デュアルウェア講習会 II
データベース操作

目標：データベースにアクセスしてテーブルを操作

目次

第 1 章

課題

データベースに表 1.1 のようなデータテーブルを作成します.

表 1.1: 課題テーブル

id	date	time	temp	humi	place
20190101160000	2019-01-01	16:00:00	20.1	50	LICTiA
20190101160001	2019-01-01	16:00:01	20.2	51	LICTiA
20190101160002	2019-01-01	16:00:02	20.3	52	LICTiA
20190101160003	2019-01-01	16:00:03	20.4	49	LICTiA
20190101160004	2019-01-01	16:00:04	20.5	51	LICTiA

プライマリキーは id とします.

第 2 章

データベース

2.1　データベースとは

　データベースとは情報を集めて使いやすいように整理したものです．コンピュータ上の情報を指すことが多いですが，紙媒体で保存された電話帳や名簿などもデータベースとよびます．

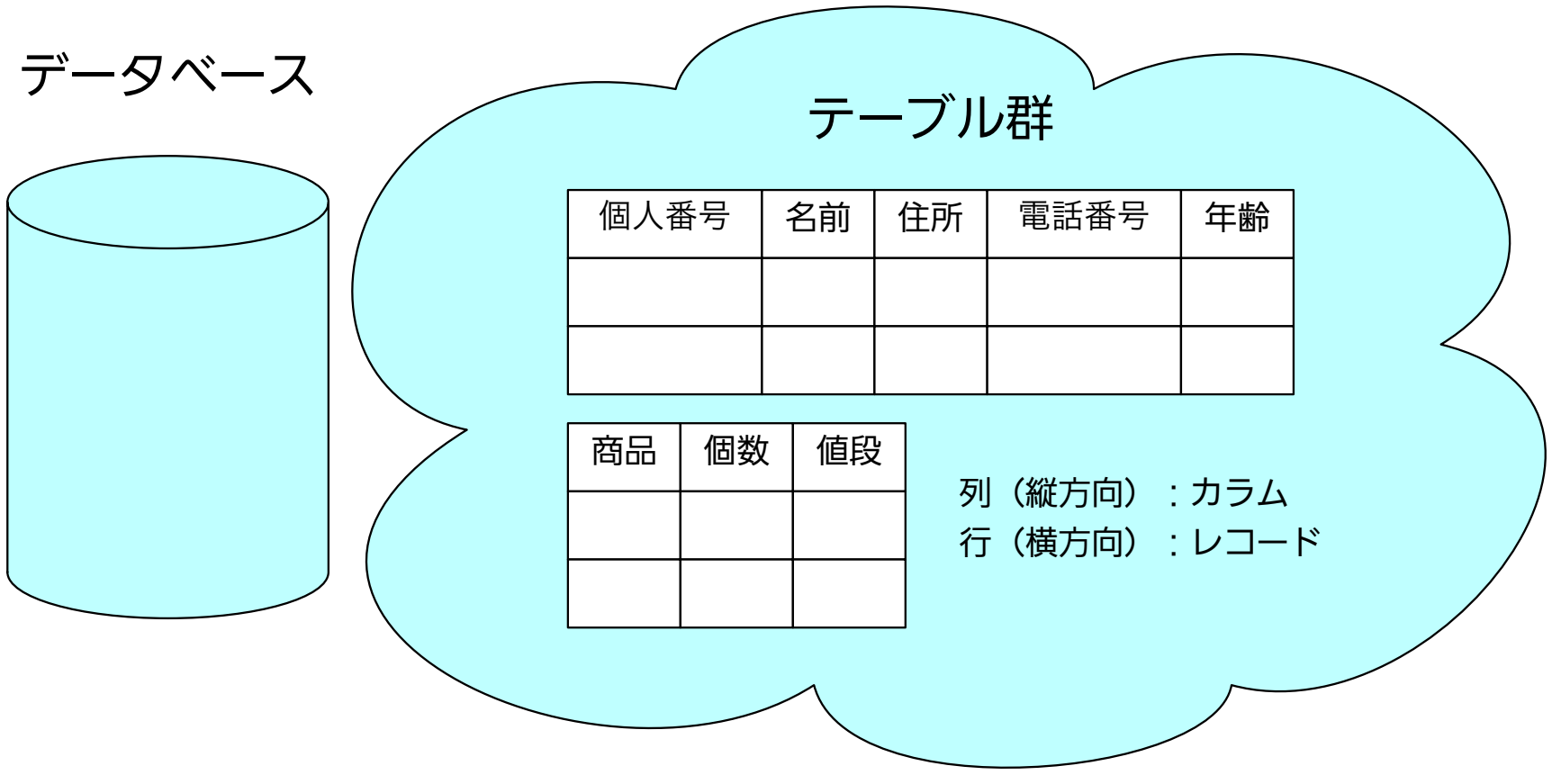

図 2.1: データベース構造

　図 2.1 はデータベースの構成になります．データベースは複数のテーブル (表) を持ち，その中にさまざまなデータを持っています．このような構造のデータベースをリレーショナルデータベースといいます．

- テーブル：
  データが登録された表のことを指します．イメージとしてはエクセルのシートのようなものと考えてください．
- レコード：
  テーブルの横の行のことを指します．1 レコードで一組のデータになります．図 1 において， 名前， 住所， 電話番号， 年齢 の 1 組でひとつのレコードになります．
- カラム：

テーブルの縦の列のことを指します．カラムには同じ属性のデータが登録されます．図 1 において， 名前 のカラムには名前が， 電話番号 のカラムには電話番号が登録されます．

- プライマリキー：
  テーブル内でレコードを一意に識別することができるように指定される項目のことを指します．この値は重複することはなくいったん入力されると基本的に変更されません．

テーブルを作成するにはカラムの個数と各カラムに何を登録するかを事前に決める必要があります．

2.2 SQL

SQL とはリレーショナルデータベースを操作するための言語です．データの検索，追加，削除などの操作ができます．データベースは SQL を使ってデータの操作を行います．操作には 4 つの種類があります（表 2.1）．

表 2.1: SQL 種類

種類	説明	コマンド例
データ定義文	オブジェクトの作成・変更・更新	`CREATE`，`DROP`
データ操作文	表に対して新しいデータの追加・更新	`INSERT`，`SELECT`，`DELETE`
データ制御文	データベースに対するアクセス権の付与削除	`GRANT`，`REVOKE`
トランザクション制御	データベースの作業を確定，または取り消し	`COMMIT`，`ROLLBACK`

本講習では主に，データ定義文，データ操作文，トランザクション制御を使いデータベースを操作します．

2.2.1 SQL 文の記述

データベースを操作するときの SQL の構文を SQL 文といいます．SQL 文を実行するときは以下のことに気を付けてください．

- SQL 文は大文字小文字を区別しない：
  SQL 文は大文字小文字を区別しません．したがって，`SLECT * FROM TEST;` と `select * from test;` は同じ意味の SQL 文と解釈されます．
- 1 行以上に渡り入力することができる：
  SQL 文を文末の `;` まで何行にまたがって記述しても問題ありません．しかし，コマンドが行をまたがっている場合はエラーになります（表 2.2）．

表 2.2: SQL 文複数行例

エラーなし		エラー検出
1 行で記述	2 行で記述	`from` が 2 行にわたっている
`SQL> select * from test;`	`SQL> select` `SQL> * from test;`	`SQL> select * fr` `SQL> om test;`

第 3 章

データベース操作

実際にデータベースを作成してデータベースを操作します．表 3.1 のようなデータテーブルを作成してデータの操作を体験します．

表 3.1: 例題テーブル

id	date	time	temp	place
20190101160000	2019-01-01	16:00:00	20.0	LICTiA
20190101160001	2019-01-01	16:00:01	20.2	LICTiA
20190101160002	2019-01-01	16:00:02	20.3	LICTiA
20190101160003	2019-01-01	16:00:03	20.2	LICTiA
20190101160004	2019-01-01	16:00:04	20.5	LICTiA

プライマリキーは id とします．

3.1 PostgreSQL

PostgreSQL とはデータベースを作成したり操作したりするオープンソースのデータベース管理システムで，世界中で多く利用されています．Linux，Winodws，macOS にも対応しており，無料で公開されています．今回はこの PostgreSQL を使ってデータベースを操作します．

3.2 Raspberry Pi に PostgreSQL をインストール

最初に，Raspberry Pi に PostgreSQL をインストールします．以下のコマンドを実行してください．

```
1 $ sudo apt-get install postgresql
```

実行すると Raspberry Pi に PostgreSQL がインストールされます．

3.3 ユーザ設定

インストールが終わったら，ユーザ設定を行います．ユーザ名は任意の名前に変更できますが，Raspberry Pi のユーザ名と同じにすると，コマンドを省略できます．以下のコマンドでユーザを作成します．このときユーザ名は pi で作成します．

```
1 $ sudo -u postgres createuser -P -s pi
```

```
2 新しいロールのためのパスワード:
3 もう一度入力してください:
```

オプション -P を付けると，パスワード設定も行えます．今回は RaspberryPi のパスワードと同じく raspberry にします．オプション -s を付けると，ユーザに管理者権限を付与できます．

3.4 データベース作成

ユーザ設定後データベースを作成します．以下このコマンドでデータベースを作成します．

```
1 $ createdb
```

createdb は引数でデータベース名を指定しますが，省略した場合，ユーザ名と同じ名前でデータベースが作成されます．今回は pi でデータベースを作成します．

PostgreSQL のユーザとデータベースが作成されたか確認をします． psql コマンドを実行して確認します．

```
1 $ psql
```

psql は PostgreSQL を起動するコマンドです．引数なしだと，Raspberry Pi と同じユーザ名で実行されます．

実行すると，プロンプトが $ から pi=# に変わります。このプロント時は PostgreSQL のコマンドが使用できます。

```
1 pi=#
```

\l コマンドを実行してデータベースの一覧を確認します．実行して以下の結果が出ることを確認してください．

```
1 pi=# \l
```

```
pi=# ¥l
                                  データベース一覧
   名前    |  所有者  | エンコーディング |  照合順序   | Ctype(変換演算子) |      アクセス権
-----------+----------+------------------+-------------+-------------------+----------------------
 pi        | pi       | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       |
 postgres  | postgres | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       |
 template0 | postgres | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       | =c/postgres          +
           |          |                  |             |                   | postgres=CTc/postgres
 template1 | postgres | UTF8             | ja_JP.UTF-8 | ja_JP.UTF-8       | =c/postgres          +
           |          |                  |             |                   | postgres=CTc/postgres
(4 行)

pi=#
```

図 3.1: データベース一覧

PostgreSQL を終了するには， \q コマンドを実行します．

```
1 pi=# \q
```

3.5　テーブル作成

データベースを作成しましたので，次にテーブルを作成します．以下のコマンドでデータベースを作成します．

```
1 CREATE TABLE [テーブル名] (
2  [カラム名] [データ型] [NULL or NOT NULL],
3  [カラム名] [データ型] [NULL or NOT NULL],
4  [カラム名] [データ型] [NULL or NOT NULL],
5  ...
6 );
```

表 3.2: CREATE コマンド

テーブル名	作成したいテーブル名
カラム名	テーブルの列の名前
データ型	カラムの型

テーブルを作成するときは，テーブル名とそのテーブルを構成するカラムの名前とそのデータ型を決めます．

3.5.1　データ型

データの型は多くの種類があります．例えば，数字を表すものにも，整数や浮動小数点など，様々な型があります．今回は代表的なものを紹介します．

- 数字（表 3.3）

表 3.3: データ型（数字）

型名	格納サイズ	型の種類	範囲
double precision	4byte	浮動小数点	15 桁精度
numeric	可変長	任意の精度を持つ数	小数点前までは 131072 桁，小数点以降は 16383 桁
smallint	2byte	整数	-32768 から +32767

- 文字列（表 3.4）

表 3.4: データ型（文字列）

型名	格納サイズ	型の説明
varchar(n)	可変長・上限 n 文字まで	上限 n 文字まで入力できる. n 文字以下の場合は入力した文字数までで保存.
char(n)	固定長・上限 n 文字まで	上限 n 文字まで入力できる. n 文字以下の場合は入力した文字数までで保存.

- 日付（表 3.5）

表 3.5: データ型（日付）

型名	大きさ	説明	範囲
date	4byte	日時のみ時刻なし	4713BC から 5874897AD

- 時刻（表 3.6）

表 3.6: データ型（時刻）

型名	大きさ	説明	範囲
time	8byte	日付なしの時刻	00:00:00〜24:00:00

3.5.2　テーブル作成

reidai テーブルを作成します．各カラムとデータ型は表 3.7 の対応になります.

表 3.7: reidai カラム名とデータ型対応

カラム名	id	date	time	temp	place
データ型	numeric	date	time	numeric	varchar(10)

reidai テーブルを作成するコマンドは以下になります.

```
1 Pi=# CREATE TABLE reidai (id numeric PRIMARY KEY,date date,time
    time,temp numeric,place varchar(10));
```

PRIMARY KEY はプライマリキーのカラムにつけるオプションです．実行したらテーブルが作成されているか確認します。 \d コマンドでテーブルの一覧を表示します．結果が以下のようになっていることを確認してください.

```
1 pi=# \d
```

```
                リレーションの一覧
 スキーマ |   名前   |      型       | 所有者
----------+--------+-----------+--------
 public   | reidai | テーブル | pi
(1 行) (1 行)
```

3.5.3 課題 1

kadai テーブルを作成します．以下のコマンドの中の #TODO# の部分を修正して， kadai テーブルを作成してください．

```
1 CREATE TABLE kadai (id numeric PRIMARY KEY,
2  date date,
3  time time,
4  temp numeric,
5  #TODO#,
6  place varchar(10));
```

カラム名とデータ型は表 3.8 になります．

表 3.8: kadai カラム名とデータ型対応

カラム名	id	date	time	temp	humi	place
データ型	numeric	date	time	numeric	numeric	varchar(10)

\d コマンドでテーブルの一覧を表示します．結果が以下のようになっていることを確認してください．

```
1 pi=# \d
```

```
                リレーションの一覧
 スキーマ |   名前   |      型       | 所有者
----------+--------+-----------+--------
 public   | kadai  | テーブル | pi
 public   | reidai | テーブル | pi
(2 行)
```

3.6　データ作成

テーブルを作成しましたが，テーブルに何もデータが入っていない状態です．テーブルにデータを挿入するには INSERT コマンドを実行します．

```
1 INSERT INTO [テーブル名 ] (カラム名1,カラム名2) VALUES ( [値1]
    [ , ‘値2’ ]・・・);
```

表 3.9: INSERT コマンド

テーブル名	データを挿入したいテーブル名
カラム名	テーブルのカラム
値	挿入するデータ

データを追加したいテーブルを指定して，カラム名と追加したいデータを指定します．

reidai テーブルの 1 行目のデータを追加する場合は以下のように実行します．

```
1 pi=# INSERT INTO reidai (id,date,time,temp,place) VALUES
    (20190101160000,'2019-01-01','16:00:00',20.0,'LICTiA');
```

追加ができると， INSERT 0 1 と表示されます．

3.6.1　例題 2

reidai テーブルの 1 行目のレコードが追加できたら，2 行目から 5 行目のレコードを追加してみてください．

3.6.2　課題 2

kadai テーブルの 1 行目から 5 行目のレコードを追加してください．

3.7　データ操作

テーブルにデータを挿入したので，挿入したデータを参照して表示をしてみましょう．テーブルのデータを参照するには， SELECT コマンドを使用します（表 3.10）．

```
1 SELECT [カラム名] FROM [テーブル] WHERE 条件式;
```

表 3.10: SELECT コマンド

テーブル名	参照したいテーブル名
カラム名	参照したいカラム
条件式	参照したいデータの条件

SELECT コマンドはテーブルにアクセスしデータの検索を行い取得するコマンドです．条件式を省略するとテーブルの全レコードが表示されます．

3.7.1　基本構文

SELECT コマンドの基本構文は以下になります.

```
1 pi=# SELECT * FROM reidai;
```

reidai テーブル内の全レコードの全カラムが表示されます．以下のように表示されることを確認してください.

```
        id        |    date    |   time   | temp | place
------------------+------------+----------+------+--------
 20190101160000 | 2019-01-01 | 16:00:00 | 20.0 | LICTiA
 20190101160001 | 2019-01-01 | 16:00:01 | 20.2 | LICTiA
 20190101160002 | 2019-01-01 | 16:00:02 | 20.3 | LICTiA
 20190101160003 | 2019-01-01 | 16:00:03 | 20.2 | LICTiA
 20190101160004 | 2019-01-01 | 16:00:04 | 20.5 | LICTiA
(5 行)
```

アスタリスクではなくカラム名を指定すれば，指定したカラムだけを表示します.

```
1 pi=# SELECT id,temp FROM reidai;
```

```
        id        | temp
------------------+------
 20190101160000 | 20.0
 20190101160001 | 20.2
 20190101160002 | 20.3
 20190101160003 | 20.2
 20190101160004 | 20.5
(5 行)
```

3.7.2　条件式

全レコードではなく指定したレコードを表示したい場合は where を使用します． where を使用すると， where 以降に書かれた条件に合ったレコードを表示するようになります．具体的には以下のように書きます.

```
1 pi=# SELECT id,temp FROM reidai WHERE id =20190101160000;
```

id =1 のように， カラム名 比較演算子 条件値 で記述します.

使用できる比較演算子は以下になります（表 3.11）.

表 3.11: 比較演算子

比較演算子	説明
=	同じ
<	小さい
>	大きい
<=	以下
>=	以上
<>，　!=	等しくない

複数の条件式を使用するには論理演算子を使用します．論理演算子を使うことにより，『〜と〜のレコードを選択』,『〜または〜のレコードを選択』といった AND/OR 検索が可能になります.

```
1 pi=# SELECT id,temp FROM reidai where id < 20190101160003 and
     temp <20.2;
```

and を使用すると， id < 20190101160003，かつ temp <20.2 を満たすレコードを表示します.

```
         id         | temp
--------------------+------
 20190101160000 | 20.0
(1 行)
```

使用できる論理演算子は主に以下になります（表 3.12）.

表 3.12: 論理演算子

比較演算子	説明
AND	かつ
OR	または

3.7.3　行のソート

SQL の出力の順番を変更することができます．変更するには order by を使用します.

```
1 pi=# SELECT * FROM reidai ORDER BY temp ASC;
```

```
       id        |    date    |   time   | temp | place
-----------------+------------+----------+------+--------
 20190101160000 | 2019-01-01 | 16:00:00 | 20.0 | LICTiA
 20190101160001 | 2019-01-01 | 16:00:01 | 20.2 | LICTiA
 20190101160003 | 2019-01-01 | 16:00:03 | 20.2 | LICTiA
 20190101160002 | 2019-01-01 | 16:00:02 | 20.3 | LICTiA
 20190101160004 | 2019-01-01 | 16:00:04 | 20.5 | LICTiA
(5 行)
```

ASC は昇順， DESC は降順になります．何も指定しないと昇順 (ASC) になります．

3.8　課題 3

コマンド SELECT * FROM kadai; を実行して，結果が以下になることを確認してください．

```
       id        |    date    |   time   | temp | humi | place
-----------------+------------+----------+------+------+--------
 20190101160000 | 2019-01-01 | 16:00:00 | 20.1 |   50 | LICTiA
 20190101160001 | 2019-01-01 | 16:00:01 | 20.2 |   51 | LICTiA
 20190101160002 | 2019-01-01 | 16:00:02 | 20.3 |   52 | LICTiA
 20190101160003 | 2019-01-01 | 16:00:03 | 20.4 |   49 | LICTiA
 20190101160004 | 2019-01-01 | 16:00:04 | 20.5 |   51 | LICTiA
(5 行)
```

値が異なっていた場合，4 章，5 章を見てデータを修正してください．

第 4 章

データ削除

レコードを削除するときは，DELETE コマンドを使用します．

```
1 DELETE FROM [テーブル名] WHERE [条件式];
```

表 4.1: DELETE コマンド

テーブル名	指定したいテーブル名
条件式	削除したいレコードの条件

指定したテーブルから指定した条件に当てはまるレコードを削除します．以下のように実行します[*1]．

```
1 pi=# DELETE FROM reidai where id =20190101160000;
```

[*1] この講習会では値を間違えて INSERT した場合，DELETE コマンドで削除してください．

第 5 章

テーブル削除

テーブルごと削除する場合は，DROP コマンドを使用します（表 5.1）.

```
1 DROP TABLE [テーブル名];
```

表 5.1: DROP コマンド

テーブル名	削除したいテーブル名

指定したテーブルを削除する場合，以下のように実行します[*1].

```
1 pi=# DROP TABLE reidai;
```

[*1] この講習会ではテーブル作成時にカラム名やデータの型を間違えたとき使用してください.

第 6 章

外部からデータベースへアクセス

この講習会では Raspberry Pi のデータベースに別の端末からネットワークを経由してアクセスします．外部からのアクセスをデータベースへ許可するには以下の個所を編集します．

6.1 postgresql.conf

`postgresql.conf` は PostgreSQL のコンフィグレーションファイルになります．このファイルの 60 行目付近の `listen_addresses` を `'localhost'` から `'*'` に変更し，コメントの `#` を外します．

`listen_addresses` は PostgreSQL に対する外部からの接続を許可するホストや，IP アドレスを設定する項目です．

Raspberry Pi で編集するには以下のコマンドを実行してください[*1]．

```
1 $ sudo sed -i -e "s/^#listen_addresses = 'localhost'/
    listen_addresses = '*'/g" /etc/postgresql/9.4/main/postgresql.
    conf
```

6.2 pg_hba.conf

`pg_hba.conf` はクライアントのアドレスとロール名を指定して，どのデータベースに対して接続を許可するのかを設定するために使用します．以下のコマンドで接続できる IP アドレスを指定できます[*2]。

```
1 $ sed -i -e "93i host all all 192.168.21.0/24 md5" /etc/
    postgresql/9.4/main/pg_hba.conf
```

ここでは `192.168.21.0/24` で，`192.168.21.xx` の IP アドレスからは接続できるように設定しています．

[*1] コマンド中の `9.4` は，PostgreSQL のバージョンなので異なる場合があります．
[*2] コマンド中の `9.4` は，PostgreSQL のバージョンなので異なる場合があります．