Unit and System Testing

Testing Principle

Standard commercial practice is to first carry out tests on a new unit of software, and only when it has passed these tests, to integrate it into the system and test the whole system with the new unit included. This principle will be followed with the Raquel DBMS.

What Constitutes a Unit in [IncrementOne] ?

The answer would appear to be that each of the 4 new 'Token' classes (for **Meta**, **<==Attribute**, **<==Key**, and **<--Remove**) constitute a unit.

However each of these classes inherits from other classes, so the base classes need to be included in the new 'Token' class units.

Furthermore, since the existing 'Token' classes need to be linked into the DBMS when an object of the RaquelDBMS class is generated, the same will apply to the 4 new units/classes, and needs to be tested.

Since each new class is a significant volume of code, it is desirable to break each one into smaller code units and test these, then build these code units up into a class for testing. This should be done recursively, because at the lowest level the code units are easy to test.

Each class can be split into 2 parts :

1. The genuine token part of a class instance. This is a straightforward part of object creation.
2. The *ExecuteTree* part of a class.

The *ExecuteTree* part of each class/unit is applied to one or more operands. Therefore they will need to be included in the final testing of *ExecuteTree*.

The *ExecuteTree* method breaks down into components that are either functional code units, or code units that build and/or re-arrange parse trees. The former can be tested on their own. The latter can be tested in 2 parts : firstly to test if the parse tree building/re-arrangement operates correctly; secondly to test if the execution of the parse tree operates correctly. Tested component parts can then combined to test whether the *ExecuteTree* method – including its operands – operates correctly.

It cannot be assumed that the tokeniser, compactor, and parser will generate valid tokens for the 4 new classes/units. This will have to be tested. Therefore each new class/unit should be tested in isolation before being tested as part of the DBMS.

What constitutes the System in [IncrementOne] ?

The answer is the entire DBMS, not the DBMS plus RaquelGUI.

The DBMS is intended to be capable of functioning correctly regardless of the application that provides it with statements and receives its responses.

Therefore it is necessary to have a test harness that handles the DBMS on its own and displays the DBMS's inputs and outputs.

It can be useful to use the RaquelGUI as a test vehicle to enter test statements into the DBMS and view the responses. Nevertheless what passes between the GUI and the DBMS needs to be checked. It should not be assumed to work perfectly with the new classes/units.

Furthermore the RaquelGUI is limited in its ability to be a useful test harness for the [IncrementOne] extension :

- The RaquelGUI only displays relvalues that are retrieved to it. So an invocation of **Meta** would also need a **<--Retrieve** assignment to be executed on the result of the **Meta** invocation in order for it to be seen in the RaquelGUI display, further complicating the test.
- The 3 assignments make changes to the DB and Meta DB. Therefore it will be necessary to view the relvar values of the DB and Meta DB directly, via a test harness, in order to check their performance.
- The RaquelGUI should always display any error messages that arise, but tests need to be carried out to ensure the RaquelGUI works correctly for the new classes/units.