データベース実習

05. テーブルの結合

株式会社ジードライブ

今回学ぶこと

- テーブルの結合(JOIN)

例題

- ある文具店で利用する販売記録テーブルを考える

販売記録 (sales)

ID	メーカー	商品名	価格	個数	日付
1001	BILOT	万年筆	10500	2	2021-07-12
1002	UNITED	名刺入れ	1260	1	2021-07-13
1003	NEVA	システム手帳	8400	2	2021-07-14
1004	UNITED	名刺入れ	1260	3	2021-07-14

テーブルの分解

- テーブル内で扱う項目が増えてくると、
  重複した項目が目立つようになってくる

販売記録 (sales)

ID	メーカー	商品名	価格	個数	日付
1001	BILOT	万年筆	10500	2	2021-07-12
1002	UNITED	名刺入れ	1260	1	2021-07-13
1003	NEVA	システム手帳	8400	2	2021-07-14
1004	UNITED	名刺入れ	1260	3	2021-07-14

同じ

テーブルの分解

- 商品の情報だけ「商品テーブル」として別テーブルに分ける
  - 販売記録のテーブルには、商品のIDのみを格納する

販売記録 (sales)

ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

関連
(リレーション)

商品情報を管理する
テーブルを別に作ろう！

商品 (goods)

ID	メーカー	商品名	価格
1	BILOT	万年筆	10500
2	UNITED	名刺入れ	1260
3	NEVA	システム手帳	8400

テーブル分解のメリット（1）

- データ量を削減できる
  - テーブル内で重複した情報を一か所に
    まとめられるため、データ量が削減できる

販売記録 (sales)

ID	商品ID	個数	日付
1001	1	2	2021-07-12
1003	2	1	2021-07-13
1004	3	2	2021-07-14
1005	2	3	2021-07-14

商品 (goods)

ID	メーカー	商品名	価格
1	BILOT	万年筆 A型	10500
2	UNITED	名刺入れ	1260
3	NEVA	システム手帳	8400

テーブル分解のメリット（2）

- 情報が局所化できる
  - 例えば、商品の情報（商品名等）に変更があった場合、一か所を変更すれば済む

販売記録（sales）

ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

ここだけ変更

商品（goods）

ID	メーカー	商品名	価格
1	BILOT	万年筆　A型	10500
2	UNITED	名刺入れ	1260
3	NEVA	システム手帳	8400

テーブル分解のメリット（3）

- データを事前登録できる
  - 例えば、まだ販売実績のない商品の情報も事前に登録しておける

販売記録（sales）

ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

「4」や「5」は無い

商品（goods）

ID	メーカー	商品名	価格
1	BILOT	万年筆 A型	10500
2	UNITED	名刺入れ	1260
3	NEVA	システム手帳	8400
4	BILOT	万年筆 B型	12600
5	BILOT	ペンケース	5250

テーブルの結合

- 販売記録を画面に表示する際には、分割したテーブルの両方から情報を取り出す必要がある

ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

＋

ID	メーカー	商品名	価格
1	BILOT	万年筆 A型	10500
2	UNITED	名刺入れ	1260
3	NEVA	システム手帳	8400
4	BILOT	万年筆 B型	12600
5	BILOT	ペンケース	5250

販売ID	メーカー	商品名	価格	個数	日付
1001	BILOT	万年筆 A型	10500	2	2021-07-12
1002	UNITED	名刺入れ	1260	1	2021-07-13
1003	NEVA	システム手帳	8400	2	2021-07-14
1004	UNITED	名刺入れ	1260	3	2021-07-14

結合の種類

- 内部結合
  - INNER JOIN (= JOIN)
- 外部結合
  - LEFT OUTER JOIN (= LEFT JOIN)
  - RIGHT OUTER JOIN (= RIGHT JOIN)

INNER JOIN (JOIN)

- 結合するテーブル双方にデータが存在する場合に、データが取得される

販売ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

＋

IDが3番の商品は存在しない

商品ID	商品名	価格
1	万年筆 A型	10500
2	名刺入れ	1260

↓

販売ID	商品ID	個数	日付	商品名	価格
1001	1	2	2021-07-12	万年筆 A型	10500
1002	2	1	2021-07-13	名刺入れ	1260
1004	2	3	2021-07-14	名刺入れ	1260

販売ID1003番の記録は取得されない

LEFT OUTER JOIN (LEFT JOIN)

- 結合される側(左側)のテーブルにあるデータは、すべて取得される

結合される側

販売ID	商品ID	個数	日付
1001	1	2	2021-07-12
1002	2	1	2021-07-13
1003	3	2	2021-07-14
1004	2	3	2021-07-14

結合する側

商品ID	商品名	価格
1	万年筆 A型	10500
2	名刺入れ	1260

IDが3番の商品は存在しない

販売ID1003番の記録も取得される

販売ID	商品ID	個数	日付	商品名	価格
1001	1	2	2021-07-12	万年筆 A型	10500
1002	2	1	2021-07-13	名刺入れ	1260
1003	3	2	2021-07-14	NULL	NULL
1004	2	3	2021-07-14	名刺入れ	1260

RIGHT OUTER JOIN (RIGHT JOIN)

- 結合する側(右側)のテーブルにあるデータは、すべて取得される

結合される側

販売ID	商品ID	個数	日付
1001	3	2	2021-07-12
1002	2	1	2021-07-13

商品ID１番が販売記録には現れていない

＋

結合する側

商品ID	商品名	価格
1	万年筆 A型	10500
2	名刺入れ	1260
3	システム手帳	8400

⬇

販売ID	商品ID	個数	日付	商品名	価格
1001	3	2	2021-07-12	システム手帳	8400
1002	2	1	2021-07-13	名刺入れ	1260
NULL	**1**	**NULL**	**NULL**	**万年筆 A型**	**10500**

販売記録には現れてないが、商品として存在するので取得される

JOINによるテーブルの結合

- 書式

```
SELECT … FROM テーブル１ JOIN テーブル２
  ON テーブル１.カラム１ = テーブル２.カラム２
```

  - JOIN を LEFT JOIN や RIGHT JOIN とすることも可能
  - テーブル１：結合される側
  - テーブル２：結合する側
  - ON の後で、結合させるカラム同士を = で示す
    ⇒ テーブル名とカラム名を . (ドット)で繋げる

例：salesテーブルのgoods_id に goodsテーブルのidを結合

```
SELECT * FROM sales JOIN goods ON sales.goods_id = goods.id;
```

JOINの注意点

- JOINは、データを取得する際に複数のテーブルを連携して取得するものであり、データベース内で複数のテーブルが実際に結合して別のテーブルになるわけではない

キーワードの順番

- 複数のキーワードを同時に指定する場合は以下の順序で指定する：

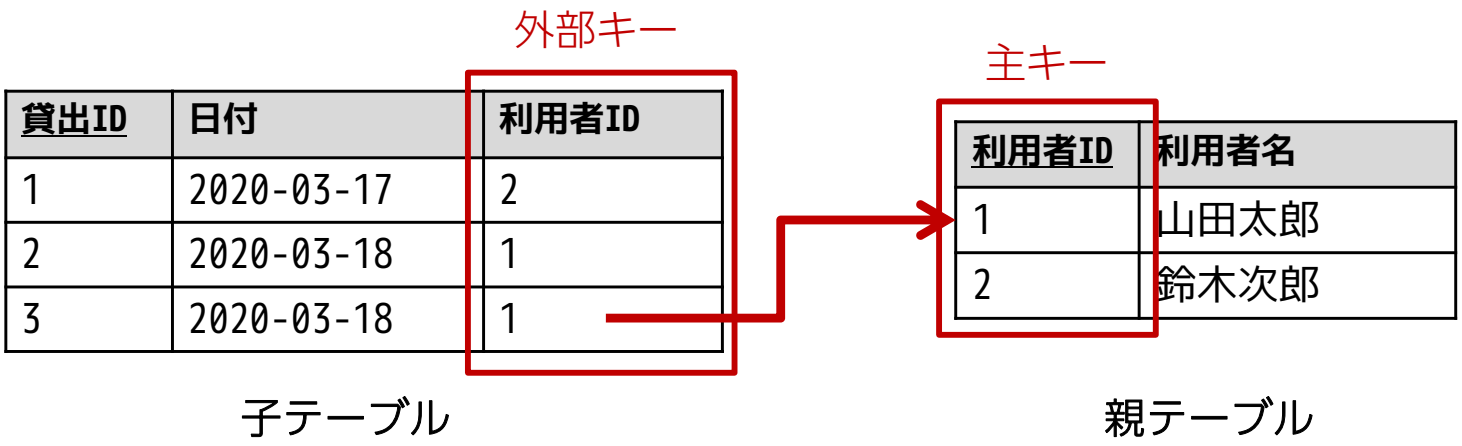
```
SELECT カラム名
  FROM テーブル名
  JOIN テーブル名 ON …
  WHERE 条件
  ORDER BY カラム名 ASC/DESC
  LIMIT 開始位置,件数
```

練習

- 練習05-1
- 練習05-2
- 練習05-3

外部キー制約

- 関連のある２つのテーブル間でデータの整合性を保つための制約
  - 別テーブルの主キーを参照するカラムに設定する
  - このとき、主キーを持つテーブルを親テーブル、外部キーを持つテーブルを子テーブルという



外部キー制約の追加

- 書式

```
ALTER TABLE 子テーブル名 ADD CONSTRAINT 外部キー名
  FOREIGN KEY (カラム名) REFERENCES 親テーブル名(カラム名)
  参照オプション;
```

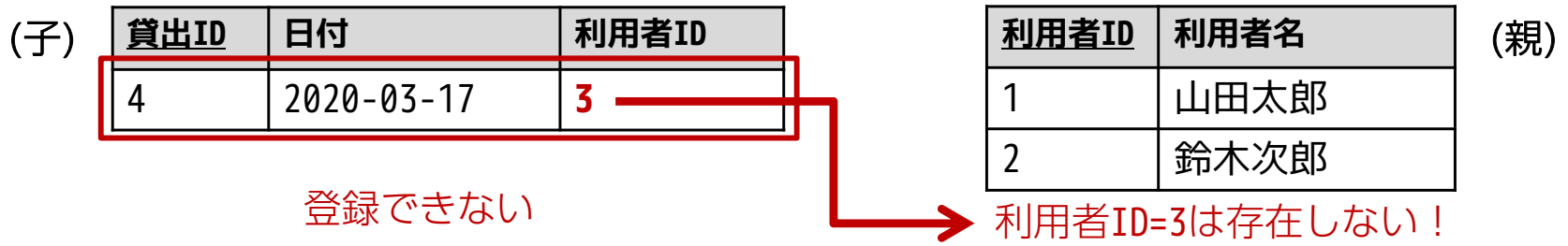
参照オプション … 親テーブルのデータを更新(UPDATE)または削除(DELETE)した場合に、そのデータを参照している子テーブルのデータがどうなるかを指定するオプション(後述)

例：loansテーブルのuser_idに外部キー制約を設定する

```
ALTER TABLE loans ADD CONSTRAINT fk_loans_users
  FOREIGN KEY (user_id) REFERENCES users (id)
  ON DELETE RESTRICT ON UPDATE RESTRICT;
```

外部キーの働き

- 親テーブルに存在しないデータを子テーブルに登録しようとするとエラーとなる



- その他の働きについては、参照オプションの内容によって変わる

参照オプション(ON DELETE)

- ON DELETEは、親テーブルのデータが削除される時の振る舞いを指定する
  - ON DELETE省略時は RESTRICT となる

オプション	振る舞い
ON DELETE RESTRICT	親データを参照している子データがある場合、エラーが発生する
ON DELETE NO ACTION	RESTRICTと同じ
ON DELETE CASCADE	親データを参照している子データも削除される
ON DELETE SET NULL	親データを参照している子データの参照カラムがnullになる

ON DELETE RESTRICT

- 子テーブルが参照しているデータを親テーブルから削除しようとするとエラーになる

(子)

貸出ID	日付	利用者ID
1	2020-03-17	2
2	2020-03-18	1
3	2020-03-18	1

参照

利用者ID	利用者名
1	山田太郎
2	鈴木次郎

(親)

削除できない
(エラーになる)

- ON DELETE NO ACTIONも同じ振る舞いとなる

ON DELETE CASCADE

- 子テーブルが参照しているデータを親テーブルから削除すると、子テーブルのデータも削除される

(子)

貸出ID	日付	利用者ID
1	2020-03-17	2
2	2020-03-18	1
3	2020-03-18	1

参照

(親)

利用者ID	利用者名
1	山田太郎
2	鈴木次郎

(子)

貸出ID	日付	利用者ID
~~1~~	~~2020-03-17~~	~~2~~
2	2020-03-18	1
3	2020-03-18	1

削除

(親)

利用者ID	利用者名
1	山田太郎
~~2~~	~~鈴木次郎~~

削除

利用者ID=2の鈴木さんを削除すると、それを参照している貸出ID=1のデータも削除される

ON DELETE SET NULL

- 子テーブルが参照しているデータを親テーブルから削除すると、子テーブルの参照情報がnullになる

(子)

貸出ID	日付	利用者ID
1	2020-03-17	2
2	2020-03-18	1
3	2020-03-18	1

参照

(親)

利用者ID	利用者名
1	山田太郎
2	鈴木次郎

(子)

貸出ID	日付	利用者ID
1	2020-03-17	null
2	2020-03-18	1
3	2020-03-18	1

(親)

利用者ID	利用者名
1	山田太郎
~~2~~	~~鈴木次郎~~

削除

利用者ID=2の鈴木さんを削除すると、それを参照している貸出ID=1のデータの利用者IDはnullになる

外部キーの確認

- 外部キーは、テーブル情報パネルのForeign Keysタブで確認できる