

データベース

第7回 リレーションスキーマの設計(1)

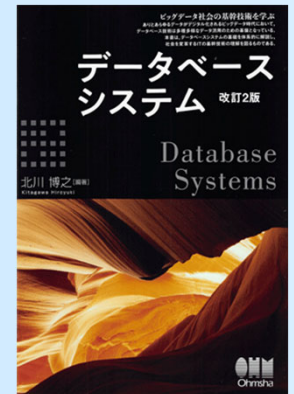
九州工業大学 情報工学部 尾下真樹

今日の内容

- 前回の復習
- リレーションスキーマの設計の概要
- 正規化の必要性
 - 更新不整合
 - 情報無損失分解と情報損失分解
- 関数従属性と多値従属性

教科書・参考書

- 「リレーショナルデータベース入門 第3版」
増永良文 著、サイエンス社（3,200円）
– 4章(4.1～4.7)
- 「データベースシステム 改訂2版」
北川 博之 著、オーム社（3,200円）
– 7章(7.2～7.3、7.9)



前回の復習

SQLによる問い合わせの記述

- SQLの基本的な書き方
 - 条件(WHERE)の書き方
 - 出力(SELECT)の書き方
 - 順序付け(ORDER BY)
 - グループ表(GROUP BY)
- 結合(JOIN)
 - 集合演算
 - 副問い合わせ(入れ子型質問)

結合

Q. 学生番号 00001 の学生が履修した科目の
科目番号、科目名、成績の一覧

```
SELECT 科目.科目番号, 科目名, 成績
FROM 科目, 履修
WHERE 科目.科目番号 = 履修.科目番号 AND
      学生番号 = '00001'
```

– 科目と履修を科目番号で等結合

科目 (科目番号, 科目名, 単位数)

履修 (科目番号, 学生番号, 成績)

科目番号が同じ
データ同士を結合

→ (科目.科目番号, 科目名, 単位数,
履修.科目番号, 学生番号, 成績)

自己結合の例

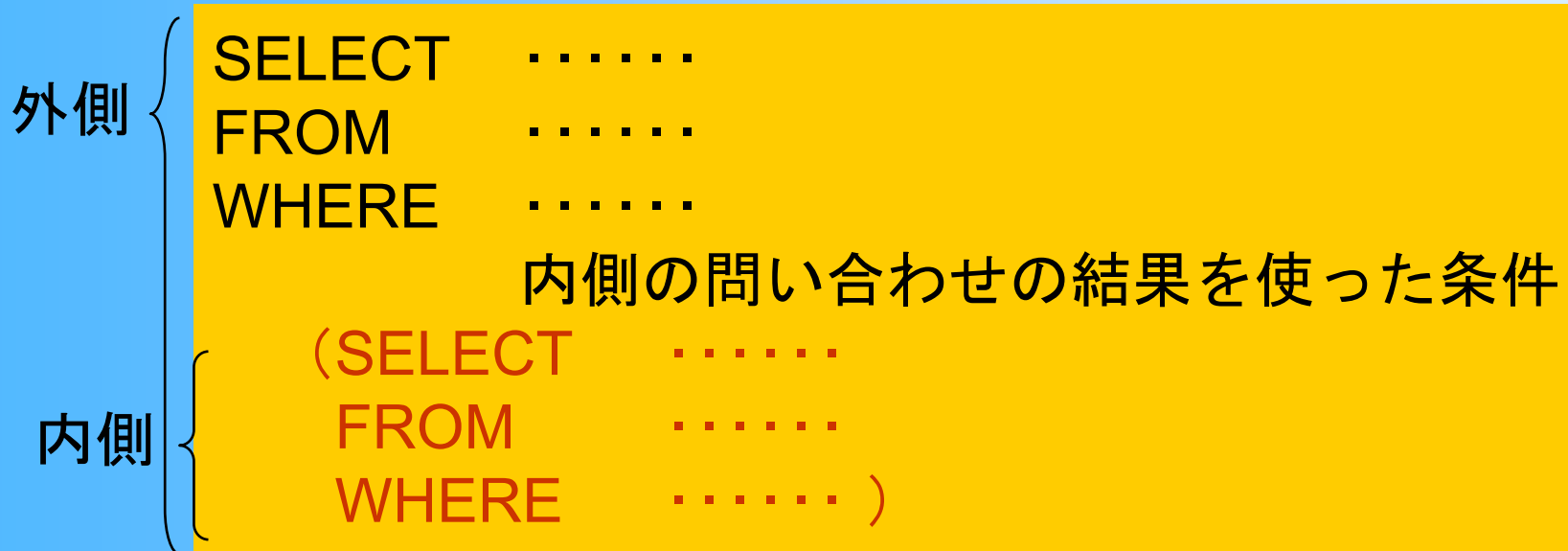
Q.科目番号 001 の科目に関して、
学生番号 001001 の学生よりも成績が良
かった学生の学生番号の一覧

```
SELECT  y.学生番号
FROM    履修 AS x, 履修 AS y
WHERE   x.科目番号 = '001' AND x.学生番号 = '001001'
        AND y.科目番号 = '001' AND y.成績 > x.成績
```

- x は科目番号001、学生番号001001の履修データ
- y は x よりも成績の良い履修データ

入れ子型質問

- 入れ子型質問の形
 - 外側の質問の中に、内側の質問がある
 - 内側の質問の結果を、外側の質問で使用
 - 最終的には、外側の質問の結果が出力される



相関を有さない入れ子型質問の例

- 相関を有さない入れ子型質問の例

Q.科目番号 001 の成績が平均点よりも高い
学生の学生番号と成績の一覧

```
SELECT  学生番号, 成績
FROM    履修
WHERE   科目番号 = '001' AND
        成績 >
        (SELECT  AVG(成績)
         FROM    履修
         WHERE   科目番号 = '001')
```

内側の問い合わせは最初に1度だけ行われる

相関を有する入れ子型質問の例

- 相関を有する入れ子型質問の例

- 学生番号001001の学生の履修で、成績が平均点よりも高い科目の科目番号、成績の一覧を出力

```
SELECT  x.科目番号, x.成績
FROM    履修 AS x
WHERE   x.学生番号 = '001001' AND
        x.成績 >
        (SELECT  AVG(y.成績)
         FROM    履修 AS y
         WHERE   x.科目番号 = y.科目番号)
```

外側の質問の
データが内側の
質問で条件とし
て使われている
＝相関を有する

- 各履修ごとに、その科目の平均点を計算

SQL記述の考え方

- よくある間違いに注意
- 複数のテーブルを用いる場合は結合が必須
- GROUP BY を用いる場合は、SELECT節には、GROUP BY に使った属性か、集約関数しか書けない
- WHERE節には、集約関数は書かない
- HAVINGには、集約関数を使った条件しか書けない

PostgreSQLによるSQL演習(宿題)

- 前回の演習が終わっていない人は、まず、終わらせる
- Moodleから演習内容が書かれたテキストファイルをダウンロード
- 演習課題のSQLを考えて、実行し、正しいデータが出力されることを確認する
- 演習のテキストファイルに、前回と同様、回答を貼り付けて、Moodleから提出

リレーションスキーマの設計の概要

スキーマとインスタンス(復習)

- リレーショナルモデルのスキーマ・インスタンス
 - スキーマ
 - リレーション名、リレーションの各属性名、ドメイン
 - キー制約、参照整合性制約、属性制約などの制約
 - インスタンス
 - 属性値の組によって表される各データの集合

従業員

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39

スキーマ

※ 実際には各種制約を含む

インスタンス

リレーションスキーマの制約(復習)

従業員

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39
004	1	柴田 勝家	60

スキーマ

インスタンス

属性値は各属性のドメイン制約に従う

主キー 外部キー

(超キー、候補キー)

参照整合性制約

リレーションスキーマは第一正規形を満たす

リレーションスキーマの設計

- リレーションスキーマの設計

- リレーショナルデータベースを作成するときには、最初にスキーマや制約を定義する必要がある
- どのようにしてスキーマを決めるべきか？

従業員

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39

部門

部門番号	部門名
1	開発
2	営業

スキーマの
設計

設計と正規化

- 基本的には、データベースで管理したいデータを表すように、スキーマを設計する
- 不適切なスキーマを設計すると、使っていくうちに不整合が生じる危険性がある
- 不整合が生じないようなスキーマを設計する必要がある
 - 不適切なリレーションを、複数のリレーションに適切に分解することで、不整合が生じないようにすることができる(=正規化)

正規化の必要性

第1正規形(復習)

- 第2回の講義で説明
- 第一正規形
 - 属性値は分解不可能な単純な値でなければならない
 - 下のようなリレーションは、第一正規形を満たす

従業員番号	部門番号	氏名	年齢
001	1	織田 信長	48
002	2	豊臣 秀吉	45
003	3	徳川 家康	39
004	1	柴田 勝家	60

第一正規形


- 第一正規形 (first normal form) 制約
 - ドメイン (属性の取りうる値) は分解不可能な単純な値でなければならないという制約
 - 各属性は、単一の値でなければいけない

第一正規形を満たさないスキーマの例 (ひとつの属性に複数の値がある)

科目番号	科目名	学生番号	履修者	成績
01	データベース	0123001	織田 信長	60
		0123002	豊臣 秀吉	70
03	コンピュータグラフィックス	0123001	織田 信長	60
		0123003	徳川 家康	75

正規形

- 更新時に不整合が発生しないような、整合性を保つリレーションスキーマの条件を定義
- 正規形の種類
 - 第1正規形
 - 第2正規形
 - 第3正規形
 - ボイス・コッド正規形
 - 第4正規形
 - 第5正規形



第1正規形→第5正規形まで、徐々に条件が厳しくなっていく

各正規形は、それよりも上の全ての正規形の条件を満たす

正規化

- 正規形を満たさないリレーションスキーマがあるときに、正規形を満たすように分解する

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120

↓ 分解

販売

商品番号	顧客	販売価格
001	A社	100
001	B社	120

顧客担当

顧客番号	社員番号
c1	s1
c2	s2

なぜ、正規化が必要か？

- 更新不整合(更新時異常)が生じる
 - 正規化されていないリレーションは、データを更新(挿入、修正、削除)しようとした時に問題(データの不整合)が生じる場合がある
- 正規化を行うことで解決
 - 更新時にデータの整合性が自動的に保たれる(整合性を壊すようなデータの追加・修正ができないような)データベースができる

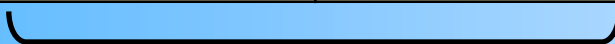
更新不整合の例

- 更新不整合が起こるスキーマの例
 - 各顧客につき、担当する社員は一人だけ、というルールがあるものとする
 - このスキーマは第1正規形は満たす

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250

主キー



更新不整合

- 更新不整合の種類
 - 修正不整合(修正時異常)
 - 挿入不整合(挿入時異常)
 - 削除不整合(削除時異常)

修正不整合

- 修正不整合(修正時異常)の例
 - ある顧客 **A社** の担当社員が、**織田 信長** から別の社員に替わると、顧客 **A社** に関する全てのデータの社員を変更する必要がある

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250

挿入不整合

- 挿入不整合(挿入時異常)の例
 - ある顧客 **D社** の担当社員 **豊臣 秀吉** が決まったとしても、顧客 **D社** との具体的な取引商品が決まっていなければ、その情報を挿入できない
 - 主キーに含まれる商品番号の属性が空値のデータを挿入することは許されないため

営業	商品番号	顧客	社員	販売価格
	—	D社	豊臣 秀吉	—

主キー —————

削除不整合

- 削除不整合(削除時異常)の例
 - 顧客 **A社** の取引商品が一時的に全てなくなった時、顧客 **A社** を社員 **織田信長** が担当している、という情報も一緒に失われてしまう

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250

更新不整合の原因

- さきほどの具体例の問題点

- 1つのリレーションに複数の異なる事象が含まれていることが原因
 - 具体例では、営業に関する販売情報と担当者情報が1つのスキーマで表現されていたことが原因
- リレーションスキーマが正規形を満たしていない
 - 正規形の定義は、次回の授業で説明

- 解決方法

- 複数のリレーションに分解する必要がある
 - 正規化の方法も、次回の授業で説明

演習問題(1)

- 更新不整合

社員(社員番号, 氏名, 部門名, 営業所名)

各部門は、それぞれ1つの営業所内にある(複数の営業所に同じ名前の部門があることはなく、部門名が決まれば営業所名が一意に決まる)。

- このリレーションにおける挿入不整合の例を説明せよ

リレーションスキーマの分解

- 正規形となるようにリレーションを分解
 - 1つのリレーションには1つの事象のみを記録するよう、適切な複数のリレーションに分解する
 - これまで問題となっていた更新異常が起こらない
 - 分解後のリレーションを自然結合することで、分解前の情報が得られる
 - 分解したことによって情報が失われることはない
(情報無損失分解)
- 誤った分解を行うと情報損失分解となる

リレーションスキーマの分解

- 分解の具体例

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250
003	D社	徳川 家康	280



分解

リレーションスキーマの分解

↑ 自然結合することで元に戻る

販売

商品番号	顧客	販売価格
001	A社	100
001	B社	120
002	A社	200
002	C社	210
003	A社	250
003	B社	250
003	D社	280

顧客担当

顧客	社員
A社	織田 信長
B社	豊臣 秀吉
C社	織田 信長
D社	徳川 家康

※ 分解後のリレーション名は適当に決める

情報損失分解

- 悪い分解の例

- 分割の仕方によっては、自然結合してももとの情報が戻らないことがある

販売

商品番号	顧客	販売価格
001	A社	100
001	B社	120
002	A社	200
002	C社	210
003	A社	250
003	B社	250
003	D社	280

商品担当

商品番号	社員
001	織田 信長
001	豊臣 秀吉
002	織田 信長
003	織田 信長
003	豊臣 秀吉
003	徳川 家康

情報損失分解

- 悪い分解の例

販売

商品番号	顧客	販売価格
001	A社	100
001	B社	120
002	A社	200
002	C社	210
003	A社	250
003	B社	250
003	D社	280

商品担当

商品番号	社員
001	織田 信長
001	豊臣 秀吉
002	織田 信長
003	織田 信長
003	豊臣 秀吉
003	徳川 家康

情報損失分解

- 悪い分解の例

- 自然結合した時、もともとは存在していなかったインスタンスが生じる

営業	商品番号	顧客	社員	販売価格
	001	A社	織田 信長	001
	001	A社	豊臣 秀吉	001
	001	B社	織田 信長	120
	001	B社	豊臣 秀吉	120
	002	A社	織田 信長	200
	002	C社	織田 信長	210
	003	A社	織田 信長	250

情報無損失分解と情報損失分解

営業

商品番号	顧客	社員	販売価格
------	----	----	------

主キー — { 商品番号, 顧客 }

※ 各顧客につき、担当する社員は一人だけ

• 情報無損失分解 (正しい分解)

販売

商品番号	顧客	販売価格
------	----	------

顧客担当

顧客	社員番号
----	------

• 情報損失分解 (悪い分解)

販売

商品番号	顧客	販売価格
------	----	------

商品担当

商品	社員番号
----	------

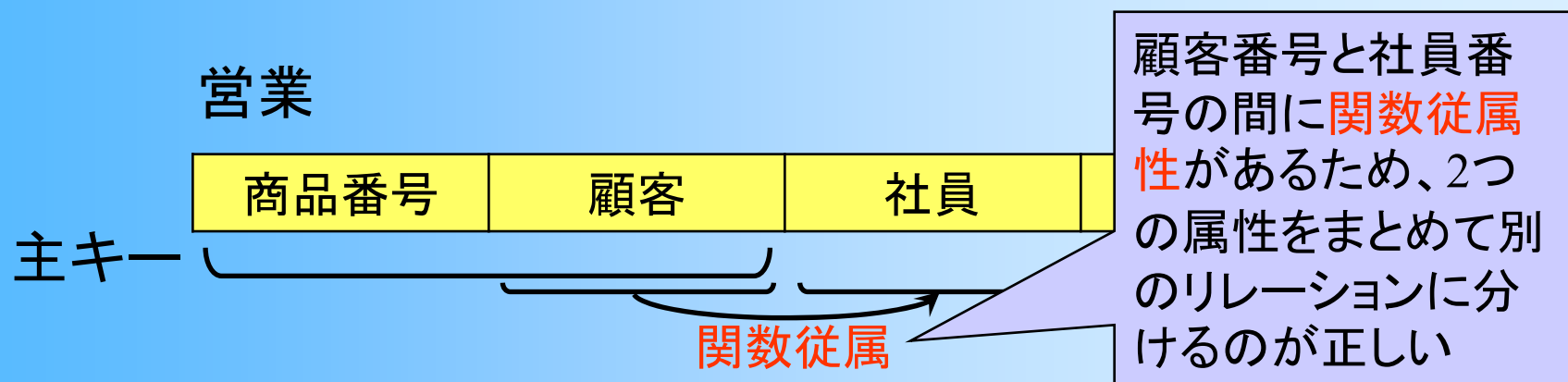
リレーションスキーマ分解のルール

- リレーションスキーマの分解
 - 更新による不整合が生じるのを防ぐ
 - 基本的には1つのリレーションに、複数の事象の情報を混在させないようにすることが必要
- 適当に分解すると、情報損失分解になってしまったため、きちんとしたルールに基づいて分解することが必要

リレーションスキーマ分解のルール

- 正規形を定義するための従属性の定義
 - 多値従属性
 - 関数従属性
 - これらはリレーションスキーマ中の属性同士の依存関係を定義するためのもの
- 正規形の定義
 - 上で定義した従属性を使って正規形を定義
 - 正規形を満たさないリレーションスキーマがあれば、満たすように分割する

情報無損失分解と情報損失分解



• 情報無損失分解 (正しい分解)

販売

商品番号	顧客	販売価格
------	----	------

営業担当

顧客	社員
----	----

• 情報損失分解 (悪い分解)

販売

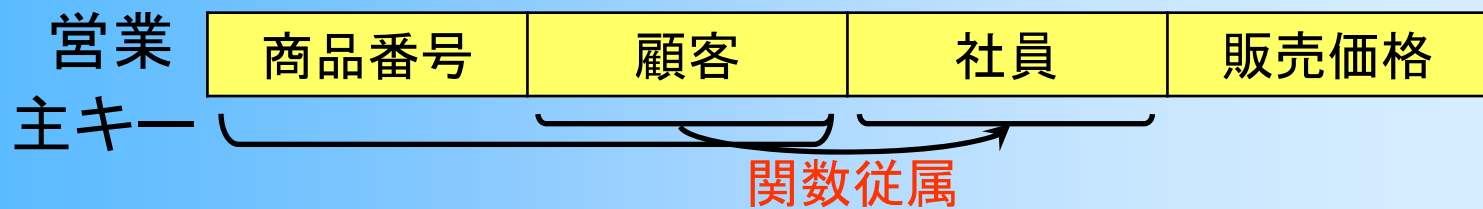
商品番号	顧客	販売価格
------	----	------

商品担当

商品	社員
----	----

ここまでのまとめ

- 悪いリレーションスキーマの例
 - 各顧客につき、担当する社員は一人だけ、というルールがあるものとする



- このままでは更新不整合が生じてしまう
- 関数従属性・多値従属性に注目して分解する

販売

商品番号	顧客	販売価格
------	----	------

営業担当

顧客	社員
----	----

関数従属性 と 多値従属性

関数従属性と多値従属性

- **関数従属性** $X \rightarrow Y$
 - 属性(の組) X が決まれば、属性(の組) Y が一意に決まる
- **多値従属性** $X \twoheadrightarrow Y$
 - ある属性(の組) X について、いくつかの属性(の組) Y が存在するときに、必ず全ての Y と $(RS - XY)$ の組み合わせが存在する
 - RS はリレーションの全ての属性

関数従属性

- 関数従属性の条件

リレーションスキーマ RS のリレーション R が以下の条件を満たす時、 Y は X に関数従属 ($X \rightarrow Y$)

$$(\forall t \in R)(\forall u \in R)(t[X] = u[X] \rightarrow t[Y] = u[Y])$$

- リレーション R において、複数のインスタンスの属性 X の値が同じであれば、属性 Y の値も同じ値になる
 - 属性 X が決まれば、属性 Y も決まる
 - 情報無損失分解の十分条件になる (必要条件ではないことに注意) (教科書「システム」p.75 定理 4.2)

関数従属性の例

- 関数従属性の例

- 各顧客につき、担当する社員は一人だけとする

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250
003	D社	徳川 家康	280

主キー

関数従属性の例

- 関数従属性の例

- 各顧客につき、担当する社員は一人だけとする

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	
001	B社	豊臣 秀吉	
002	A社	織田 信長	
002	C社	織田 信長	210
003	A社	織田 信長	
003	B社	豊臣 秀吉	
003	D社	徳川 家康	

例えば、顧客がA社であれば、担当社員は常に織田 信長となっている

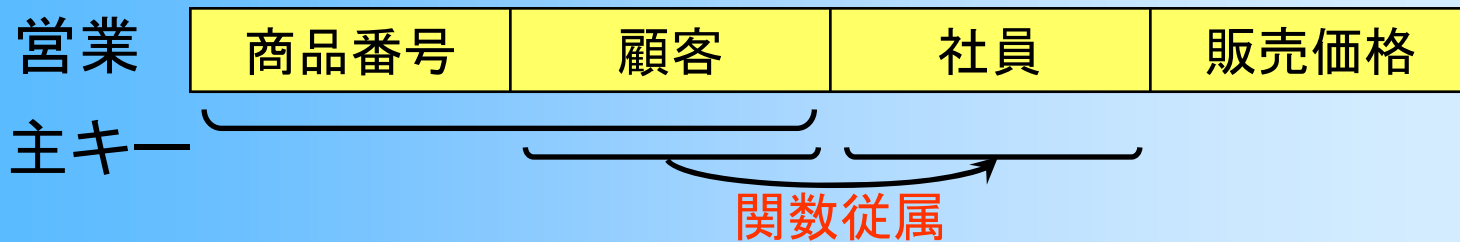
このスキーマには、
顧客 → 社員
という関数従属性がある

主キ

関数従属

(関数従属の属性で分解できる)

関数従属性にもとづく分解の例



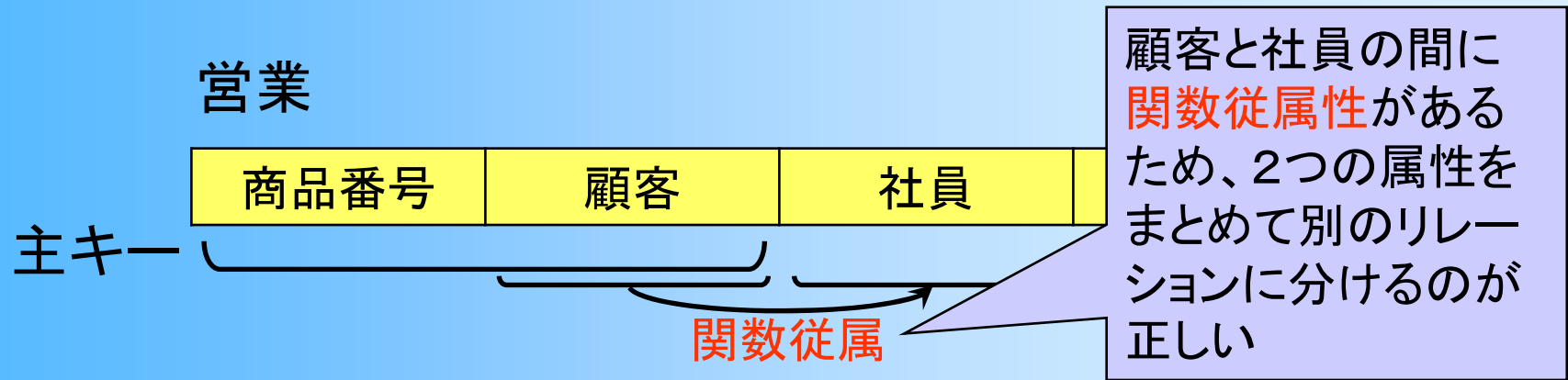
販売

商品番号	顧客	販売価格
001	A社	100
001	B社	120
002	A社	200
002	C社	210
003	A社	250
003	B社	250
003	D社	280

顧客担当

顧客	社員
A社	織田 信長
B社	豊臣 秀吉
C社	織田 信長
D社	徳川 家康

情報無損失分解と情報損失分解



• 情報無損失分解 (正しい分解)

販売

商品番号	顧客	販売価格
------	----	------

営業担当

顧客	社員
----	----

• 情報損失分解 (悪い分解)

販売

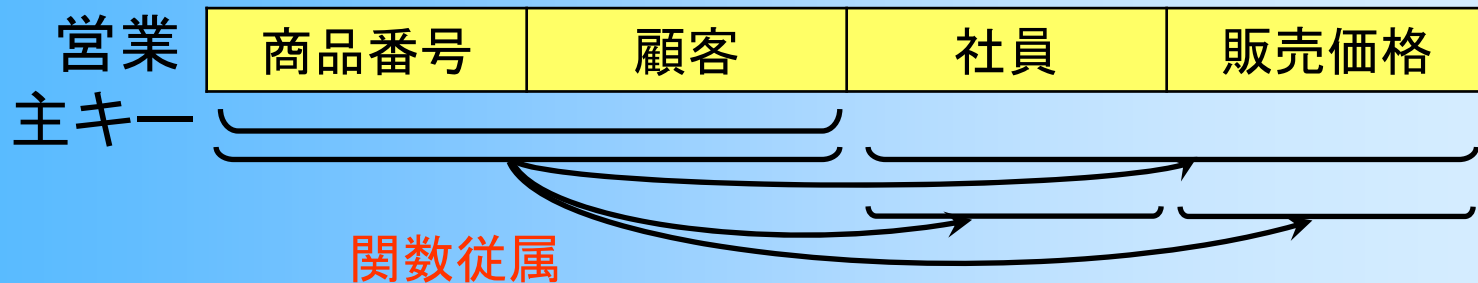
商品番号	顧客	販売価格
------	----	------

商品担当

商品番号	社員
------	----

関数従属性とキー属性の関係

- ある属性(属性の集合) → リレーションの他の全ての属性 の関数従属性があれば、その属性はキー属性(超キー属性)になる



{ 商品番号, 顧客 } → { 社員番号, 販売価格 }

{ 商品番号, 顧客 } → 社員番号

{ 商品番号, 顧客 } → 販売価格

– 上記の例のような**自明な関数従属性**が存在する

キー制約(復習)

- キー

- 複数のインスタンスが同一の属性値をもつことがないような属性、あるいは、属性の集合

- キー制約

- キー制約が指定された属性(属性の集合)は、複数のインスタンスが同一の属性値を持つことが許されない
- キー制約を持つ属性(属性の集合)の属性値が指定されれば、リレーション中のひとつのインスタンスを特定できる

キーの種類(復習)

- 超キー

- 複数のインスタンスが同一の属性値をもつことがないような属性、あるいは、属性の集合
- キーの条件を満たす、全ての属性、属性の集合

- 候補キー(単にキーと呼ぶこともある)

- 最小の超キー(候補キーの部分集合が超キーとならない)

- 主キー

- 候補キーのうち、属性値が空値にならず管理上適当なもの

超キー

- 例: 学生(学生番号、氏名、住所、学科)

主キー

候補キー

※ どの属性が超キーになるかはデータに依存

※ 超キーは他にも存在

多値従属性

- 情報無損失分解の必要十分条件
- 多値従属性の定義

リレーションスキーマ RS のリレーション R が、以下の条件を満たす時、 X は Y に多値従属 ($X \twoheadrightarrow Y$)

$$(\forall t \in R)(\forall u \in R)(t[X] = u[X] \rightarrow (\exists v \in R)(\exists w \in R)$$

$$((t[X] = u[X] = v[X] = w[X]) \wedge$$

$$t[Y] = v[Y] \wedge t[RS - XY] = w[RS - XY] \wedge$$

$$u[Y] = w[Y] \wedge u[RS - XY] = v[RS - XY]))$$

多値従属性

- 情報無損失分解の必要十分条件
- 多値従属性の定義

リレーションスキーマ RS のリレーション R が、以下の条件を満たす時、 X は Y に多値従属 ($X \twoheadrightarrow Y$)

$t(x, y1, z1)$, $u(x, y2, z2)$ というインスタンスが存在すれば、
 $v(x, y1, z2)$, $w(x, y2, z1)$ のようなインスタンスも存在する、
という条件を満たす

多値従属性の例

- プロジェクトごとに社員とミーティング日が決まっているものとする

プロジェクト

プロジェクト	社員	ミーティング日
p1	織田 信長	月曜日
p1	豊臣 秀吉	月曜日
p1	織田 信長	木曜日
p1	豊臣 秀吉	木曜日
p2	織田 信長	月曜日
p2	徳川 家康	月曜日
p2	織田 信長	金曜日
p2	徳川 家康	金曜日

主キー

多値従属性の例

- プロジェクトごとに社員とミーティング日が決まっているものとする

プロジェクト	プロジェクト	社員	ミーティング日	
	p1	織田 信長	月曜日	<i>t</i>
	p1	豊臣 秀吉	月曜日	<i>w</i>
	p1	織田 信長	木曜日	<i>v</i>
	p1	豊臣 秀吉	木曜日	<i>u</i>
	p2	織田 信長	月曜日	

プロジェクト p1 には、社員 **織田 信長**, **豊臣 秀吉** が所属
プロジェクト p1 のミーティング日は、**月曜日**と**木曜日**
プロジェクト p1 に関する 4つのインスタンスが存在する

多値従属性の例

- プロジェクトごとに社員とミーティング日が決まっているものとする

プロジェクト

プロジェクト	社員	ミーティング日
p1	織田 信長	月曜日
p1	豊臣 秀吉	月曜日
p1	織田 信長	木曜日
p1	豊臣 秀吉	木曜日
p1	織田 信長	月曜日
p1	徳川 家康	月曜日

t

w

v

u

$t(x, y_1, z_1)$, $u(x, y_2, z_2)$ という
インスタンスが存在すれば、
 $v(x, y_1, z_2)$, $w(x, y_2, z_1)$ のよう
なインスタンスも存在する

織田
徳川

X, Y は複数の属性の組でも良く、
 $Z = RS - XY$ であることに注意

多値従属性の記述方法

- 多値従属性の記述方法

- この例の場合、以下の多値従属性があると言える
 - どの書き方でも同じ意味になる

プロジェクト →→ 社員

プロジェクト →→ ミーティング日

プロジェクト →→ 社員 | ミーティング日

プロジェクト

プロジェクト	社員	ミーティング日
--------	----	---------

多値従属性にもとづく分解の例

プロジェクト

プロジェクト	社員	ミーティング日
主キー		

プロジェクト番号 →→ 社員番号 | ミーティング日



分解

参加社員

プロジェクト	社員
p1	織田 信長
p1	豊臣 秀吉
p2	織田 信長
p2	徳川 家康

ミーティング日

プロジェクト	ミーティング日
p1	月曜日
p1	木曜日
p2	月曜日
p2	金曜日

多値従属性と関数従属性の関係

- 関数従属性は多値従属性の特殊な形
- 関数従属性を満たす場合は、多値従属性も満たす
 - $X \rightarrow Y$ ならば $X \rightarrow \rightarrow Y$
 - 関数従属性は、多値従属性の Y が一通りしかない(一意に決定する)ケース
 - 関数従属性を満たしていれば、多値従属性も満たす
 - 逆は成り立たないことに注意

多値従属性と関数従属性の例

- 関数従属性が成立すれば、多値従属性も成立

顧客 → 社員 → 顧客 →→ 社員

顧客番号ごとに社員番号が一意に
決まる特殊な例

営業

商品番号	顧客	社員	販売価格
001	A社	織田 信長	100
001	B社	豊臣 秀吉	120
002	A社	織田 信長	200
002	C社	織田 信長	210
003	A社	織田 信長	250
003	B社	豊臣 秀吉	250

関数従属性と多値従属性のまとめ

- **関数従属性** $X \rightarrow Y$
 - 属性(の組) X が決まれば、属性(の組) Y が一意に決まる
- **多値従属性** $X \twoheadrightarrow Y$
 - ある属性(の組) X について、いくつかの属性(の組) Y が存在するときに、必ず全ての Y と $(RS - XY)$ の組み合わせが存在する
 - RS はリレーションの全ての属性
 - 関数従属性は多値従属性の特殊なものになる
 - Y が常に1種類のみ存在するもの

関数従属性の判定

- その属性にどのような値が入るかという条件によって決まる
 - スキーマだけでは分からず、データベースに格納されるデータについての予備知識が必要
 - 今回の例では、1件の顧客につき担当者1人というきまりなど
 - スキーマ設計を行う時には、どのような関数従属性が存在するかを調べて、書き出す必要がある

関数従属性の理論

- 関数従属性の導出
 - 与えられた関数従属性から全ての関数従属性を求める
- 論理的に含意する (logically imply)
 - $\{A \rightarrow B, B \rightarrow C\}$ ならば $A \rightarrow C$ となる
 $\{A \rightarrow B, B \rightarrow C\} \vdash A \rightarrow C$
- 閉包 (closure)
 - ある関数従属性の集合 F から導出される全ての関数従属性の集合 F^+

関数従属性の理論

- アームストロングの公理系
 - 反射律
 - Y が X の部分集合であれば、 $X \rightarrow Y$
 - 例: 学生番号、氏名 \rightarrow 氏名
 - 増加律
 - $X \rightarrow Y$ であれば、 $X \cup A \rightarrow Y \cup A$
 - 例: 学生番号 \rightarrow 学科 なら、学生番号、氏名 \rightarrow 学科、氏名
 - 推移律
 - $X \rightarrow Y$ かつ $Y \rightarrow Z$ であれば、 $X \rightarrow Z$
 - 例: 学生番号 \rightarrow 学科番号 かつ 学科番号 \rightarrow 学科名 なら、学生番号 \rightarrow 学科名
 - 関数従属性 $X \rightarrow Y$ に、アームストロングの公理を適用した結果も、必ず関数従属性になる(=健全で完全(sound and complete))

まとめ

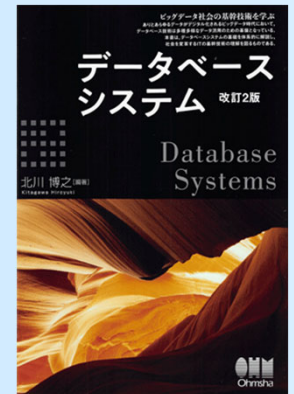
- 前回の復習
- リレーションスキーマの設計の概要
- 正規化の必要性
 - 更新不整合
 - 情報無損失分解と情報損失分解
- 関数従属性と多値従属性

次回予告

- 次回:正規形と正規化
 - 第2正規形、第3正規形、ボイス・コッド正規形、第4正規形、第5正規形
- 次々回:リレーションスキーマの設計
 - 概念設計と論理設計
 - 実体関連モデルによるスキーマの設計
 - 正規化によるスキーマの設計

教科書・参考書

- 「リレーショナルデータベース入門 第3版」
増永良文 著、サイエンス社（3,200円）
 - 4章(4.9~4.13)
- 「データベースシステム 改訂2版」
北川 博之 著、オーム社（3,200円）
 - 7章(7.4~7.11)
 - 教科書によって正規形の定義の表記が異なるが、本講義では両方の内容を説明



中間授業アンケート

- Moodle上で中間授業アンケートを実施
 - 本アンケートの回答は、今後の授業の改善に役立てますので、なるべく回答をしてください
 - アンケート回答は強制ではありませんので、回答するかどうかは自由です
また、回答の有無や、回答の内容は、成績評価には一切影響しません
 - 期末アンケートは、別途、実施されます
意見等があれば、期末アンケートの方に書いても構いません