Databases - Lab Sheet 3

Gordon Royle

School of Mathematics & Statistics
University of Western Australia

Student ids

Getting the ids is a *projection* of the table S onto the column `sid`.

$$\rho_{\texttt{sid}}\,(S)$$

Names of the male students

This needs a selection to extract the rows corresponding to the male students, and then a projection onto the `name` column, but all the information comes from the table S.

$$\rho_{\texttt{name}}\left(\sigma_{\texttt{gender}='M'}\left(S\right)\right)$$

Class list for Databases

This expressions needs all three tables to get the student name, the unit name and the grades.
They can be joined using the natural join ($\bowtie$) but then the rows correspoinding to Databases must be extracted.
The column called `name` needs to be qualified because `S.name` is the student's name, while `U.name` is the unit name.

$$\pi_{\texttt{S.name,G.grade}}\left(\sigma_{\texttt{U.name='Databases'}}\left(S \bowtie G \bowtie U\right)\right)$$

Translating SQL to RA

This is easy as it just uses a join, a selection and a projection. As S and G have just one column name in common - that is, `sid`, the natural join will be fine.

$$\pi_{\texttt{sid}}\left(\sigma_{\texttt{grade}<50}\left(S \bowtie G\right)\right)$$

The query gets the IDs of anyone who has failed a unit (any unit).

Translating RA to SQL

The expression involves those rows of the natural join (i.e. $\bowtie$) relating to the unit CITS1402,

```
SELECT *
FROM S NATURAL JOIN G
WHERE G.uid = 'CITS1402';
```

As there is no projection onto a particular set of columns at the end, all the columns are selected and so `SELECT *` is the easiest way to do that (or give the names of all the columns).

Average population

One aggregate function applied to every row of one table:

```
SELECT AVG(Population)
FROM Country;
+------------------+
| AVG(Population)  |
+------------------+
|   25434098.1172  |
+------------------+
1 row in set (0.40 sec)
```

Total population

One aggregate function applied to every row of one table, and then renamed

```
SELECT SUM(Population)
AS worldPopulation
FROM Country;
+-----------------+
| worldPopulation |
+-----------------+
|      6078749450 |
+-----------------+
```

English countries

We need to join `Country` and `CountryLanguage` and then pull out the English-speaking countries, before applying the aggregate operators.

```
SELECT MIN(Population),
       MAX(Population),
       AVG(Population)
FROM   CountryLanguage L
       JOIN Country C
         ON L.CountryCode = C.Code
WHERE  L.Language = 'English'
       AND L.IsOfficial = 'T';
+-----------------+-----------------+-----------------+
| MIN(Population) | MAX(Population) | AVG(Population) |
+-----------------+-----------------+-----------------+
|               0 |       278357000 |   10435427.2727 |
+-----------------+-----------------+-----------------+
```

Life expectancy

This is a straightforward selection of rows from `Country` where the `Continent` is Asia, and then finding the minimum value of one of the columns.

```
SELECT MIN(LifeExpectancy)
FROM   Country
WHERE  Continent = 'ASIA';
+----------------------+
| MIN(LifeExpectancy)  |
+----------------------+
|                 45.9 |
+----------------------+
```

How many aggregate functions?

There are 17 different function *names*, (but 14 different functions)

12.19.1 GROUP BY (Aggregate) Functions

Table 12.24 Aggregate (`GROUP BY`) Functions

Name	Description
`AVG()`	Return the average value of the argument
`BIT_AND()`	Return bitwise and
`BIT_OR()`	Return bitwise or
`BIT_XOR()`	Return bitwise xor
`COUNT(DISTINCT)`	Return the count of a number of different values
`COUNT()`	Return a count of the number of rows returned
`GROUP_CONCAT()`	Return a concatenated string
`MAX()`	Return the maximum value
`MIN()`	Return the minimum value
`STD()`	Return the population standard deviation
`STDDEV_POP()`	Return the population standard deviation
`STDDEV_SAMP()`	Return the sample standard deviation
`STDDEV()`	Return the population standard deviation
`SUM()`	Return the sum
`VAR_POP()`	Return the population standard variance
`VAR_SAMP()`	Return the sample variance
`VARIANCE()`	Return the population standard variance