リレーショナルデータモデル（Relational Data Model）

- E. F. Coddにより1969年に提案されたデータモデル
- ネットワークデータモデルやハイアラキカルデータモデルと比較して、**データ独立性の達成が容易**
- データベースの構成単位は表形式のリレーションであり、操作が単純・明解
- 現在では標準的なデータモデルとなっており、Oracle, Microsoft SQL Serverなど多数のDBMSがある

:リレーショナルデータモデル(1)

- 表形式でデータを表現(表をリレーションと呼ぶ)、データの単位はタップル(表の行に対応)
- リレーションは属性(表の列に対応)を持ち、各属性は同じドメイン(同一の型のデータの集合)のデータを持つ
- タップルの各成分はそれぞれあるドメインの要素である

商品

商品番号	商品名	在庫
G1	冷蔵庫	5
G2	テレビ	7
G3	エアコン	2

顧客

顧客番号	顧客名
C1	竹中
C2	松田

リレーション名 → 注文

タップル

リレーション

注文番号	商品番号	顧客番号	個数
O1	G1	C1	1
O2	G2	C1	3
O3	G2	C2	2
O4	G3	C2	1

ドメイン

dom(商品番号)={G1,G2,G3}
dom(顧客番号)={C1,C2}
dom(商品名)={冷蔵庫,テレビ,エアコン}
dom(顧客名)={竹中,松田}
dom(在庫)={整数}
dom(個数)={整数}

リレーショナルデータモデル(2)

- リレーション
  - リレーションはタップルの「集合」であるので、**要素の重複がない**（一つのリレーションが同じタップルを複数個要素として持つことはない）
- ドメイン
  - ドメインとはデータ型の拡張概念であり、リレーションの属性同士は、**ドメインが同じときのみ比較可能**である
  - ドメインとデータ型の違い：

    前の例で、「商品番号」と「顧客番号」は、データ型はどちらも文字列であるが、 両者は意味的に異なり、比較できない

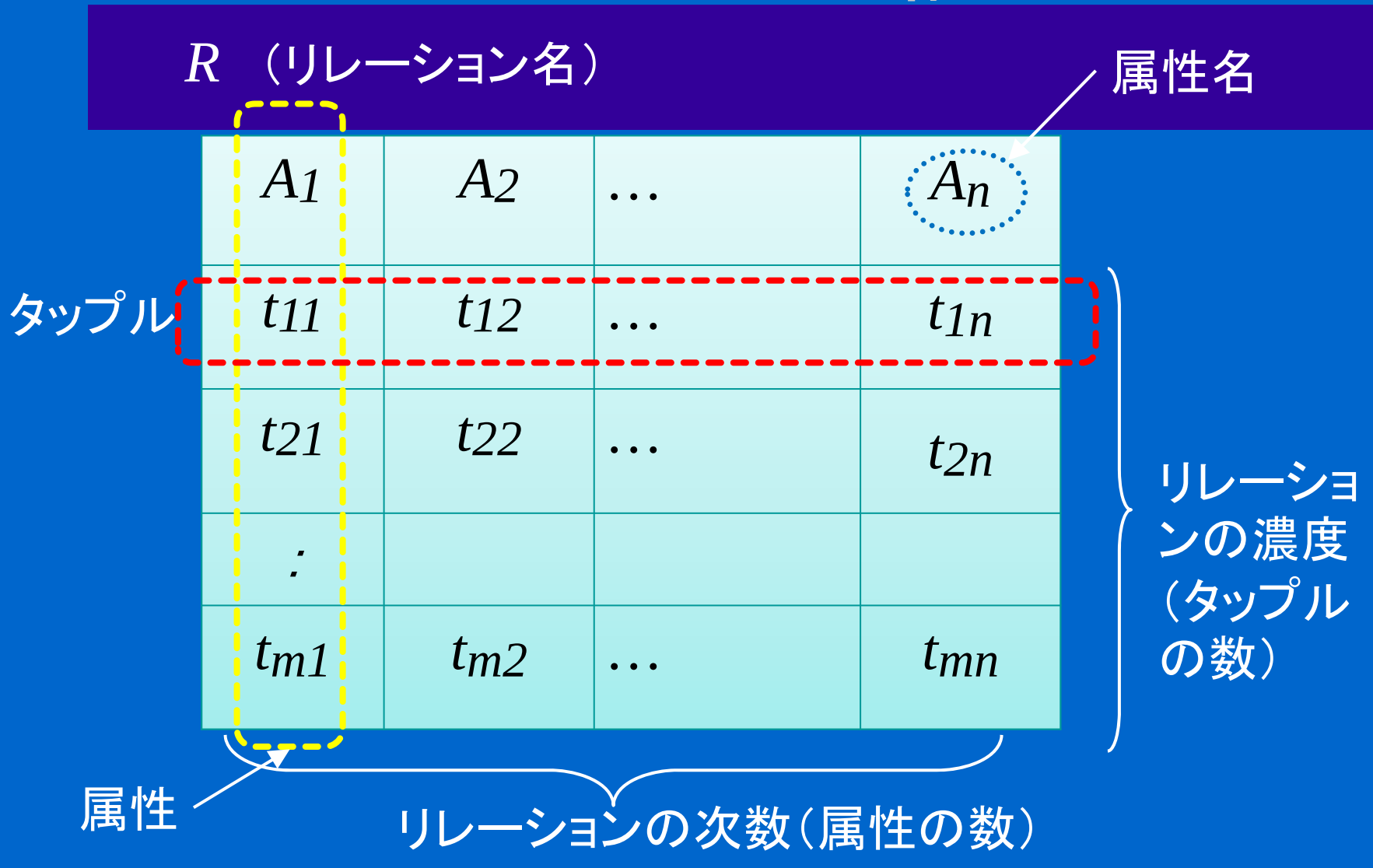
    → ドメインが異なると考える
- 属性
  - リレーションの持つ性質や特徴を表現したもの
  - ドメインと属性の違い：

    同じドメインであっても、属性が異なることがある

    「商品」の「在庫」と「注文」の「個数」はドメインは同じで属性は異なる

まとめ： **データ型 ⊃ ドメイン ⊃ 属性**

リレーションの構造



$t_i = (t_{i1}, t_{i2}, \ldots, t_{in})$: n-タップル（n個組）、 t_{ij}: タップルt_iの第j成分

リレーションの形式的定義(1)

- リレーション(relation):ドメイン$D_1, D_2, \ldots, D_n$上のリレーションRとは、直積$D_1 \times D_2 \times \cdots \times D_n$の有限部分集合
  - 直積(Cartesian product): $D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$
- ドメイン(domain): 同じ型(type)を持つデータの集合
- タップル(tuple): リレーションの要素。タップルは$t = (d_1, d_2, \ldots, d_n)$とn個組で表す(*n*-タップルとも言う)
- リレーションの属性(attribute): 各リレーションでそのリレーションを構成しているドメインを名前(属性名と呼ぶ)で指定する。ある属性A_iで指定されたドメインを**dom(A_i)**と表す

リレーションの形式的定義(2)

- リレーションの**次数**(degree): リレーションが持つ属性の数
- リレーションの**濃度**(cardinality): リレーションが要素として持つタップルの数
- タップルの**成分**(component): タップルが持つ個別のデータ。属性により指定される(例:リレーションRの要素であるタップルtでの属性A_iに対応する成分は**$t[A_i]$**と表す)

リレーションの条件

- リレーショナルデータモデルでは、すべてのリレーションは第一正規形でなくてはならない
- **第一正規形**(1NF: first normal form) : あるリレーションを構成しているすべてのドメインが**単純(simple)**であるとき、そのリレーションは第一正規形であるという
- 単純なドメインの定義:
  1. そのドメインが他のドメインの**直積**(の部分集合)ではなく、かつ、
  2. そのドメインが他のドメインの**巾集合**(の部分集合)ではない

正規化(normalization):

- 第一正規形でないリレーションを第一正規形に変換

→ 直積および巾集合を単純な形に分解する

正規化(Normalization)

直積の正規化

- 属性の分解であり、リレーションの次数（属性の数）は増えるが、リレーションの濃度（タップルの数）は変わらない
- 正規化の例：
  - 住所のように、郵便番号、都道府県、市町村などのデータの組で表されている属性を、それぞれの項目ごとに別の属性に分解
  - 座標のように、(緯度，経度)の組で表されている属性を、それぞれの項目ごとに別の属性に分解

    （注）緯度と経度は、それぞれさらに分解可能である（例えば、N31は、北緯を表すNと緯度を表す31にさらに分解できる）

巾集合の正規化

- 属性の変換とタップルの分解であり、リレーションの濃度（タップルの数）が増えるが、リレーションの次数（属性の数）は変わらない
- 正規化の例：
  - 世帯のように、構成員の集合値を取る属性を各構成員を表す属性に変換することで、タップルを分解

正規化の例(直積)
(normalization of Cartesian product)

住所録(正規化前)

氏名	住所
大阪太郎	560-8531大阪府豊中市待兼山町
神戸次郎	657-8501兵庫県神戸市灘区六甲台町

住所録(正規化後)

氏名	郵便番号	都道府県	市	区	町村
大阪太郎	560-8531	大阪府	豊中市		待兼山町
神戸次郎	657-8501	兵庫県	神戸市	灘区	六甲台町

dom(住所)=dom(郵便番号)×dom(都道府県)×dom(市)×dom(区)×dom(町村)

正規化の例（巾集合）
(normalization of power set)

世帯（正規化前）

世帯	家族
世帯1	{大阪太郎、大阪春子、大阪一太郎}
世帯2	{神戸次郎、神戸夏子、神戸小次郎}

$dom(家族)=2^{dom(構成員)}$

巾集合(power set)とは、ある集合のすべての部分集合からなる集合

例　ドメインD={1,2,3}とすると、Dの巾集合

2^D={{}, {1}, {2}, {3}, {1,2}, {2,3}, {1,3}, {1,2,3}}

となる

世帯（正規化後）

世帯	構成員
世帯1	大阪太郎
世帯1	大阪春子
世帯1	大阪一太郎
世帯2	神戸次郎
世帯2	神戸夏子
世帯2	神戸小次郎

リレーションスキーマとインスタンス

- リレーションスキーマ(relation schema):リレーションの骨組みだけの情報。時間に対して不変。太字で表記(例:　リレーションスキーマ**R**)
- インスタンス(instance):リレーションの実現値(ある時点での値)。時間によって変化。厳密には時刻をつけて記述。
- 超キー(super key):その値だけ決まればタップルを一意に識別できる属性の集合。
  リレーションスキーマ**R**の任意のインスタンスRにおいて $(\forall t, t' \in R)(t[K] = t'[K] \Rightarrow t = t')$
- リレーションはタップルの集合で同じタップルは2個以上ないので、リレーション**R**の**全属性集合Ω_R**={A1, A2, …, An}は常に超キー。
- 候補キー(candidate key):超キーの中で**極小のもの**(これ以上属性を減らすと超キーとならないもの)。主キー(primary key)は候補キーのうちの一つ(そのリレーションのキーとして適切なもの)

R(リレーションスキーマ)

商品番号	商品名

主キー:　商品番号
(商品名が同じタップルが複数あるため商品名だけではタップルが一意に決まらないが、商品番号だと一意に決まる)

R_t(時刻tでのインスタンス)

商品番号	商品名
G1	冷蔵庫
G2	冷蔵庫
G3	テレビ

主キーの例

商品

商品番号	商品名
G1	冷蔵庫
G2	テレビ
G3	エアコン

顧客

顧客番号	顧客名
C1	竹中
C2	松田

注文

注文番号	商品番号	顧客番号	個数
O1	G1	C1	1
O2	G2	C1	3
O3	G2	C2	2
O4	G3	C2	1

商品番号、顧客番号、注文番号（属性名に下線）は主キー

リレーショナルデータベースの言語

次のような言語がある

1. **データ操作言語（DML: Data Manipulation Language）**
   - データベースの問合せや更新を行うための言語
2. **データ定義言語（DDL: Data Definition Language）**
   - データベースの設計を行うための言語
3. **データ制御言語（DCL: Data Control Language)**
   - データベースのデータに対するアクセスや更新の制御を行うための言語

リレーショナルデータベースの操作言語 (Data Manipulation Language)

リレーショナルデータベースを操作するための言語としては、次のようなものがある

- **リレーショナル代数(Relational Algebra)**
  - リレーショナルデータベースの操作を理論的に説明するための言語
  - 計算機上で実装するためのものではない
- **SQL**
  - 計算機上で実装されるための言語であり、多くの処理系が存在する

リレーショナル代数 (Relational Algebra)

- リレーショナル代数は、集合論と述語論理学に基礎をおいている
- リレーション（タップルの集合）を被演算子（operand）とする、次の2種類の演算（operation）がある
  - 集合演算（Set Operation）
  - リレーショナル代数に特有の演算

集合演算

- 和集合演算(Union) $R \cup S$
- 差集合演算(Difference) $R - S$
- 共通集合演算(Intersection) $R \cap S$
- 直積演算(Cartesian product) $R \times S$

（和集合・差集合・共通集合演算では、**RとSのリレーションスキーマが和両立**である必要があることに注意）

$$R \cup S \stackrel{def}{=} \{t | t \in R \vee t \in S\}$$

$$R - S \stackrel{def}{=} \{t | t \in R \wedge t \notin S\}$$

$$R \cap S \stackrel{def}{=} \{t | t \in R \wedge t \in S\}$$

$$R \times S \stackrel{def}{=} \{(r, s) | r \in R \wedge s \in S\}$$

和両立（Union Compatible）

リレーションスキーマ***R***(A1, A2, .., An)と***S***(B1,B2, …, Bm)が**和両立**とは、次の2つの条件を満たしているときをいう

1. $n = m$
2. 各i $(1 \leq i \leq n)$に対してdom(Ai) =dom(Bi)が成立する

集合演算の例で使うリレーション

emp1

empno	ename	deptno	salary	roomno
E02	Morgan	D01	70	R03
E03	Robert	D01	80	R02
E05	Lincoln	D02	90	R11
E06	Taylor	D01	80	R12
E93	Matsuda	D91	90	R91

emp2

empno	ename	deptno	salary	roomno
E01	Smith	D01	100	R01
E04	Washington	D02	120	R10
E05	Lincoln	D02	90	R11
E91	Suzuki	D91	140	R91
E92	Tanaka	D91	130	R91
E93	Matsuda	D91	90	R91

emp1 ∪ emp2

empno	ename	deptno	salary	roomno
E01	Smith	D01	100	R01
E02	Morgan	D01	70	R03
E03	Robert	D01	80	R02
E04	Washington	D02	120	R10
E05	Lincoln	D02	90	R11
E06	Taylor	D01	80	R12
E91	Suzuki	D91	140	R91
E92	Tanaka	D91	130	R91
E93	Matsuda	D91	90	R91

emp1 — emp2

empno	ename	deptno	salary	roomno
E02	Morgan	D01	70	R03
E03	Robert	D01	80	R02
E06	Taylor	D01	80	R12

emp1 ∩ emp2

empno	ename	deptno	salary	roomno
E05	Lincoln	D02	90	R11
E93	Matsuda	D91	90	R91

リレーショナル代数に特有の演算(1)

- 射影演算(projection) $\pi_{A1,A2,...,An}(R)$
- θ—選択演算(selection) $\sigma_{Ai\theta Aj}(R)$
- θ—結合演算(join) $R \bowtie_{A_i \theta B_j} S$
- 自然結合演算(natural join) $R \bowtie S$
- 商演算(division) $R \div S$

リレーショナル代数に特有の演算(2)

R

A_1	A_2	…	A_n
t_{11}	t_{12}	…	t_{1n}
t_{21}	t_{22}	…	t_{2n}
:			
t_{m1}	t_{m2}	…	t_{mn}

θ—選択
（例）
$\sigma_{A2=t12}(R)$

S

B_1	B_2	…	B_n
t_{11}	t_{12}	…	t_{1n}
t_{21}	t_{22}	…	t_{2n}
:			
t_{m1}	t_{m2}	…	t_{mn}

射影
（例）$\pi_{A1}(R)$

結合

$$R \bowtie_{A_i \theta B_j} S$$

リレーショナル代数に特有の演算(3): 定義

$$\pi_X(R) \stackrel{def}{=} \{u | t \in R \wedge u = t[X]\}$$

$$\sigma_{A_i \theta A_j}(R) \stackrel{def}{=} \{t | t \in R \wedge t[A_i]\ \theta\ t[A_j]\}$$

(ただし，A_i と A_j は $\theta-$ 比較可能)

$$R \bowtie_{A_i \theta B_j} S \stackrel{def}{=} \{v | v \in R \times S \wedge v[R.A_i]\ \theta\ v[S.B_j]\}$$

$$R \bowtie_{A_i \theta B_j} S \stackrel{def}{=} \sigma_{R.A_i\ \theta\ S.B_j}(R \times S)$$

(ただし，A_i と B_j は $\theta-$ 比較可能)

$$R \bowtie S \stackrel{def}{=} \pi_{R.A_1,\cdots,R.A_n,S.D_l,\cdots,S.D_{m-l}}(\sigma_{R.C_1=S.C_1,\cdots,R.C_l=S.C_l}(R \times S))$$

$$R \div S \stackrel{def}{=} \{v | v \in \pi_{A_1,\cdots,A_{n-m}}(R) \wedge (\forall u \in S)((v,u) \in R)\}$$

$$R \div S \stackrel{def}{=} \pi_{A_1,\cdots,A_{n-m}}(R) - \pi_{A_1,\cdots,A_{n-m}}((\pi_{A_1,\cdots,A_{n-m}}(R) \times S) - R)$$

演算の例で使うリレーション

emp

empno	ename	deptno	salary	roomno
E01	Smith	D01	100	R01
E02	Morgan	D01	70	R03
E03	Robert	D01	80	R02
E04	Washington	D02	120	R10
E05	Lincoln	D02	90	R11
E06	Taylor	D01	80	R12
E91	Suzuki	D91	140	R91
E92	Tanaka	D91	130	R91
E93	Matsuda	D91	90	R91

dept

deptno	dname	manager
D01	Account	E01
D02	Personnel	E04
D91	Database	E91

$\pi_{deptno}(emp)$ （射影）

deptno
D01
D02
D91

←演算結果はリレーションであるためタップルの重複が取り除かれていることに注意

$\sigma_{salary>=100}(emp)$ （選択）

empno	ename	deptno	salary	roomno
E01	Smith	D01	100	R01
E04	Washington	D02	120	R10
E91	Suzuki	D91	140	R91
E92	Tanaka	D91	130	R91

$$emp \bowtie_{emp.deptno=dept.deptno} dept$$

（結合）

emp. empno	emp.ename	emp. deptno	emp. salary	emp. roomno	dept. deptno	dept. dname	dept. manager
E01	Smith	D01	100	R01	D01	Account	E01
E02	Morgan	D01	70	R03	D01	Account	E01
E03	Robert	D01	80	R02	D01	Account	E01
E04	Washington	D02	120	R10	D02	Personnel	E04
E05	Lincoln	D02	90	R11	D02	Personnel	E04
E06	Taylor	D01	80	R12	D01	Account	E01
E91	Suzuki	D91	140	R91	D91	Database	E91
E92	Tanaka	D91	130	R91	D91	Database	E91
E93	Matsuda	D91	90	R91	D91	Database	E91

$emp \bowtie dept$（自然結合）

empno	ename	deptno	salary	roomno	dname	manager
E01	Smith	D01	100	R01	Account	E01
E02	Morgan	D01	70	R03	Account	E01
E03	Robert	D01	80	R02	Account	E01
E04	Washington	D02	120	R10	Personnel	E04
E05	Lincoln	D02	90	R11	Personnel	E04
E06	Taylor	D01	80	R12	Account	E01
E91	Suzuki	D91	140	R91	Database	E91
E92	Tanaka	D91	130	R91	Database	E91
E93	Matsuda	D91	90	R91	Database	E91

↑

重複する属性が削除されていることに注意

商演算の意味

- 直積の逆演算（R×S÷SはRになる）
- Sの要素をすべて含むようなRのタップル（Sと共通の属性を除く）を返す

store

store_name	goods
A	bread
A	butter
A	milk
B	bread
B	jam
C	coffee

shopping_list

goods
bread
butter
milk

store ÷ shopping_list

store_name
A

商演算の実行結果(1)

$$R \div S = \pi_X(R) - \pi_X((\pi_X(R) \times S) - R)$$

(a)

(b)

(c)

(d)

R

X	Y
a	1
a	2
b	1

S

Y
1
2

(a)

X	Y
a	1
a	2
b	1
b	2

(b)

X	Y
b	2

(c)

X
b

(d)

X
a

商演算の実行結果(2)

(a) $\pi_X(R) \times S$の結果:「商店」と「商品」のすべての組合せを生成する

(b) $(\pi_X(R) \times S) - R$の結果:「商店」と「商品」のすべての組合せから現在の「商店」ー「商品」の表に含まれているものを除く(現在の「商店」では販売されていない「商品」が求められる)

(c) $\pi_X((\pi_X(R) \times S) - R)$の結果:販売されていない「商品」を持つ「商店」を求める

(d) $\pi_X(R) - \pi_X((\pi_X(R) \times S) - R)$の結果:「商店」のうちで販売されていない「商品」を持たないものを求める(すべての「商品」を販売している「商店」を求める)