IQS Demo Session

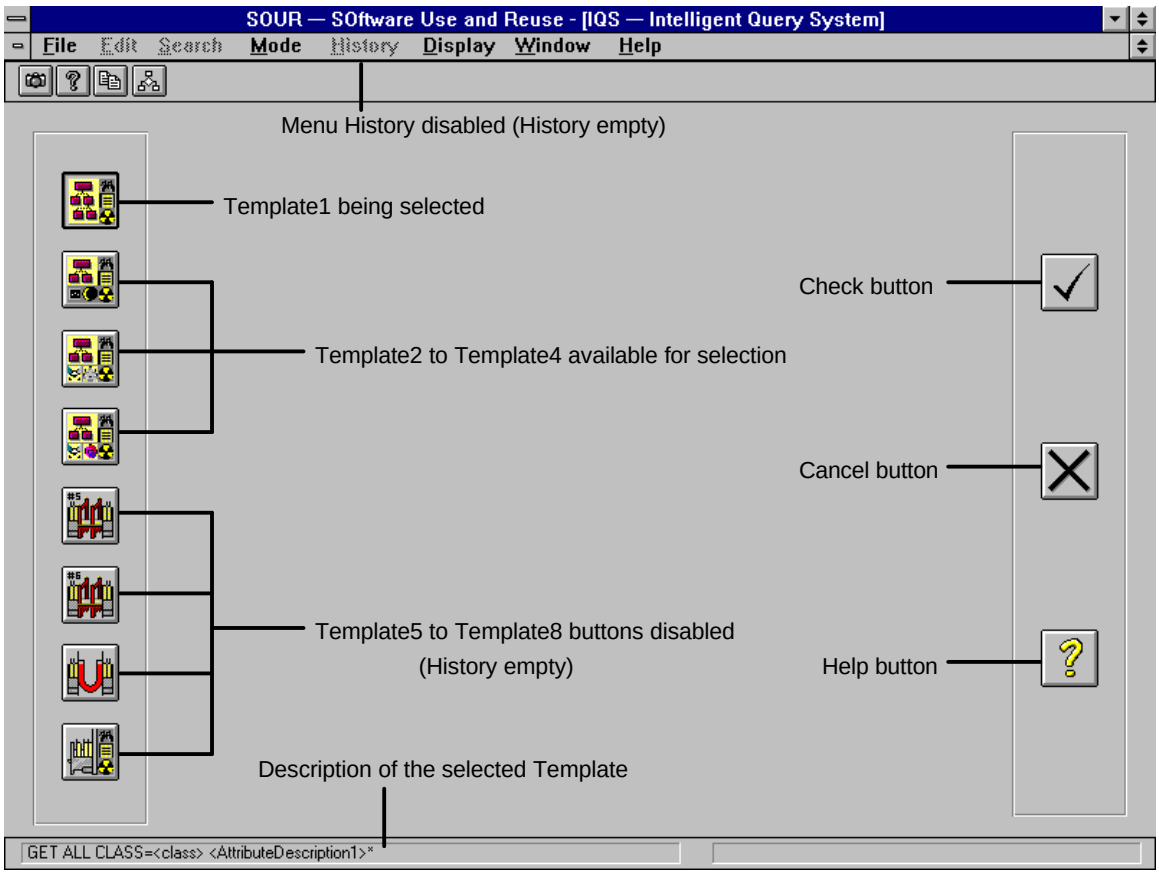
The IQS Demo Session basicaly consists on a sequence of screenshots showing the full use of the main capabilities of both the IQS Assisted and Batch Modes.

The Assisted Mode Demo Session covers all the query *Templates*[1]. Results visualization based on the Result Manager facilities and switching into and back from the Batch Mode are also explored.

The Batch Mode Demo Session mainly focuses on History manipulation (importing, joining, etc) as well as on showing an alternate and less restrictive way to make queries.

Assisted Mode IQS Demo Session

Assisted Mode is the default mode when entering the IQS subsystem. **Screenshoot 0** presents the IQS Assisted Mode front-end for an empty History state. Note that because the History is empty, the related operations are unavailable (History Menu is disabled). Also, for this reason, the last four *Templates* (which are based on re-using objects from the History) are not yet choosable.



Screenshoot 0 - IQS Assisted Mode front-end

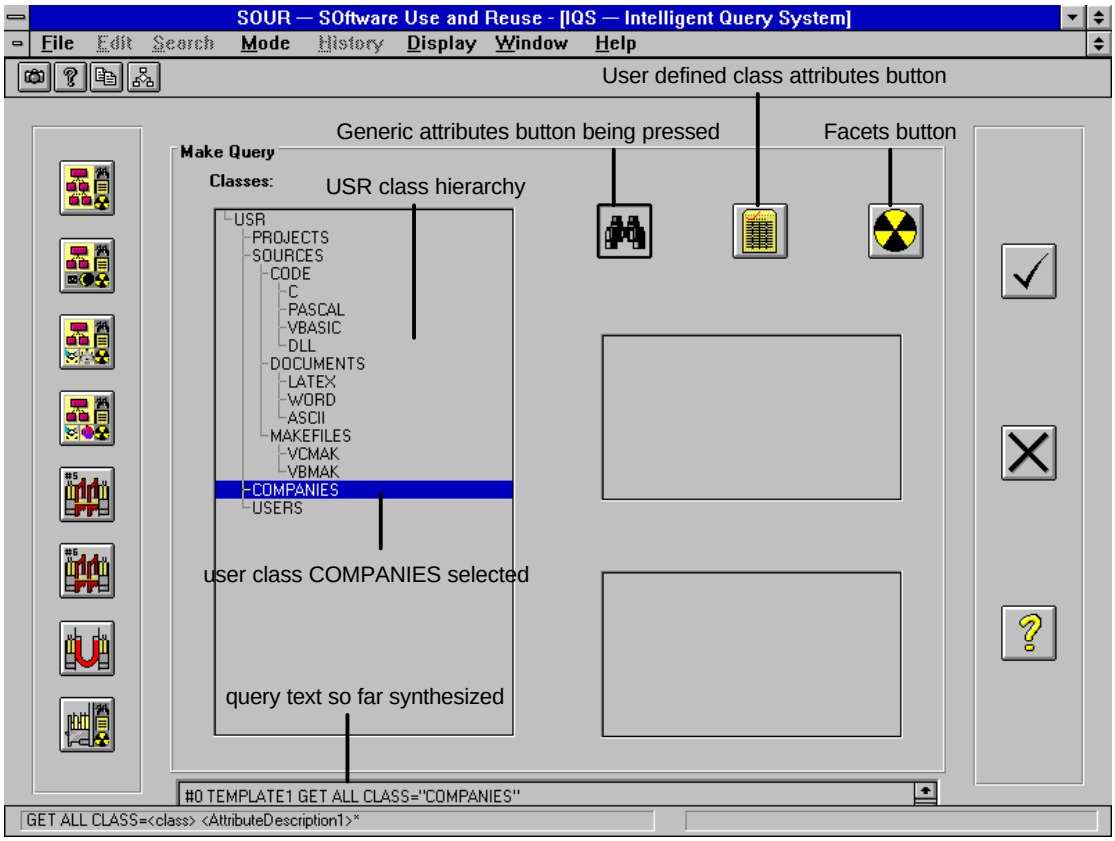
[1]Recall the meaning of each *Template* from the section ?? of the IQS Functional Specifications document.

Template1 demo

Just after having pressed the *Template1* button, the USR class hierarchy is presented, in order to let a user class to be choosen. This class selection is common to all the *kernel-Templates* (*Template1* to *Template4*) and only happens once per query. The objects belonging to that class are the ones to be successively refined during the querying process (basicaly, *kernel-Templates* differ on the refinement criteria).

During any *Template* solving, if the Check button is enabled, then there's already a solution to the query, being that the set of objects solving the query sinthesized so far (this temporary solution can be visualized through Result Manager[2]).

Screenshoot 1 shows IQS's *Template1* desktop after selecting the class `COMPANIES` and about to starting the refinement by *generic attributes*. The possible refinement mechanisms for the *Template1* queries are *generic attributes*, *user defined class attributes* and *facets*, each one being accessible by a specific interface button.

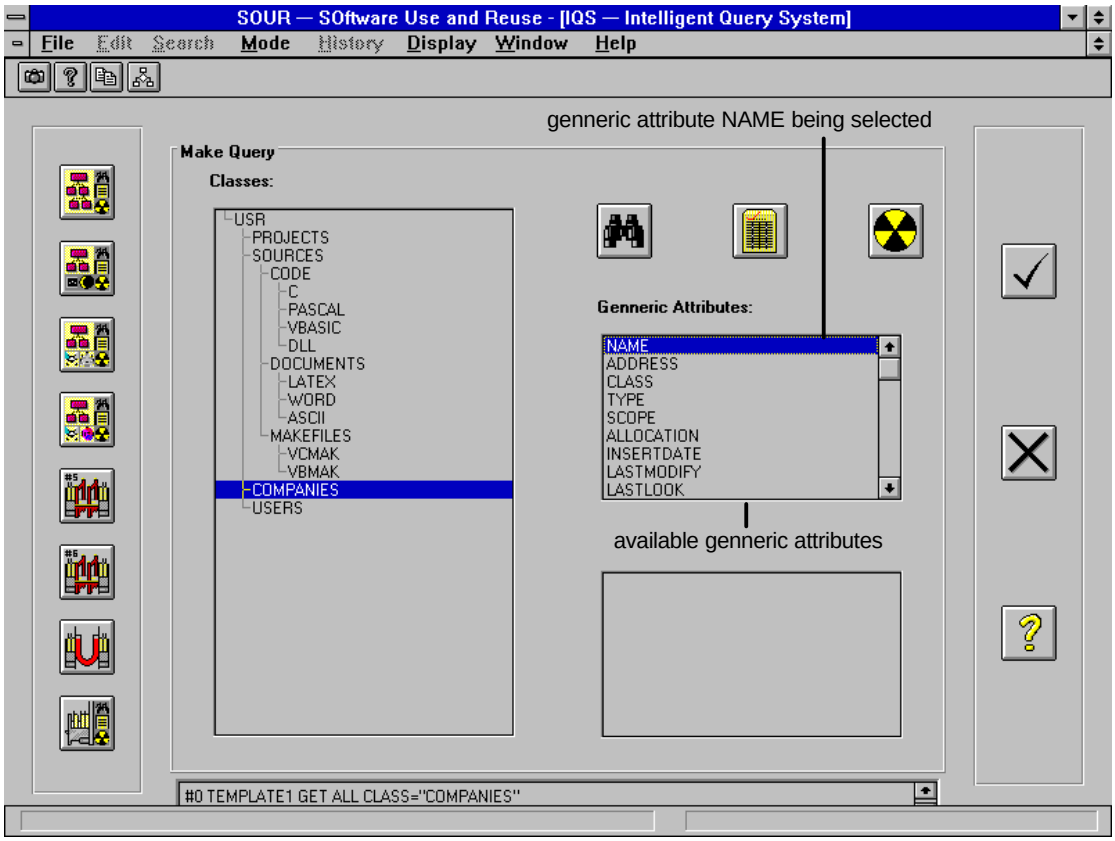


Screenshoot 1: starting refinement of the class `COMPANIES` objects by *generic attributes*.

Having choosen the refinement through *generic attributes*, one can proceed by specifying the name of a *generic attribute* and then the respective value (this is also valid for *user defined class attributes* or *facets*).

[2]refer to section ?? for more details.

Screenshoot 1.1 shows the selection of the *generic attribute* `NAME` from a list-box (**Generic Attributes**) of all the *generic attribute* names available to choose. That list was only made visible after having pressed the *generic attribute's* button, which, at any time, is only enabled if there are *generic attributes* remaining to choose (repeated filtering by the same attribute or facet is not allowed[3]). This is also true for the *user defined class attributes* button and *facets* button (in fact, it is perfectly legal to interrupt the refinment by a certain attribute or facet by pressing a button concerning other attribute or facet).

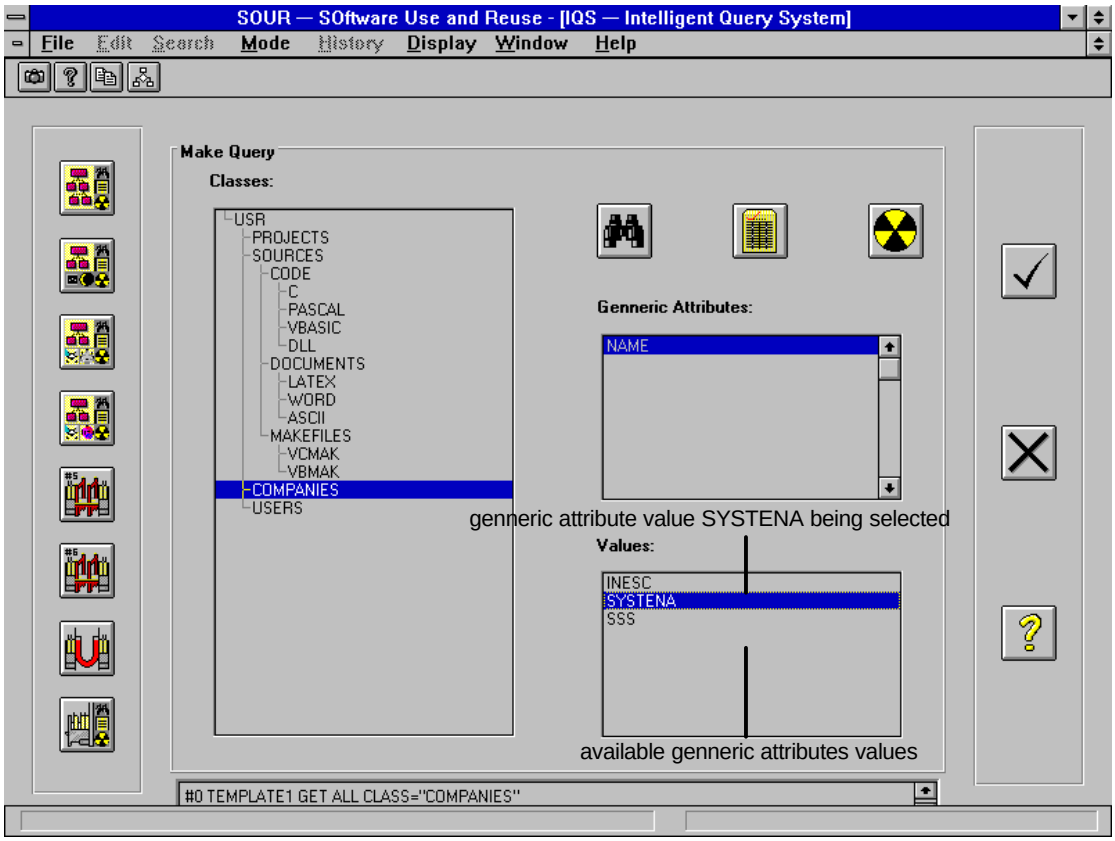


Screenshot 1.1: Choosing the *generic attribute* `NAME`.

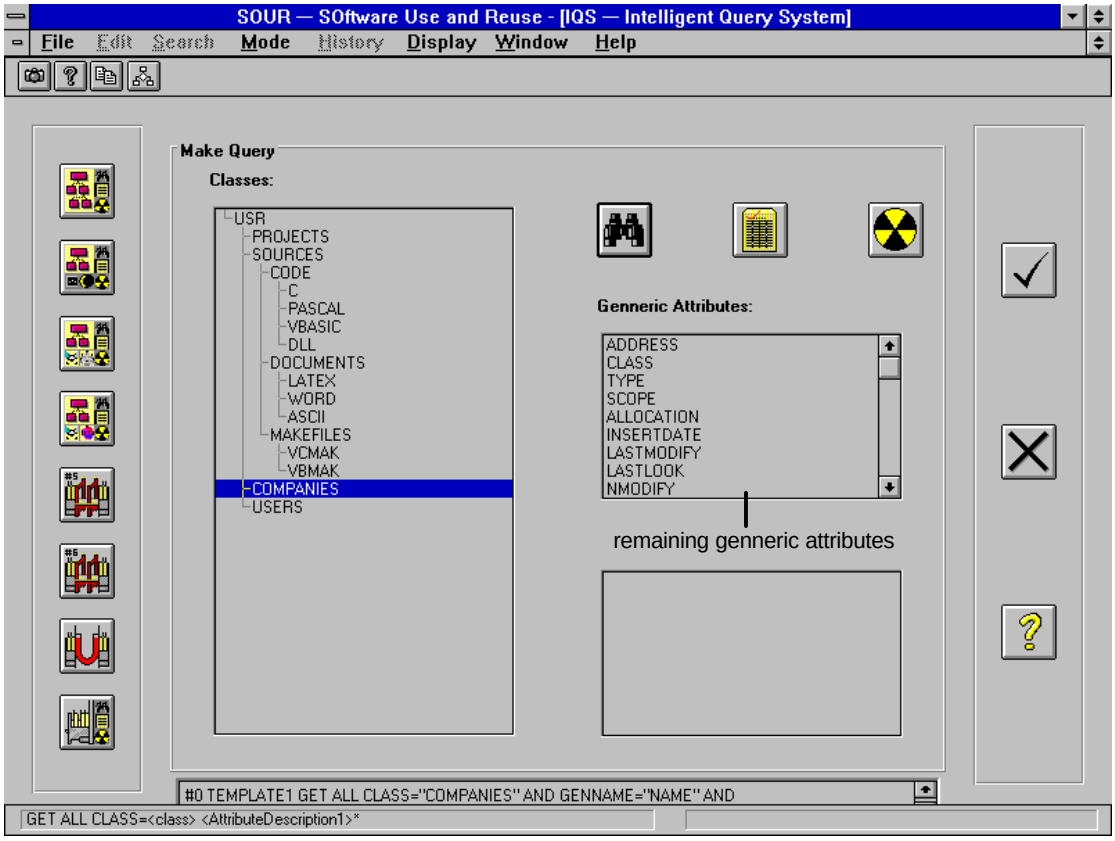
Having selected a name from the **Generic Attributes** list box, the union of all the values relative to that name can be found at the **Values** list box. The selection of a certain value from this list box will make the actual solution to keep only the objects associated with that value, for the name previously choosen. **Screenshoot 1.2** shows this situation for the particular case of the value `SYSTENA` of the *generic attribute* `NAME`.

Screenshoot 1.3 shows the remaining *generic attribute* names, after having completed the filtering through the *generic attribute* `NAME`.

[3]recall the subject of avoiding redundancy at section ?? of the IQS Functional Specification Manual.



Screenshot 1.2: Choosing the value `SYSTENA` for the *generic attribute* `NAME`.

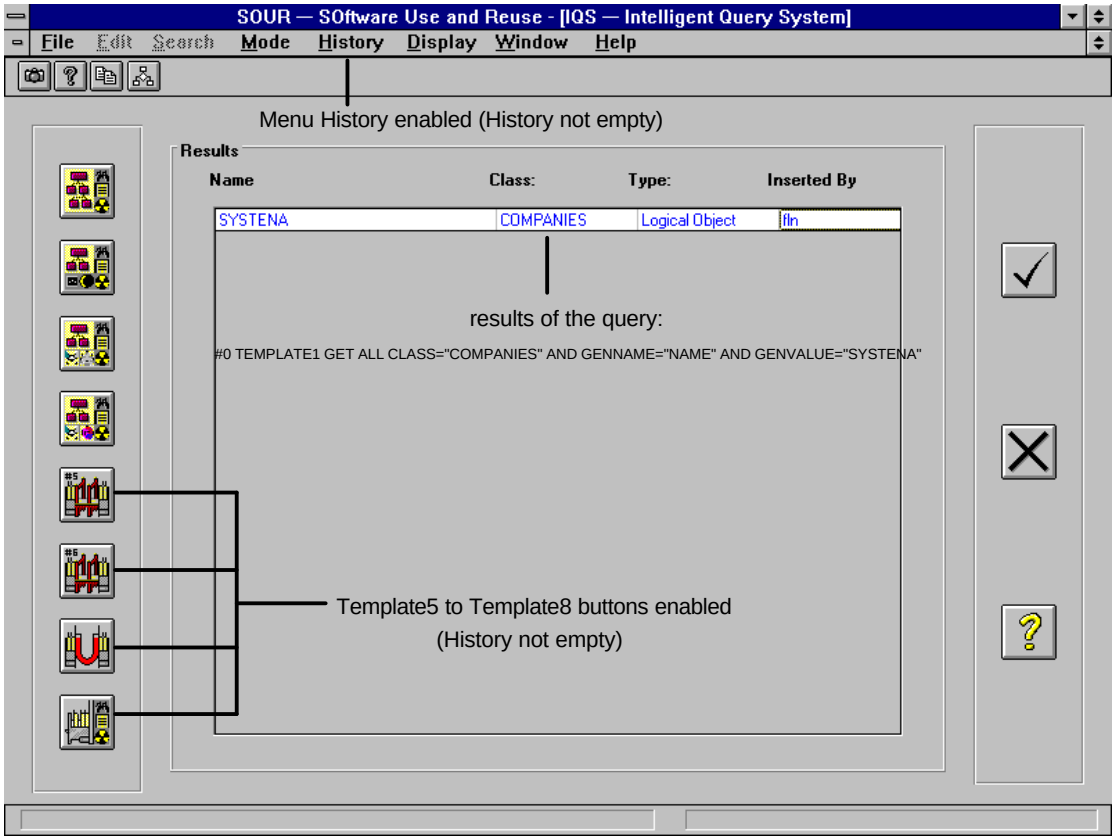


Screenshot 1.3: Remaining *generic attributes*.

Filtering could now proceed by chosing another *generic attribute* or *user defined class attribute* or *facet*, and the respective value. If one decided to terminate the query (by pressing the `CHECK` button), a table with some basic information about the objects found would be presented, as depicted in **Screenshot 1.4**.

At that state, it is also possible (and most of the times, desirable) to call the Result Manager in order to show in a graphic way the objects and some specific information related (the procedure to do that is the same as the one for a temporary solution or for any query kept by the History; this facilitie will be explored at section ??).

Note also that it is already possible to make a *non-Kernel* query because once the previous query terminated successfully, the History is no longer empty.



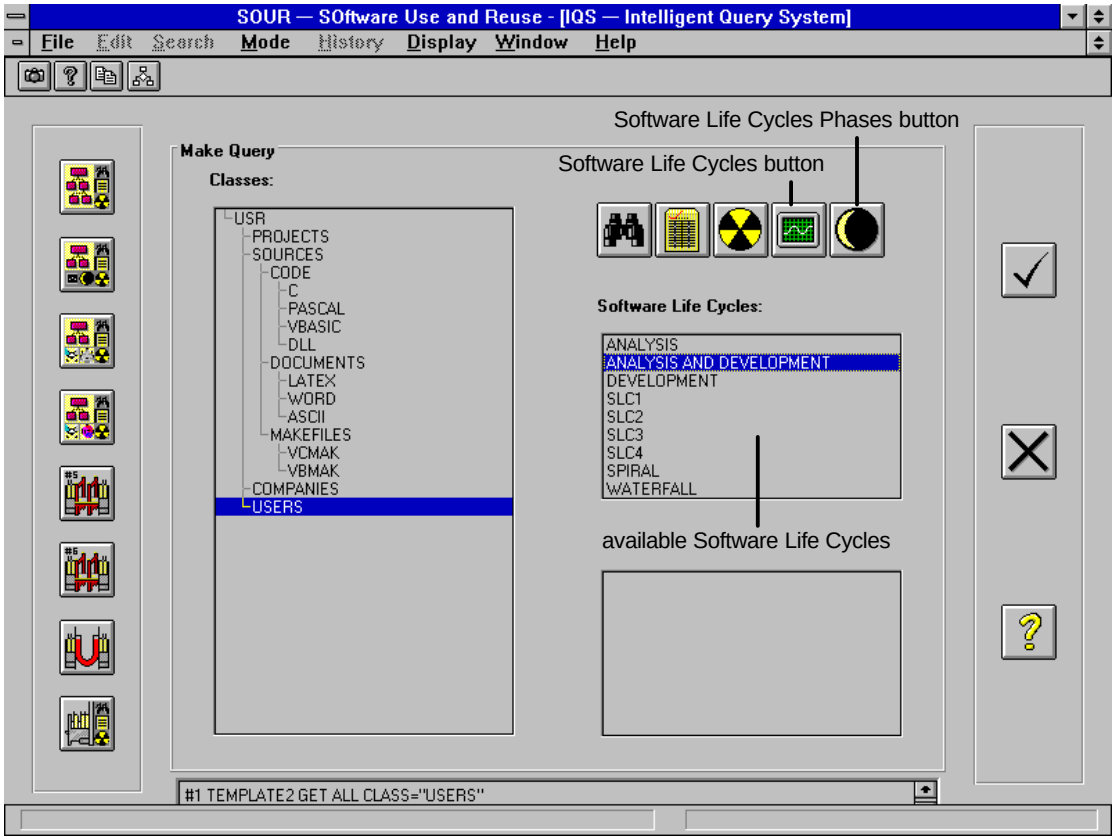
Screenshot 1.4: Results table.

Template2 demo

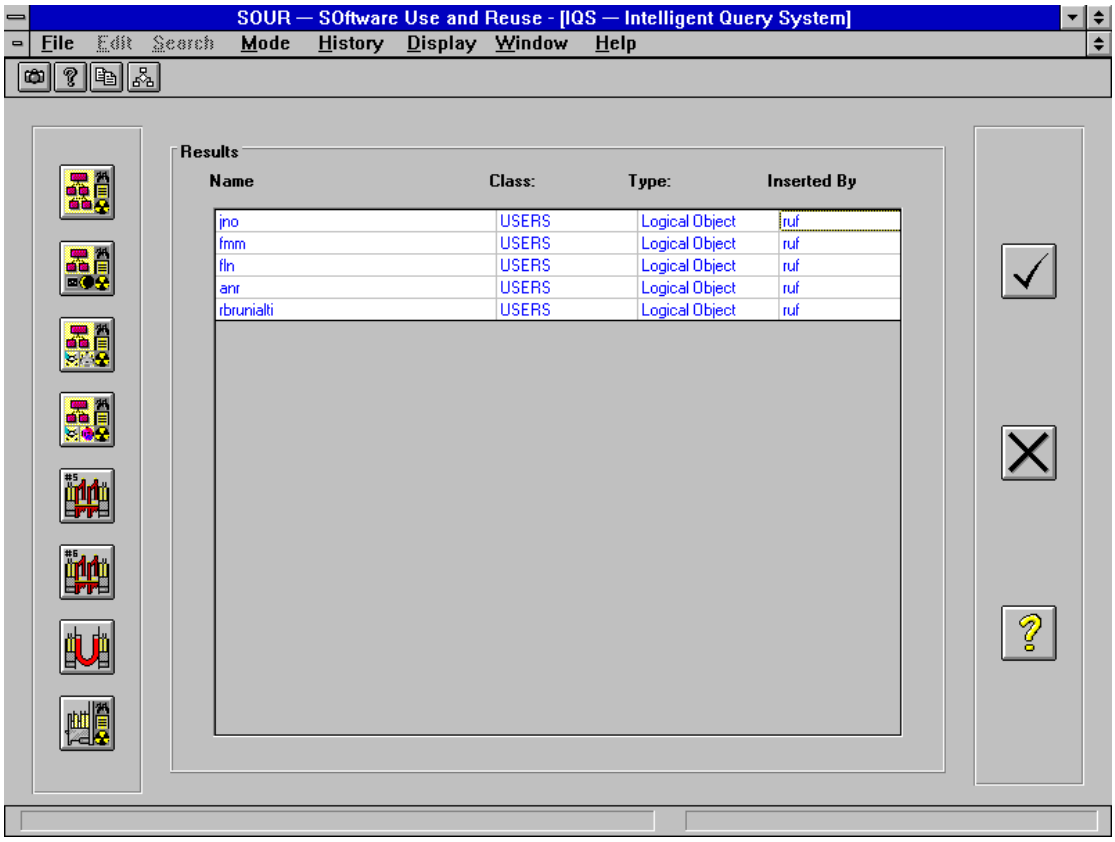
Similarly to *Template1*, *Template2* queries also start by choosing a user class from the USR class hierarchy, but this time just after having pressed the *Template2* button. The refinement of the resulting objects will be based not only in *generic attributes*, *user defined class attributes* or *facets*, but also in the specification of a *software life cycle* or one or more *software life cycle phases*.

Note that because the ERA layout does not provide for the association of an object with a certain *software life cycle phase*, what really happens when choosing a *phase* is the implicit refinement of the present query solution by the *cycle(s)* containing that *phase*. Also, if one decides to refine the objects by directly choosing a *software life cycle*, then no future refinement by *software life cycle* or *software life cycle phase(s)* should be allowed, once the ERA schema also prevents an object from having more than one *software life cycle* associated.

Screenshoot 2.1 presents the *Template2* front-end concerning the interactive solving of the query `#1 TEMPLATE2 GET ALL CLASS="USERS" AND SLCNAME="ANALYSIS AND DEVELOPMENT"`, intended to retrieve the objects belonging to the `USERS` class and associated with the *software life cycle* `ANALYSIS AND DEVELOPMENT`. **Screenshoot 2.2** shows those objects.



Screenshot 2.1: Refining by the `ANALYSIS AND DEVELOPMENT` *software life cycle*



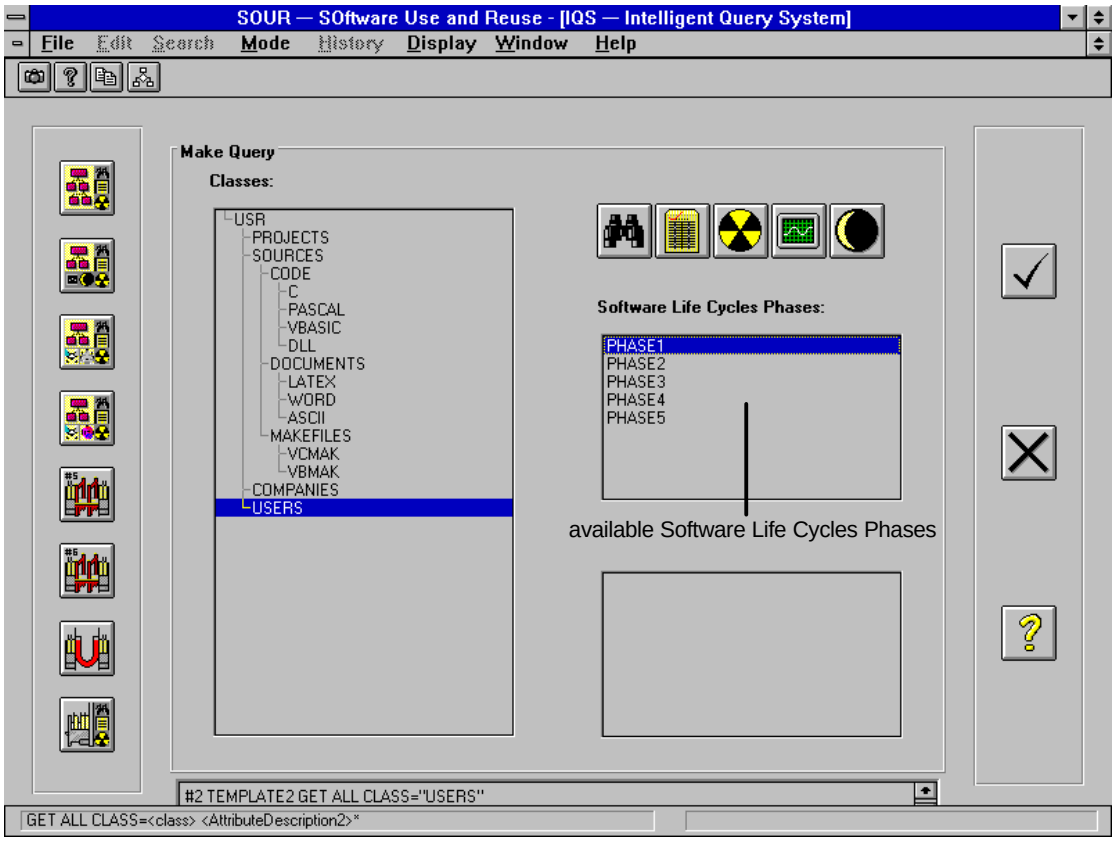
Screenshot 2.2: final results of the query `#1 TEMPLATE2 GET ALL CLASS="USERS" AND SLCNAME="ANALYSIS AND DEVELOPMENT".`

An alternative way of getting exactly the same results as the previous query would be to select the objects by choosing one or more *software life cycle phase(s)*, these phases exclusively belonging to the *cycle* `ANALYSIS AND DEVELOPMENT`. Screenshoots 2.3 to 2.? illustrate this procedure.

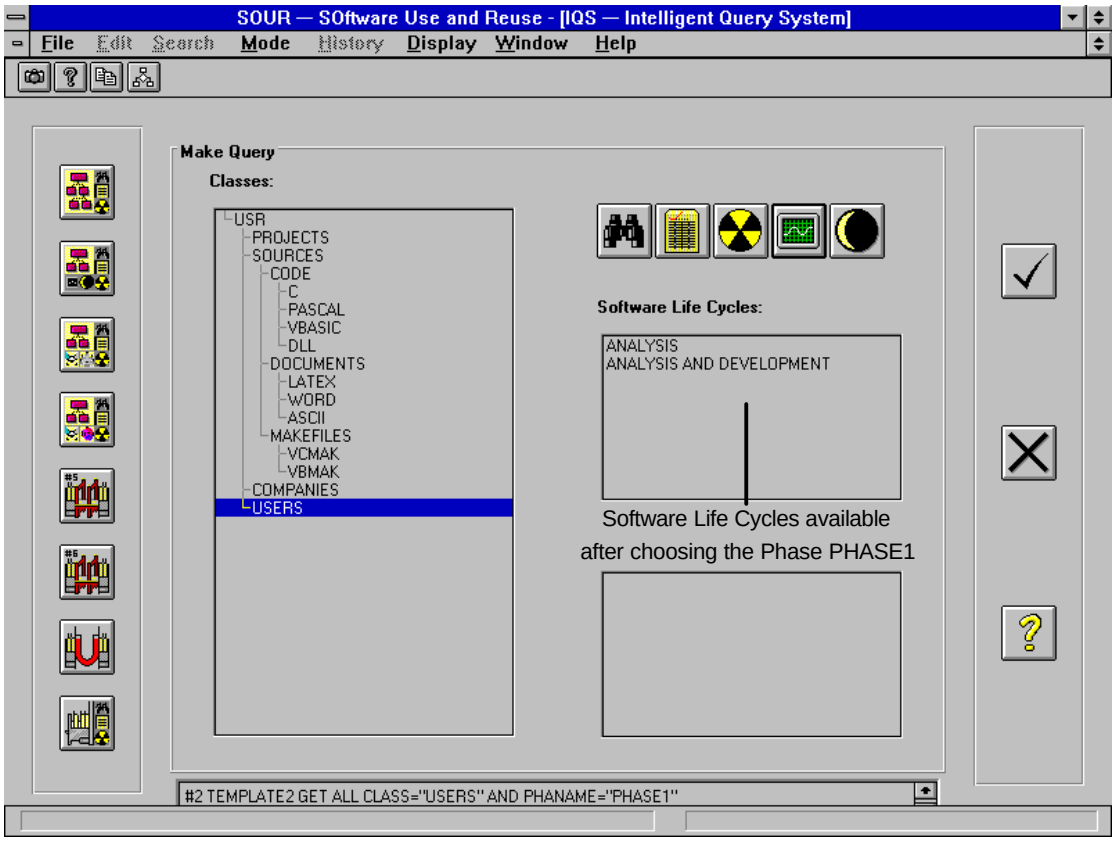
Screenshoot 2.3 shows the choosing of the *software life cycle phase* `PHASE1` from the union of all the *software life cycle phases* belonging to all the *software life cycles* remaining.

By choosing a particular *software life cycle phase*, one can expect to keep in the query solution only the objects for whom there are *software life cycles* associated and containing that *phase*. In this context, it is logical to preserve in the *software life cycles* list only the *cycles* containing that *phase*, as well it is also logical to keep in the *software life cycle phases* list only the remaining *phases* (note that the *phase* just choosen will not be considered anymore), contained by those *cycles*.

Screenshoot 2.4 shows that, after the **Screenshoot 2.3** procedure, the only *software life cycles* that contained the *phase* `PHASE1` were `ANALYSIS` and `ANALYSIS AND DEVELOPMENT`.



Screenshot 2.3: filtering by the *software life cycle phase* `PHASE1`.



Screenshot 2.4: available *software life cycles* after refining by `PHASE1`

Screenshot 2.5: refining by `PHASE3`, after refining by `PHASE1`

At this point, it should be obvious that filtering by a certain *software life cycle phase* is the only way to choose objects simultaneously belonging to more than one *software life cycle*, these *cycles* being the ones containing that *phase*. However, if theres only one *software life cycle* containing the *phase* just choosen, then, this is equivalent to explicitly select that *cycle*. This is what happens at **Screenshoot 2.5**, were the *phase* `PHASE3` is choosen (among the phases remaining from **Screenshoot 2.3**) and actually only the cycle `ANALYSIS AND DEVELOPMENT` contains that phase.

Althought not presented here, the solution to the query

```
#2 TEMPLATE2 GET ALL CLASS="USERS" AND PHANAME="PHASE1" AND
PHANAME.="PHASE3"
```

is the same as to the query

```
#1 TEMPLATE2 GET ALL CLASS="USERS" AND SLCNAME="ANALYSIS AND
DEVELOPMENT".
```

presented at **Screenshoot 2.2**.