

テーブル設計・正規化

データベースのテーブルについて
まだ説明していなかったことについて補足します。

- ・主キー 外部キー 制約について
- ・データ型
- ・テーブル設計
カラムの分割 正規化

教科書のP8～P17あたりに該当しますので
また時間があるときにそちらの説明も見てみてください。
今回は授業用に少し改変して話させていただきます。

■ 主キー

テーブルの列を選んで設定します。

テーブルには必ず主キーを設定しなければならず、
主キーがないテーブルはRDBでは認められません。

サンプルデータベースのbooksテーブルでは「isbn」列に主キーが設定されている
その他のテーブルの主キーも確認してみましょう

●書籍情報テーブル (books)

列名	データ型	概要
isbn *	CHAR(17)	ISBN コード
title		書名
price	INT	価格
publish	VARCHAR(30)	出版社
publish_date	DATE	刊行日
category_id	CHAR(5)	分類 ID

この本では*が目印です

・書籍情報テーブル(books)

isbn	title	price	publish	publish_date	category_id
4-0010-0000-0	ハムスターの観察	900	山田出版	2010-11-01	Z9999
4-0010-0000-1	PHPドリル	5100	山田出版	2013-01-14	P1111
4-0010-0000-4	フェレットの観察	1000	山田出版	2012-10-26	Z9999
4-0010-0000-5	らくだの観察日記	1100	山田出版	2012-12-24	Z9999
4-0010-0000-6	あひるの観察日記	700	山田出版	2012-11-15	A9999
4-0010-0000-7	かえるの観察日記	800	山田出版	2013-01-15	H9999
4-0010-0000-9	SQL入門	5500	山田出版	2012-10-30	S2222
4-7980-0522-3	JSPリファレンス	1800	秀和システム	2010-04-19	J4444
4-7980-0945-8	PHP5サンプル集	3000	秀和システム	2012-11-01	P1111
4-7981-0722-0	XML辞典	3300	翔泳社	2011-09-16	X3333

主キーに設定された列は特別な役割を担います。
テーブルの中からデータを一意に特定するという役割です。
イメージは↓のような感じ

isbn 「4-0010-0000-7」といえば？

・書籍情報テーブル(books)

	isbn 	title	price	publish	publish_date	category_id
▶	4-0010-0000-0	ハムスターの観察	900	山田出版	2010-11-01	Z9999
	4-0010-0000-1	PHPドリル	5100	山田出版	2013-01-14	P1111
	4-0010-0000-4	フェレットの観察	1000	山田出版	2012-10-26	Z9999
	4-0010-0000-5	らくだの観察日記	1100	山田出版	2012-12-24	Z9999
	4-0010-0000-6	あひるの観察日記	700	山田出版	2012-11-15	A9999
	4-0010-0000-7	かえるの観察日記	800	山田出版	2013-01-15	H9999
	4-0010-0000-9	SQL入門	5500	山田出版	2012-10-30	S2222
	4-7980-0522-3	JSPリファレンス	1800	秀和システム	2010-04-19	J4444
	4-7980-0945-8	PHP5サンプル集	3000	秀和システム	2012-11-01	P1111
	4-7981-0722-0	XML辞典	3300	翔泳社	2011-09-16	X3333

かえるの観察日記というタイトルで
価格が800円
山田出版から出ていて
出版日は2013-01-15
カテゴリIDはH9999

主キー列の値が1つ決まれば、1レコードについて特定できます。

この状態を実現するために主キーになる列は「UNIQUE制約(ほかのデータと重複してはいけない)」

「非NULL制約(NULLであってはならない)」を満たしている必要があります。

「UNIQUE制約(ほかのデータと重複してはいけない)」

「非NULL制約(NULLであってはいけない)」を守ればどんな列でも主キーに設定することができますが、書籍タイトルや氏名などに設定すると問題が発生する場合があります。

・書籍情報テーブル(books)

	isbn	title	price	publish
▶	4-0010-0000-0	ハムスターの観察	900	山田出版
	4-0010-0000-1	PHPドリル	5100	山田出版
	4-0010-0000-4	フェレットの観察	1000	山田出版
	4-0010-0000-5	らくだの観察日記	1100	山田出版
	4-0010-0000-6	あひるの観察日記	700	山田出版
	4-0010-0000-7	かえるの観察日記	800	山田出版
	4-0010-0000-9	SQL入門	5500	山田出版
	4-7980-0522-3	JSPリファレンス	1800	秀和システム
	4-7980-0945-8	PHP5サンプル集	3000	秀和システム
	4-7981-0722-0	XML辞典	3300	翔泳社

・アンケート回答テーブル(quest)

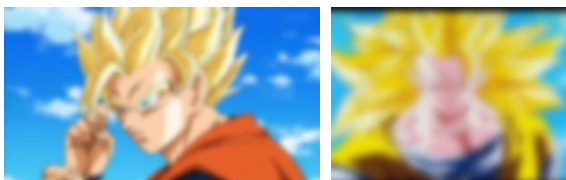
	id	name	name_kana	sex	prefect
▶	1	山田太郎	ヤマダタロウ	男	東京都
	2	井上茉莉	イノウエマリ	女	神奈川県
	3	上原遙	ウエハラハルカ	女	茨城県
	4	江本修子	エモトシュウコ	女	東京都
	5	小野博美	オノヒロミ	女	神奈川県
	6	河合太郎	カワイタロウ	男	東京都
	7	矢口一樹	ヤグチカズキ	男	茨城県
	8	有木守	アリキマモル	男	千葉県
	9	吉岡毅	ヨシオカツヨシ	男	埼玉県
	10	和田駿	ワダシュン	男	東京都

まったく同じタイトルの別の本がでてきた場合2冊目はUNIQUE制約に引っかかってしまいレコードの追加ができません。
おなじようにすでに山田太郎さんがいるところに2人目の山田太郎さんを追加することはできません。

「山田太郎2」「山田太郎3」のように工夫して入力することでこの問題は一時的に回避することができますが、

データベースの運用としてはこの方法はよくない選択です。

主キーには重複しないIDやコードを利用しましょう



複数の列の組み合わせを主キーとする複合キーというものもあります

●タイムカードテーブル (time_card)

列名	データ型	概要
s_id *	CHAR(7)	社員コード
r_date *	DATE	日付
work_time	DECIMAL	勤務時間

組み合わせが一つずつ違う
値を特定するためには
「s_id」と「r_date」が必要



・タイムカードテーブル(time_card)

s_id	r_date	work_time
DA00001	2012-11-01	9
DA00001	2012-11-02	8
DA00001	2012-11-03	10
DA00001	2012-11-04	9
DA00001	2012-11-07	8
DA00001	2012-11-08	8
DA00001	2012-11-09	8
DA00001	2012-11-10	8
DA00001	2012-11-11	9
DA00001	2012-11-14	8
DA00001	2012-11-15	10

主キーがあることにより
テーブル内に重複したデータが存在するのを防ぎ
データを矛盾なく保存することが出来るようになります。

■ 外部キー

外部キーも列に対して設定できるものです。
テーブル同士をつなぐリレーションシップで使用する列となります。
サンプルデータベースのemployeeテーブル、time_cardテーブル、depart
テーブルで外部キーについて確認しましょう

主キー  外部キー  とさせていただきます。

↓ 次ページで確認

●社員テーブル (employee)

列名	データ型	概要
s_id *	CHAR(7)	社員コード
l_name	VARCHAR(20)	社員名 (氏)
f_name	VARCHAR(20)	社員名 (名)
l_name_kana	VARCHAR(100)	社員名 (氏カナ)
f_name_kana	VARCHAR(100)	社員名 (名カナ)
sex	SMALLINT	性別 (1: 男、2: 女、3: その他)
class	VARCHAR(20)	役職 (部長、課長、主任、担当、アシスタント)
depart_id	CHAR(3)	所属部署コード
b_id	CHAR(7)	上司コード (社員コードに対応。上司がない社員は NULL)
last_update	DATE	最終更新日
retired	SMALLINT	退職済みか (1: 退職済み 0: 在職)

●タイムカードテーブル (time_card)

列名	データ型	概要
s_id *	CHAR(7)	社員コード
r_date *	DATE	日付
work_time	DECIMAL	勤務時間

●所属部署テーブル (depart)

列名	データ型	概要
depart_id *	CHAR(3)	所属部署コード
depart_name	VARCHAR(20)	所属部署名

●タイムカードテーブル (time_card)

s_id	r_date	work_time
DA00001	2012-11-01	9
DA00001	2012-11-02	8
DA00001	2012-11-03	10
DA00001	2012-11-04	9
DA00001	2012-11-07	8
DA00001	2012-11-08	8
DA00001	2012-11-09	8
DA00001	2012-11-10	8
DA00001	2012-11-11	9
DA00001	2012-11-14	8
DA00001	2012-11-15	10

●所属部署テーブル (depart)

depart_id	depart_name
E01	第一営業部
E02	第二営業部
J01	人事部
S01	経営企画部
S02	総務部

●社員テーブル (employee)

s_id	l_name	f_name	l_name_kana	f_name_kana	sex	class	depart_id	b_id	last_update	retired
AI00001	相沢	聡	アイザワ	サトシ	1	部長	S02	NULL	2012-02-24	0
DA00001	大門	一郎	ダイモン	イチロウ	1	担当	J01	AI00001	2011-08-10	0
FU00001	藤井	雄太	フジイ	ユウタ	1	主任	E01	SE00001	2011-04-10	0
FU00002	藤岡	幸太郎	フジオカ	コウタロウ	1	担当	E01	FU00001	2011-06-12	0
HA00001	速水	和幸	ハヤミ	カズユキ	1	部長	E01	NULL	2007-02-03	1
HA00002	葉山	俊輔	ハヤマ	シュンスケ	1	アシスタント	E01	FU00002	2012-12-11	0
KA00001	川口	裕子	カワグチ	ユウコ	2	アシスタント	S01	KI00001	2012-12-28	1
KA00002	加藤	昭雄	カトウ	アキオ	1	アシスタント	S01	KI00001	2011-11-11	0
KA00003	神田	佐知子	カンダ	サチコ	1	アシスタント	J01	KA00003	2012-01-28	0
KI00001	木村	一郎	キムラ	イチロウ	1	担当	S01	TA00001	2011-07-12	0
NA00001	中野	康代	ナカノ	ヤスヨ	2	アシスタント	S02	TA00002	2012-04-17	1

3つのテーブルが外部キーに設定された列を通してつながっているのが確認できたかと思います。

このようなつながりを「**リレーションシップ**」といいます。

外部キーの役割はテーブル同士をつなぐことと主キーと同様、データの矛盾を防ぐことです。

外部キーを設定する列には参照制約(テーブル間で対応するデータがあること)が課されます

参照制約については次ページで具体的な例で説明します

●社員テーブル (employee)

列名	データ型	概要
s_id *	CHAR(7)	社員コード
l_name	VARCHAR(20)	社員名 (氏)

・タイムカードテーブル(time_card)

s_id	r_date	work_time
DA00001	2012-11-10	9

参照制約により以下の操作はできないようになっていきます！

employeeテーブルに存在しない社員の勤怠記録の追加

work_time	DECIMAL	勤務時間
-----------	---------	------

●所属部署テーブル (depart)

列名	データ型	概要
depart_id *	CHAR(3)	所属部署コード
depart_name	VARCHAR(20)	所属部署名

所属部署テーブル(depart)

depart_id	depart_name
E01	第一営業部
E02	第二営業部
J01	人事部
S01	経営企画部
S02	総務部

・社員テーブル(employee)

s_id	l_name	f_name	l_name_kana	f_name_kana	sex	class	depart_id	b_id	last_update	retired
AI00001	相沢	聡	アイザワ	サトシ	1	部長	S02	NULL	2012-02-24	0
1					1	担当	J01	AI00001	2011-08-10	0
1					1	主任	E01	SE00001	2011-04-10	0
1					1	担当	E01	FU00001	2011-06-12	0
1					1	部長	E01	NULL	2007-02-03	1
1					1					
2					1					
1					1					
1					1					
1					1					
2					2	アシスタント	S02	TO000002	2012-04-17	1

社員 相沢聡 さんのレコードの削除

※time_cardテーブルに相沢さんのs_idを参照しているレコードがあるため、完全に消したい場合はtime_cardテーブルの相沢さんに関するレコードをすべて削除したのち、employeeテーブルの相沢さんのレコードが削除可能となります

departテーブルに存在しないような部署例えば「Z01」などのdepart_idを設定した社員の追加

外部キー(参照制約)があることにより
テーブルにおかしなデータが紛れ込むのを防ぎ
矛盾のないデータベースを作ることが出来るようになります。

■ データ型

本研修で今まで説明をしてきませんでした、
データベースのテーブルには様々なデータ型を設定することが出来ます。
格納したいデータに応じて適切なデータ型を選ぶことが必要になります。

id列 ⇒ 数値型

名前列 ⇒ 文字列型

登録日列 ⇒ 日付型

id		名前		登録日	
1	数値のみOK	竈門炭治郎	文字列のみOK	2023/05/02	日付のみOK
2		竈門禰豆子		2023/05/02	
3		我妻善逸		2023/05/02	
4		嘴平伊之助		2023/05/02	
5		サイコロステーキ先輩		2023/05/02	

■ テーブル設計 カラムの分割

テーブルの列はデータの意味が壊れてしまわない程度に細かく分割することをおすすめします。

例えば

「氏名」という一つの列にするのではなく「氏」と「名」で列を分ける

「住所」という一つの列にするのではなく「県名」「市町村名」「番地・マンション名」に分ける

といった具合です。

なぜかというと

複数の列をつなぎ合わせることはCONCAT関数などを使えば簡単に行えるのに
対して、一つの列を分解することは、SQLの力だけでは難しいからです。

あらかじめ、扱いやすいようにデータを格納しておきましょう。

id	氏名
1	坂上田村麻呂
2	ヘリコプター

どこで区切るかわからない！

id	氏	名
1	坂上	田村麻呂
2	ヘリコ	プター

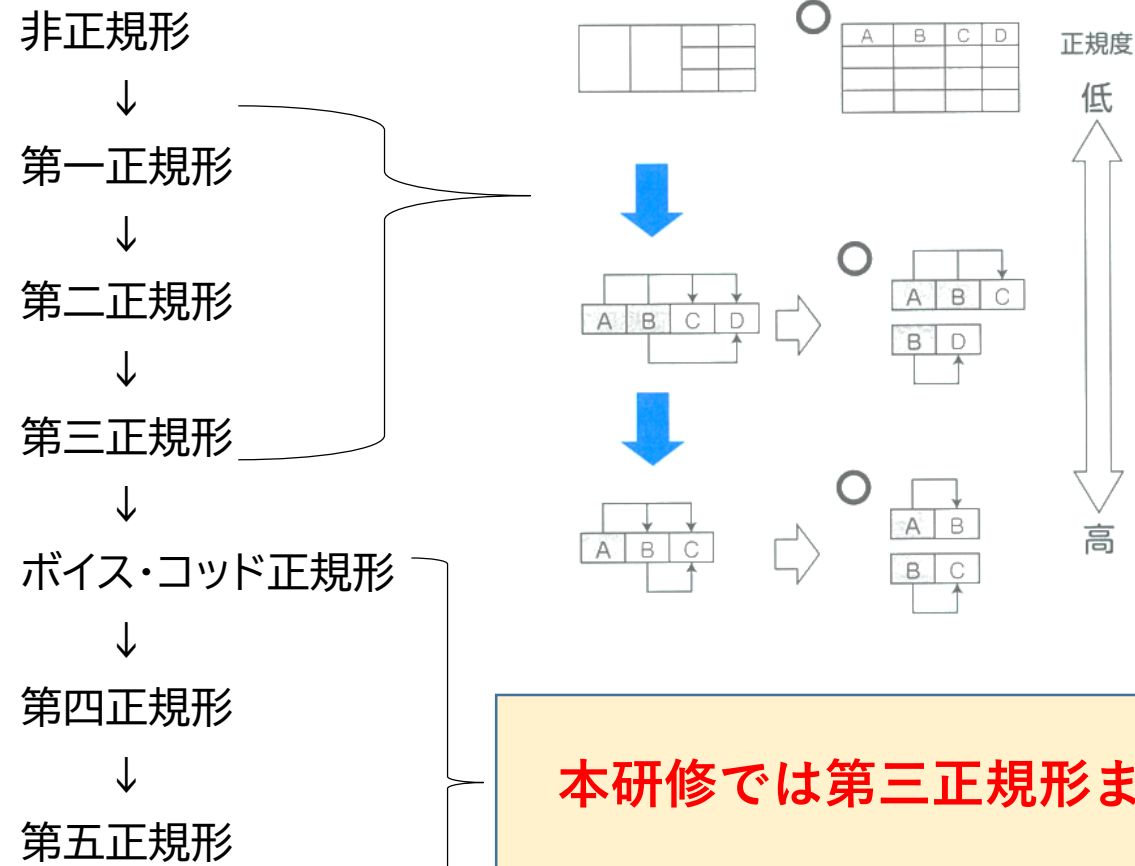
CONCAT関数を使えばすぐにくっつけられる

id	氏名
1	坂上田村麻呂
2	ヘリコプター

■ テーブル設計 正規化

正規化とは

⇒ データベースで保持するデータの冗長性を排除し、一貫性と効率性を保持するための
テーブル構成を作ることです。正規化はレベルが設定されており、下記のようになっています。



本研修では第三正規形までを扱います

■ 非正規形のテーブル

●非正規化状態のテーブル（注文書テーブル）

注文番号	顧客番号	顧客名	顧客住所	商品番号 1	商品名 1	単価 1	購入数 1	商品番号 2	...
P00001	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0001	どっさりカイロ	300	10	P0002	Windows Versa XP
P00002	USR002	薄井浩	静岡県鎌ヶ谷市 静波 9-8-98	P0001	どっさりカイロ	300	5		
P00003	USR003	日尾晃子	埼玉県新田原市 三次 1-1-11	P0002	Windows Versa XP	21500	2	P0003	サンタのカッコ
P00004	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0010	豪華おせち	15000	1		

すべての情報がテーブルに詰め込まれたような状態

リレーショナルデータベースのテーブルに登録することも難しく、テーブルと呼ぶのも難しいただの表のようなもの

「注文書テーブル」という名前がついているが、中を見てみると、「顧客の情報」「商品の情報」など、さまざまな内容が詰まっている

また、テーブルに**繰り返し項目**があり、注文できる商品数がテーブル定義によって決まってしまうという

不便な部分がある。

このようなテーブルはテーブル設計初心者が作ってしまいがちです。

[問題点]

- ・値の更新によって矛盾が生じることがある
- ・行の削除によって情報が損失することがある
- ・注文があるまですべての情報はデータベースに登録できない
- ・注文できる商品の数がテーブルの列の数に制限される

注文されたことのない顧客
注文されたことのない商品
等はテーブルに登録できない

●非正規化状態のテーブル（注文書テーブル）

注文番号	顧客番号	顧客名	顧客住所	商品番号 1	商品名 1	単価 1	購入数 1	商品番号 2	...
P00001	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0001	どっさりカイロ	300	10	P0002	Windows Versa XP
P00002	USR002	薄井浩	静岡県鎌ヶ谷市 静波 9-8-98	P0001	もっさりカイロ	300	5		
P00003	USR003	日尾晃子	埼玉県新田原市 三次 1-1-11	P0002	Windows Versa XP	21500	2	P0003	サンタのカッコ
P00004	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0010	豪華おせき				

入力中の打ち間違えで情報が矛盾
P0001は「どっさりカイロ」？
それとも「もっさりカイロ」？

レコードを削除することで
データベース上で「山田奈美」という顧客がい
た情報が消失してしまう

非正規形 → 第一正規形に

<条件>

- ・テーブルが繰り返し項目を持たないこと

<方法>

- ・キーを設定する
- ・繰り返し項目の排除 ⇒ テーブル分割
- ・計算で求まる列を削除する

繰り返し項目（商品情報）を
別テーブルに！

●非正規化状態のテーブル（注文書テーブル）

注文番号	顧客番号	顧客名	顧客住所	商品番号 1	商品名 1	単価 1	購入数 1	商品番号 2	...
P00001	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0001	どっさりカイロ	300	10	P0002	Windows Versa XP
P00002	USR002	薄井浩	静岡県鎌ヶ谷市 静波 9-8-98	P0001	どっさりカイロ	300	5		
P00003	USR003	日尾晃子	埼玉県新田原市 三次 1-1-11	P0002	Windows Versa XP	21500	2	P0003	サンタのカッコ
P00004	USR001	山田奈美	千葉県小金井市 福山 0-0-00	P0010	豪華おせち	15000	1		

■ 第一正規形のテーブル

[改善点]

商品の購入数がテーブル定義によって制限されることはなくなった



[問題点]

- ・値の更新によって矛盾が生じることがある
- ・行の削除によって情報が損失することがある
- ・注文があるまですべての情報はデータベースに登録できない

●第1正規化後のテーブル

注文番号 	顧客番号	顧客名	顧客住所
P00001	USR001	山田奈美	千葉県小金井市福山 0-0-00
P00002	USR002	薄井浩	静岡県鎌ヶ谷市静波 9-8-98
P00003	USR003	日尾晃子	埼玉県新田原市三次 1-1-11
P00004	USR001	山田奈美	千葉県小金井市福山 0-0-00

▲注文テーブル

注文番号 	商品番号 	商品名	単価	購入数
P00001	P0001	どっさりカイロ	300	10
P00001	P0002	Windows Versa XP	21500	1
P00002	P0001	どっさりカイロ	300	5
P00003	P0002	Windows Versa XP	21500	2
P00003	P0003	サンタのカッコ *	5000	1
P00004	P0010	豪華おせち	15000	1

▲注文明細テーブル

第二正規形へすすむために、新用語「関数従属(かんすうじゅうぞく)」

関数従属性

⇒ある列の値が特定の列の値によって決まること

今の第一正規形のテーブルの関数従属を整理するとこんな感じ

●第1正規形のテーブル

注文番号	顧客番号	顧客名	顧客住所
PO0001	USR001	山田奈美	千葉県小金井市福山 0-0-00
PO0002	USR002	薄井浩	静岡県鎌ヶ谷市静波 9-8-98
PO0003	USR003	日尾晃子	埼玉県新田原市三次 1-1-11
PO0004	USR001	山田奈美	千葉県小金井市福山 0-0-00

▲注文テーブル

注文番号	商品番号	商品名	単価	購入数
PO0001	P0001	どっさりカイロ	300	10
PO0001	P0002	Windows Versa XP	21500	1
PO0002	P0001	どっさりカイロ	300	5
PO0003	P0002	Windows Versa XP	21500	2
PO0003	P0003	サンタのカッコ	5000	1
PO0004	P0010	豪華おせち	15000	1

▲注文明細テーブル

下の注文明細テーブルは
部分関数従属の状態

主キーがすべての列に関数従属している状態
⇒完全関数従属

主キー列が複数あって、関数従属が分かれている状態
⇒部分関数従属

第一正規形 → 第二正規形に

<条件>

1. 表が第一正規形の条件を満たす
2. 完全関数従属する

<方法>

部分関数従属をなくす
(完全関数従属にする)

● 第1正規形のテーブル

注文番号	顧客番号	顧客名	顧客住所
P00001	USR001	山田奈美	千葉県小金井市福山 0-0-00
P00002	USR002	薄井浩	静岡県鎌ヶ谷市静波 9-8-98
P00003	USR003	日尾晃子	埼玉県新田原市三次 1-1-11
P00004	USR001	山田奈美	千葉県小金井市福山 0-0-00

▲注文テーブル

注文番号	商品番号	商品名	単価	購入数
P00001	P0001	どらりカイロ	300	10
P00001	P0002	Windows Versa XP	21500	1
P00001	P0003	パソコン	300	5
P00001	P0004	Windows Versa XP	21500	2
P00001	P0005	パソコン	5000	1
P00004	P0010	豪華おせち	15000	1

▲注文明細テーブル

部分関数従属を別テーブルに！

■ 第二正規形のテーブル

[改善点]

- ・値の更新によって矛盾が生じることがある **が一部改善**
- ・行の削除によって情報が損失することがある **が一部改善**
- ・注文があるまですべての情報はデータベースに登録できない **が一部改善**

[問題点]

- ・値の更新によって矛盾が生じることがある
- ・行の削除によって情報が損失することがある
- ・注文があるまですべての情報はデータベースに登録できない

商品テーブルができたことによって
商品名を打ち間違えてレコードごとに情報
が矛盾することがなくなった

商品についてはあらかじめデータベース
に登録しておくことも可能になった

注文が取り消しになっても、商品情報が
消えることがなくなった

●第二正規形後のテーブル

注文番号	顧客番号	顧客名	顧客住所
PO0001	USR001	山田奈美	千葉県小金井市福山 0-0-00
PO0002	USR002	薄井浩	静岡県鎌ヶ谷市静波 9-8-98
PO0003	USR003	日尾晃子	埼玉県新田原市三次 1-1-11
PO0004	USR001	山田奈美	千葉県小金井市福山 0-0-00

▲注文テーブル

注文番号	商品番号	購入数
PO0001	P0001	10
PO0001	P0002	1
PO0002	P0001	5
PO0003	P0002	2
PO0003	P0003	1
PO0004	P0010	1

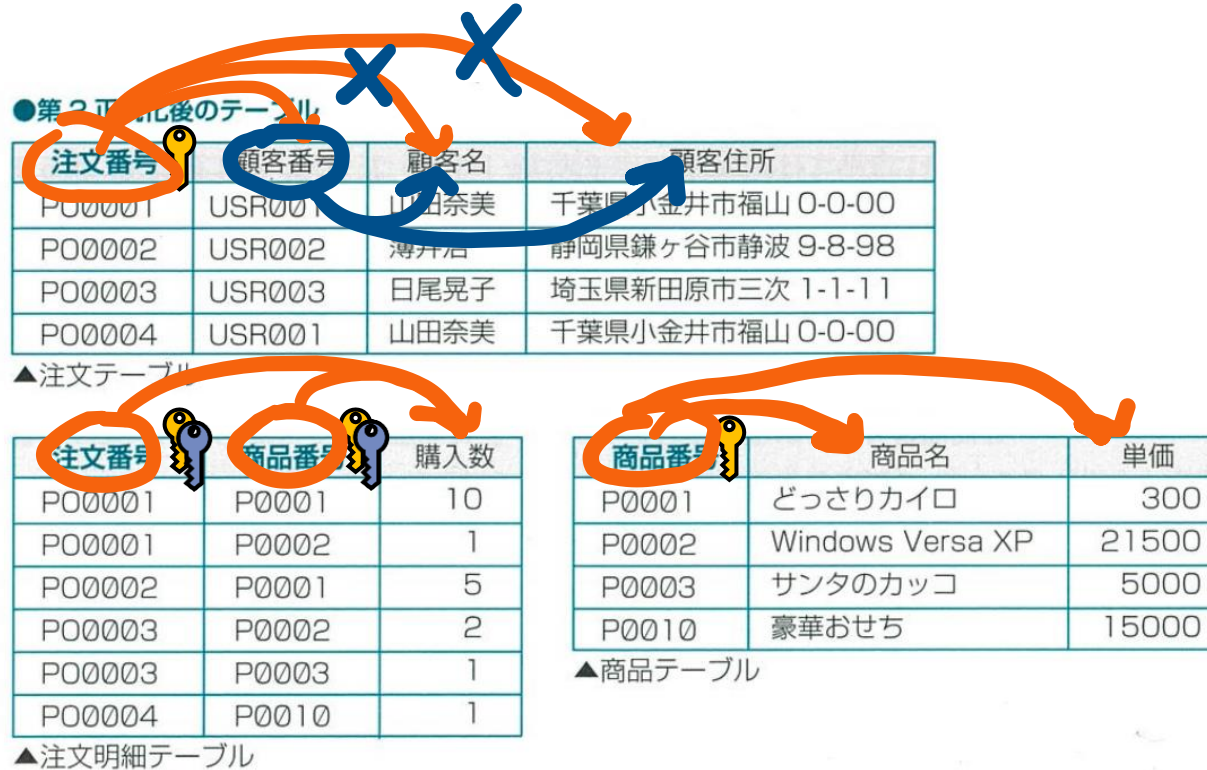
▲注文明細テーブル

商品番号	商品名	単価
P0001	どっさりカイロ	300
P0002	Windows Versa XP	21500
P0003	サンタのカッコ	5000
P0010	豪華おせち	15000

▲商品テーブル

商品に関する情報はこの部分だけ

ここまでで一つ伏せていた情報があります



実は注文テーブルの関数従属について顧客名や顧客住所は注文番号に関数従属しているのではなく顧客番号に関数従属しています。

主キー以外の部分で現れる関数従属を**推移的関数従属**といいます。

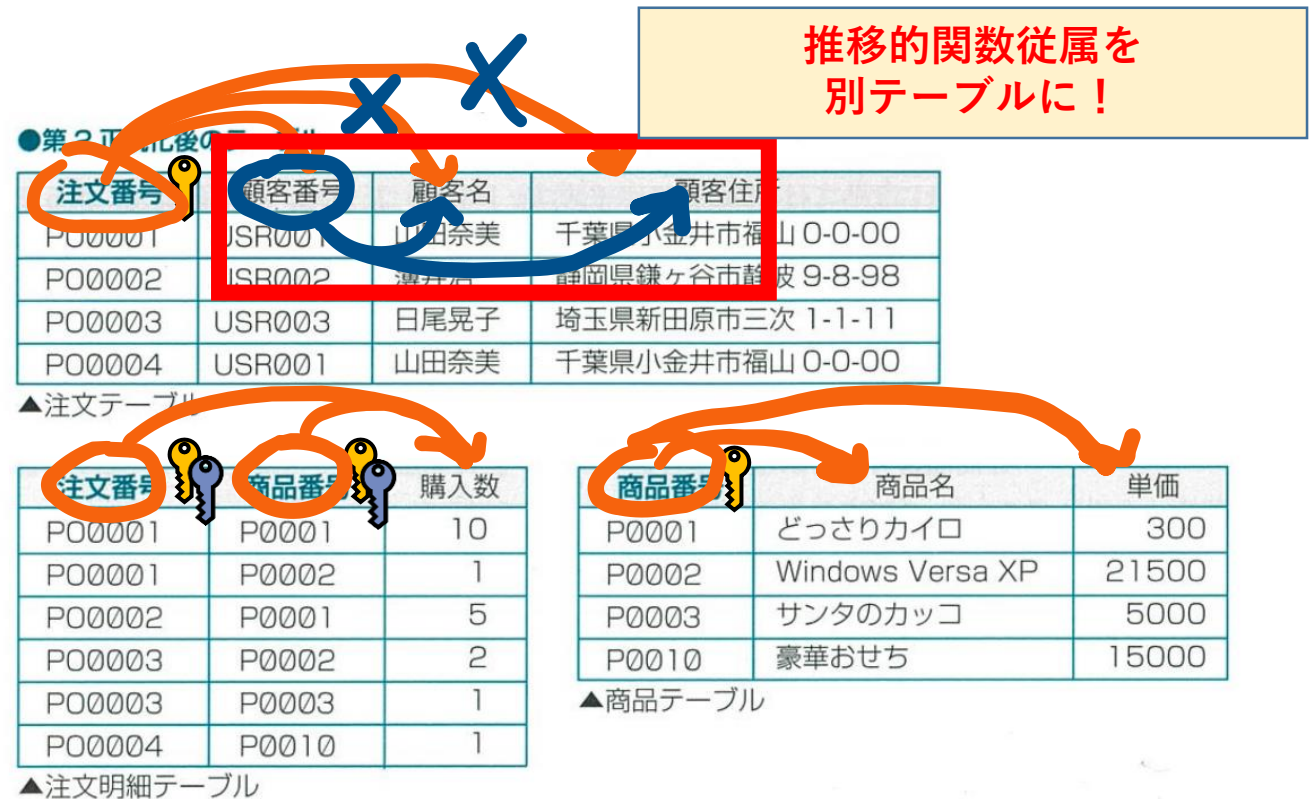
第二正規形 → 第三正規形に

<条件>

1. 表が第二正規形の条件を満たす
2. 推移的関数従属がないこと

<方法>

推移的関数従属をなくす



■ 第三正規形のテーブル

[改善点]

- ・値の更新によって矛盾が生じることがある **が改善**
- ・行の削除によって情報が損失することがある **が改善**
- ・注文があるまですべての情報はデータベースに登録できない **が改善**

●第3正規化後のテーブル

注文番号 	顧客番号 
P00001	USR001
P00002	USR002
P00003	USR003
P00004	USR001


▲注文テーブル

顧客番号 	顧客名	顧客住所
USR001	山田奈美	千葉県小金井市福山 0-0-00
USR002	薄井浩	静岡県鎌ヶ谷市静波 9-8-98
USR003	日尾晃子	埼玉県新田原市三次 1-1-11

▲顧客テーブル

注文番号 	商品番号 	購入数
P00001	P0001	10
P00001	P0002	1
P00002	P0001	5
P00003	P0002	2
P00003	P0003	1
P00004	P0010	1

▲注文明細テーブル

商品番号 	商品名	単価
P0001	どっさりカイロ	300
P0002	Windows Versa XP	21500
P0003	サンタのカッコ	5000
P0010	豪華おせち	15000

▲商品テーブル

顧客情報はあらかじめデータベースに登録することができる

結婚などで顧客名が変わったときでも顧客テーブルのレコードを1つ修正するだけでOK

注文時に顧客名の打ち間違いなどで情報が矛盾することがない

■ 正規化まとめ

正規化を進めることで、データの矛盾をなくし、使いやすいテーブルを設計することができます。

ただし、テーブルを分割することで、結合演算の頻度は上がります。
(ほしい情報がさまざまなテーブルに分散しているため)

テーブル設計する際には、
ひとまず、第三正規形の形まで持っていき、
必要に応じて、第二正規形に落とすといったアプローチがおすすめです。