


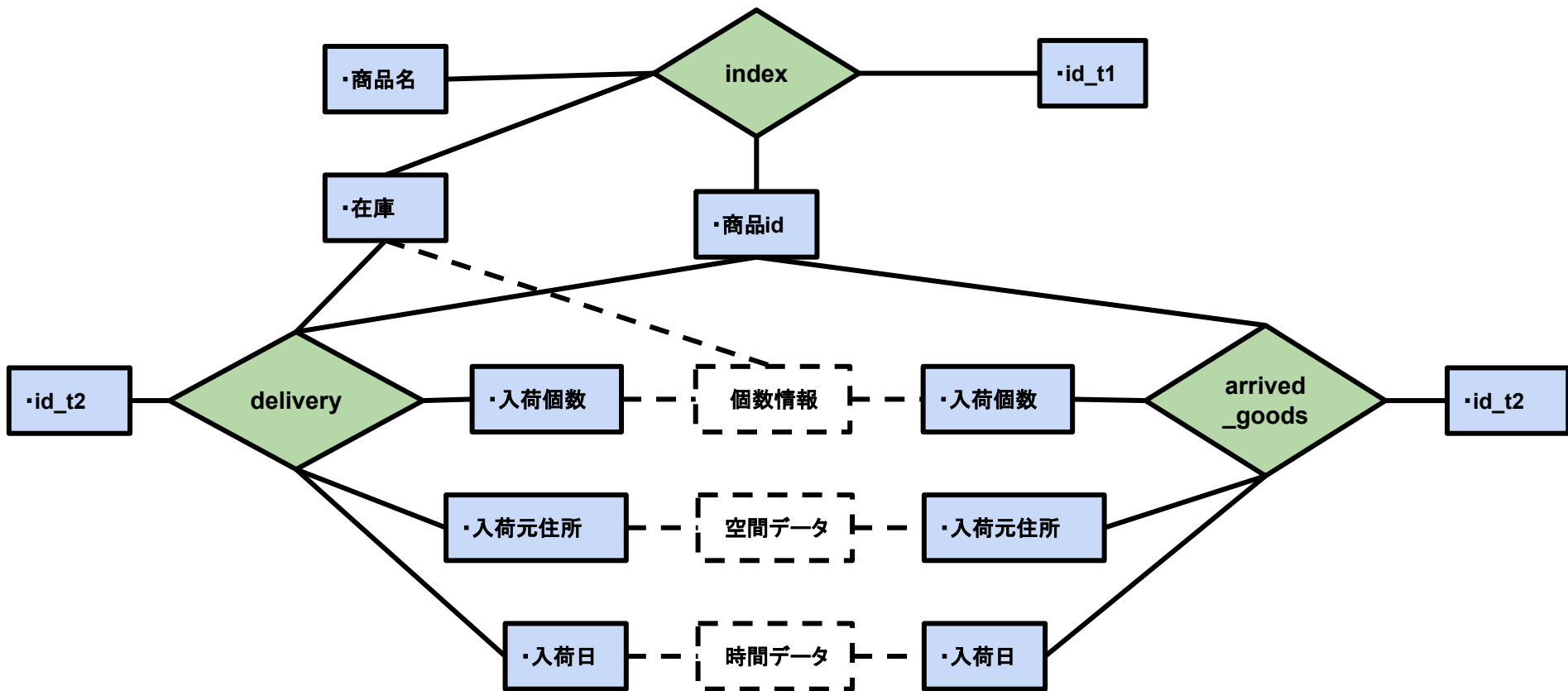
個人中間 レポート

1922074 木村太紀



1. データベース構築の目的とER図

今回は架空の商店における商品取引台帳を3つのテーブルにて再現した。



1_2.DB構築SQL及びにCSV(Github)

Table1:「index.sql」、「index.csv」

▪[index.sql](#)

▪[index.csv](#)

Table2:「arrived_goods.sql」、「arrived_goods.csv」

▪[arrived_goods.sql](#)

▪[arrived_goods.csv](#)

Table3:「delivery.sql」、「delivery.csv」

▪[delivery.sql](#)

▪[delivery.csv](#)

2_1.作成テーブル概要(Table1、以下全てUTF-8)

Table1:「index」(30 rows)

商品の基本情報をまとめたテーブル。

- ・id_t1: INTEGER NOT NULL PRIMARY KEY
- ・商品名: VARCHAR(6) NOT NULL
- ・商品id: VARCHAR(3) NOT NULL
- ・在庫: INTEGER NOT NULL

2_2.作成テーブル概要(Table2)

Table2:「arrived_goods」(30 rows)

商品の入荷情報をまとめたテーブル。

空間データ:入荷元住所、時間データ:入荷日

- ・id_t2: INTEGER NOT NULL PRIMARY KEY
- ・商品id: VARCHAR(3) NOT NULL
- ・入荷個数: INTEGER NOT NULL
- ・入荷元住所: VARCHAR(34) NOT NULL
- ・入荷日: VARCHAR(5) NOT NULL

2_3.作成テーブル概要(Table3)

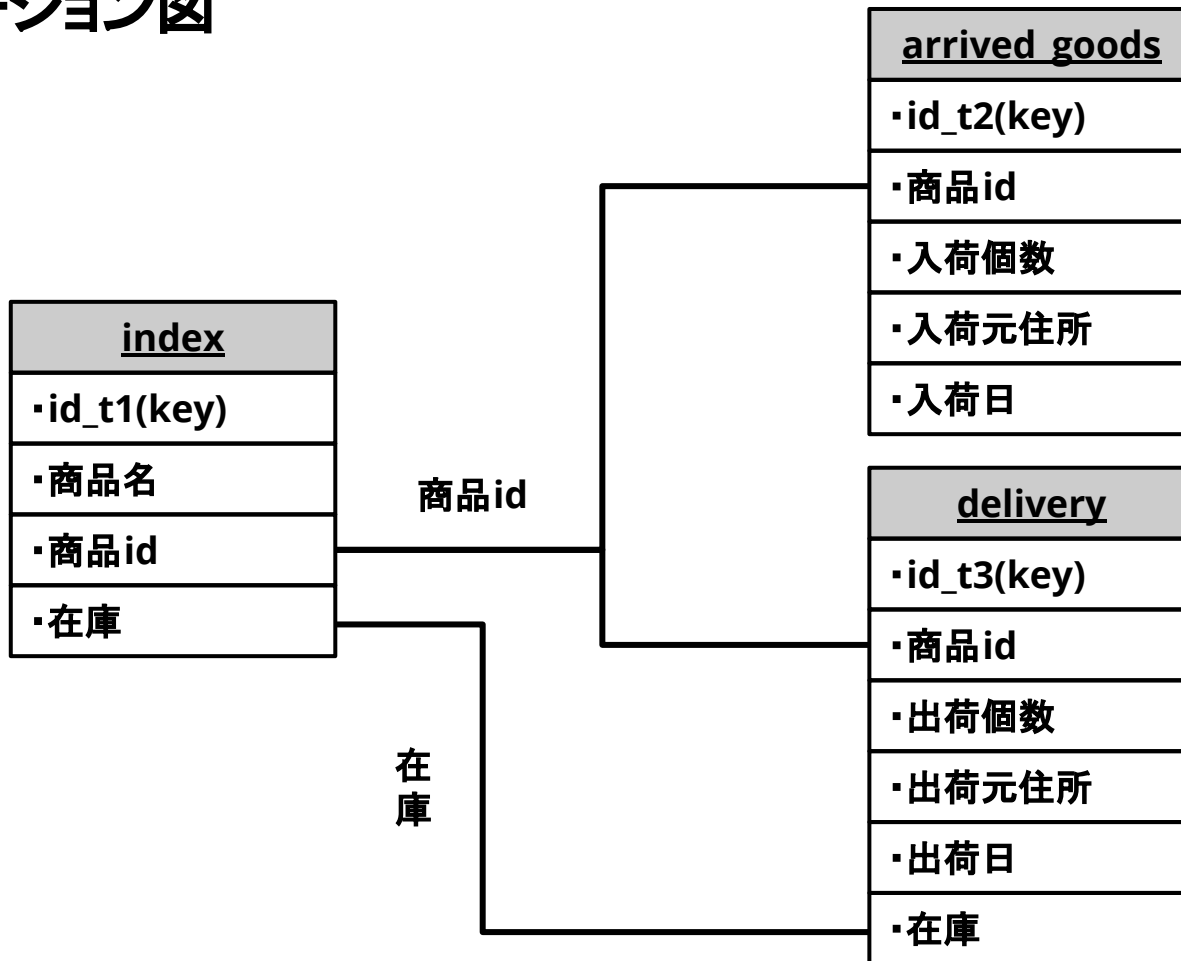
Table3:「delivery」(30 rows)

商品の出荷情報をまとめたテーブル。

空間データ:出荷先住所、時間データ:出荷日

- ・id_t3:INTEGER NOT NULL PRIMARY KEY
- ・商品id:VARCHAR(3) NOT NULL
- ・出荷個数:INTEGER NOT NULL
- ・出荷元住所:VARCHAR(23) NOT NULL
- ・出荷日:VARCHAR(5) NOT NULL
- ・在庫:INTEGER NOT NULL

3.リレーション図



4_1.実行クエリとその結果1(時間データ)

「時間データ取得クエリ」

```
SELECT index.商品id,index.商品  
名,arrived_goods.入荷日,delivery.出荷日  
FROM  
index  
JOIN  
arrived_goods  
ON  
index.商品id=arrived_goods.商品id  
RIGHT JOIN  
delivery  
ON  
index.商品id=delivery.商品id  
;
```

・[時間データ取得クエリ\(Github\)](#)

商品id	商品名	入荷日	出荷日
1	ルコソコラン	10/05	11/04
10	フォセイ	04/22	03/15
11	コロイト	10/26	3/9
12	ココロイト	06/17	04/29
13	コマダイト	10/07	06/21
14	パイライト	01/26	07/04
15	フガナイ	06/07	05/02
16	スガナイ	07/24	11/26
17	チャナイ	07/27	03/25
18	チュナイ	08/04	04/03
19	チャナイ	07/04	03/14
2	ローコドット	08/24	06/30
20	ジャナイト	05/10	04/29
21	グアルイト	07/24	11/20
22	アルイト	04/09	10/16
23	パイライト	09/06	07/23
24	ペウイト	10/10	3/5
25	ウニコニード	06/27	05/11
26	テラダイト	06/19	05/15
27	ランスイト	12/13	2/2
28	ノスピネルト	05/17	04/13
29	イオンイト	12/08	05/25
3	アコバイト	02/16	06/01
30	コーバイト	11/12	7/18
4	ラメンイト	03/17	02/21
5	シメンイト	08/31	08/18
6	クロイト	10/03	08/27
7	ルスダイト	04/04	03/17
8	スピイト	02/28	03/07
9	スコサイト	10/27	6/16

(30 rows)

4_2.実行クエリ2(空間データ)

「空間データ取得クエリ」

```
SELECT index.商品id,index.商品名,arrived_goods.入荷元住所,delivery.出荷元住所  
FROM  
index  
JOIN  
arrived_goods  
ON  
index.商品id=arrived_goods.商品id  
RIGHT JOIN  
delivery  
ON  
index.商品id=delivery.商品id  
;
```

・[空間データ取得クエリ\(Github\)](#)

4_2.実行結果2(空間データ)

商品id	商品名	入荷元住所	出荷元住所
1	ル	兵	京
10	コ	加	都
11	ソ	古	京
12	コ	川	京
13	エ	市	京
14	イ	平	京
15	イ	市	京
16	イ	市	京
17	イ	市	京
18	イ	市	京
19	イ	市	京
2	イ	市	京
20	イ	市	京
21	イ	市	京
22	イ	市	京
23	イ	市	京
24	イ	市	京
25	イ	市	京
26	イ	市	京
27	イ	市	京
28	イ	市	京
29	イ	市	京
3	イ	市	京
30	イ	市	京
4	イ	市	京
5	イ	市	京
6	イ	市	京
7	イ	市	京
8	イ	市	京
9	イ	市	京
(30 rows)			

4_3.実行クエリとその結果3(在庫の整合性を確認)

「在庫の整合性を確認」

SELECT index.商品id,index.商品
名,arrived_goods.入荷個数,delivery.出荷個
数,index.在庫

FROM

index

JOIN

arrived_goods

ON

index.商品id=arrived_goods.商品id

RIGHT JOIN

delivery

ON

index.商品id=delivery.商品id

;

・[在庫の整合性を確認\(Github\)](#)

商品id	商品名	入荷個数	出荷個数	在庫
1	ルコソコ	50	3	47
10	フォレコ	36	6	30
11	コサコ	42	31	11
12	コロコ	30	15	15
13	マローダ	48	13	35
14	バイラ	33	8	25
15	フガナ	19	3	16
16	スピン	50	0	50
17	ガナ	49	26	23
18	チユナ	41	11	30
19	チア	18	3	15
2	ロアー	21	9	12
20	ジグ	19	6	13
21	グアラ	32	11	21
22	アルバ	27	2	25
23	パイラ	44	34	10
24	ペウタ	34	1	33
25	ウコニ	45	12	33
26	テラ	20	19	1
27	ランス	13	11	2
28	ノスピ	14	4	10
29	イオ	30	19	11
3	アオン	41	7	34
30	コラ	5	2	3
4	シメン	44	11	33
5	シメン	50	4	46
6	クル	37	14	23
7	ルス	25	6	19
8	スビ	40	4	36
9	スコ	22	2	20
(30 rows)				

5_1.レポート(データ抽出、在庫10個以上を抽出)

「在庫が10個以上」

```
SELECT index.商品id,index.商品  
名,arrived_goods.入荷個数,delivery.出荷個  
数,index.在庫 FROM
```

```
index
```

```
JOIN
```

```
arrived_goods
```

```
ON
```

```
index.商品id=arrived_goods.商品id
```

```
RIGHT JOIN
```

```
delivery
```

```
ON
```

```
index.商品id=delivery.商品id
```

```
WHERE
```

```
index.在庫>=10
```

```
;
```

[・在庫が10個以上\(Github\)](#)

商品id	商品名	入荷個数	出荷個数	在庫
1	ルコソコ	50	3	47
10	フレコ	36	6	30
11	フレコ	42	31	11
12	フレコ	30	15	15
13	フレコ	48	13	35
14	フレコ	33	8	25
15	フレコ	19	3	16
16	フレコ	50	0	50
17	フレコ	49	26	23
18	フレコ	41	11	30
19	フレコ	18	3	15
2	ルコソコ	21	9	12
20	フレコ	19	6	13
21	フレコ	32	11	21
22	フレコ	27	2	25
23	フレコ	44	34	10
24	フレコ	34	1	33
25	フレコ	45	12	33
28	フレコ	14	4	10
29	フレコ	30	19	11
3	ルコソコ	41	7	34
4	フレコ	44	11	33
5	フレコ	50	4	46
6	フレコ	37	14	23
7	フレコ	25	6	19
8	フレコ	40	4	36
9	フレコ	22	2	20

(27 rows)

5_2.レポート(データ抽出、東京が出荷先に含まれる物を抽出)

「東京が出荷先に含まれる」

```
SELECT index.商品id,index.商品  
名,arrived_goods.入荷元住所,delivery.出荷元  
住所 FROM  
index  
JOIN  
arrived_goods  
ON  
index.商品id=arrived_goods.商品id  
RIGHT JOIN  
delivery  
ON  
index.商品id=delivery.商品id  
WHERE  
delivery.出荷元住所  
LIKE  
'%東京%'  
;
```

・東京が出荷先住所に含まれる ([Github](#))

商品id	商品名	入荷元住所	出荷元住所
錠 20	ジャグイト	福井県越前市小杉町1-4-1	東京都渋谷区上原4丁目433番地20号
(1 row)			

5_3.レポート(データ抽出、商品名にダイトを含む鉱物を入荷している場所と、その購入者)

「商品名にダイトを含む鉱物を入荷している場所と、その購入者」

```
SELECT index.商品id,index.商品名,arrived_goods.入荷元住所,delivery.出荷元住所 FROM
index
JOIN
arrived_goods
ON
index.商品id=arrived_goods.商品id
RIGHT JOIN
delivery
ON
index.商品id=delivery.商品id
WHERE
index.商品名
LIKE
'%ダイト'
;
```

・[商品名にダイトを含む鉱物関連\(Github\)](#)

5_3.抽出結果(商品名にダイトを含む鉱物を入荷している場所と、その購入者)

商品id	商品名	入荷元住所	出荷元住所
鉱 8	スピダイト	岩手県釜石市小佐野 7階	京都府相楽郡和束町湯船 414番地14号
鉱 12	スコマダイト	東京都福井県西京前川市市辺小市伏小郡佐富見杉川辺 4-7-5 フリード小佐野町 7階	東京都府相楽郡和束町湯船 414番地14号
鉱 13	マダイト	東京都福井県西京前川市市辺小市伏小郡佐富見杉川辺 3-12-8 ピソプランコ・富士町 914号室	東京都府相楽郡和束町湯船 414番地14号
鉱 20	ジャンダイト	東京都福井県西京前川市市辺小市伏小郡佐富見杉川辺 1-4-1	東京都府相楽郡和束町湯船 414番地14号
鉱 27	ジラ	東京都福井県西京前川市市辺小市伏小郡佐富見杉川辺 5-13-2 T O P ・ 永田 3階	東京都府相楽郡和束町湯船 414番地14号

(5 rows)

6.考察

今回はPostgreSQLとVisualStudio Codeを用いてデータベースの管理・構築を行った。

今回分かった事として、SQL文を書く場合は、まず一度SQLと接続が可能なテキストエディタ(VisualStudio Code等)を用いて書くのが最も効率が良い事がある。

ubuntu等のターミナルに直接書く場合だとエラーを起こすインデント等の判別が分かり辛い場合がある為、様々な補足機能を導入出来るエディタを通して書く事により、不要なエラー処理を減らす事が可能である。

またDB構築をする前にリレーション図や実際のDB構築目標を図に起こす事で、視覚的にも効率的なDB管理が行える。

6.参考文献、使用データ

- ・テストデータジェネレータ
- ・それっぽい名前ジェネレータ
- ・乱数ジェネレータ
- ・Webジェネレータ(日時)
- ・住所データのランダム生成