データベース
第9回

土田　隼之

授業計画

	週	授業内容・方法	週ごとの到達目標
後期	1週	データベースの概要	データベースの役割、データベースの学術利用、業務利用、その意義と用途を理解できる。
	2週	データベースのための基礎理論	集合とその演算、組（タプル）、組の集合としてのリレーションなど、データベースのための基礎理論を理解できる
	3週	リレーショナルデータモデルとリレーショナル代数	RDBMSで利用されるデータモデルであるリレーショナルデータモデルとデータ操作のためのリレーショナル代数を理解できる。
	4週	SQL(1)	RDBMSの利用全般に用いられる言語SQLの基本を理解できる。リレーションへのデータ登録・削除・更新、簡単な問
	5週	SQL(2)	
	6週	RDBMSの内部構成	RDBMSの内部構成、および大量のデータの中から目的とするデータに素早くアクセスする仕組みであるインデックスを理解できる。
	7週	問合せ最適化	RDBMSで、SQL問合せを実行するための実行プランを生成するための問合せ最適化が理解できる。
	8週	中間試験	中間試験
	9週	プログラムからのRDBMSの利用	汎用プログラミング言語で書かれたプログラムからRDBMSを利用する方法が理解できる。
	10週	正規化	ーションの更新時に発生しうるデータの不整合、およ 決策であるリレーションの正規化が理解できる。
	11週	データモデリング	データ 作業で
	12週	SQL(3)	る。 SQLにおける問合せを行う高度なselect文を理解できる。
	13週	トランザクションと同時実行制御	アプリケーションがデータベースにアクセスする単位であるトランザクションの概念、および複数のトランザクションを正常に実行するための基礎理論を理解できる。
	14週	NoSQLデータベースとビッグデータ(1)	ビッグデータを扱うため開発された新しいデータベースであるNoSQLの基礎を理解できる。主にNoSQLの概観と、ビッグデータを扱うためのデータモデルや実行制御理論を理解できる。

来週に答案返却と、問題解説をおこないます。

先に、第10週の内容をやります
#ローカル環境のMySQLの説明を年明けにしてから

リレーショナルデータモデルと第1正規形

過去の講義

データベースにデータを蓄えるには、データやデータ同士の関係をモデル化して、蓄えるための枠組みをあらかじめ用意しなければならない。そのため、漠然と存在するデータ間の関係を把握する必要がある。リレーショナルデータモデルでは、下記のようにデータや関係を表現する。

・リレーション:表の各行は、共通の構造をしたタプル
　　各行は、横に並んでいるそれぞれのデータが分解できない単純な値
・第一正規形:単純な値でできている表
・非第一正規形:表の中の項目が表やリストになっている

5,”明石”など

第一正規形

料理名	値段(円)
チャーハン	800
蟹チャーハン	1300
天津飯	900
餃子	500

非第一正規形

料理名		値段(円)
中華料理	チャーハン	800
	天津飯	900
イタリアン	パスタ	500
ミネラルウォーター		400

行が共通のタプル構造ではない

第一正規形と更新時異常

以前に、リレーショナルモデルの基本となる第一正規形を示した。第一正規形では、**更新時異常**と呼ばれる問題が存在することが知られている。以下の履修情報リレーション(学生が受講する科目と科目の教員、および学生の所属する学科と学科長のリレーション)で例示する。

履修情報

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

a)学生が決まれば学科が定まり、学科から学科長が定まる

1)挿入時異常

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2
S4	null	null	設計	C2

学生「S4」が学科「設計」に加わったとする。しかし、履修科目が決まっていないならば、属性{科目}が主キーであり、科目がnullのタプルを挿入できない

主キーはnull不可

a)学生は1つの学科に所属し、各学科には1人の学科長がいる
b)各科目は、1人あるいは複数の教員が担当
c)学生の履修は{科目,教員}の組(例:プログラム,P1)で定まる
1人の学生が同じ科目を複数履修はしない {学生,科目名}で{教員}が決まる。({教員,学科,学科長}が決まる)
d)教員は複数の科目を担当できる。異なる教員が同一科目を担当する可能性が有る。(情報工学,P2),(情報工学,P3)
#主キーは、{学生,科目名}となる。

第一正規形と更新時異常2

以前に、リレーショナルモデルの基本となる第一正規形を示した。第一正規形では、**更新時異常**と呼ばれる問題が存在することが知られている。

履修情報

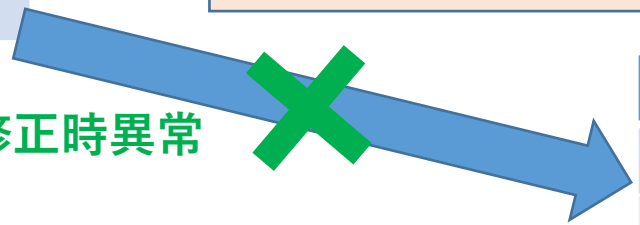
学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

2)削除時異常

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

学生「S3」が科目「設計演習」の履修登録を取り消したとする。タプル全体が削除され、学生「S3」の情報も失われてしまう。

3)修正時異常



学生	科目名	教員	学科	学科長
S1	プログラム	P1	設計	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

学生「S1」の学科が「情報」から「設計」に変更になったとする。変更があれば、該当するすべてのタプルを修正する必要がある。この場合「S1」の「プログラム」「情報工学」の両方を修正することとなる。

学生S1は(所属)学科が設計なのか？情報なのか？

更新時異常と情報無損失分解

不整合発生の原因は、2つの情報(学生の履修登録、学生の学科)を1つのリレーションとしたことである。過去に扱った「射影演算」で適切なリレーションに分割して解消できる。なお、自然結合演算により復元可能である。

履修情報

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

「学生」に関する自然結合で復元

無損失分解

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2

履修情報

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S1	情報工学	P3	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P2	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

S1,情報工学,P3のレコードは分割前は存在していない

分割が適切でないと正しく復元できない
→復元できてない

「科目」に関する自然結合で復元

履修科目

学生	科目名	学科	学科長
S1	プログラム	情報	C1
S1	情報工学	情報	C1
S2	プログラム	情報	C1
S2	情報工学	情報	C1
S3	設計演習	設計	C2

学科情報

科目名	教員
プログラム	P1
情報工学	P2
情報工学	P3
設計演習	P4

更新時異常と情報無損失分解

不整合発生の原因は、2つの情報(学生の履修登録、学生の学科)を1つのリレーションと

き

データベースでは、なんらかの事象(例:履修情報)を複数の表(例:履修登録表、学生学科表)に分割して管理する。
分割された表を結合する際に、関数の概念が用いられる。

S3	設計演習	P4	設計	C2

「学生」に関する自然結合で復元

分割が適切でないと
正しく復元できない
→復元できてない

「科目」に関する自然結合で復元

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2

履修科目

学生	科目名	学科	学科長
S1	プログラム	情報	C1
S1	情報工学	情報	C1
S2	プログラム	情報	C1
S2	情報工学	情報	C1
S3	設計演習	設計	C2

学科情報

科目名	教員
プログラム	P1
情報工学	P2
情報工学	P3
設計演習	P4

関数

リレーション間の関係に制約を持たせることで、
データを効率的に扱う方法

・データベースでは、正規化や関数従属性(外部キーなど)が重要である。
関数はそれらの基礎であり、具体例としては以下のようなものがある。

例:販売実績(顧客ID,顧客名,商品名,数量,単価,金額)リレーションの関数
#関数の定義は次スライド

購入IDが決まるとレコードが一つに決まる

f1:**購入ID**
→**顧客ID,顧客名,商品名、数量**
単価、金額
f2:**顧客ID→顧客名**

顧客IDが決まると、顧客名が一つに決まる

購入ID	顧客ID	顧客名	商品名	数量	単価	金額
O01	C01	明石太郎	ノート	10	110	1100
O02	C02	神戸次郎	鉛筆	20	80	1600
O03	C01	明石太郎	けしごむ	5	70	350
O04	C03	加古川三郎	定規	1	100	100

関数

リレーション間の関係に制約を持たせることで、
データを効率的に扱う方法

・データベースでは、正規化や関数従属性(外部キーなど)が重要である。
関数は、下記のような定義となる。

D_1はあるドメイン(例:顧客名)
a_1はあるドメインの属性(例:明石、神戸、兵庫、加古川)

前提条件:$X=D_1\times\cdots\times D_n, Y=E_1\times\cdots\times E_n$を
それぞれ、属性系列$a=a_1,\cdots,a_n$と$b=b_1,\cdots,b_n$の直積集合とする。

XからYへの関数f:X→Y(あるいは、属性系列のみを明示してf:a→b)とは、次の二つの性質を満たすXからYへの対応関係である。

顧客IDに対応する顧客名は何かある

1)Xの任意のタプルにYのあるタプルが少なくとも一つは対応する。つまり、任意のx∈Xに対して、xfyとなるy∈Yが存在する。

顧客IDに対応する顧客名は1種類

2)Xのタプルxが Yの二つ以上のタプルyに対応することがない。すなわち、x1,x2∈X,y1,y2∈Yに対し、x1fy1,x2fy2の時、x1=x2ならばy1=y2である。

fがXからYへの関数のとき、xfyをf(x)=yと書き、xはfによりyに対応付けられる(写像される)と言う。

タプルとリレーション

過去の講義

・データベースでは、その要素の種類を表した属性などと呼ばれる情報を付加して、扱われることが多い。

例:料理名ドメイン、人名ドメイン

・同じ属性(料理名など)の要素の集合をドメインと呼ぶと前スライドで述べた。ドメインD1,D2,…,Dnの直積集合をD1×D2×…×Dnとすると、直積集合の要素(d1,…,dn)をタプル、その部分集合をリレーションと呼ぶ。

(天津飯,900)

各表のデータは「全ての組合せ」の部分集合

属性名→カラム名、列名
#例:「値段(円)」

料理名	値段(円)
チャーハン	800
蟹チャーハン	1300
天津飯	900
餃子	500

タプル→行、レコード
#例:(チャーハン,800)

リレーション→表

リレーションと集合

過去の講義

・データベースでは、リレーショナル代数を用いる。リレーショナル代数は、リレーション上の演算からなるがリレーションは集合の一種である。

集合演算のようなもの

表とほとんど同じ意味

リレーションとリレーション代数の例

店舗Aのメニュー Ra

料理名	値段(円)
チャーハン	800
蟹チャーハン	1300
天津飯	900
餃子	500

店舗Bのメニュー Rb

料理名	値段(円)
チャーハン	800
中華飯	750
マーボ飯	900
餃子	500

共通集合演算
(リレーショナル代数)

店舗Aと店舗Bで共通のメニュー

料理名	値段(円)
チャーハン	800
餃子	500

リレーションであるRaとRbに対して、リレーショナル代数である共通集合演算を適用し、「店舗Aと店舗Bで共通のメニュー」というリレーションを生成している。

・集合は”もの”の集まりであり、集合に属する対象を要素などと呼ぶ。データベースでは、同一の型や属性を持った要素の集合に対して、ドメインという用語が使われる。

例:料理名ドメイン、人名ドメイン

関数

リレーション間の関係に制約を持たせることで、
データを効率的に扱う方法

・データベースでは、正規化や関数従属性(外部キーなど)が重要である。
関数は、下記のような定義となる。

D_1はあるドメイン(例:顧客名)
a_1はあるドメインの属性(例:明石、神戸、兵庫、加古川)

前提条件:$X=D_1\times\cdots\times D_n, Y=E_1\times\cdots\times E_n$を
それぞれ、属性系列$a=a_1,\cdots,a_n$と$b=b_1,\cdots,b_n$の直積集合とする。

XからYへの関数f:X→Y(あるいは、属性系列のみを明示してf:a→b)とは、次の二つの性質を満たすXからYへの対応関係である。

顧客IDに対応する顧客名は何かある

1)Xの任意のタプルにYのあるタプルが少なくとも一つは対応する。つまり、任意のx∈Xに対して、xfyとなるy∈Yが存在する。

顧客IDに対応する顧客名は1種類

2)Xのタプルxが Yの二つ以上のタプルyに対応することがない。すなわち、x1,x2∈X,y1,y2∈Yに対し、x1fy1,x2fy2の時、x1=x2ならばy1=y2である。

fがXからYへの関数のとき、xfyをf(x)=yと書き、xはfによりyに対応付けられる(写像される)と言う。

関数従属性と正規形

情報無損失分解は関数従属性と関連している。
以下、Rをリレーションスキーマ、A,B,CをRの属性の任意の部分集合とする。

・関数従属性:

A:決定項(例:学生)

B:被決定項(例:学科、学科長)

Rの任意のタプルにおいて、Aの値が等しければBの値が等しいという制約がなりたつ場合、関数従属性A→Bが成り立つという。
#候補キーの値に対してタプルが一意に定まる
→Aを候補キーとすると、すべての属性がAに関数従属している。

候補キーは、ほかのスーパーキーを含んでいないため、「候補キーを決定項とする関数従属性は完全関数従属性」である。

A⊃A’

・完全関数従属性:
関数従属性A→Bで、Aの任意の真部分集合A’について、関数従属性A’→Bが存在しないとき、完全関数従属性A→Bが成り立つという。

これ以上カラムを減らすと、一意に決まらない

関数従属性と正規形2

関数従属性では以下の公理系が成り立つ。
・アームストロングの公理系

Rをリレーションスキーマ、A,B,CをRの属性の任意の部分集合とする。

(1)反射律:BがAの部分集合であるならば、A→B
　⇒被決定項が決定項の部分集合であれば必ず関数従属する　例:{氏名,住所}→{氏名}

(1)を「自明な関数従属性」という

(2)添加律:A→Bならば、AUC→BUC

{学生}→{学科}なら、{学生,科目名}→{学科,科目名}

(3)推移律:A→BかつB→Cならば、A→C

{学生}→{学科},{学科}→{学科長}より、{学生}→{学科,学科長}

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

候補キー{学生,科目名}なので、
{学生,科目名}→{教員,学科,学科長}が成り立つ

{学生,科目名}のいずれが欠けても候補キーにならないので、完全関数従属性となる。

・完全関数従属性:
関数従属性A→Bで、Aの任意の真部分集合A'について、関数従属性A'→Bが存在しないとき、完全関数従属性A→Bが成り立つという。

これ以上カラムを減らすと、一意に決まらない

課題9　締切:12/28

9-1)学生の履修登録に関するリレーションスキーマ 履修登録(学生,科目,教科書,出版社)があり、以下制約があるとする。このとき、以下の問に答えよ。

教科書P108の問1

①学生は任意の科目を履修できる。

②科目ごとに1冊の教科書が指定されている。ただし、科目が異なっても教科書が異なるとは限らない。

③教科書は出版社をもつ。ただし、教科書が異なっても出版社が異なるとは限らない。

(1)このリレーションスキーマのリレーションでは、どのような更新時異常が発生するか、リレーションの事例を示し、挿入時異常、削除時異常、修正時異常に分けて事例を述べよ。

(2)このリレーションスキーマのもつ関数従属性を示せ。ただし、自明な関数従属性は除くものとする。また、一つの決定項に対して被決定項が複数存在する場合には、一つの関数従属性として示してよい。

Heathの定理

定理の内容:リレーションスキーマR(A,B,C)でA→Bが成り立つならば、Rはリレーションスキーマ R1(A,B)とR2(A,C)に情報無損失分解できる。

{学生}→{学科,学科長}とHeathの定理から、
リレーション(学生,学科,学科長)と(学生,科目,教員)に情報無損失分解できることがわかる。

履修情報

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2

{科目}のみに従属している属性はないので、
{科目}に対し{学生}、{教員、学科、学科長}は一意に決まらない

履修科目

学生	科目名	学科	学科長
S1	プログラム	情報	C1
S1	情報工学	情報	C1
S2	プログラム	情報	C1
S2	情報工学	情報	C1
S3	設計演習	設計	C2

学科情報

科目名	教員
プログラム	P1
情報工学	P2
情報工学	P3
設計演習	P4

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S1	情報工学	P3	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P2	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

第二正規形

定義:リレーションスキーマRにおいて、以下の２つの条件が成り立つ場合にRは第二正規形であるという。
(1)Rは第一正規形である。
(2)すべての非キー属性(候補キーに属してない属性)は各候補キーに完全関数従属している。

スキーマ全体の候補キーは{学生、科目名}だが、
{学生}→{学科、学科長}となっている。
候補キーを減らしても関数従属するので(2)を満たさない
#学科、学科長は非キー属性

履修情報

学生	科目名	教員	学科	学科長
S1	プログラム	P1	情報	C1
S1	情報工学	P2	情報	C1
S2	プログラム	P1	情報	C1
S2	情報工学	P3	情報	C1
S3	設計演習	P4	設計	C2

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2

{学生}に非キー属性{学科,学科長}が関数従属しているので第二正規形ではない
→非キー属性{学科,学科長}が(候補キーから減らした{学生}に関数従属なので)候補キーに完全関数従属してない

第三正規形

定義:リレーションスキーマRにおいて、以下の２つの条件が成り立つ場合に、Rは第三正規形であるという。
(1)Rは第二正規形である。
(2)すべての非キー属性はどの候補キーにも推移的に関数従属しない。

第二正規形から、推移的関数従属性を取り除いたものが第三正規形
推移的関数従属性:推移律(A→BかつB→Cならば、A→C)が成り立つときA→Cを推移的関数従属性という。

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2

学生→学科→学科長
#推移的に関数従属している

挿入時異常

学生・学科

学生	学科	学科長
S1	情報	C1
S2	情報	C1
S3	設計	C2
null	機械	C3

推移的関数従属性が無くなるまで分割

学科

学科	学科長
情報	C1
設計	C2

学生

学生	学科
S1	情報
S2	情報
S3	設計

学科「機械」が新設され、学科長「C3」が決まったとしても、学生が配属されるまで、学科と学科長の情報が登録できない

課題9　締切:12/28

9-2)教科書P108の設問2

9-1)の(1)の事例として作成したリレーションについて、以下の問いに答えよ。

(1)このリレーションを第二正規形に正規化したリレーションを示せ。

(2)(1)で作成したリレーションが第三正規形になっているかを確認し、なっていない場合には第三正規形に正規化せよ。

第三正規形と更新時異常

第三正規形まで満たせば、更新時異常が発生しないか考える。
これまでの条件に、各教員は1科目のみ担当するという制約を追加する。
→①{学生,科目}→{教員},②{教員}→{科目}であり、{学生,教員}も候補キーとなる。
　スキーマの候補キーが{学生,科目},{学生,教員}となる。

スキーマに非キー属性(どんな候補キーにも含まれていない属性)が無いため、第三正規形となる

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

履修登録

学生	科目名	教員
S1	プログラム2	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

教員「P1」の科目名が「プログラム」から「プログラム2」に変更になったとする。変更があれば、該当するすべてのタプルを修正する必要がある。

更新時異常

学生、科目、教員が1つのリレーションに存在することが原因

ボイス・コッド正規形

定義:リレーションスキーマRのすべての関数従属性A→Bにおいて、以下のいずれかが成り立つ場合にRはボイス・コッド正規形であるという。
(1)A→Bは自明な関数従属性である。
(2)AがRのスーパーキーである。

自明な関数従属性とは、反射律が成り立つとき
反射律:BがAの部分集合であるならば、A→B
⇒被決定項が決定項の部分集合であれば必ず関数従属する
例:{氏名,住所}→{氏名}

自明ではない関数従属性の決定項は候補キーを含む属性のみであることが要求されている。
{教員}→{科目名}という関数従属性があるが、{教員}はスーパーキーではない。
スーパーキー:「タプルを一意的に特定できる属性や属性の集合」
候補キー:スーパーキーのうち、スーパーキーに他のスーパーキーが含まれていないもの

分割後は関数従属性{学生,科目}→{教員}が保存されない
#{学生,科目}から{教員}が一意に決まらない

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

履修登録

学生	科目名	教員
S1	プログラム	P1
S1	情報工学	P2
S2	プログラム	P1
S2	情報工学	P3
S3	設計演習	P4

候補キー{学生,教員}

学生	教員
S1	P1
S1	P2
S2	P1
S2	P3
S3	P4

教員	科目名
P1	プログラム
P2	情報工学
P3	情報工学
P4	設計演習

スパーキー以外に関数従属する被決定項(科目名)を分離して分割

課題9　締切:12/28

(余裕があれば)9-3)

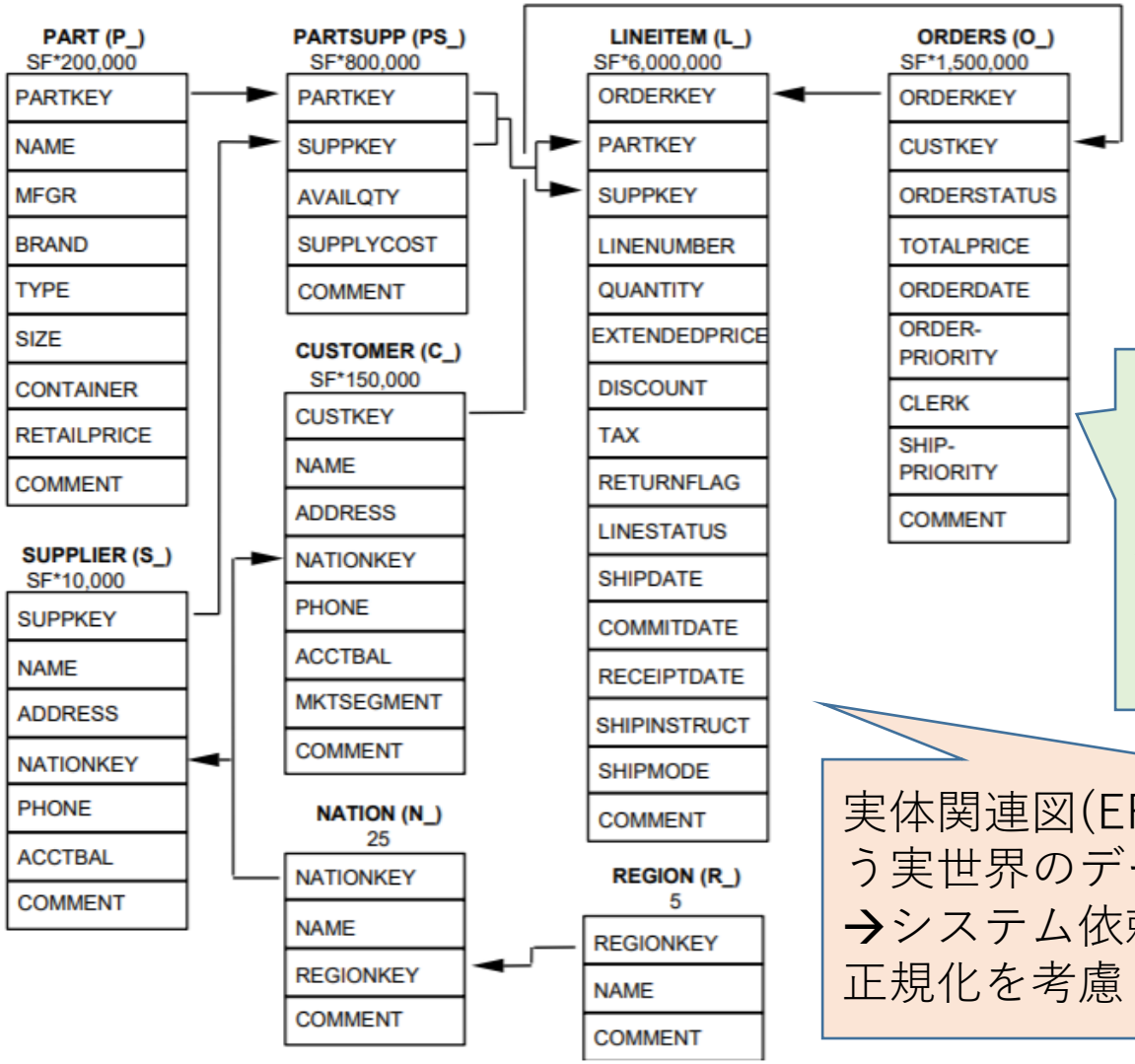
授業で使用する教科書のリレーションスキーマ 教科書(科目,教員,教科書)において、各科目は複数の教員が担当し、各教員が任意の教科書を使用する。ただし、各教科書は特定の科目のみで使用され、教員は複数の科目を担当できるものとする。なお、各教員がある科目で使用する教科書は高々1とする。

(1)このリレーションスキーマは第三正規形になるが、ボイスコッド正規形にはならない。その理由を述べよ。また、更新時異常が起こる事例を例示せよ。

(2)リレーションスキーマ教科書をボイスコッド正規形に正規化せよ。

データモデリング

実社会の中でデータベース化したい範囲からデータ項目を抽出・整理して、データベースの適切な構造を決定することを、データモデリングという。



データベースで広く使われているベンチマークであるTPC-Hのテーブル
#ビジネス分析を模擬
例:直近数年は注文が来てないが、過去はたくさん注文が来ていた企業や地域を抽出,注文額の上位100社

実体関連図(ER図):データベースが扱う実世界のデータ間の関係を表現
→システム依頼者の要望を聞いて、正規化を考慮しながら、ER図を作成

データベース設計

以下の手順により、一般的にデータベースの設計を行う

概念設計:システム構築対象となる実世界からデータ項目を抽出
　　データ間の関係を整理
　　データモデルに依存しない形でモデル化(ER図が一般的)

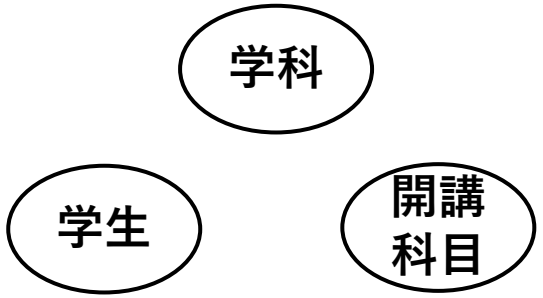
論理設計:概念モデルを、対象とするデータモデルに適合させ、
　　データベースが十分な性能を発揮するようにデータ構造を調整
　　例:リレーションの集合にデータを変形させ、索引や制約を設定

物理設計:システムに要求される性能要件を満たすように、
　　ハードディスクを含めたハードウェア構成や、
　　各リレーションのデータを記憶させる記憶装置・位置の割り当て
　　使用するDBMSの選定

データモデリングの方法1

例題:ある大学の履修管理システムの構築
要求1:履修管理システムは大学のすべての学生と開講科目を管理し、どの学生がどの科目をどの学期に履修したかを検索表示できる。
要求2:履修管理システムは学生の履修履歴を管理し、学生ごと、科目ごとに検索表示できる。

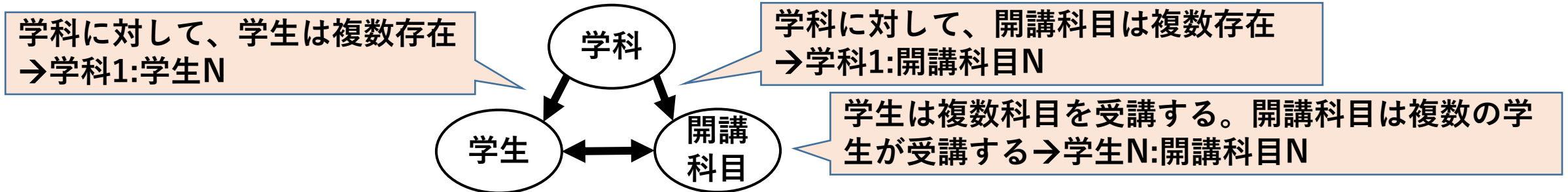
手順1:実体の抽出
実体とは、実世界のデータをモデル化する際のデータの単位であり、リレーショナルデータベースではリレーションに対応する。実世界での物理的な実体を伴うもの(例:学生、学科、開講科目)に着目することを考える。
#学生個人名や学科名などの固有名詞ではなく、総称を挙げる



データモデリングの方法2

例題:ある大学の履修管理システムの構築

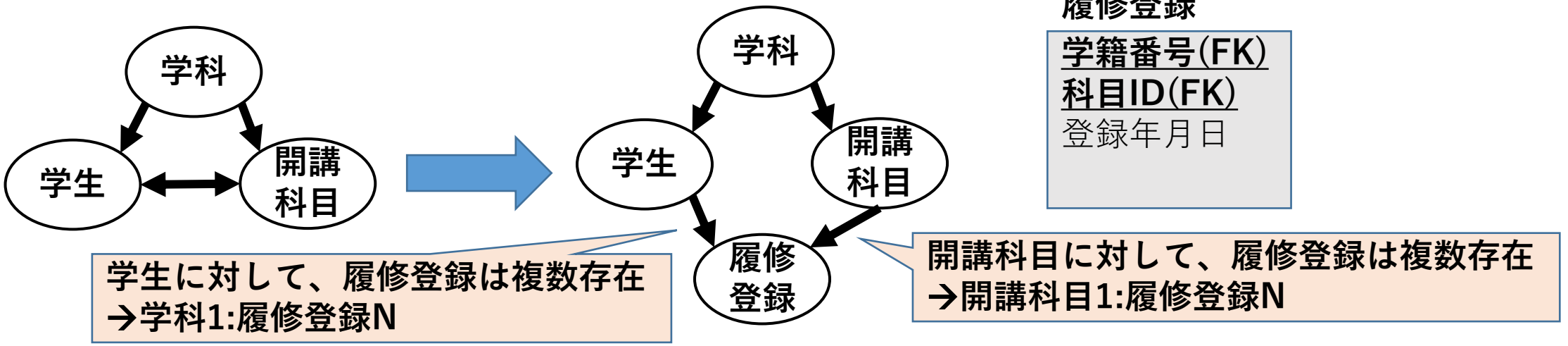
手順2:実体間の関連の設定
2つの実体間の関連を、リレーショナルデータベースでは外部キーで表現する。関連は２つの実体がどのような関係にあるかを考えると設定しやすい。
A):「学生」と「学科」は「所属する・所属される」の関係にある。1学生は必ず1学科に所属し、1学科には複数の学生が所属するため、学科と学生は「1対多」の関連になる。
B):「学生」と「開講科目」は「受講する・される」の関係にある。1学生は複数の開講科目を受講し、1開講科目は複数の学生に受講されるため、学生と開講科目の関連は「多対多」である。



データモデリングの方法3

例題:ある大学の履修管理システムの構築

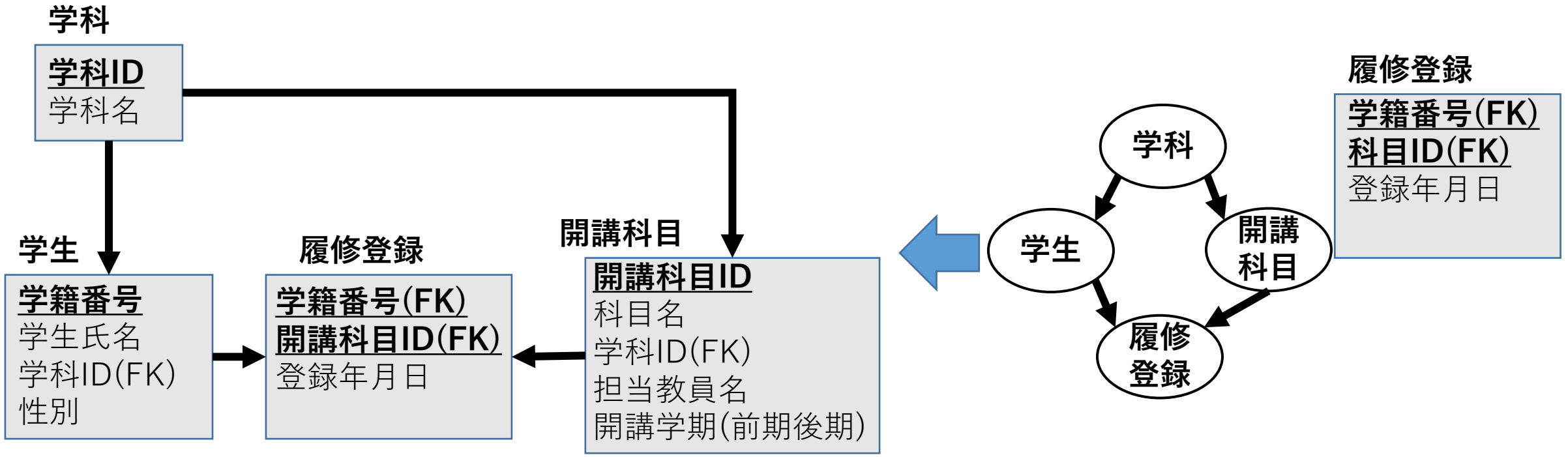
手順3:多対多の関連の分割
実体間の関連に「多対多」がある場合には、実体の間に新たな実体を加えて複数の「1対多」に分解する。
例:「学生」と「開講科目」の間に「履修登録」を追加する。「学生」と「履修登録(ある学生がある科目を履修という情報)」の関連は「１対多」であり、「履修登録」と「開講科目」も同様に「１対多」である。



データモデリングの方法4

例題:ある大学の履修管理システムの構築

手順4:キーと属性の設定
各実体に対し、キーと属性を設定する。キーはその実体を一意に識別するためのデータ項目であり、リレーショナルデータベースでは主キーとなる。
例:「開講科目」に対し「開講科目ID」、「学生」に対し「学籍番号」等



データモデリングの方法5

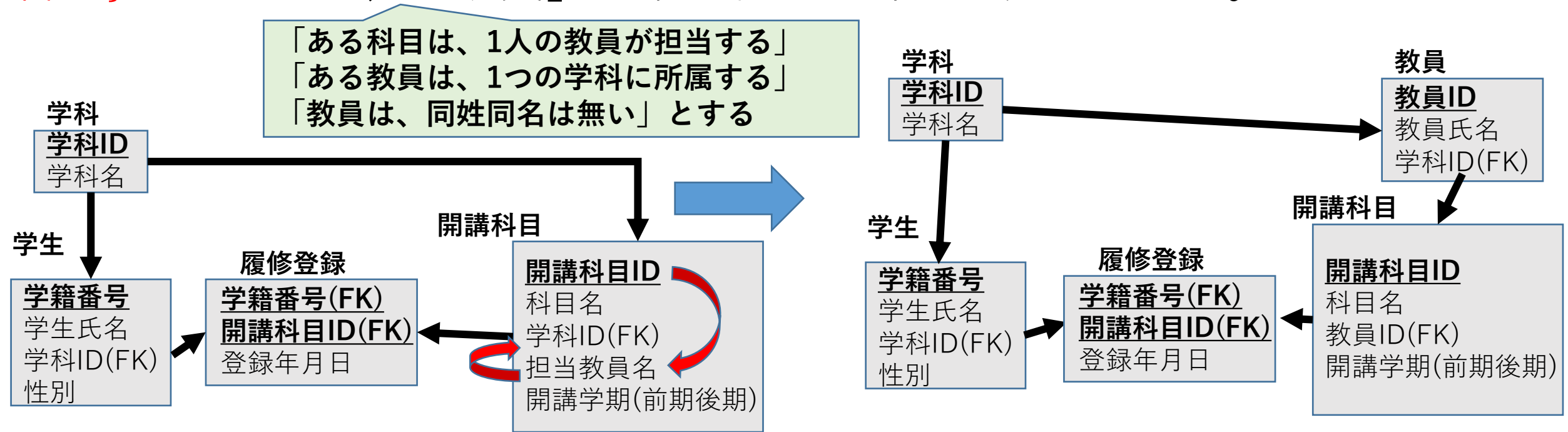
例題:ある大学の履修管理システムの構築

手順5:正規化

第3正規形まで正規化するのが一般的である。

#理由の一つとして、正規化を進めると表分割されていくが、データを復元するためには多数の(処理時間が大きい)表結合(JOIN)をする必要があるため。

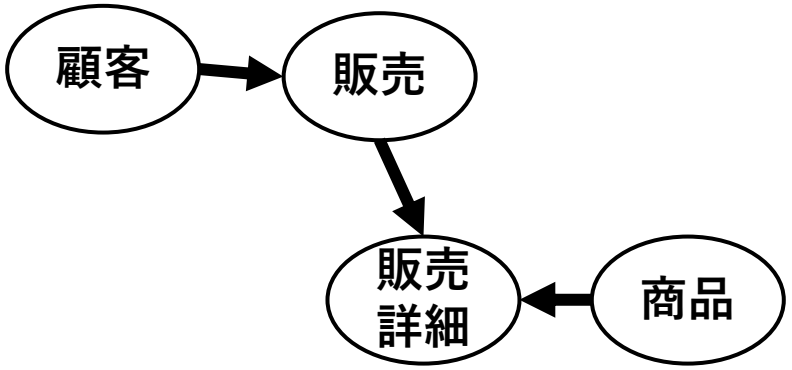
例:「開講科目」の中に推移的関数従属性{開講科目ID}→{担当教員名}→{学科ID}があるので、「教員」を取り出すと第3正規形となる。

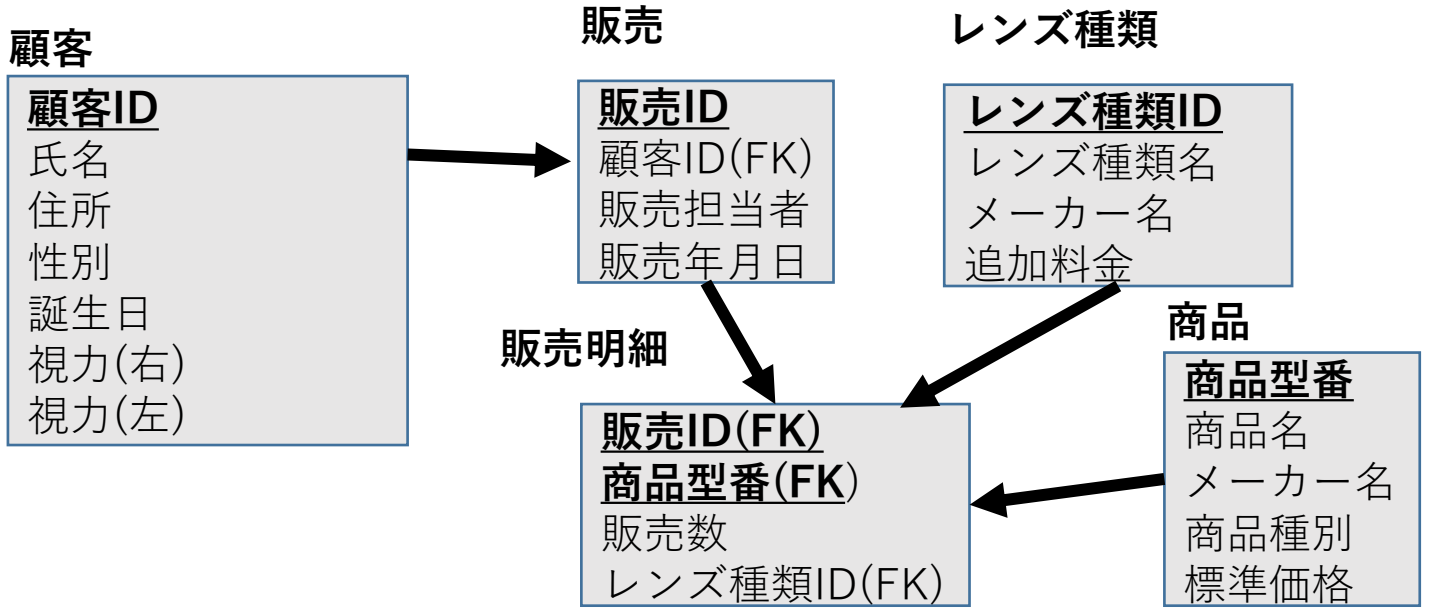


データモデリングの例(眼鏡店の販売管理)

例題:眼鏡店の顧客管理と販売管理のシステム

要求(1):眼鏡と眼鏡用品を販売する。販売する商品は複数の製造メーカーから仕入れており、商品型番によって一意に決まる。
要求(2):購入客は、登録用紙に氏名、住所、性別、誕生日を記入し顧客登録
要求(3):顧客単位で購入履歴を管理。購入履歴や過去視力結果を参照したい
要求(4):眼鏡はフレームごとに値段が決まり、使用するレンズの種類によって追加料金がかかる。





課題10　締切:1/5

10-1)図書館の書籍貸出業務を支援するデータベースシステムを構築したい。
以下の要求を満たす実体関連図を作成せよ。

要求(1):図書館の利用者は、事前に利用者登録を行えば、受付で書籍を借りることができる。

要求(2):利用者は、書籍を最大２週間の期間借りることができる。ただし、同時に借りられる書籍は10冊以内とする。

要求(3):利用者の要望や、書籍の破損状態に応じ、書籍入荷や廃棄を行う。

要求(4):利用者は図書館内の端末で蔵書検索を行い、書籍情報と保管されている書架を知ることができる。

10-2)以下を抽出するSQLを記載せよ。なお、表名や属性名は自ら考案すること。

a)「2週間を超えて貸し出されている本」

b)「累積の貸出本数の多い利用者」

```sql
create table kashidashi(
kashidashi_id char(5) not null,
user_id char(5) not null,
kashidashi_date date,
primary key (kashidashi_id));

insert into kashidashi values('K01','U01','2021-12-01');
insert into kashidashi values('K02','U01','2021-12-21');

select * from kashidashi where kashidashi_date >= (NOW() - INTERVAL 14 day);
select * from kashidashi where kashidashi_date < (NOW() - INTERVAL 14 day);
```

日付型はdate

'YYYY-MM-DD'

貸し出し日>=今日-14日
#2週間以内の貸し出し

貸し出し日<今日-14日
#2週間超過の貸し出し

実行 (Ctrl-Enter) MySQLを学ぶ | プログラミング力診断

出力 入力 コメント 0

```
kashidashi_id   user_id kashidashi_date
K02     U01     2021-12-21
kashidashi_id   user_id kashidashi_date
K01     U01     2021-12-01
```