

4.8.1 従業員の検索の入力フォーム (employee_search_form.php)

...

4.8.2 従業員の検索結果の表示 (employee_search_result.php)

場合分けや複数の SQL を使い分ける場合は、その説明を行う。

前のページのフォームに対して入力された、以下の情報を受け取る。プログラムの??行目で、データを取得して、変数に代入する。

- 部門番号 (dept_no) \$dept_no

プログラムの?? ~ ??行目で、従業員の検索を行うための SQL 文を作成する。変数 \$dept_no に文字列「ALL」が格納されている場合は、全ての部門の従業員を表示するための SQL を実行する。

ソースコード 4.5: 全ての従業員を表示するための SQL

```
1 select id, department.name, employee.name, age from employee, department where employee
   .dept_no = department.dept_no order by id";
```

変数 \$dept_no に「ALL」以外の文字列が格納されている場合は、部門番号が \$dept_no に等しい従業員のみを表示するための SQL を実行する。SQL 文の \$dept_no には、その変数の値が挿入される。

ソースコード 4.6: 指定された部門番号に所属する従業員を検索するための SQL

```
1 select id, department.name, employee.name, age from employee, department where employee
   .dept_no = department.dept_no and employee.dept_no = '$dept_no' order by id";
```

4.8.3 従業員のデータ検索の実行例

複数の条件を組み合わせた検索機能など、機能が正しく動作していることを示すために複数の実行例を示した方がよい場合は、複数の実行例を示す。ただし、必要以上に多数の実行例（全ての条件の組み合わせのバリエーションなど）を示す必要はないので、いくつかの代表的な実行例を示す。

図?? ~ ?? は、部門のみを条件とする検索の実行例として、部門が「総務」である従業員を検索する操作を行ったときの、一連のスクリーンショットを示したものである。

図?? ~ ?? は、部門と年齢を組み合わせた条件とする検索の実行例として、部門が「開発」で、かつ年齢が 20 歳以上 30 歳以下の従業員を検索する操作を行ったときの、一連のスクリーンショットを示したものである。

...

...

第5章 まとめ

まとめや感想として、何か書きたいこと（工夫した点や、詰まった点、反省点など）があれば、自由に記述する。何もない場合は、この章は省略して構わない。