データベース
第1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、簡単な問合せなど、基本的なSQLの使い方を理解できる。
	5週	SQL(2)	RDBMSの利用全般に用いられる言語SQLを作成できる。SQLにおける問合せを行うselect文を理解できる。
	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	リレーションの更新時に発生しうるデータの不整合、およびその解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	実社会の中でデータベース化したい範囲を決定し、データ項目を抽出・整理して適切なデータ構造を決定する作業であるデータモデリングが理解できる。
	12週	SQL(3)	RDBMSの利用全般に用いられる言語SQLを作成できる。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。

注意点と成績評価について(シラバス抜粋)

注意点	少なくとも1つ以上のプログラミング言語を習得しておくことを履修条件とする。 データ構造とアルゴリズム関連の知識を有していることが望ましい。 合格の対象としない欠席条件(割合) 1/3以上の欠課

欠席回数には注意してください。

評価割合

	試験	発表	相互評価	態度	ポートフォリオ	その他	合計
総合評価割合	60	0	0	0	40	0	100
基礎的能力	0	0	0	0	0	0	0
専門的能力	60	0	0	0	40	0	100
分野横断的能力	0	0	0	0	0	0	0

課題点です。
提出期限に遅れると、各回の課題点の最高点を1点減点します。

この講義でやること

・**<u>データベースの構成技術を理解する</u>**

例: SQL、データ格納形式、問合せ最適化など

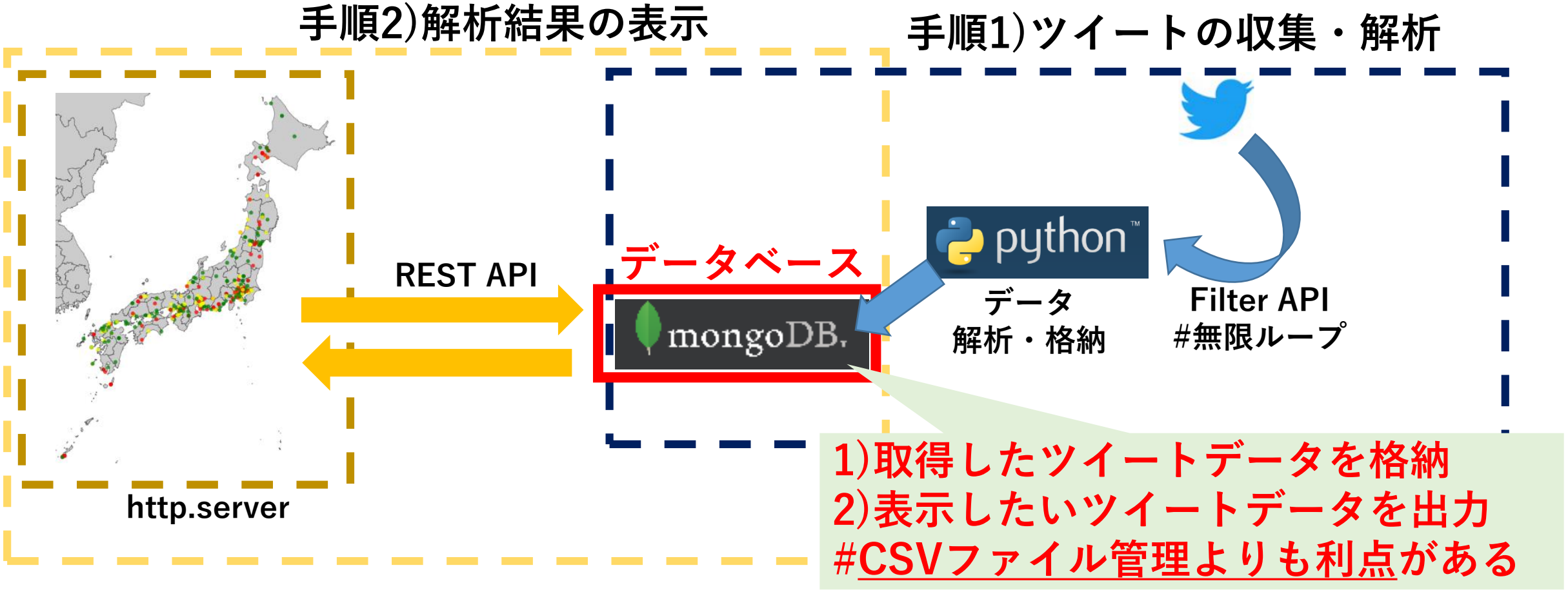
・**<u>データベースを学ぶ意味</u>**

-企業の在庫管理や、金融機関の口座管理など、世の中でデータベースは広く使われている。正規化、冗長化、トランザクション、SQL、問合せ最適化、データベースの格納形式などの構成技術を学ぶことで、データベースを用いたシステムを正しく構築できるようになる。
なお、インターネットやIoT、M2Mなどで大量に生成されるビッグデータを扱うための技術についても取り扱う。

データベースの使用例(Webアプリ)

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般に使われるソフトウェア(ミドルウェア)

データベースの使用例(ツイッター分析)



データベースの使用例:ツイッター分析

「きのこの山たけのこの里国民総選挙」と題して、
2018年にツイッター連動の人気投票が実施された

データベースの使用例:ツイッター分析

ツイートで各党を応援
(#きのこ党, #たけのこ党, #どっちも党)

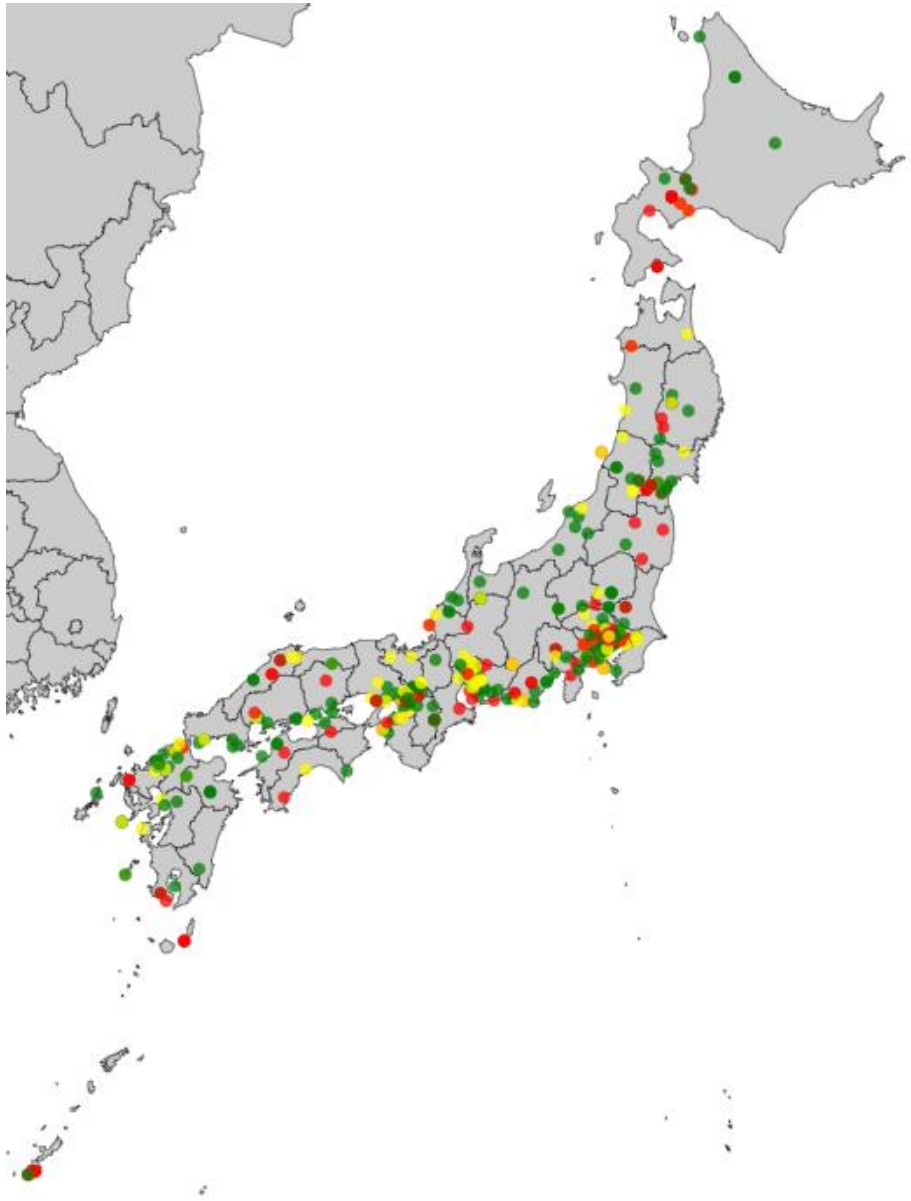
データベースの使用例:ツイッター分析


```
text: "#きのこ党 マニフェスト【きのこの山ワールド　開発！】に投票しました。皆様も清き一票を！ ki-07 https://t.co/6lx9PY..."
source: "<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter fo..."
truncated: false
in_reply_to_status_id: null
in_reply_to_status_id_str: null
in_reply_to_user_id: null
in_reply_to_user_id_str: null
in_reply_to_screen_name: null
> user: Object
geo: null
coordinates: null
v place: Object
    id: "2a16e2975d9264fa"
    url: "https://api.twitter.com/1.1/geo/id/2a16e2975d9264fa.json"
    place_type: "city"
    name: "寝屋川市"
    full_name: "大阪 寝屋川市"
    country_code: "JP"
    country: "日本"
  v bounding_box: Object
      type: "Polygon"
    v coordinates: Array
      v 0: Array
        v 0: Array
            0: 135.586933
            1: 34.728425
```

```
truncated: false
in_reply_to_status_id: null
in_reply_to_status_id_str: null
in_reply_to_user_id: null
in_reply_to_user_id_str: null
in_reply_to_screen_name: null
> user: Object
> geo: Object
v coordinates: Object
    type: "Point"
  v coordinates: Array
      0: 131.9
      1: 43.1333
> place: Object
```

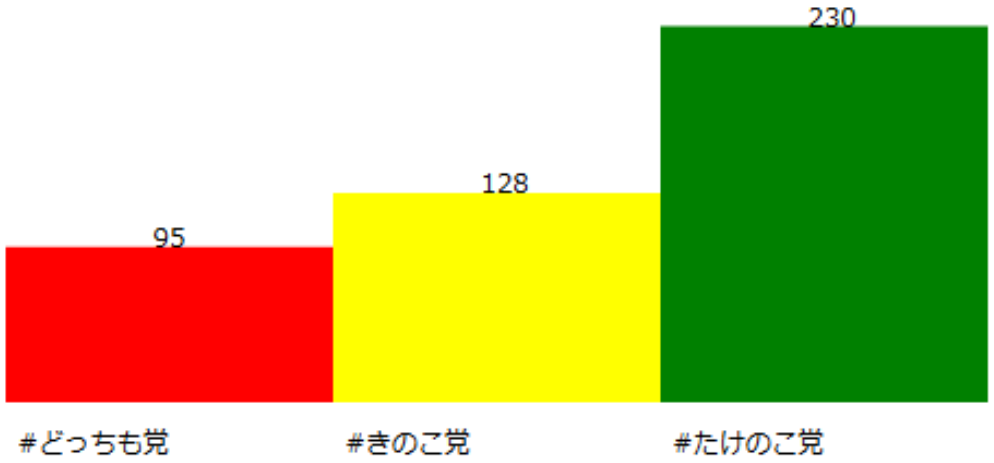
データベースの使用例:ツイッター分析

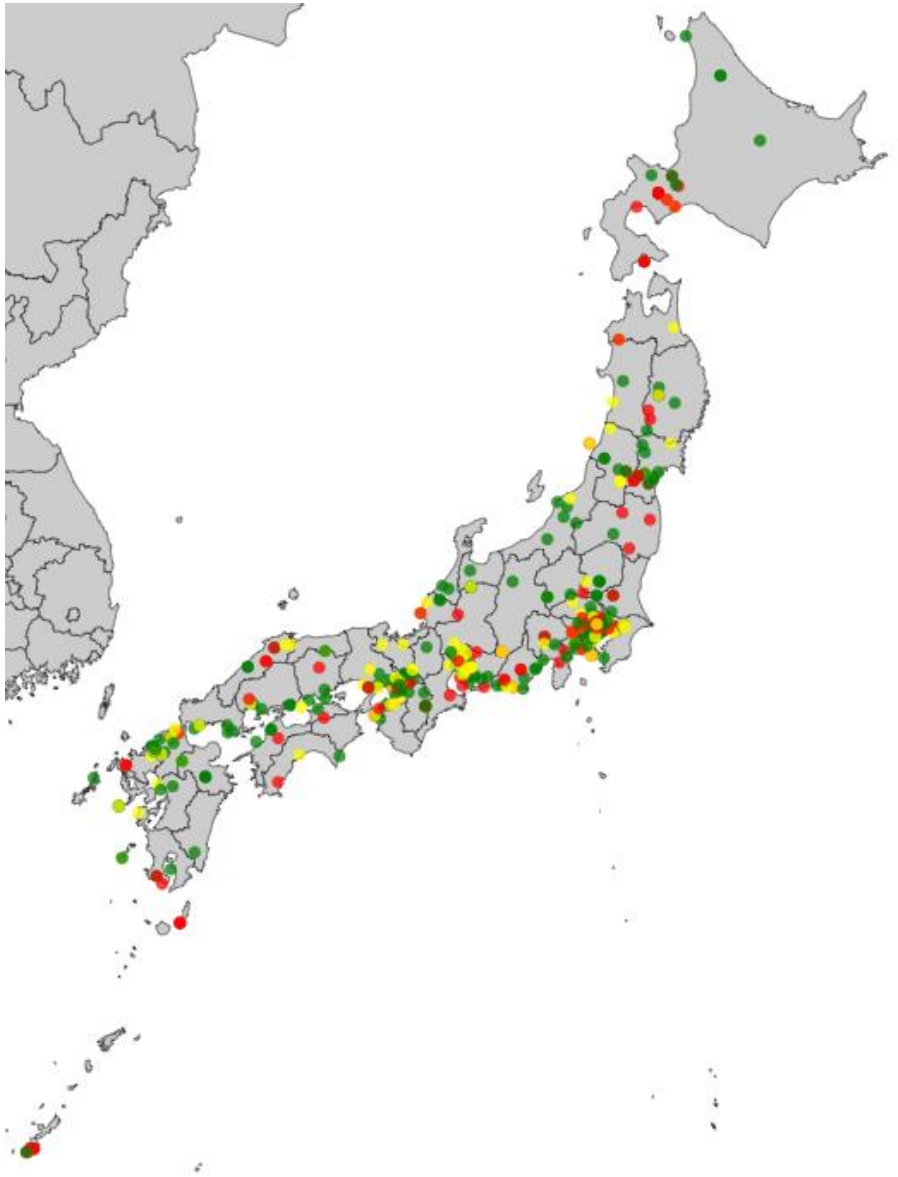
ハッシュタグ解析して、
現状調査(地理分布)



データベースの使用例:ツイッター分析

たけのこ党が優勢！





データベースの使用例(Webアプリ)

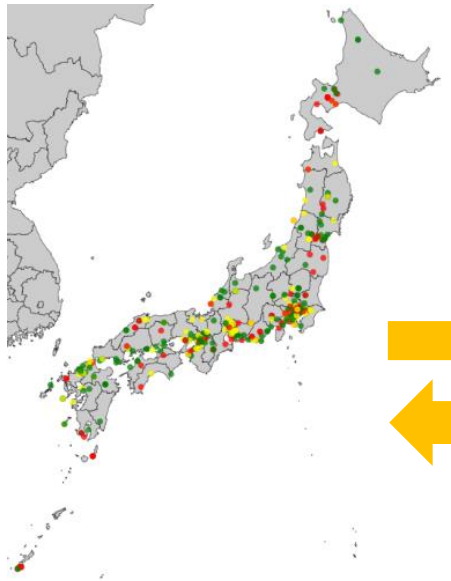
再掲示

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般に使われるソフトウェア(ミドルウェア)

データベースの使用例(ツイッター分析)

手順2)解析結果の表示

手順1)ツイートの収集・解析



http.server

REST API

データベース

mongoDB.

python

データ
解析・格納

Filter API
#無限ループ

1)取得したツイートデータを格納
2)表示したいツイートデータを出力
#CSVファイル管理よりも利点がある

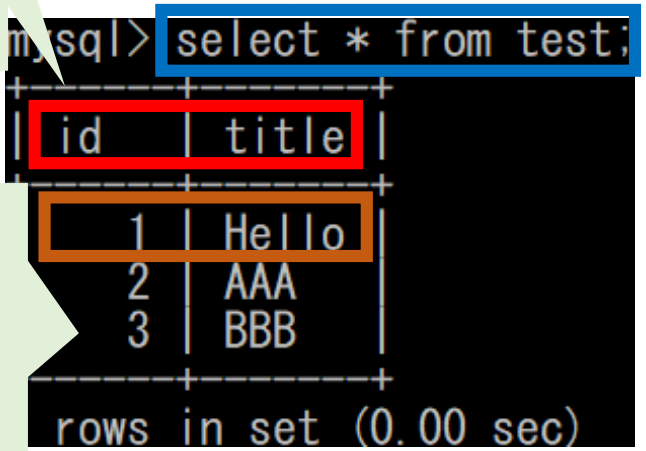
データベースでの表現形式

- データを表形式(縦(列方向)のカラム、横(行方向)のレコード)で表現する。

データベースでは、縦横の表形式でデータを扱う
最上段の横:カラム名(例:id,title)
属性名とも呼ばれ、データ値
(例:1,2)の意味(例:識別番号)を示す

最上段より下の各横一行が、個々のデータ(例:1,Hello)を表す。
データ値は、各カラムの値であり、
idが1で、titleがhello
idが2で、titleがAAAのデータが格納されている

データベースの操作例
#SQL(Structured Query Language)でデータベースを操作(詳細は後程)
select * from test;



データベースの操作画面例

- 表同士を組合わせて、必要なデータを取り出すことができる。

```
mysql> select * from test;
+------+-------+
| id   | title |
+------+-------+
|    1 | Hello |
|    2 | AAA   |
|    3 | BBB   |
+------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select * from test2;
+------+-------+
| id   | value |
+------+-------+
|    1 |    10 |
|    2 |     5 |
|    3 |     1 |
|    3 |     2 |
+------+-------+
4 rows in set (0.00 sec)
```

データベースの操作例
#SQLでデータベースを操作
左:test表とtest2表を表結合
右:test2のvalueが5以上のみ抽出し、表結合

データ値が同じである複数の表を組合わせて、一つの表とできる

```
mysql> select * from test,test2 where test.id=test2.id;
+------+-------+------+-------+
| id   | title | id   | value |
+------+-------+------+-------+
|    1 | Hello |    1 |    10 |
|    2 | AAA   |    2 |     5 |
|    3 | BBB   |    3 |     1 |
|    3 | BBB   |    3 |     2 |
+------+-------+------+-------+
4 rows in set (0.00 sec)
```

```
mysql> select * from test,test2 where test.id=test2.id and test2.value>=5;
+------+-------+------+-------+
| id   | title | id   | value |
+------+-------+------+-------+
|    1 | Hello |    1 |    10 |
|    2 | AAA   |    2 |     5 |
+------+-------+------+-------+
2 rows in set (0.00 sec)
```

全てのデータを一つの表で扱うと、読出し・書込みが非効率
#使わないデータも処理の必要

データベースの目的と役割

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般的にデータベースが使われる。

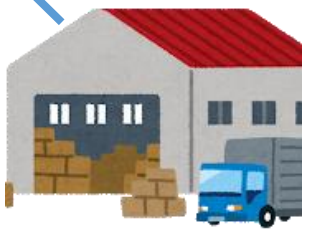
1)データ整合性・耐障害(耐故障),2)共通インタフェース

select * from ～

update table ～

例:売上分析システム

例:Web通販の在庫管理



蓄積された売上データを統一的なインタフェースで取得・分析できる
→2)共通インタフェース
#SQL: select * from table1 where ～

同時に買っても在庫が一致
「在庫が無いのに販売」が起きない
→1)データ整合性・耐障害

データベースの目的と役割

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般的にデータベースが使われる。

1)データ整合性・耐障害(耐故障)

同時に買っても在庫が一致
「在庫が無いのに販売」が起きない
→1)データ整合性・耐障害

受注表

受注ID	顧客名	商品ID	販売個数
201	A	101	1
202	A	102	1
203	B	101	1

在庫表

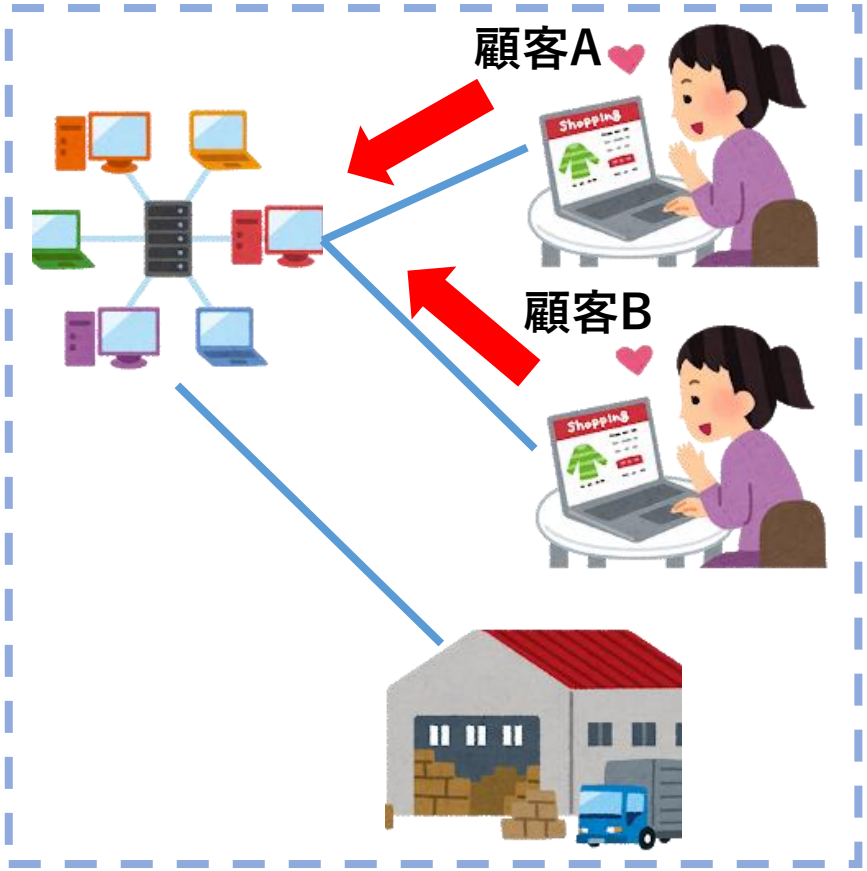
商品ID	商品名	在庫個数
101	コート	3
102	手袋	2
103	靴下	1

②在庫有れば、受注処理

商品ID	商品名	在庫個数
101	コート	2
102	手袋	2
103	靴下	1

受注処理の手順
①在庫表をロックし、他から更新不可へ
②コートの在庫があるのを確認し、1減らし受注処理
③在庫表のロックを解除

例:Web通販の在庫管理



データベースの目的と役割

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般的にデータベースが使われる。

1)データ整合性・耐障害(耐故障)

振込や引き落としが同時に行われても残高は常に一致
処理中に日本中が停電でも、残高が一致
処理が全て行われるか、全く行われないかのどちらか
#振込:「A口座は減って、B口座は増える」の両方実行
→1)データ整合性・耐障害

正常処理

時刻

口座A 残高	口座B 残高
100	80
70	
	110

振込処理は2処理(口座A:減、口座B:増)で構成

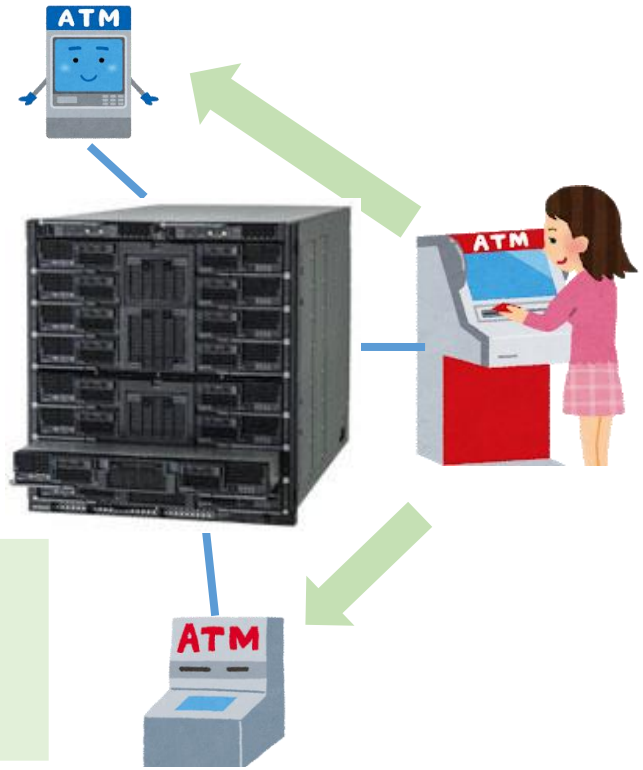
異常処理の例

口座A 残高	口座B 残高
100	80
70	

停電や故障

両方の処理が実行できない場合は、元の状態へ戻す(ロールバック)

例:金融機関ATMでの振込処理



データベースの目的と役割

- 処理開始時にbegin実行することで、begin開始時までデータ状態を巻き戻せる(トランザクション)。

時刻

異常処理の例

口座A 残高	口座B 残高
100	80
70	

停電や故障

両方の処理が実行できない場合は、元の状態へ戻す(ロールバック)

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from test;
+------+-------+
| id   | title |
+------+-------+
|    1 | Hello |
|    2 | AAA   |
|    3 | BBB   |
+------+-------+
```

```
mysql> update test set title = 'CCC' where id = 3;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from test;
+------+-------+
| id   | title |
+------+-------+
|    1 | Hello |
|    2 | AAA   |
|    3 | CCC   |
+------+-------+
```

「idカラムの値」が3のレコードの「titleカラムの値」をCCCに更新

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from test;
+------+-------+
| id   | title |
+------+-------+
|    1 | Hello |
|    2 | AAA   |
|    3 | BBB   |
+------+-------+
```

ROLLBACKでbegin処理開始時に戻る

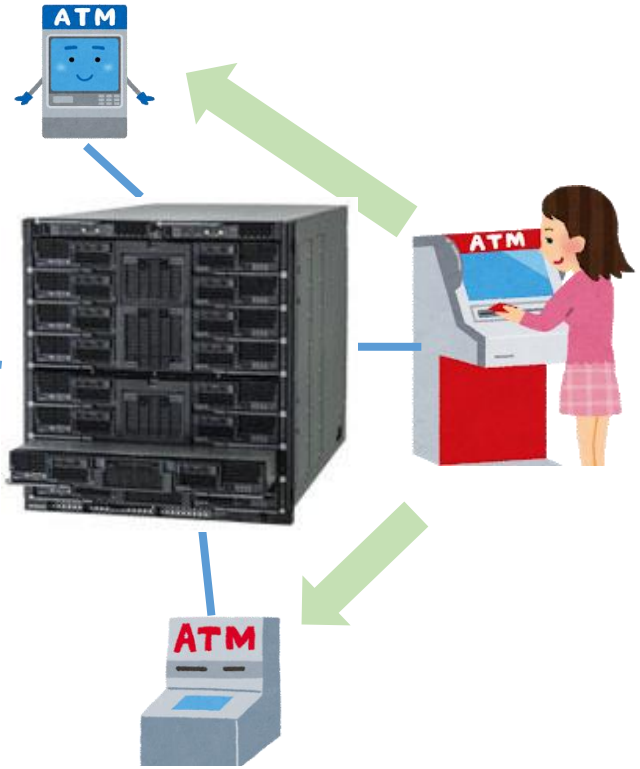
データベースの目的と役割

- (大量の)データを整合性を確保しつつ、(同一or異なる)多数のプログラムから操作する場合に、一般的にデータベースが使われる。

3)冗長化・並列化

データベースを使うと、過去に広く使われた実績のある3)冗長化・並列化の仕組みが手軽に使える
自分で作るよりも簡単で確実

ディスクアレイ(SSD・HDDの集合体)やブレードサーバ(サーバの集合体)が金融機関など大規模システムでは使われる

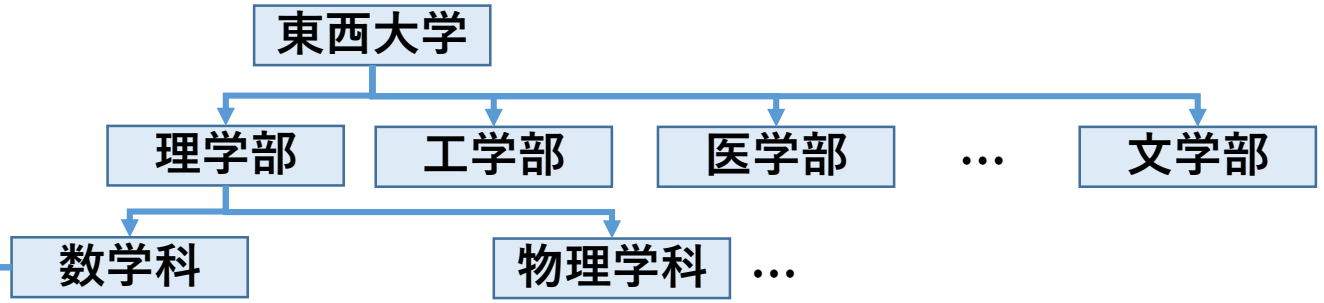


データモデルとデータベースの種類1

- 階層型データベース(DB):データ間の親子関係から、木構造を構成

1対Nの親子関係により整理し、親ノードから子ノードを辿ってデータアクセスする。N対Mの親子関係が有る場合や、階層的な関係が定義できない場合にうまく扱えない。

203	数学太郎	2020	男
204	数学花子	2020	女
⋮			



- リレーショナルDB:表形式に表現されるリレーションでデータ管理

現在、最も広く利用されており、Oracle,MySQL,SQL Serverなどがある。SQLはリレーショナルデータベースのための言語。

受注表

受注ID	顧客名	商品ID	販売個数
201	A	101	1
202	A	102	1
203	B	101	1

在庫表

商品ID	商品名	在庫個数
101	コート	3
102	手袋	2
103	靴下	1

データモデルとデータベースの種類2

- 半構造データベース:格納するテーブル定義を事前規定せず、保存するデータに応じてデータ構造を毎回定義し、保存する。

→〇事前にデータ構造を定義するのが難しいデータの管理を容易化する

×データ毎にテーブル定義情報が必要で、読込み等の処理が低速の傾向

```
<?xml version=“1.0” encording=“utf-8”?>
<students>
 <student><student_ID>B123<¥student_ID><name>DB太郎<¥name> …<¥student>
…
<¥student>
```

student:学生レコード,カラム名がstudent_ID,nameなど

データモデルとデータベースの種類3

- NoSQLデータベース:SQLを使わないデータベースの総称

例えば、Webゲームでのユーザログイン処理では、ユーザ名と対応するパスワードを高速に取得できる必要がある。

→このような場合、キーバリューストアと呼ばれるキー(例:ユーザ名)に対応するバリュー(例:パスワード)が高速に取得できるデータベースの一種が使用されることが有る。

#SQL処理(表結合、条件によるレコード抽出など)は不要のため。

NoSQLの詳細は第十四回などで扱う。

データベースの歴史1

- データとプログラムの分離

1946年に、プログラムを主記憶に格納・実行することで汎用的なデータ処理が可能なストアードプログラム方式のコンピュータが開発された。初期のコンピュータでは、プログラムとデータが明確に分けられておらず、同様のデータであってもプログラム毎に個別にデータを用意する必要があり、その管理が煩雑という問題があった。

- 多数のデータに効率的にアクセスできるようにする仕組み

1968年に、データを階層的に管理するデータベースが開発された(IBM)。

宇宙船開発(アポロ計画)での膨大な(数十万件の)部品データを管理するため

データベースの歴史2

- リレーショナルデータベース

階層型DBでは、物理的なシステムとデータとの独立性が低く、汎用的なデータベースとは言えない側面があった。1970年に、コッド博士が提案したリレーショナルモデルに基づいたデータベースでは、数学的に定義することで物理的なデータの記憶方法からの独立性を高めている。

#レコードの物理的な並び順などは、リレーショナルでは原則考慮不要

- ＮｏＳＱＬの登場

インターネットやIoT、M2Mの普及に伴い、世の中で生成されるデータが増大した。大量かつ次々と生み出されるようなデータを「ビッグデータ」と呼ぶが、このような大量データを既存のデータベース技術で扱うことが困難である。そこで、従来のSQLを利用するリレーショナルモデルにとらわれず、スケーラビリティに富む技術が提案されている。これらのSQLを利用しないデータベースの総称をNoSQLデータベースと呼ぶ。

課題1-1 締切(10/19)

- データベースが使われている事例を2つ探して、それぞれの概要を100文字程度で説明してください。

リレーションと集合

・関係データベースでは「集合の一種であるリレーション」と「リレーション上の演算であるリレーション代数」が使われる。

集合演算のようなもの

表とほとんど同じ意味
次スライドのタプルなどの概念があるのが一般的な集合と異なる

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、ドメイン(例:料理名ドメインはチャーハン、蟹チャーハン等)という用語が使われる。

タプルとリレーション

・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。

・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、各ドメインから1つずつ選んだ(属性の)値の組(d1,…,dn)をタプル、その部分集合をリレーションと呼ぶ。

例:(d1,d2)だと(天津飯,900)。
d1はチャーハンなどが入る

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

SQLの概要

リレーショナルデータベースを利用するためには、一般に、リレーショナルデータベースを操作する標準言語であるSQLと呼ばれる言語を使う。
#他には、C言語からのカーソル操作→検索結果を順に取得できる仕組み

1)SQL(Structured Query Languageの略or何らかの頭文字ではない)
2)SQLで、データベースにデータを挿入(登録)が行える
3)SQLで、データベースから目的のデータを参照(取り出し)が行える
4)SQLで、データを挿入するための枠組み(表定義)が定められる

コッド博士が在籍していたIBM社の関係データベースSystemRで使われていた言語(SEQUEL:Structured English Query Language 構文化英文問合せ言語)がもとになっている。RDBMS向けにいくつか言語が開発されていたが、標準化された。現在でも、仕様追加がなされている。

SQLの仕様

リレーショナルデータモデルにはリレーション、タプル、属性と呼ばれた概念があった。一方、同様の概念をSQLではテーブル(表)、行、列と呼ぶ。

・SQLでは、テーブルの行と列には順序がある。
#リレーションと同様、基本的にはテーブルの行の順序には特別な意味は無いが、検索結果として値の小さい順などに並び変えたテーブルでは行の順序には意味がある。
・SQLでは、英文のようにデータベースに対する操作命令を表現する。
・あらかじめ決められた単語(select, from, whereなど)が200語以上あり、予約語と呼ぶ。予約語以外の単語や文字列はテーブル名や列名に使える。

```
insert into item values('A01','オフィス用紙A4',2000);
insert into item(item_id, item_name) values('A03', "オフィス用紙A4");

select * from item;
select * from item where item_id ='A01';
select * from item where price >1000;
```

基本SQL:テーブルとテーブル定義1

・テーブルはリレーションに相当する(列は属性に、行はタプルに対応)。
テーブル定義では「データの格納型の種類や数、整合性制約」を定める。

・データベースに実際に作成したテーブルは実テーブルと呼び、実テーブルから関係演算などにより作り出されたテーブルは導出テーブルと呼ぶ。
・下記は、商品テーブルの例である。「item」はテーブル名であり、商品番号(item_id)、商品名(item_name)、価格の列(price)をもち、5行分のデータが格納されている。

商品テーブル

商品番号	商品名	価格
A01	オフィス用紙A4	2000
A02	オフィス用紙A3	4000
A03	オフィス用紙B5	1500
B01	トナーカートリッジ黒	25000
C01	ホワイトボード	14000

```
create table item(item_id char(3)not null,
                  item_name varchar(20),
                  price int,primary key (item_id));
```

基本SQL:データ挿入(格納)

・create table文により作成されたテーブルにはデータを挿入されるまでは1行も中身は入っていない。データを挿入するにはinsert文を使う。insert文の構文は下記である。

[ ]は、オプションであり、
指定しなくても文が実行可能

insert into テーブル名 [(列名1,…,列名m)] values (値1,…,値m)

item_id,item_name,priceにそれぞれ対応

```
insert into item values('A01','オフィス用紙A4',2000);
insert into item(item_id, item_name) values('A03', "オフィス用紙A4");
```

priceはnullとしてデータ格納

item_id	item_name	price
A01	オフィス用紙A4	2000
A03	オフィス用紙A4	NULL

基本SQL:データ参照

・データベースに登録されているデータはselect文により参照することができる。このselect文では、いろいろな条件を与えて条件に一致するデータを検索することができるため、select文によるデータ参照のことを問合せ(クエリ,query)と呼ぶ。
#データ更新など、SQLによるデータ操作のための要求全般を問合せと呼ぶことも有る。

[ ]は、オプションであり、
指定しなくても文が実行可能

select 列名1,…,列名m from テーブル名 [where 条件]
select * from テーブル名 [where 条件]

*は、列名全てと等価

```
insert into item values('A01','オフィス用紙A4',2000);
insert into item(item_id, item_name) values('A03', "オフィス用紙A4");

select * from item;
select * from item where item_id ='A01';
select * from item where price >1000;
```

item_id=‘A01’

```
item_id item_name       price
A01     オフィス用紙A4   2000
A03     オフィス用紙A4   NULL
item_id item_name       price
A01     オフィス用紙A4   2000
item_id item_name       price
A01     オフィス用紙A4   2000
```

price >1000

課題1-2 締切:10/19

身近な事柄を参考にして、簡単なテーブルを考え、テーブルを作成するためのcreate文を記載せよ。create文とinsert文を実行した後に、参照SQLを実行してください。実行後のスクショを提出ください。

https://paiza.io/ja/projects/newで簡易なMySQLが実行可能である。

(第二回時間あれば)リレーションと整合性制約

・現実世界のデータをデータベースに収める際に、データベース自身の整合性(データが矛盾しない)を保つためには、データやデータ間に存在している元々の条件や制約を、データベースにおいても再現するのが望ましい。

例えば、個人を識別している学生の学籍番号は各学校において重複することはない。電話番号も異なる契約に同じ番号が割り振られることは無い。
生年月日の数字は、月ならば1から12までの整数である。
→このような制約を満たさないデータをデータベースの格納時にはじく事で、整合性がとれたデータのみを保存することができる。

データベースに課されるこのような制約のことを**整合性制約**と呼ぶ。

リレーションと整合性制約(キー制約)

ある大学で使用されている下記のリレーションを考える。属性「学籍番号」では、重複した属性値(学籍番号の値)は存在しないはずである。一般に、リレーションでは空値にならない候補キーを1つ選択し、主キーと呼ぶ。

空値:一部の属性値が未決で、とりあえず空欄にしたい場合、空値(null value)とする。
スーパーキー:あるリレーションにおける任意のリレーションに対し「タプルを一意的に特定できる属性や属性の集合」を、スーパーキーという。

主キーを構成する属性名の下にアンダーラインをひく

学籍番号が決まると、氏名や所属学部なども自動的にわかる

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm

上記でのスーパーキー:{学籍番号}、{学籍番号,所属学部}、{学籍番号,氏名}、{学籍番号,電話番号}、{学籍番号,所属学部,氏名}、{学籍番号,所属学部,電話番号}、{学籍番号,氏名,電話番号}、{学籍番号,所属学部,氏名,電話番号}の8つ。

リレーションと整合性制約(キー制約)2

ある大学で使用されている下記のリレーションを考える。属性「学籍番号」では、重複した属性値(学籍番号の値)は存在しないはずである。一般に、リレーションでは空値にならない候補キーを1つ選択し、主キーと呼ぶ。

空値:一部の属性値が未決で、とりあえず空欄にしたい場合、空値(null value)とする。
スーパーキー:あるリレーションにおける任意のリレーションに対し「タプルを一意的に特定できる属性や属性の集合」を、スーパーキーという。
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を、候補キー、あるいは単にキーと呼ぶ。

主キーを構成する属性名の下にアンダーラインをひく

学籍番号が決まると、氏名や所属学部なども自動的にわかる

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm

上記でのスーパーキー:{学籍番号}、{学籍番号,所属学部}、{学籍番号,氏名}、{学籍番号,電話番号}、{学籍番号,所属学部,氏名}、{学籍番号,所属学部,電話番号}、{学籍番号,氏名,電話番号}、{学籍番号,所属学部,氏名,電話番号}の8つ。
上記での候補キー:{学籍番号}のみ

リレーションと整合性制約(参照整合性制約)

一方のリレーションの属性値が他方のリレーションの属性値を参照している事が有る。その場合、片方の属性値が対応する主キーの属性値のいずれかの値になっていなくてはならないという制約(参照整合性制約)がある。

外部キー:他リレーションの主キーにある属性値をとる属性(空値もとる)
#外部キーは他リレーションの属性の集合からなる主キーの場合もある

「所属学部」の属性値は「学部コード」の属性値を参照している。
→「学部コード」に無い値を「所属学部」の属性値にするのは不可

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm

学部コード	学部名	学部長	住所
01	文学部	渡邉一郎	東京都千代田区丸の内A
02	理工学部	西園公平	東京都千代田区丸の内B
03	情報工学部	中田元	東京都千代田区丸の内C

リレーションと整合性制約(参照整合性制約)

データベースでは、なんらかの事象(例:学生情報)を複数の表(例:学生表、学部表)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。
#詳細は、また後で

「所属学部」の属性値は「学部コード」の属性値を参照している。
→「学部コード」に無い値を「所属学部」の属性値にするのは不可

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm

学部コード	学部名	学部長	住所
01	文学部	渡邉一郎	東京都千代田区丸の内A
02	理工学部	西園公平	東京都千代田区丸の内B
03	情報工学部	中田元	東京都千代田区丸の内C

リレーショナルデータモデルと第1正規形

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

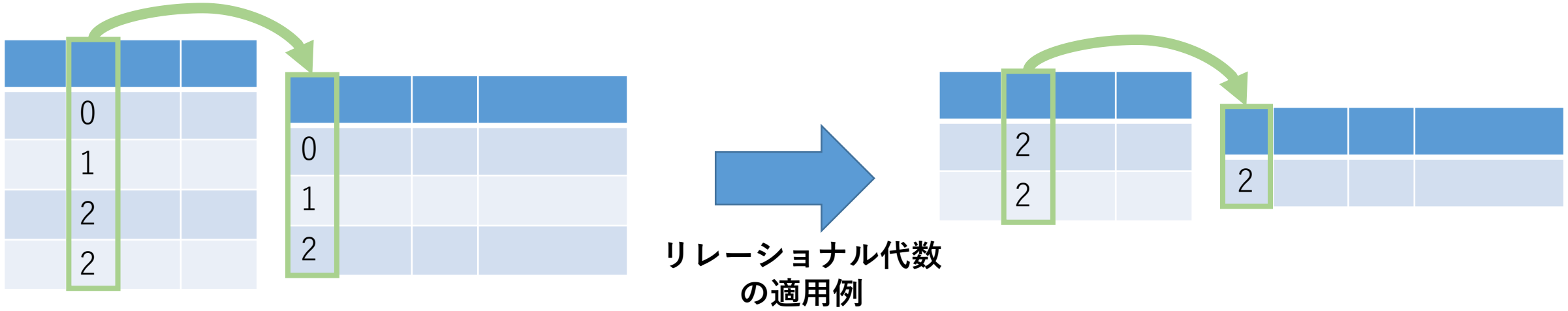
非第一正規形

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

リレーショナル代数

・リレーショナルデータベースでは、データモデルはリレーションの集合として表現される。複数のリレーションを外部キーと主キーによって関連付けることで、データベース内のデータ間の関係を管理する。
・ユーザがデータベースから目的のデータを取得するためには、格納されているリレーションから、目的とするデータを含む新たなリレーションを生成して、取得する。→この操作のための演算体系をリレーショナル代数という。



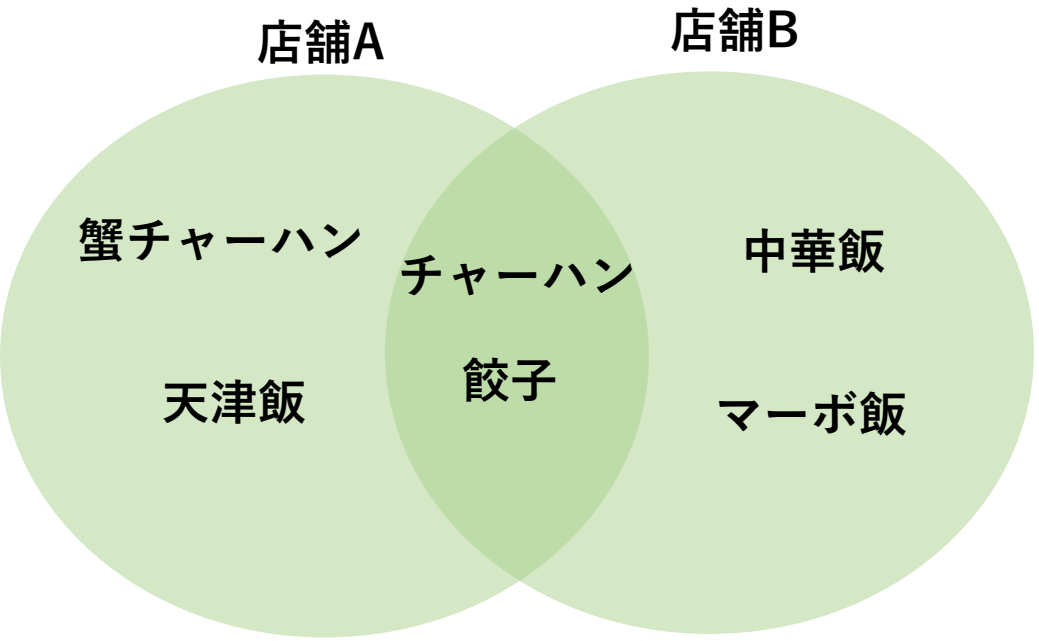
集合演算

・リレーショナル代数の演算は2種類(**集合演算**・関係演算)に分類できる。
・集合論で定義された集合演算には、和集合演算、差集合演算、共通集合演算、直積演算がある。
a)和集合、共通集合、差集合は「属性が一致しているリレーション間」(和両立が成り立つリレーション間)のみに適用できる。
b)直積演算は、リレーション間の属性が一致していなくても適用できる。

和両立:2つのリレーションR,Sが有る場合に、RとSの属性が一致することを表す概念。以下のように定義される。
リレーションR($A_1,A_2,\cdots,A_n$)とS($B_1,B_2,\cdots,B_n$)が次の2条件を満たすとき、リレーションRとSは和両立である。
1)RとSの次数(属性の数)が等しい(つまり、n=m)
2)各i($1 \leqq i \leqq n$)に対し、A_iとB_iのドメイン(属性の取り得る範囲)が等しい

集合演算(和集合演算)

・和両立である2つのリレーションRとSに対して、RかSのどちらかに含まれるタプルからなる新たなリレーションを生成する演算を和集合演算と呼ぶ。その演算結果として得られるリレーションを和あるいは和集合と呼びRUSで表す。



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

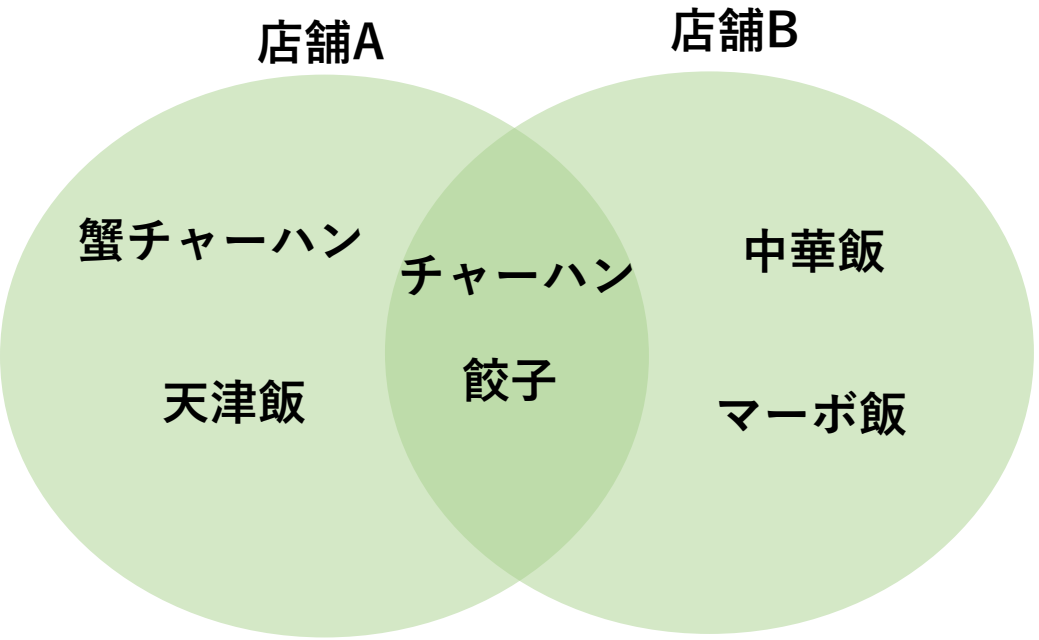
料理名	値段(円)
チャーハン	800
中華飯	750
マーボ飯	900
餃子	500

どちらかの店舗で扱っているメニュー

料理名	値段(円)
チャーハン	800
蟹チャーハン	1300
天津飯	900
中華飯	750
マーボ飯	900
餃子	500

集合演算(共通集合演算)

・和両立である2つのリレーションRとSに対して、RとSの両方に共通するタプルからなるリレーションを生成する演算を共通集合演算と呼ぶ。また、その演算結果として得られるリレーションを共通あるいは積集合と呼び、R∩Sで表す。



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

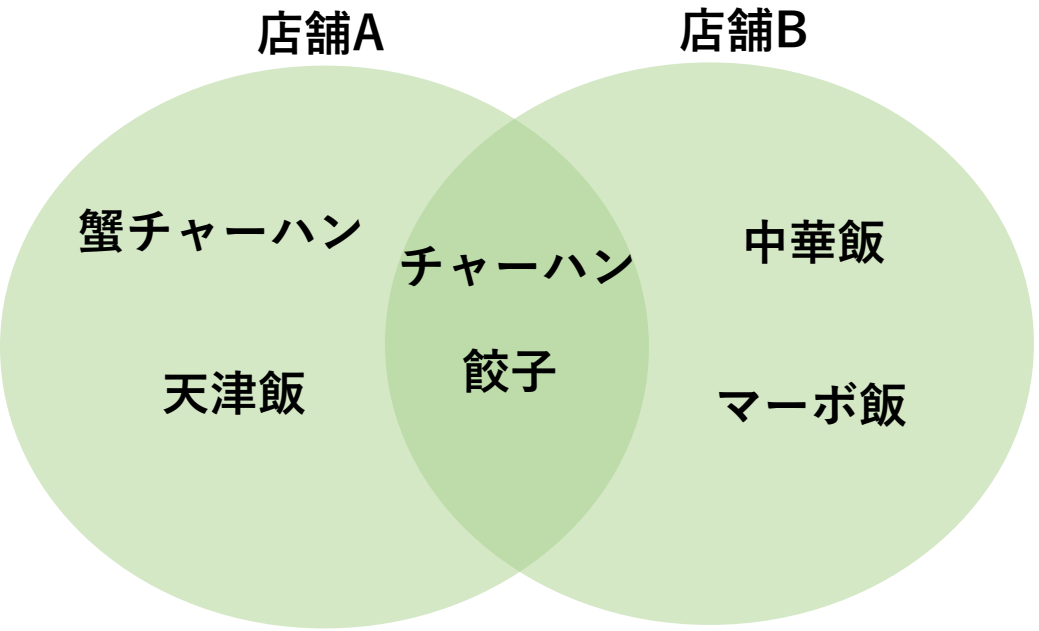
料理名	値段(円)
チャーハン	800
中華飯	750
マーボ飯	900
餃子	500

両方の店舗で扱っているメニュー

料理名	値段(円)
チャーハン	800
餃子	500

集合演算(差集合演算)

・和両立である2つのリレーションRとSに対し、片方のリレーションのみに存在するタプルからなるリレーションを求める演算を差集合演算と呼ぶ。またその演算結果として得られるリレーションを差あるいは差集合と呼び、R-S(Rだけに存在するタプルのリレーション)およびS-R(Sだけに存在するタプルのリレーション)と表す。



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

Ra-Rb
店舗Raのみで扱っているメニュー

料理名	値段(円)
蟹チャーハン	1300
天津飯	900

Rb-Ra
店舗Rbのみで扱っているメニュー

料理名	値段(円)
中華飯	1300
マーボ飯	900

集合演算(直積演算)

・2つの異なるリレーションRとSに対して、すべてのタプルの組合せを求める演算を直積(あるいはデカルト積)と呼び、その演算結果をR×Sで表す。

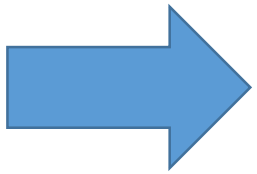
和両立でなくとも、直積演算は適用可能

販売店Ra

販売店	責任者
東京	太田
大阪	大津
福岡	小川

仕入れ商品Rb

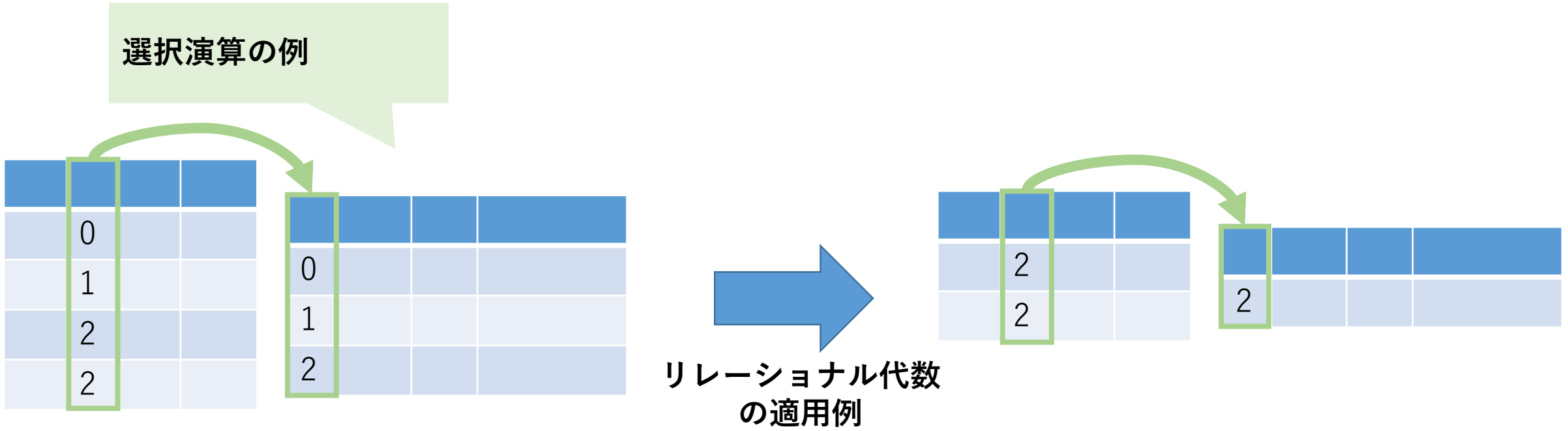
仕入製品	製造工場
TV01	花巻
TV02	熊本
PV02	静岡



仕入製品	製造工場	販売店	責任者
TV01	花巻	東京	太田
TV01	花巻	大阪	大津
TV01	花巻	福岡	小川
TV02	熊本	東京	太田
TV02	熊本	大阪	大津
TV02	熊本	福岡	小川
PV02	静岡	東京	太田
PV02	静岡	大阪	大津
PV02	静岡	福岡	小川

関係演算

・リレーショナル代数の演算は2種類(集合演算・**関係演算**)に分類できる。
・リレーショナルデータモデルに固有の主な関係演算としては、選択演算、射影演算、結合演算が有る。



関係演算(選択演算)

・リレーションRの中から、条件式Fを満たすタプルを抽出する演算を選択(selection)と呼び、$\sigma_F(R)$で表す。条件式Fとしては、Rの属性A_i,A_jや定数値cに対して、比較演算子θ(=,≠,<,>,≧,≦)を用いて、$A_i \theta A_j$(例:$A_i=A_j$)や$A_i \theta c$(例:$A_i=5$)といった比較演算式を指定する。

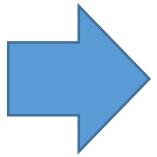
・比較演算式で比較される2つの値は比較可能(ドメインが等しく、比較結果の真偽が常に定まる)でなければならない。

θ比較可能:2つの値が比較可能であること

・条件式Fとして、比較演算を論理積(∧)や論理和(∨)、否定(¬)で組合わせたものを指定できる。

学生一覧　Ra

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm



学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde

$\sigma_{所属学部=01}(Ra)$

関係演算(射影演算)

・リレーションRから、指定した属性集合βのみを持つリレーションを抽出する演算を射影(projection)と呼び、$\pi_{\beta}(R)$で表す。βはRの属性の部分集合である。射影はβで指定された属性集合以外を取り除き、テーブルを縦方向に切り出す演算である。

学生一覧　Ra

学籍番号	所属学部	氏名	電話番号
130101	01	高橋一郎	03-1234-xyza
130102	01	鈴木花子	03-5678-bcde
120201	02	山田太郎	03-9012-fghi
120202	02	小鳥遊次郎	03-3456-jklm

$\pi_{学籍番号,所属学部}(Ra)$

学籍番号	所属学部
130101	01
130102	01

関係演算(結合演算)

・結合(join)とは、2つのリレーションRとSを条件式Fに従って1つのリレーションに統合する演算であり、その演算結果を$R \bowtie_F S$で表す。条件式Fは選択演算の場合と同様の条件式を指定する。その結果得られるリレーションは、RとSの直積R×Sに対して条件式Fを満足するタプルのみを残す選択演算を行った結果と等しい。

・どちらか一方のリレーションのみに存在するタプルを演算結果に含めないことから、結合のことを内部結合(inner join)とも呼ぶ。

学籍番号	学部番号	氏名
130101	01	高橋一郎
120201	02	山田太郎

学部番号	学部名
01	文学部
02	理工学部
03	情報工学部

外部キーの属性間で結合する場合が多いですが、ドメインがあっていれば外部キーでなくても結合可能

学籍番号	学部番号	氏名	学部番号	学部名
130101	01	高橋一郎	01	文学部
120201	02	山田太郎	02	理工学部

「学部コード」が03のタプルは、両方のリレーションに無いから結果に含まれない

関係演算(自然結合演算)

・等結合(「2つのリレーションの対応する属性の値が等しい結合条件」を用いる結合)で得られるリレーションでは、結合条件式に用いた属性が重複する。等結合の結果から重複する属性を射影で取り除いた結果を求める結合演算を、自然結合と呼ぶ。

学籍番号	学部番号	氏名
130101	01	高橋一郎
120201	02	山田太郎

学部番号	学部名
01	文学部
02	理工学部
03	情報工学部

結合演算

学籍番号	学部番号	氏名	学部番号	学部名
130101	01	高橋一郎	01	文学部
120201	02	山田太郎	02	理工学部

自然結合演算

学籍番号	学部番号	氏名	学部名
130101	01	高橋一郎	文学部
120201	02	山田太郎	理工学部

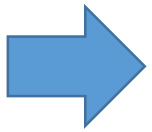
関係演算(外部結合演算)

・結合演算(内部結合)の一種である等結合では、一方のリレーションRのタプルのうちで、もう一方のリレーションSに対応するタプルがないものは、演算結果であるリレーションに含まれない。これに対し、そのような場合でもRのタプルを演算結果に含め、対応するSの属性値を”null”とする等結合を外部結合(outer join)と呼ぶ。”null”は、その値が空値であることを表す。

・外部結合は、2つのリレーションを残すかにより、左外部結合、右外部結合に分類できる。両方のリレーションのタプルを全て残す完全外部結合もある。

学籍番号	学部番号	氏名
130101	01	高橋一郎
120201	02	山田太郎

学部番号	学部名
01	文学部
02	理工学部
03	情報工学部



右外部結合演算

学籍番号	学部番号	氏名	学部番号	学部名
130101	01	高橋一郎	01	文学部
120201	02	山田太郎	02	理工学部
null	null	null	03	情報工学部

課題2-1　締切:10/26

2-1)下記における、スーパーキー、候補キーをすべてあげよ
さらに、下記において、属性値間の外部キーの関係にあるものを1つあげよ

商品(商品ID,商品名,単価)
顧客(顧客ID,顧客名)
売上集計(顧客ID,商品ID,売上合計)

主キーを構成する属性名の下にアンダーラインをひく

2-2)下記に示すリレーションRaとRbに対して、和集合と共通集合を求めよ。

A店舗取り扱い商品リストRa

商品番号	商品名	値段
R001	プリンタ1	49,800
C001	パソコン1	148,000
H002	HDD2	9,800
M001	Camera1	38,800
M005	Camera5	49,800

B店舗取り扱い商品リストRb

商品番号	商品名	値段
R001	プリンタ1	49,800
C003	パソコン3	188,000
H001	HDD1	39,800
M002	Camera2	58,800
M005	Camera5	49,800