2016-06-14 Exam Exercises and Solutions

Exercises

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 number of `Units` is always positive, that is $\mathrm{Units} \geq 1$.

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

a. 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.

b. Find the `ID` , `Description` and `UnitPrice` of the products that *aren't* stocked at all, anywhere.

> You can load a sample RelaX dataset with the gist ID `126fcdb8c1bedc5080270dff5f642186`

Solutions

1. Relational Algebra

Question (a)

Insight

We must first identify the relations which contain all the required data. As it turns out, we need all of them:

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

The easiest query consists of:

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.

Answer

The following expression identifies the desired data:

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

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

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

We then need to select the relevant attributes, via a **projection** on $\mathbf{r}$:

$$\Pi_{\text{PRODUCTS.ID, Description, UnitPrice, Address}}(\mathbf{r})$$

RelaX Code

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

Question (b)

Insight

When tackling these queries **subtractions** are required. We find those tuples which *don't* meet our criteria and we remove them from the set of all the 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.

Answer

Let $\mathbf{r}$ be the relation which includes the data of all those products we aren't interested in:

$$\mathbf{r} = \Pi_{\text{ID, Description, UnitPrice}}(\text{PRODUCTS} \bowtie_{\text{ID}=\text{ProductID}} \text{STOCKS})$$

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, $\mathbf{r}$, from the set of all products (stocked and otherwise):

$$\text{PRODUCTS} - \mathbf{r}$$

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)
```