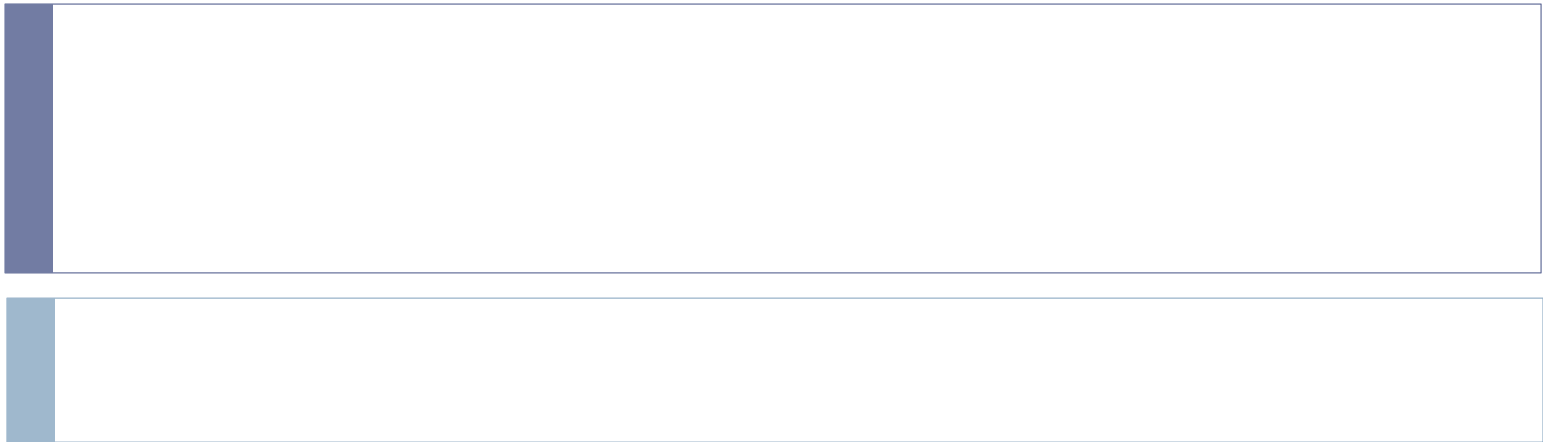


# SQL補講(6月)



補講

# 目次

---

- ▶ 正規化
- ▶ 結合
- ▶ timestamp

# 正規化

- ▶ 例えば小売業の会社で以下のような売上のテーブルを使っているとしても実務的には問題ない

id	user_name	item_name	price	created_at	updated_at
1	tomihara	おにぎり	120	2024/06/01	2024/06/01
2	ueda	本	600	2024/06/02	2024/06/02
3	higuchi	ジュース	110	2024/06/03	2024/06/05
4	higashi	おかし	160	2024/06/04	2024/06/04
5	tomihara	本	3000	2024/06/05	2024/06/07

- ▶ ただし、このままだと、以下のような不具合が起こってしまう可能性がある。
  - ▶ 「tomihara」を「tomita」と間違えて入力してしまうかもしれない。
  - ▶ 同性の「tomihara」がいた時にこのままでは区別出来ない。
  - ▶ 「本」を「雑誌」に変えようとした時に今まで入力されたすべての「本」を変える必要がある。
  - ▶ おにぎりの金額が変更になった時、今までの癖で古い金額を入力してしまうかもしれない。
  - ▶ 売上が発生しない限り、見込みのお客さんや新商品の情報を登録できない。
  - ▶ などなど

- 
- ▶ 前ページの不具合が起きないようにするために行うのが正規化というテーブルを分割するルールです。
  - ▶ 正規化は、初めてテーブルを作成する人でも、大ベテランの人でも同じ様にテーブルを分割して効率のいいテーブルを作成することができるようにするためのルールです。
  - ▶ もちろん分割したテーブルをつないで元のテーブルに戻すことができます。(情報無損失分解)
    - ▶ これを、結合といいます。
  - ▶ 内部結合、(左右)外部結合というものがあります。

# 前提

---

- ▶ 3つのテーブルは正規化済みである。
- ▶ USERSテーブル
  - ▶ ユーザー(お客さん)の情報を集めたテーブル
  - ▶ idは主キーで自動採番
  - ▶ created\_at, updated\_atについては省略しています。

id(PK)	user_name	created_at	updated_at
1	tomihara		
2	higuchi		
3	ueda		
4	higashi		

- ▶ user\_nameの変更はこのテーブルで変更すればいい。
- ▶ userの登録と売上が全く違うテーブルで管理されることになる。

## ▶ ITEMSテーブル

### ▶ 商品の情報を集めたテーブル

id(PK)	item_name	price	created_at	updated_at
1	おにぎり	120		
2	本	600		
3	おかし	110		
4	ジュース	160		
5	めがね	3000		

- ▶ item\_nameが変更になったらこのテーブルのカラム値を変更すればいい。
- ▶ item\_priceが変更になったらこのテーブルのカラム値を変更すればいい。

---

## ▶ SALESテーブル

### ▶ 商品の売上情報を集めるテーブル

- ▶ 売上が発生する毎に1行(レコード)のデータが追加される

id(PK)	user_id	item_id	created_at	updated_at
1	1	1		
2	3	2		
3	2	4		
4	4	3		
5	1	2		



# 内部結合

- ▶ 内部結合というのは結合するテーブルのどちらにもあるものだけを結合する方法のことです。
- ▶ USERSテーブルとSALESテーブルをid(USERSテーブル)とuser\_id(SALESテーブル)を使って内部結合すると以下ようになります。

id(PK)	user_id	item_id
1	1	1
2	3	2
3	2	4
4	4	3
5	1	2

SALESテーブル

id(PK)	user_name
1	tomihara
2	higuchi
3	ueda
4	higashi

USERSテーブル

id(PK)	user_id	user_name	item_id
1	1	tomihara	1
2	3	ueda	2
3	2	higuchi	4
4	4	higashi	3
5	1	tomihara	2

- ▶ SQLでは以下のように記述します。
- ▶ 今回は左側のSALESテーブルと右側のUSERSテーブルをusers\_idを使って結合。
- ▶ SALESテーブルにUSERSテーブルのusers\_nameカラムの情報を追加して表示しています。

▶ SELECT

s.id,s.user\_id,u.user\_name,s.item\_id

FROM

SALES as s

主(左側)になるテーブル

(INNER) JOIN

USERS as u

結合(右側)するテーブル

ON

s.users\_id = u.id

# 複数テーブルの結合

---

- ▶ 2つ以上のテーブルを結合したい時はJOIN句以降を追加します。

- ▶ SELECT

```
s.id,s.user_id,u.user_name,s.item_id,i.item_name  
FROM SALES as s  
JOIN USERS as u ON s.user_id = u.id  
JOIN ITEMS as i ON s.item_id = i.id;
```

# データの追加


- ▶ SALESテーブルに以下のようなデータを追加した時、どの様になるでしょうか？

id(PK)	user_id	item_id
6	5	6

- ▶ 内部結合の結果は同じになります。
- ▶ 今回追加したuser\_idが「5」のデータがUSERSテーブルには無いためどちらにもあるuser\_idが表示される内部結合では結果に変化はありません。

# 外部結合

- ▶ 左外部結合を使うと以下の事ができます。
  - ▶ SALESテーブルはすべて表示する。
  - ▶ USERSテーブルにあるものはuser\_nameを表示する。
- ▶ 左なのは、今回のSQLでSALESテーブルを左(主)に置いているからです



id(PK)	user_id	item_id
1	1	1
2	3	2
3	2	4
4	4	3
5	1	2
6		6

SALESテーブル

id(PK)	user_name
1	tomihara
2	higuchi
3	ueda
4	higashi

USERSテーブル

- 
- ▶ SELECT  
    s.sale\_id, s.user\_id, u.user\_name, s.item\_id  
FROM  
    SALES as s  
LEFT JOIN  
    USERS as u  
ON  
    s.user\_id = u.id;
  - ▶ USERSテーブルにない「id」が「5」の行の「user\_name」にはnullが入っています
  - ▶ もちろんLEFT JOINの部分をRIGHT JOINとするとINNER JOINのときと同じ結果になります。

## 参照制約(外部キー制約)

- ▶ SALESテーブルにデータを追加する時に、USERSテーブルのidにデータがあるものしか登録できないようにすることができます。
- ▶ CREATE TABLE SALES\_LIMIT (  
    id                INT AUTO\_INCREMENT PRIMARY KEY,  
    user\_id          VARCHAR(20) NOT NULL,  
    item\_id          VARCHAR(20) NOT NULL,  
    created\_at       TIMESTAMP,  
    updated\_at       TIMESTAMP,  
    FOREIGN KEY (user\_id) REFERENCES USERS(id)  
);

USERSテーブルのid



# TIMESTAMP関連

- ▶ 残念ながらH2にはTIMESTAMP型をDATE型やTIME型に変換する機能はありません。文字列の切り出しを使って必要な部分を切り出すことになります。
  - ▶ 日付部分のみ
    - ▶ `SELECT LEFT(created_at,10) FROM SALES;`
  - ▶ 時間部分のみ
    - ▶ `SELECT SUBSTRING (created_at,12,8) FROM SALES;`
- ▶ 今日より前(今日を含む)のデータは以下のようなSQLです
  - ▶ `SELECT * FROM ITEMS`  
`WHERE LEFT(created_at,10) <= CURRENT_DATE;`
- ▶ 1日前から今日まで(1日前を含まない)のデータは以下のようなSQLです
  - ▶ `SELECT * FROM ITEMS`  
`WHERE LEFT(created_at,10) > (CURRENT_DATE-1);`