2016-06-14 Exam Exercises

The following exercises and their solutions were originally authored by *Professor Maria de Marsico*, for the "Basi di Dati" exam session on June, the 14th, 2016.

Their translation is ongoing.

1. Relational Algebra

Consider a database with the following schema, describing a manufacturer's operations:

```
PRODUCTS (ID, Description, UnitPrice)
WAREHOUSES (ID, Address)
STOCKS (ProductID, WarehouseID, Units)
```

`STOCKS` instances describe how many *products' units* are stocked and in which *warehouses*. When a product is stocked, its `Units ⩾ 1`.

Write **relational algebra expressions** for the following queries:

1. For each product whose stock is equal or larger than 10 units, *in any warehouse*, get both:
    - the product's data (`ID`, `Description` and `UnitPrice`)
    - the addresses of all the warehouses where at least 10 product units are stocked.
2. Find the `ID`, `Description` and `UnitPrice` of the products that *aren't* stocked at all, anywhere.

You can load a sample RelaX dataset with this gist ID: `126fcdb8c1bedc5080270dff5f642186`

1.1. Answer

We must identify the relations containing the needed data. We require all of them, as:

- `STOCKS` contains the number of stocked units for each product
- `PRODUCTS` holds the data for products details
- `WAREHOUSES` includes the addresses

The easiest query involves:

1. joining the three relations together, via **theta joins** where appropriate
2. performing a **selection** on the resulting relation, by filtering those tuples whose `Units` are equal or higher than 10
3. using a **projection** to pick out the attribute values we require

> `PRODUCTS` and `WAREHOUSES` both feature an `ID` attribute, although these identify tuples in different relations, with different meanings. It wouldn't make sense to perform a **natural join** between them.

Let r identify the desired data:

$$r = \sigma_{Units \geq 10} \text{ STOCKS} \bowtie_{WarehouseID=ID} \text{WAREHOUSES} \bowtie_{ProductID=PRODUCTS.ID} \text{PRODUCTS}$$

A less efficient *alternative*, due to more *joins*, could be:

$$r = \sigma_{Units \geq 10} (\text{WAREHOUSES} \bowtie_{ID=WarehouseID} \text{STOCKS} \bowtie_{ProductID=PRODUCTS.ID} \text{PRODUCTS})$$

We then need to select the relevant attributes, via a **projection** on r:

$$\pi_{PRODUCTS.ID,\ Description,\ UnitPrice,\ Address} (r)$$

RelaX Code

```
r = σ Units ⩾ 10 STOCKS ⨝ WarehouseID = ID WAREHOUSES ⨝ ProductID =
PRODUCTS.ID PRODUCTS
π PRODUCTS.ID, Description, UnitPrice, Address (r)
```

1.2. Answer

This class of relational algebra problems is best handled with **subtractions**. The tuples that *don't* meet the selection criteria are first collected and then removed from the set of all the candidate tuples.

In this case we don't need to query the `WAREHOUSES` relation, seeing as it contains no relevant data for our purposes.

Products that *aren't* stocked *don't appear* in `STOCKS` instances; there are no such tuples whose `Units` value is `0`.

Let r be the relation which includes the data of all those products we aren't interested in:

$$r = \pi_{ID,\ Description,\ UnitPrice}\,(\mathrm{PRODUCTS} \bowtie_{ID=ProductID} \mathrm{STOCKS})$$

We are selecting **all** the tuples that match stocked products, referenced in `STOCKS` via the `ProductID` attribute. *Unstocked* products, absent from `STOCKS`, won't be included in the *join*.

We finally **subtract** the data of all stocked products, r, from the set of all products (stocked and otherwise):

$$\mathrm{PRODUCTS} - r$$

The initial **projection** ensures that the two relations' schemas are **compatible**, as required by the **subtraction**.

RelaX Code

```
PRODUCTS - π ID, Description, UnitPrice (PRODUCTS ⨝ ID = ProductID STOCKS)
```