CS252 Fundamentals of Relational Databases — Worksheet 3

Working with a Database in Rel

If you get any unpleasant error messages you may wish to study the Rel forum at http://shark.armchair.mb.ca/~dave/relforum/index.php, in the section named Report a Bug.

1. Start up Rel. Instructions are at http://www.dcs.warwick.ac.uk/~acristea/courses/CS252/rel. Before you proceed to Step 2, note that you are welcome to “cheat” by using the scripts provided for this step at this web page.

2. Here is the supplier-and-parts database from Chris Date’s Introduction to Database Systems (8th edition), as shown on the inside back cover of that book (except that we have converted the attribute names to mixed case and changed the US spelling “Color” to the British spelling “Colour”):

<table>
<thead>
<tr>
<th></th>
<th>S#</th>
<th>Sname</th>
<th>Status</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S1</td>
<td>Smith</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>2</td>
<td>S2</td>
<td>Jones</td>
<td>10</td>
<td>Paris</td>
</tr>
<tr>
<td>3</td>
<td>S3</td>
<td>Blake</td>
<td>30</td>
<td>Paris</td>
</tr>
<tr>
<td>4</td>
<td>S4</td>
<td>Clark</td>
<td>20</td>
<td>London</td>
</tr>
<tr>
<td>5</td>
<td>S5</td>
<td>Adams</td>
<td>30</td>
<td>Athens</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P#</th>
<th>Pname</th>
<th>Colour</th>
<th>Weight</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P1</td>
<td>Nut</td>
<td>Red</td>
<td>12.0</td>
<td>London</td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
<td>Bolt</td>
<td>Green</td>
<td>17.0</td>
<td>Paris</td>
</tr>
<tr>
<td>3</td>
<td>P3</td>
<td>Screw</td>
<td>Blue</td>
<td>17.0</td>
<td>Oslo</td>
</tr>
<tr>
<td>4</td>
<td>P4</td>
<td>Screw</td>
<td>Red</td>
<td>14.0</td>
<td>London</td>
</tr>
<tr>
<td>5</td>
<td>P5</td>
<td>Cam</td>
<td>Blue</td>
<td>12.0</td>
<td>Paris</td>
</tr>
<tr>
<td>6</td>
<td>P6</td>
<td>Cog</td>
<td>Red</td>
<td>19.0</td>
<td>London</td>
</tr>
</tbody>
</table>

The following instructions assume you would like to set up at least one of these relvars manually rather than by using the provided scripts. In any case, you are strongly advised to investigate Tutorial D’s VAR statement for defining relvars, at least to familiarize yourself with the syntax.

Execute a Rel VAR statement for each of S, P and SP. Use INTEGER as the declared type for STATUS and QTY, RATIONAL for WEIGHT, and CHAR for all the other attributes. Feel free to use lower case or mixed case to suit your own taste for attribute and relvar names, but do not otherwise change any of the given names.

To help you with the syntax for a database relvar declaration, here is one for Hugh Darwen’s enrolment example:

```sql
VAR enrolment BASE RELATION
  { StudentId CHAR, CourseId CHAR, Name CHAR }
KEY { StudentId, CourseId } ;
```

CS252 Worksheet 3: Working with a Database in Rel — Page 1 of 3
**Tutorial D** requires at least one **KEY** constraint to be specified for each relvar. One key for each for S, P and SP is shown by underlining the attribute name in the table. No other **KEY** constraints are needed.

"Populate" (as they say) each relvar with the values shown in Date's tables. There are several ways of achieving this. Choose whichever you prefer from the following:

a. Add an **INIT** ( ... ) specification to the end of the **VAR** statement before you execute it. Inside the parens, write a **RELATION** { ... } expression, using a **TUPLE** expression for each required tuple, as in the enrolment literal used in Worksheet 1.

b. Execute the **VAR** statement without an **INIT** ( ... ) specification. The implied initial value is the empty relation of the appropriate type. You can see this by asking Rel for the current value of the relvar. For example, to get the current value of S, just type S into the lower pane and click Run (F5).

Now use an assignment statement of the form

\[
\text{varname} := \text{RELATION-expression}
\]

to populate the relvar. Check that Rel has indeed assigned the correct value to it.

c. Use Rel **INSERT** statements to populate the relvar piecemeal, perhaps one tuple at a time. Having typed in the first **INSERT** statement. Here is the general form of an **INSERT** statement to insert a single tuple:

\[
\text{INSERT varname RELATION} \{ \text{TUPLE} \{ \ldots \} \};
\]

Note that the source operand is still a relation, not just a tuple, hence the need to “wrap” the **TUPLE** expression inside **RELATION** { }.

3. Informally, we refer to S as suppliers, P as parts and SP as shipments. Predicates for these relvars are:

**S**: Supplier S# is named Sname, has status Status and is located in city City.

**P**: Part P# is named Pname, is coloured Colour, weighs Weight and is located in city City.

**SP**: Supplier S# ships part P# in quantities of Qty.

What, then, is the predicate for the expression S JOIN SP JOIN P? What do you expect to be the result of that expression? What is its degree? Does Rel give the result you expected? Explain what you see.

4. Attempt to insert a tuple into SP with supplier number S1, part number P1 and quantity 100. Explain the result of your attempt.

5. [see next page]
5. Get Rel to evaluate each of the following expressions. For each one, write down the corresponding predicate and also give an informal interpretation of the query in the style of those given in Exercise 6 below.

a. SP WHERE P# = 'P2'
b. S { ALL BUT Status }
c. SP { S#, Qty }
d. S MATCHING ( SP WHERE P# = 'P2' )
e. P NOT MATCHING ( SP WHERE S# = 'S2' )
f. S { City } UNION P { City }
g. S { City } MINUS P { City }
h. S { S#, City } COMPOSE P { P#, City }
i. ( S RENAME ( City AS SC ) ) { SC } JOIN
   ( P RENAME ( City AS PC ) ) { PC }

6. Write Tutorial D expressions for the following queries and get Rel to evaluate them:

a. Get all shipments.
b. Get supplier numbers for suppliers who supply part P1.
c. Get suppliers with status in the range 15 to 25 inclusive.
d. Get part numbers for parts supplied by a supplier in Paris.
e. Get part numbers for parts not supplied by any supplier in Paris.
f. Get city names for cities in which at least two suppliers are located.
g. Get all pairs of part numbers such that some supplier supplies both of the indicated parts.
h. Get the total number of parts supplied by supplier S1.
i. Get supplier numbers for suppliers with a status lower than that of supplier S1.
j. Get supplier numbers for suppliers whose city is first in the alphabetic list of such cities.
k. Get part numbers for parts supplied by all suppliers in London.
l. Get supplier-number/part-number pairs such that the indicated supplier does not supply the indicated part.
m. Get all pairs of supplier numbers, Sx and Sy say, such that Sx and Sy supply exactly the same set of parts each.