# 2：データの操作(DML)入門2

製作：清水健二

## テーブルのデータを更新する

### データの更新(UPDATE)

データを更新する際には、UPDATE文を使用します。「UPDATE」の後に更新先のテーブル名を指定し、更新する列と値をSET句で指定します。
更新する行は、WHERE句で条件指定します。**WHERE句を省略すると、全ての行が更新されてしまいますので、必ずWHERE句を付けてください。**

```
 UPDATE [データベース名].[テーブル名] SET [列名] = [値]（※複数ある場合は「,」で区切る） WHERE [条件(※条件の表記は、SELECT文の場合と同じ)];
```

ここでは、引き続き`World`データベースを使用します。下記のコマンドを実行してください。

```
 USE World;
```

**ex.　国コード「JPN」の人口を「127,094,745」人に変更する**

まず、SELECT文で登録時の人口「 126,714,000」人であることを確認します。

```
SELECT Name,Population FROM Country WHERE Code = 'JPN';
```

次にUPDATE文を使用します。

```
UPDATE Country SET Population=127094745 WHERE Code = 'JPN';
```

再度SELECT文を実行して人口が変更されたことを確認してください。

```
SELECT Name,Population FROM Country WHERE Code = 'JPN';
```

Joe Biden

**ex. 国コード「USA」の大統領を「~~Donald Trump~~」に変更する**

まず、SELECT文で登録時の大統領「George W. Bush」であることを確認します。

```
SELECT HeadOFState FROM Country WHERE Code = 'USA';
```

次にUPDATE文を使用します。

```
UPDATE Country SET HeadOFState='Donald Trump' WHERE Code = 'USA';
```

再度SELECT文を実行して大統領が変更されたことを確認してください。

```
SELECT HeadOFState FROM Country WHERE Code = 'USA';
```

# テーブルに新規データを挿入する

## データの挿入(INSERT INTO)

データの登録には、INSERT文を使用します。「INSERT INTO」の後に登録先のテーブルおよび列名を指定し、VALUES句の後に各列の値を指定します。

```
INSERT INTO [データベース名].[テーブル名]([列名(※複数ある場合は「,」で区切る)])
VALUES ([列に対応する値(※複数ある場合は「,」で区切る)]);
```

ここでは以前に作成したデータベース`sample`(※環境によって名前が違います)を使用します。

```
USE sample;
```

データベース内に`data`テーブルがあることを確認します。

```
SHOW TABLES;
```

`data`テーブルのカラムの内容を確認します。

```
SHOW COLUMNS FROM data;
```

ここでは、下記のデータを`data`テーブルに挿入します。

| code | name | amount |
|---|---|---|
| 1 | pencil | 12 |

次のSQL文を実行します。

```
INSERT INTO data (code,name,amount) VALUES (1,'pencil',12);
```

SELECT文で値が挿入されたことを確認します。

```
SELECT * FROM data;
```

## 主キーは重複できない(PRIMARY KEY制約)

ちなみに上記のINSERT文をもう一度実行するとエラーが表示されます。これは主キーとして設定している`code`に対してPRIMARY KEY制約がかかっているためです。そのため、2件目以降を登録する場合は重複しない主キーを記述する必要があります。

## 2件目以降の登録

主キーが重複しなければ複数件INSERTを実行できます。
ここでは下記のデータを登録します。

| code | name | amount |
|---|---|---|
| 2 | notebook | 4 |
| 3 | eraser | 20 |

下記のSQL文を実行します。

```
INSERT INTO data
(code,name,amount)
VALUES
(2,'notebook',4),
(3,'eraser',20);
```

SELECT文で値が挿入されたことを確認します。

```
SELECT * FROM data;
```

# テーブルのデータを削除する

## データの削除(DELETE)

データを削除する際には、DELETE文を使用します。FROM句の後に削除先のテーブル名を指定し、削除する行はWHERE句で条件指定します。こちらも**WHERE句を省略すると、全ての行が削除されてしまいますので、必ずWHERE句を付けてください。**

```
DELETE FROM [データベース名].[テーブル名] WHERE [条件(※条件の表記は、SELECT文の場合と同じ)];
```

**ex.`data`テーブルの`code`が3のデータのみ削除する**
まず、codeが3のデータを確認します。

```
SELECT * FROM data WHERE code=3;
```

内容を確認したら下記のSQLを実行してデータを削除してください。

```
DELETE FROM data WHERE code=3;
```

再度SELECT文を実行してデータが削除されたことを確認します。

```
SELECT * FROM data;
```

**ex.データを全件削除する**
下記のSQL文を実行してください。

```
DELETE FROM data;
```

このように簡単にデータを全件削除できてしまうため、大事なデータの削除する場合は必ずWHERE句をつけて、レコードを選択してから実行してください。

## 【補足】存在チェックフラグをつける

DELETE文で削除したデータは元に戻せないうえ、件数によっては処理に時間がかかります。そこで開発現場では、存在チェックフラグというTRUEかFALSEの真偽値のカラムを作り、それをUPDATE文でFALSEと処理することで疑似的に削除したように扱います。この方法ならばフラグを変更するだけでデータの表示、非表示が変更できます。
**ex.コード1の存在チェックフラグ(is_exist)をFALSEに更新する**

```
UPDATE data SET is_exist = false WHERE code =1;
```

SELECT文でデータを表示する際は、下記のようにWHERE句で存在チェックフラグがTRUEのものだけ表示させます。
ex.存在チェックフラグ(is_exist)がtrueのものだけ表示する

```
SELECT * FROM data WHERE is_exist = true;
```