

# 複雑なUPDATE, DELETE, INSERT カラムの制約

1日目

2日目

3日目

4日目

5日目

6日目

7日目

8日目

9日目

10日目

11日目

12日目

13日目

14日目

15日目

16日目

17日目

18日目

19日目

20日目

21日目

複雑なUPDATE, DELETE, INSERT

これまで、副問い合わせ、JOIN、ウィンドウ関数など複雑なSQLを**SELECT**で利用してきた



同様に、副問い合わせ、JOIN、ウィンドウ関数の結果は、**UPDATE**、**DELETE**、**INSERT**でも利用して、より複雑な(便利な)更新、削除、挿入処理を行える

# UPDATE処理の応用

UPDATE処理内で、テーブル間の連結や絞込みなどの処理をして更新をします

## UPDATE文1

UPDATEでは、直後にテーブル名を指定して、その後にSETと更新するカラムを、で区切って更新する値を設定します

UPDATE テーブル名

```
SET COLUMN1=〇〇, COLUMN2= × × ;
```

## UPDATE文2(特定の行を更新)

UPDATEの直後にWHEREを記述すると、更新する対象の行を絞り込んで特定の行に対して更新ができる

UPDATE テーブル名

```
SET COLUMN1=〇〇, COLUMN2= × ×
```

```
WHERE name="Taro";
```

# 1. 副問い合わせでの絞込み条件をUPDATEに用いる

SELECTで取得されるカラムを用いることができる。SELECTをUPDATEに変更して、その直後にSELECTで取得対象のテーブルを記述する。その後に、**SET**を記述して「更新するカラム=更新する値」を記述する

```
SELECT
  lastName, firstName
FROM employees AS emp
WHERE
  emp.office_code IN (SELECT office.office_code FROM offices WHERE office.country = 'USA');
```

```
UPDATE employees AS emp
SET emp.office_name = UPPER(emp.office_name)
WHERE
```



**employees** テーブルの **office\_name** を大文字 にして更新

```
emp.office_code IN (SELECT office.office_code FROM offices WHERE office.country = 'USA');
```

## 2. 他のテーブルと連結して連結されたテーブルの値を用いる

SELECTをUPDATEに変更して、その直後に更新対象のテーブルを記述する。その後に、INNER JOINを記述して、他のテーブルと紐づける。

SETを記述して「更新するカラム=他のテーブルから取得した値」を記述する

```
SELECT
```

```
  *
```

```
FROM employees AS emp
```

```
INNER JOIN departments AS dp
```

```
ON dp.id = emp.department_id AND emp.department_id = 2;
```



INNER JOINをUPDATEとSETの間に記述する

```
UPDATE employees AS emp
```

```
INNER JOIN departments AS dp
```

```
ON dp.id = emp.department_id AND emp.department_id = 2
```

```
SET emp.department_name = dp.name
```

### 3. WITHを利用して紐づけて更新

WITHを利用して、一時テーブルと紐づけることで複雑な更新処理を簡単に実装できるようになる

```
WITH tmp_table AS(
    SELECT cst.id AS id, SUM(od.order_amount * od.order_price) AS payment
    FROM customers AS cst
    INNER JOIN orders AS od
    ON cst.id = od.customer_id
    GROUP BY cst.id
)
UPDATE customers
INNER JOIN tmp_table
ON customers.id = tmp_table.id
SET customers.payment = tmp_table.payment;
```

WITHと作成した中間テーブル(tmp\_table)とINNER JOINで紐づける。SETで更新処理をする

# DELETE処理の応用

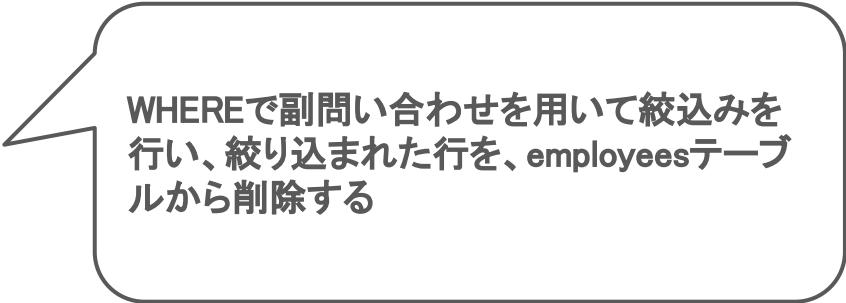
DELETE処理内で、複雑な絞込みを行い絞り込まれた対象のテーブルの行を削除する

## DELETE文の基本的な書き方

```
DELETE FROM テーブル名 WHERE name="Taro";
```

## 副問い合わせを用いたDELETE文

```
DELETE FROM  
  employees  
WHERE  
  office_code IN (SELECT  
    office_code  
  FROM  
    offices  
  WHERE  
    country = 'USA');
```



WHEREで副問い合わせを用いて絞込みを行い、絞り込まれた行を、employeesテーブルから削除する



# INSERT処理の応用

副問い合わせや、JOINで連結した結果をテーブルに格納する

## INSERT文の基本的な書き方

INSERT INTO テーブル名(カラム1, ...) VALUES (値1, ...);

## SELECTの結果をINSERTでテーブルに挿入する

**INSERT INTO** tmp\_customers

**SELECT**

od.id, CONCAT(ct.last\_name, ct.first\_name), od.order\_date

**FROM**

orders AS od

**INNER JOIN**

customers AS ct

**ON**

od.customer\_id = ct.id;

INNER JOINで、テーブル同士を紐づけ、その結果をINSERTでテーブルに挿入する

# カラムの制約

# NOT NULL制約(NULL値を入れない)

NOT NULL制約を追加すると、特定のカラムにNULLを入れることができなくなる  
(**DEFAULT**とともに利用されることが多いです)

```
name VARCHAR(100) DEFAULT '名無し' NOT NULL
```

**NULLの値を挿入できず、値を指定せずINSERTした場合は、「名無し」が格納される**

```
CREATE TABLE users(  
  id INT PRIMARY KEY,  
  name VARCHAR(255) DEFAULT '名無し' NOT NULL,  
  age INT  
);
```

INSERT INTO users(id, age) VALUES(1, 23); # idとageに値を入れるとnameを指定しないとデフォルトの¥名無しが入る

```
SELECT * FROM users; # 1 名無し 23
```

# UNIQUE制約(同じ値を入れない)

UNIQUE制約を追加すると、特定のカラムに対して重複を許さなくできるようになる  
(NULLは重複とカウントされず、複数入れることができる)

```
name VARCHAR(100) UNIQUE
```

UNIQUE制約をつけると,nameカラムには同じ値が入れられなくなる

```
CREATE TABLE users(  
  id INT PRIMARY KEY,  
  name VARCHAR(255) UNIQUE,  
  age INT  
);
```

```
INSERT INTO users VALUES(1, "Taro", 23);
```

```
INSERT INTO users VALUES(2, "Taro", 24);# nameが重複してエラーになる
```

# CHECK制約(指定した制約を自由にいれる)

CHECK制約を追加すると、特定のカラムに対して指定した条件を満たす値しか格納することができなくなる

ageが18以上でない場合は、エラーになるCHECK制約をつける

```
age INT CHECK(age >= 18)
```

ageが18以上かつcityがTokyoでない場合は、エラーになるCHECK制約をつける

```
CONSTRAINT chk_person CHECK (age>=18 AND city='Tokyo')
```

```
CREATE TABLE persons (  
  id INT PRIMARY KEY,  
  name varchar(255) CHECK(),  
  age int,  
  city varchar(255),  
  CONSTRAINT chk_person CHECK (age>=18 AND city=Tokyo)  
);
```

```
INSERT INTO persons VALUES(1, "Taro", 17, "Tokyo"); # ageが18未満のためエラー
```

# 主キー制約(PRIMARY KEY)

主キー制約をつけると特定のカラムに対して、NOT NULL制約、UNIQUE制約が付き、インデックスも付与される

## idカラムを主キーに設定する

```
id INT PRIMARY KEY
```

## idとnameカラムを主キーに設定する

```
PRIMARY KEY(id, name)
```

idとnameはNULLとして設定できず、同じ値の組み合わせを格納できない

```
CREATE TABLE persons(  
  id INT,  
  name VARCHAR(255),  
  PRIMARY KEY(id, name)  
);
```