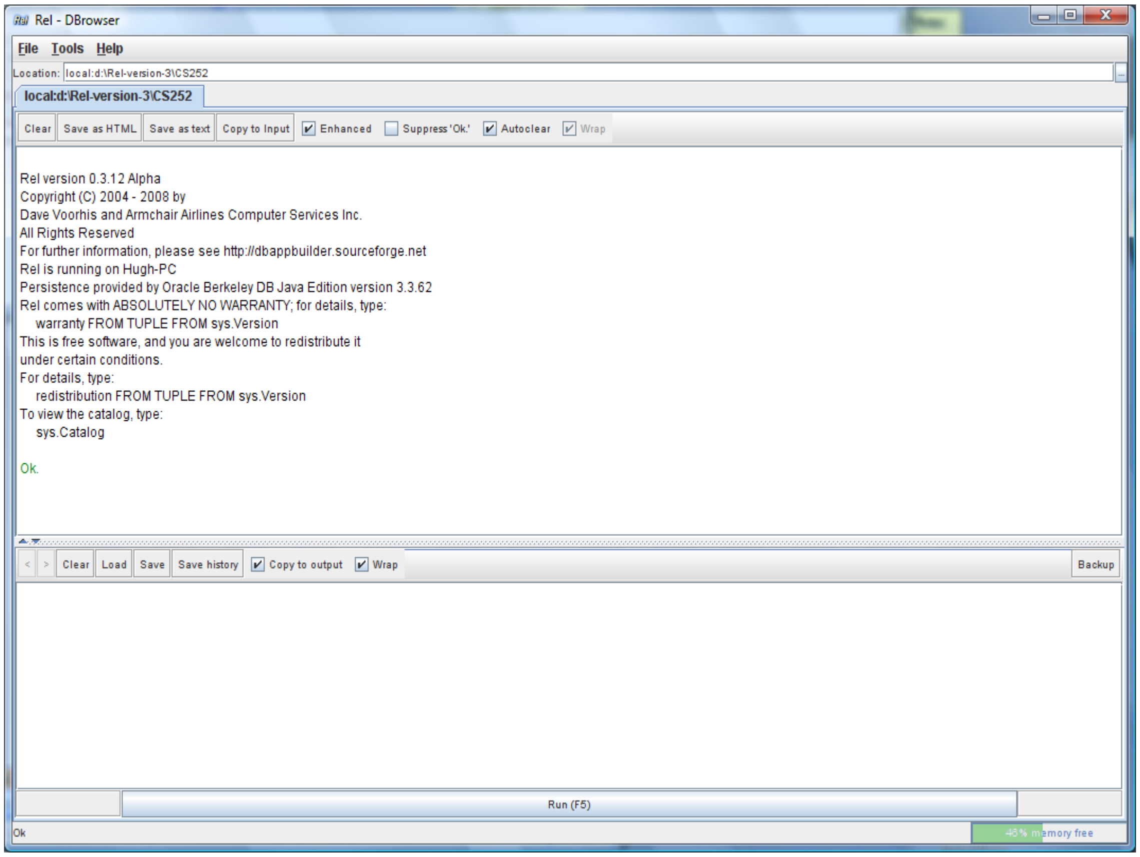
CS252 Fundamentals of Relational Databases — Worksheet 1

Getting Started with *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/.
2. Familiarise yourself with the way of working and the things you can do in *Rel*. You should be looking at a window something like this (which was obtained in Windows Vista):



```
Rel version 0.3.12 Alpha
Copyright (C) 2004 - 2008 by
Dave Voorhis and Armchair Airlines Computer Services Inc.
All Rights Reserved
For further information, please see http://dbappbuilder.sourceforge.net
Rel is running on Hugh-PC
Persistence provided by Oracle Berkeley DB Java Edition version 3.3.62
Rel comes with ABSOLUTELY NO WARRANTY; for details, type:
    warranty FROM TUPLE FROM sys.Version
This is free software, and you are welcome to redistribute it
under certain conditions.
For details, type:
    redistribution FROM TUPLE FROM sys.Version
To view the catalog, type:
    sys.Catalog

Ok.
```

- Note the layout of the window: a lower pane into which you can type statements to be executed, an upper pane in which results are displayed, and the movable horizontal bar between the panes.
- Note the ▲ and ▼ at the left-hand end of the horizontal bar, allowing you to let one or the other pane occupy the whole window for a while.
- See what is available on the Tools menu and perhaps choose your preferred font. *Alert:* At the time of writing the chosen font appears to be used only when you do not select Enhanced on the output pane.
- Note the < and > to the left of the menu on the input (lower) pane. These are greyed out initially but after you have executed a couple of statements you will be able to use them to recall previously executed statements to the input pane.

- Note the toolbars on both panes. As you do the exercises, decide which options suit you best. Note that you can save the contents of either pane into a local file, and that you load the contents of a local file into the input area.
- Note the check boxes on the right of the toolbars. They are fairly self-explanatory, apart from "Enhanced", which we will examine later.
- The box at the top of the upper pane, labelled "Location:", identifies the directory containing the database you are working with. You can switch to another directory by clicking on the little button to the right of the box, labelled with three dots (…).

3. Type the following into the lower pane:

    ```
    output 2+2 ;
    ```

    Execute what you have typed, either by clicking on Evaluate (F5) shown at the bottom of the window or by pressing F5.

    Now delete the semicolon and try executing what remains. (If necessary, use the < button on the lower pane to recall the statement.) You will see how *Rel* handles errors.

    Now strike out the word `output` and do not put back the semicolon. What happens when you execute that? (i.e., just `2+2`).

    You have now learned:

    - that in *Rel* (as in **Tutorial D**) every executable *statement* is terminated by a semicolon;
    - that *Rel* allows you to obtain the result of evaluating an *expression* by using an `output` statement;
    - that *Rel* treats an attempt to "execute" an expression `x` as shorthand for the statement `output x ;` — the absence of the semicolon signals to *Rel* that you are using this convenient shorthand.

4. This exercise is merely to alert you to a certain awkwardness in *Rel* that has no real importance but might cause you to waste a lot of time if you are not warned about it. It's the same as Step 3 except that instead of `2+2` you type `2+2.0`. Look closely at what happens. It doesn't work!

    Rel, like some other languages, treats `INTEGER` and `RATIONAL` as distinct types. If you want to do arithmetic on rational numbers, both operands must be rational numbers. Literals denoting rational numbers are distinguished from those denoting integers by the presence of a decimal point, and *Rel* follows the convention in the English-speaking community of using a full stop for this purpose (as opposed to the comma that is used throughout most of Europe, for example).

    Now try this: `1/2` (i.e., the integer 1 divided by the integer 2). And then this: `1.0/2.0`.

    You have now learned that (a) the operands of dyadic arithmetic operators in *Rel* must be of the same type, and (b) the type of the result of an invocation of such an operator is always of the same type as the operands. **Tutorial D** is silent on such issues, because they are orthogonal to what **Tutorial D** is really intended for (teaching relational theory). But every implementation of **Tutorial D** has to address them somehow.

    Fortunately, arithmetic is orthogonal to relational theory and there is no need for us to be bothered by *Rel*'s behaviour here. You have possibly already learned that the same problems do not arise in SQL, where `1/2`, `1/2.0` and `1.0/2.0` are all equivalent, in spite of the fact that `INTEGER` and `REAL` (SQL's counterpart of **Tutorial D**'s `RATIONAL`) are also distinct types in SQL.

5. Now try the following compound statement:

```
begin ;
VAR x integer init(0) ;
x := x + 1 ;
output x ;
end ;
```

Why do we have to write `output x ;` in full here, instead of just `x`?

Now write the fourth line in uppercase: `OUTPUT X ;` What happens?

Try `OUTPUT x ;` instead. What have you learned about *Rel*'s rules concerning *case sensitivity*?

6. Now you can start investigating *Rel*'s support for relations (though not relational databases yet). First, see how *Rel* displays a relation (i.e., the result of evaluating a relation expression) in its upper pane. *Rel* supports two styles of presentation, depending on whether the “Enhanced” option is checked.

With “Enhanced” unchecked (it is usually checked to start with), get *Rel* to evaluate the following relation expression (a literal which we shall call `enrolment`):

```
RELATION {
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  },
TUPLE { StudentId 'S1', CourseId 'C2', Name 'Anne'  },
TUPLE { StudentId 'S2', CourseId 'C1', Name 'Boris' },
TUPLE { StudentId 'S3', CourseId 'C3', Name 'Cindy' },
TUPLE { StudentId 'S4', CourseId 'C1', Name 'Devinder' } }
```

See Lecture HACD.2, Slides 9-12. Look closely at the output. Is it identical to the input?

Next, without altering the contents of the lower pane, turn “Enhanced” back on. Note the effect on the display in the output pane.

Now delete all the tuple expressions, leaving just `RELATION { }`. What happens when *Rel* tries to evaluate that?

Now use < to recall the original `RELATION` expression to the input pane and re-evaluate it with “Enhanced” *off*. Use copy-and-paste to copy the result to the input pane, then delete all the `TUPLE` expressions, to leave this:

```
RELATION {StudentId CHARACTER, CourseId CHARACTER,
          Name CHARACTER} { }
```

Study the result of that in the output pane, first with “Enhanced” off, then with it on.

What conclusions do you draw from all this, about *Rel* and **Tutorial D**?

From now on you can run with “Enhanced” either on or off, according to your own preference.

Next, enter the following literal, perhaps by using the < button to recall `enrolment` and editing it:

```
RELATION {
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  },
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  }
 }
```

Before you press Evaluate (F5), think about what you expect to happen. Does the result meet your expectation? How do you explain it?

Use < again to recall the `enrolment` literal. Insert the word `WITH` at the beginning, add `AS enrolment : enrolment` at the end, to give:

```
WITH RELATION {
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  },
TUPLE { StudentId 'S1', CourseId 'C2', Name 'Anne'  },
TUPLE { StudentId 'S2', CourseId 'C1', Name 'Boris' },
TUPLE { StudentId 'S3', CourseId 'C3', Name 'Cindy' },
TUPLE { StudentId 'S4', CourseId 'C1', Name 'Devinder' } }
AS enrolment : enrolment
```

and evaluate that.

How do you understand what you have just done? (`WITH` isn't taught in the lectures. In case you aren't clear, try this in *Rel*: `WITH 2+2 as four, four+four as eight : eight + four`. Note carefully that the introduced names, `four` and `eight`, are local only.)

By inspection of `enrolment` only, write down all the cases you can find of two students such that there is at least one course they are both enrolled on.

7. For this exercise you will need to continue using < to recall your previous command (now including the definition of the introduced name `enrolment`) and overtype as necessary. Use `enrolment` to investigate the relational operator known as **projection (*see Lecture HACD.4, slides 18-21*).** The projection of a given relation over a specified subset of its attributes yields another relation. In **Tutorial D** a projection is specified by writing a list of attribute names, enclosed in braces `{}` and separated by commas, after the operand relation. The key words `ALL BUT` can optionally precede the list of attribute names, inside the braces.

   How many distinct projections can be obtained from `enrolment`? Obtain as many of these as you wish, trying both the "inclusion" method and the "exclusion" method using `ALL BUT`.

8. Still using `enrolment`, investigate the relational operator known as **rename (*see Lecture HACD.4, slides 13-14*).** The renaming of a given relation returns that relation with one or more of its attributes renamed. In **Tutorial D** a renaming is specified by writing `RENAME ( *old* AS *new*, ... )` after the operand relation.

   At the moment you should have this in your input pane:

```
WITH RELATION {
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  },
TUPLE { StudentId 'S1', CourseId 'C2', Name 'Anne'  },
TUPLE { StudentId 'S2', CourseId 'C1', Name 'Boris' },
TUPLE { StudentId 'S3', CourseId 'C3', Name 'Cindy' },
TUPLE { StudentId 'S4', CourseId 'C1', Name 'Devinder' } }
AS enrolment : enrolment
```

   Replace the single word (`enrolment`) that follows the colon by a renaming of `enrolment` such that the result has attribute name `SID1` instead of `StudentId`, `N1` instead of `Name`, and is otherwise the same as `enrolment` itself. Replace the : that ends the `WITH` specification by a comma and add `AS E1 : E1` at the end. The result should look like this:

```
WITH RELATION {
TUPLE { StudentId 'S1', CourseId 'C1', Name 'Anne'  },
TUPLE { StudentId 'S1', CourseId 'C2', Name 'Anne'  },
TUPLE { StudentId 'S2', CourseId 'C1', Name 'Boris' },
TUPLE { StudentId 'S3', CourseId 'C3', Name 'Cindy' },
TUPLE { StudentId 'S4', CourseId 'C1', Name 'Devinder' } }
AS enrolment ,
<your renaming of enrolment, as specified> AS E1 : E1
```

Evaluate that to check that you wrote the renaming correctly.

9. Again replace the : by a comma and this time add a similar renaming of `enrolment,` using `SID2` and `N2` instead of `SID1` and `N1` for the new attribute names, and add `AS E2 : E1 JOIN E2` at the end. You are investigating the operator called `JOIN` **(*see Lecture HACD.4, slides 10-12, 15-17, and 22*).**

   How do you interpret the result? How many tuples does it contain? Replace the key word `JOIN` by `COMPOSE` **(*see Lecture HACD.6, slides 7-8*)**. How do you interpret *this* result? How many tuples are there now? How do you account for the difference?

10. Add `WHERE NOT ( SID1 = SID2 )` to end of the expression you evaluated in Step 9 **(*see Lecture HACD.5, slides 3-6*)**. Examine the result closely. Now place parentheses around `E1 COMPOSE E2` and evaluate again. Confirm that you get the same result.

    Repeat the experiment, replacing `WHERE NOT ( SID1 = SID2 )` by `{ SID1 }`. Do you get the same results this time? If not, why not?

    What does all this tell you about operator precedence rules in **Tutorial D**?

    Why was it probably a good idea to add that `WHERE` invocation? Did it completely solve the problem? If not, can you think of a better solution?

    What connection, if any, do you see between this exercise and Exercise 6?

11. Close *Rel* by clicking on File/Exit.

    Write a summary of your experiences, good *and* bad, with this worksheet. Feel free to comment on *Rel*, **Tutorial D**, and the worksheet, as you wish.

End of Worksheet