

# データベース設計論

## 第4回 制約と操作体系

---

2019/10/29

# リレーションの制約

- リレーションは(少なくとも) **第一正規形**を満たさなければならない
- リレーションは **一貫性制約**を満たさなければならない
  - 一意性制約(unique constraint)
  - 参照制約(referential constraints)

# 第一正規形であるとは

- ドメインはシンプル(simple)でなければならない
- シンプルである, とは?
  - A) ドメインが, あるドメインの直積であってはいけない
  - B) ドメインが, あるドメインのべき集合であってはならない

## 非第一正規形の例

社員番号	社員名	...
0650	(鈴木, 一郎)	
1508	(浜崎, アユ)	...

A)を侵害している例

社員番号	社員名	趣味
0650	鈴木一郎	{野球, 盆栽, コイン収集}
1508	浜崎アユ	{作詞, ショッピング}

B)を侵害している例

# 正規化(normalization)

社員番号	社員名	...
0650	(鈴木, 一郎)	
1508	(浜崎, アユ)	...

A)を侵害している例

複数の属性に分解する

社員番号	社員(姓)	社員(名)	...
0650	鈴木	一郎	
1508	浜崎	アユ	...

社員番号	社員名	趣味
0650	鈴木一郎	{野球, 盆栽, コイン収集}
1508	浜崎アユ	{作詞, ショッピング}

複数のタプルに分解する

社員番号	社員名	趣味
0650	鈴木一郎	野球
0650	鈴木一郎	盆栽
0650	鈴木一郎	コイン収集
1508	浜崎アユ	作詞
1508	浜崎アユ	ショッピング

# リレーションの制約

- リレーションは(少なくとも) **第一正規形**を満たさなければならない
- リレーションは **一貫性制約**を満たさなければならない
  - 一意性制約(unique constraint)
  - 参照制約(referential constraints)

# 主キー

## 一意性制約(unique constraint)

- タプルの重複は許されない

## 主キー(Primary Key)

1. リレーションスキーマの**部分属性リスト**でその属性値がタプルを一意に識別し、かつ**極小**であるもの

↑  
「最小」ではない。  
属性のひとつがかけるとその性質がなくなる

2. **キー制約: Key constraint**

主キーの属性値に空値(NULL)は許されない

# 外部キー (foreign key)

社員

社員番号	社員名	趣味
0650	鈴木一郎	野球
0650	鈴木一郎	盆栽
0650	鈴木一郎	コイン収集
1508	浜崎アユ	作詞
1508	浜崎アユ	ショッピング

分解

社員

社員番号	社員名
0650	鈴木一郎
1508	浜崎アユ

趣味

社員番号	趣味
0650	野球
0650	盆栽
0650	コイン収集
1508	作詞

## 参照制約

- 外部キー
  - 他のリレーションのあるタプルを参照する属性
  - 参照するタプルの主キーの値を持つ
- 参照制約
  - 外部キーの属性値は、参照する主キーの値かあるいは空値(NULL)しか許されない

社員

社員番号	社員名
0650	鈴木一郎
1508	浜崎アユ

趣味

社員番号	趣味
0650	野球
0650	盆栽
0650	コイン収集
1508	作詞
1508	ショッピング



# 外部キー (foreign key)

## 学生

学籍番号	氏名
g0720501	赤井かな
g0720502	伊藤緑
g0720503	内田洋子
g0720504	内村亜衣
...	...

## 授業

授業番号	授業名
PR001	プログラミング実習
IT002	情報理論
AI003	人工知能論

## 履修

授業番号	学籍番号	成績
PR001	g0720501	
PR001	g0720504	
IT002	g0720502	
IT002	g0720507	
AI003	g0720503	

# リレーションスキーマの表記の仕方

- 主キーの下にはアンダーラインを引く
- 外部キーの下には点線を引く

社員




<u>社員番号</u>	社員名
0650	鈴木一郎
1508	浜崎アユ

趣味

<u>社員番号</u>	趣味
0650	野球
0650	盆栽
0650	コイン収集
1508	作詞
1508	ショッピング

# 演習1: 以下のテーブルを第1正規形に直しましょう

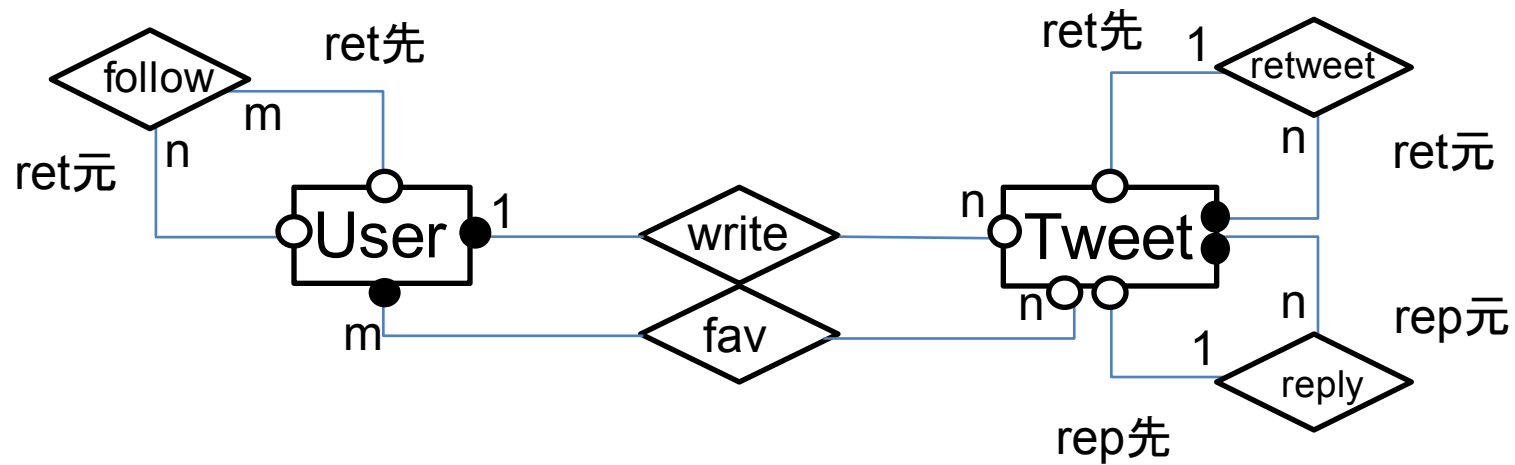
- 栄養成分(商品名, 原材料, アルコール分, エネルギー, タンパク質, 脂質, 糖質, 食物繊維, ナトリウム, プリン体, 賞味期間)

商品名	原材料	アルコール分	エネルギー	たんぱく質	脂質	糖質	食物繊維	ナトリウム	プリン体	賞味期間
 ザ・プレミアム・モルツ	麦芽、ホップ	5.5%	47kcal	0.4～0.6g	0g	3.8g	0～0.1g	0～7mg	約9.5mg	9ヶ月
 ザ・プレミアム・モルツ〈黒〉	麦芽、ホップ	5.5%	52kcal	0.4～0.7g	0g	4.6g	0.1～0.5g	0～7mg	約9.4mg	9ヶ月
 ザ・プレミアム・モルツ〈コクのブレンド〉	麦芽、ホップ	5.5%	49kcal	0.4～0.7g	0g	3.8g	0～0.5g	0～7mg	約9.5mg	9ヶ月

# ER図から リレーションを設計する

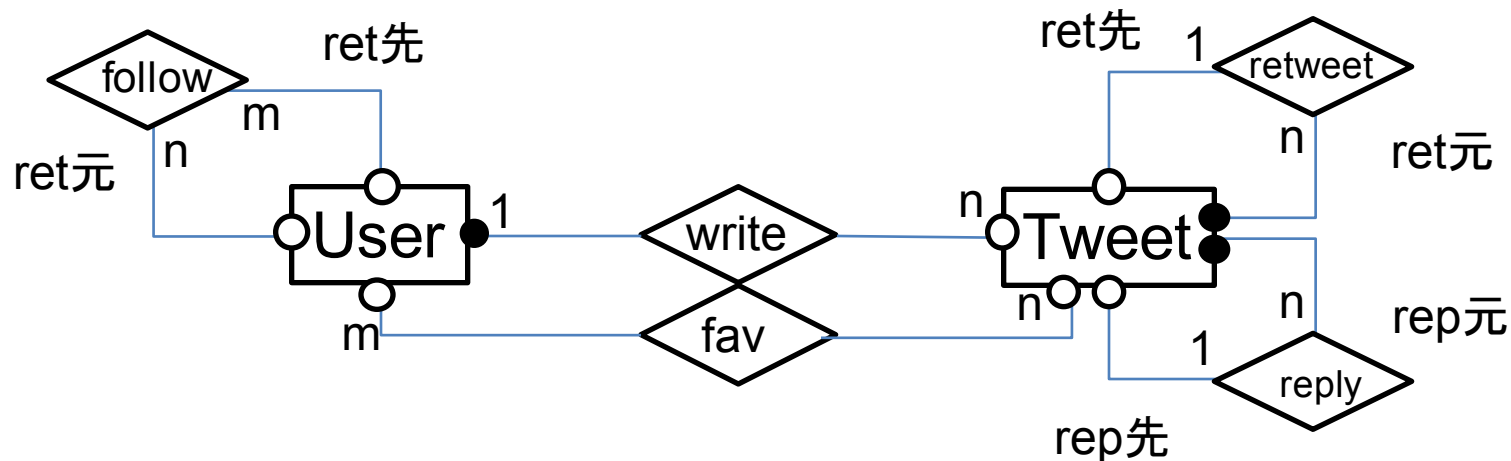
---

# 例として使う ER図



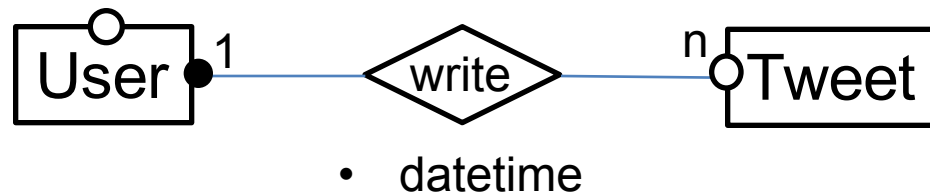
# 手順1: 実体をリレーションスキーマにする

- user(account, name, email)
- tweet(id, content)



## 手順2: 1:nの関係の場合の対処

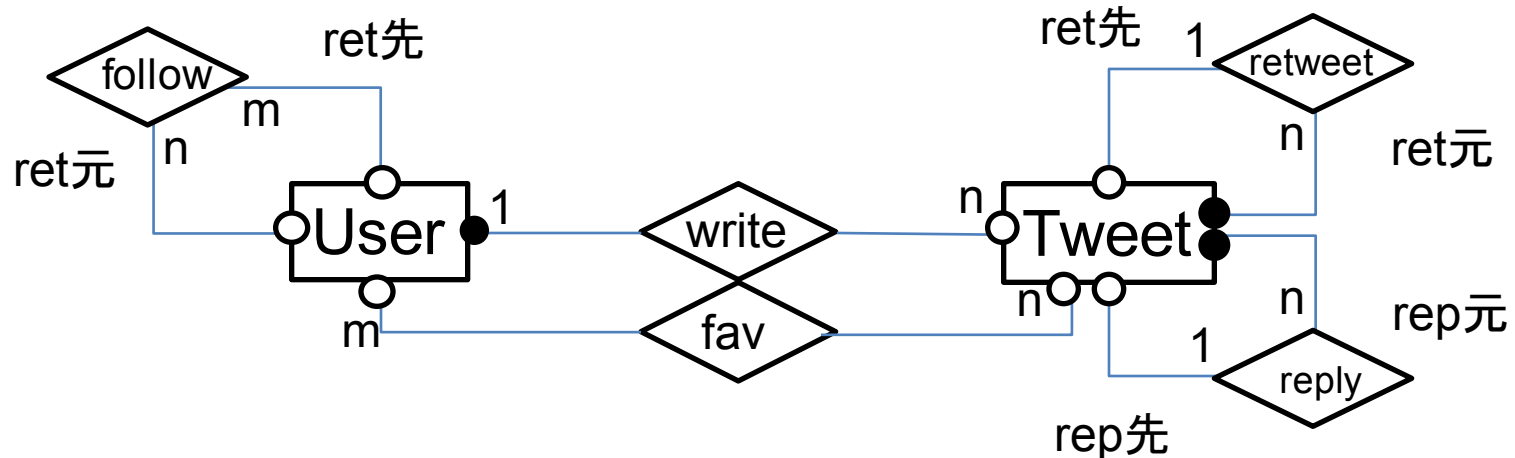
- n側の実体に対するリレーションスキーマに1側の主キーを外部キーとして追加
- 関連に属性がついていたら、それもn側の実体に追加
  - user(account, name, email)
  - tweet(id, content, account, datetime)



# 他の1:nの関係もリレーションに反映

- user(account, name, email)
- tweet(id, content, account, datetime, retweeted\_id)

ret元のid (retweeted\_id) をret先の外部キーとする

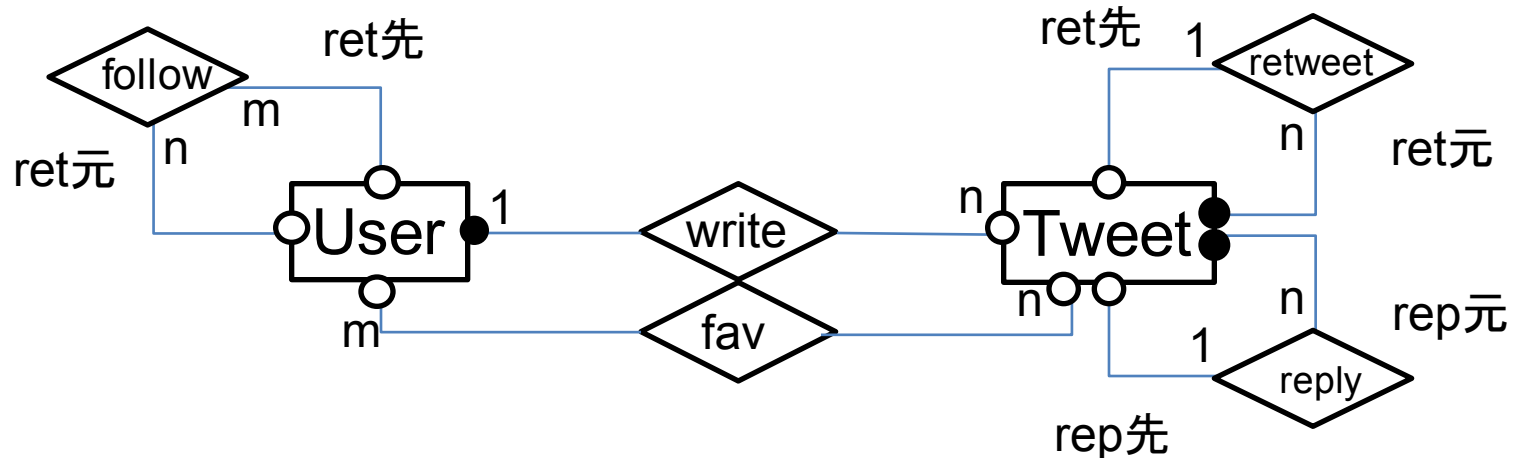




# 他の1:nの関係もリレーションに反映

- user(account, name, email)
- tweet(id, content, account, datetime, retweeted\_id, replied\_id)

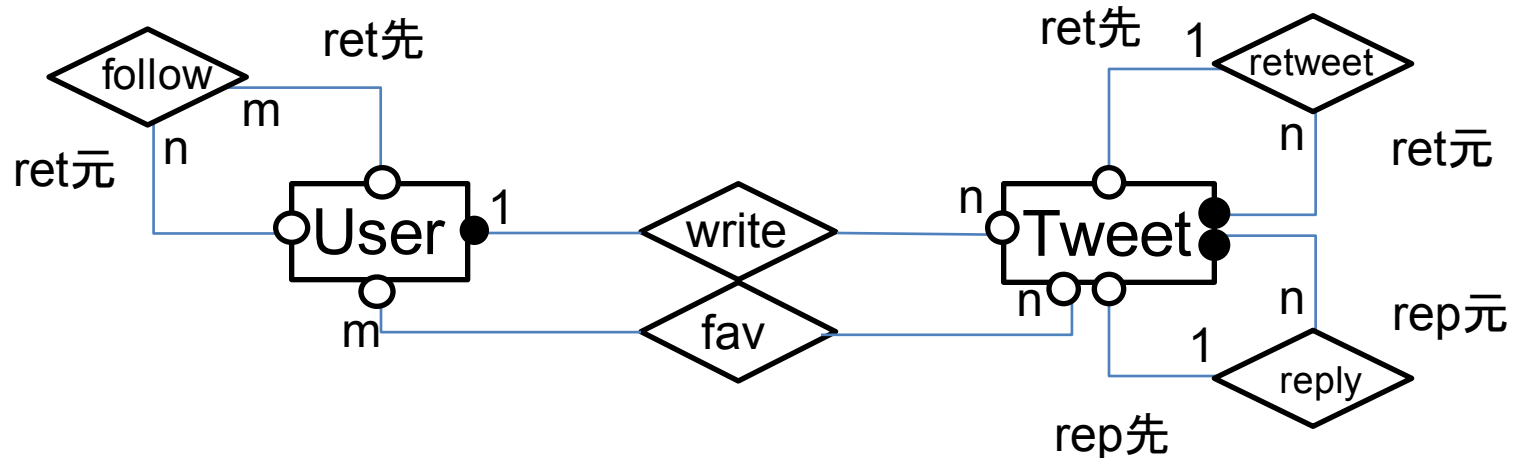
rep元のid(replied\_id)をret先の外部キーとする



# 他の1:nの関係もリレーションに反映

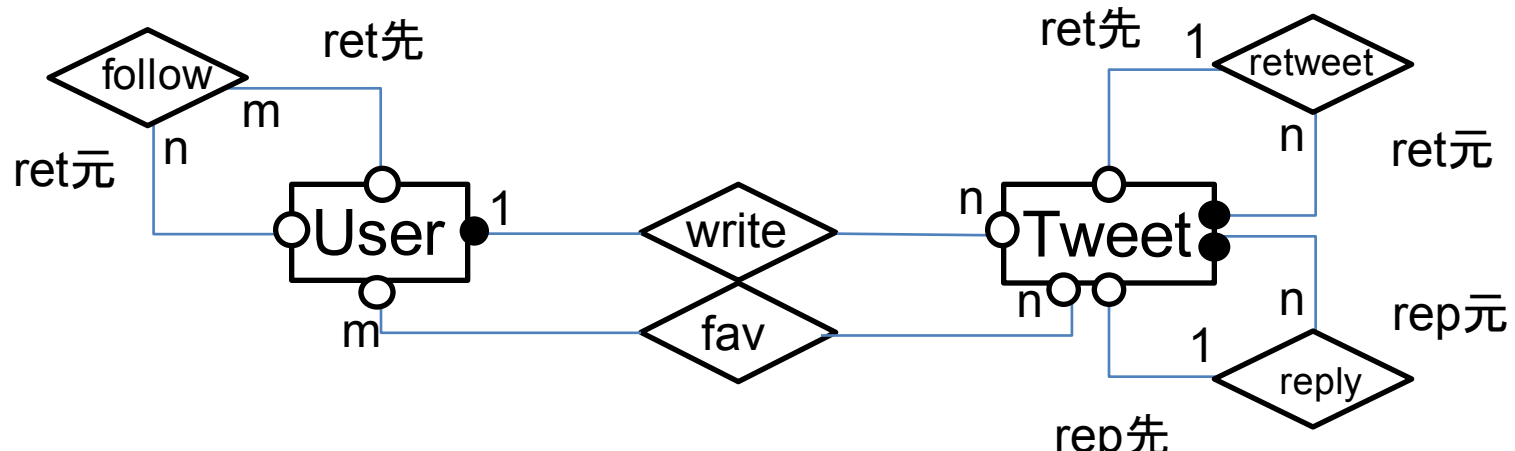
- user(account, name, email)
- tweet(id, content, account, datetime, retweeted\_id, replied\_id)

rep元のid(replied\_id)をret先の外部キーとする



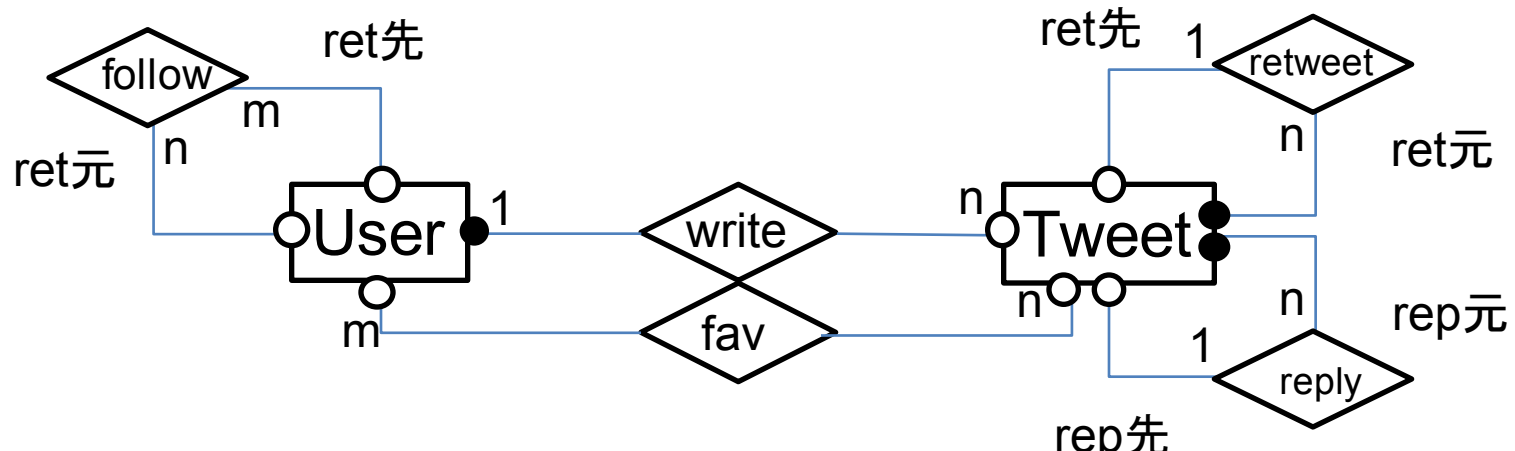
## 手順3: m:n 関係の対処

- 関連に対するリレーションスキーマを作る
- 二つの実体の主キーを追加し、これらを外部キーとする
- user(account, name, email)
- tweet(id, content, account, datetime, retweeted\_id, replied\_id)
- favorite(account, tweet\_id, datetime)



## 手順3: m:n 関係の対処

- 関連に対するリレーションスキーマを作る
- 二つの実体の主キーを追加し、これらを外部キーとする
- user(account, name, email)
- tweet(id, content, account, datetime, retweeted\_id, replied\_id)
- favorite(account, tweet\_id, datetime)
- follow(follower\_account, followee\_account)



# リレーショナルデータベースモデルの操作体系

- **関係論理** (第一階述語論理に基づく)

- $P(t)$ を述語論理とする時, それを $P(t)$ が真となるものの集合 $\{t \mid P(t)\}$ を求める

$P(t) \equiv$  “ $t$ はAB型である”  
 $\{t \mid P(t)\}$  : AB型の人集合

- 非手続的言語
- **SQL**のベースとなる操作体系

- **関係代数**

- 集合に対する演算の組合せで必要な集合を求める

$A - B, A \cup B, A \cap B, A \times B, \neg A,$   
 $\sigma_C A, \pi_a A, \delta A, A \bowtie B$

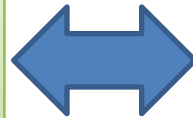
- 手続き的言語
- 関係論理と等価
- 問合せ実行プランの生成に必要な体系

# 操作体系の関係

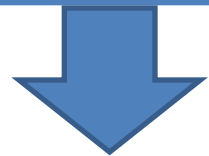
関係完備  
relational complete

RCで書いた式は  
RAでも書くことができる  
RAで書いた式は  
RCでも書くことができる

関係論理  
relational calculus  
(RC)



関係代数  
relational algebra  
(RA)



関係完備

SQL

# 関係代数

- リレーションを対象にした演算の組合せで問合せ (query) を表す
- 演算子
  - 和( $A \cup B$ ), 差( $A - B$ ), 交差( $A \cap B$ )
  - 直積 ( $A \times B$ )
  - 射影( $\pi_L(R)$ ), 選択 ( $\sigma_C(R)$ )
  - 結合( $A \bowtie_C B$ )
  - 商( $A \div B$ )

関係代数のために  
導入された演算子

# SQL

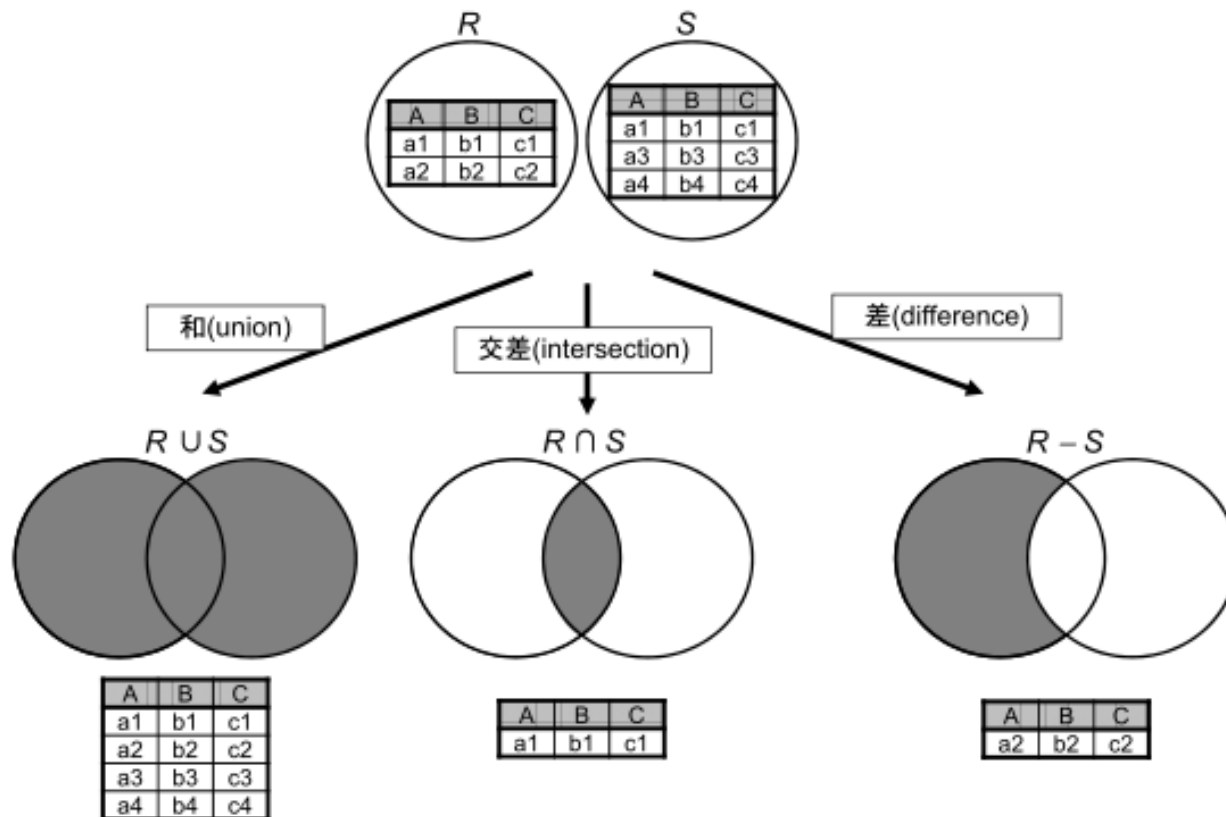
- 関係論理に基づいたデータベース問合せ言語
- ISO国際標準で規格化されている

```
SELECT 属性名, 属性名, ...  
FROM <リレーション名>, <リレーション名>  
WHERE <検索条件>
```



# 関係代数の演算子

- 和, 差, 交差



# 射影(projection) $\pi_L(R)$

- $R_2 = \pi_L(R_1)$ 
  - $L$ は $R_1$ から選んだ属性のリスト
  - $R_2$ は $R_1$ の各タプルの $L$ にある属性を指定された順番で抜き出したもの
- 例2) userの名を一覧を求める

$\pi_{name}(user)$

user(account, name, email)

account	name	email
ariyoshihiroiki	有吉弘行	ariyoshi@example.com
RyoNishikido_JP	錦戸 亮	nishikido@aaa.net
okazaki_taiiku	岡崎体育	taiiku@okazaki.jp

# 射影(projection) $\pi_L(R)$

- 射影演算は以下の関係論理式とSQL文で表すことができる

- 関係論理式

$$L = \{l_1, \dots, l_n\} \text{として}$$

$$\{t \mid s \in R \wedge t.l_1 = s.l_1 \wedge \dots \wedge t.l_n = s.l_n\}$$

- SQL文

SELECT  $l_1, \dots, l_n$  FROM R

# 選択 (selection ) $\sigma_C(R)$

- $R_2 = \sigma_C(R_1)$ 
  - $C$ は $R_1$ の属性を参照する条件
  - $R_2$ は条件 $C$ を満たすような $R_1$ のタプルすべて
- 例1) nameが岡崎体育であるuserのタプル

$\sigma_{name='岡崎体育', user}$

user(account, name, email)

account	name	email
ariyoshihiroiki	有吉弘行	ariyoshi@example.com
RyoNishikido_JP	錦戸 亮	nishikido@aaa.net
okazaki_taiiku	岡崎体育	taiiku@okazaki.jp

# 選択 (selection ) $\sigma_C(R)$

- 選択演算は以下の関係論理式とSQL文で表すことができる

- 関係論理式

$C$ は $R$ に対する論理式 $C(R)$ とする  
 $\{t | t \in R \wedge C(R)\}$

- SQL文

```
SELECT *  
FROM R  
WHERE C
```

該当するタプルの全ての属性をもとめたいときには「\*」と書きます。

# 演算子の組合せ

- 関係代数の演算子の出力はリレーションなので出力結果に対して演算を適用できる

- 例)  $R_2 = \sigma_{name='岡崎体育'}(user)$

account	name	email
ariyoshihiroiki	有吉弘行	ariyoshi@example.com
RyoNishikido_JP	錦戸 亮	nishikido@aaa.net
okazaki_taiiku	岡崎体育	taiiku@okazaki.jp

$$R_3 = \pi_{account}(R_2)$$

account
okazaki_taiiku

上記の処理をまとめて書ける

$$R_3 = \pi_{account}(\sigma_{name='岡崎体育'}(user))$$

# 演算子の組合せ

- 例3) 岡崎体育のアカウント名を求める

$$\pi_{account}(\sigma_{name='岡崎体育'}(user))$$

対応するSQL文

```
SELECT u.account  
FROM user u  
WHERE u.name = '岡崎体育'
```

単一のリレーションに関する関係論理式は  
選択演算と射影演算の組合せで表すことができる

# $\theta$ -結合(theta-join) $R_1 \bowtie_C R_2$

- 二つのリレーション  $R_1, R_2$  の各タプルのうち、条件  $C$  を満たす組合せを求める
- 例

$U = \pi_{account, name} user$

account	name
ariyoshihiroiki	有吉弘行
RyoNishikido_JP	錦戸 亮
okazaki_taiiku	岡崎体育

$T = \pi_{account, content} tweet$

account	content
RyoNishikido_JP	これリプライ?
okazaki_taiiku	リツイートです
RyoNishikido_JP	そうなの?

$U \bowtie_{U.account=T.account} T$

U.account	U.name	T.account	T.content
Ryonishikido_JP	錦戸 亮	Ryonishikido_JP	これリプライ?
okazaki_taiiku	岡崎体育	okazaki_taiiku	リツイートです
Ryonishikido_JP	錦戸 亮	Ryonishikido_JP	そうなの?



## $\theta$ -結合(theta-join) $R_1 \bowtie_C R_2$

- 例4) 錦戸亮の全ツイートの時刻とツイート内容を求める
  1.  $R_1$ : name='錦戸 亮'であるユーザ
    - $R_1 = \sigma_{name='錦戸 亮', user}$
  2.  $R_2$ :  $R_1$ に対応するtweetのタプルを求める
    - $R_2 = R_1 \bowtie_{user.account=tweet.account} tweet$
  3.  $R_3$ :  $R_2$ からdatetimeとcontentを射影する
    - $R_3 = \pi_{datetime, content} R_2$

$$\pi_{datetime, content}((\sigma_{name='錦戸 亮', user} \bowtie_{user.account=tweet.account} tweet))$$

# SQL→関係代数→実行プラン

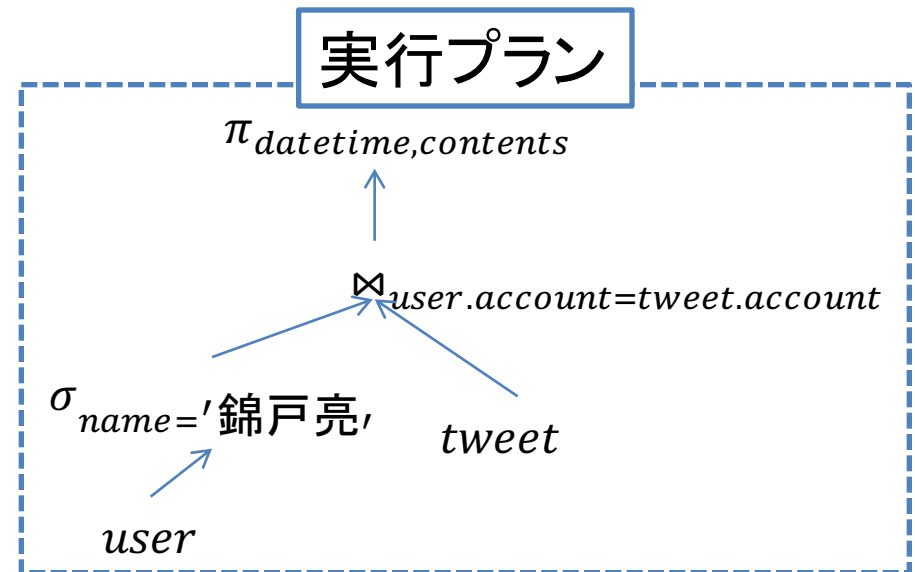
- 利用者が指定する問合せは非手続的

```
SELECT t.datetime, t.content
FROM user u, tweet t
WHERE u.name = '錦戸亮'
      and u.account = t.account
```

- DBMSはそれと等価な関係代数式を求める

```
 $\pi_{datetime, contents}(\sigma_{name='錦戸亮', user} \bowtie_{user.account=tweet.account} tweet)$ 
```

- 関係代数式から実行プランを求め、最適なプランに書き換えて実行する



※DBMSは各演算子のための実行プログラムをいくつか用意しており、最適なプログラムを選ぶ