

Project: DRA FRONT END FILTER PROJECT

Title: Errors in the Specifications

Ref: DS/FMU/FEF/002 *Issue: Revision : 1.9* *Date:* 5 December 2009

Status: Informal *Type:* Technical Note

Keywords:

Author:

<i>Name</i>	<i>Location</i>	<i>Signature</i>	<i>Date</i>
G. M. Prout	WIN01		

Authorisation for Issue:

<i>Name</i>	<i>Function</i>	<i>Signature</i>	<i>Date</i>
R.B. Jones	HAT Manager		

Abstract: A record of errors found in the FEF specifications, and changes made from Annex 2 of the ITT for the DRA front end filter project RSRE 1C/6130.

Distribution: HAT FEF File

0 DOCUMENT CONTROL

0.1 Contents List

0	DOCUMENT CONTROL	2
0.1	Contents List	2
0.2	Document Cross References	3
0.3	Changes History	3
0.4	Changes Forecast	3
1	GENERAL	4
1.1	Scope	4
1.2	Introduction	4
2	ERRORS IN SPECIFICATION OF 28/2/92	4
3	ERRORS IN SPECIFICATION OF 18/3/92	5
4	ADDITIONS AND CHANGES TO ANNEX 2	5
4.1	Specification of 28/2/92	5
4.2	Specification of 18/3/92	6
5	FURTHER ISSUES	6
6	PHASE TWO SPECIFICATION PROBLEMS	6
6.1	Output Filter	6
6.1.1	Definitions	6
6.1.2	Select Query Filter	6
6.2	Transformations	6
6.2.1	<i>SELECT</i> s in <i>FROM</i> -list	6
6.2.2	Column Existence	7
6.2.3	<i>SELECT</i> s in Expressions	7
6.2.4	Directory and Table Classes	7
6.2.5	Ordering of Output Tables	7
6.2.6	<i>simplify_{ands}</i> and <i>simplify_{ors}</i>	7
7	PROBLEMS DETECTED DURING PHASE 2 PROOFS	7
7.1	<i>HideDerTableData</i>	8
7.2	<i>Where</i>	8
7.3	<i>Group</i>	8
7.4	<i>Alltuples</i>	8
7.5	<i>ExistsTuples/SingleValue</i>	9
7.6	<i>SetFuncAllAnd/SetFuncAllOr/BinOpAnd/BinOpOr</i>	9
8	PHASE 2 ISSUES	9
8.1	Ordering of Tables	9
8.2	Modelling Issues	9

0.2 Document Cross References

- [1] DS/FMU/017. *Secure Database Technical Proposal*. High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.
- [2] DS/FMU/FEF/026. *Critical Requirements on the SWORD Query Transformations*. R.D. Arthan, ICL Secure Systems, WIN01.
- [3] DS/FMU/FEF/032. *Table Computations for SWORD*. R.D. Arthan, ICL Secure Systems, WIN01.
- [4] DS/FMU/FEF/034. *Phase II Proof Strategy*. R.D. Arthan, ICL Secure Systems, WIN01.
- [5] *Invitation to Tender RSRE 1c/6130*. DRA, December 1991.

0.3 Changes History

0.4 Changes Forecast

None

1 GENERAL

1.1 Scope

This document provides a record of the errors found in the SSQL specifications provided by DRA. It also indicates alterations made from the original specification.

1.2 Introduction

The basis of contract RSRE 1C/6130 is the Secure Database Technical Proposal, [1]. The SSQL specifications referred to in that document are those of the Annexes to the ITT, [5]. Since ICL's bid was accepted, we have been supplied with two new versions of specifications of secure SSQL. This document outlines errors found in these two new specifications. This document also indicates changes made from the ITT specifications, including substantial additions to those original specifications.

2 ERRORS IN SPECIFICATION OF 28/2/92

Page	Description
8,9	<i>col_name</i> in all three <i>Set_clause</i> functions should be <i>Col_name</i> .
25	<i>nullValue</i> missing from error list.
26	<i>Val</i> may be of type <i>Code</i> .
27	<i>Float</i> already used as an identifier.
28	Spurious reference to column existence classification.
28	<i>TableSpec</i> has spurious argument <i>Class</i> .
30	<i>_lub_</i> for 'worths' should be of type <i>Worth</i> \rightarrow <i>Worth</i> \rightarrow <i>Worth</i> .
33	Type error in second definition of <i>_**_</i> (<i>a</i> is not a list; should be nil).
34	<i>projectTuples</i> should take a <i>Table_spec</i> not a <i>TableSpec</i> . Brackets missing around <i>visible spec</i> .
37	All references to <i>Action</i> should be to <i>Query</i> , except <i>action</i> and <i>guaranteed_action</i> which should be removed. Missing arguments from <i>update</i> .
38	<i>Action</i> no longer a syntactic function.
39	<i>set_func_all</i> and <i>set_func_distinct</i> call <i>Value</i> which has missing <i>v</i> argument.
41	Error in description of <i>Table_spec</i> .
45,46	In <i>Query delete</i> and <i>Query update</i> , argument to <i>changeSpec</i> has incorrect type.

Table 1: Errors in 28/2/92 spec

3 ERRORS IN SPECIFICATION OF 18/3/92

Page	Description
7	Constructor <i>vase</i> should be <i>case</i> .
7	<i>row_existence</i> and <i>joined_row_existence</i> missing from list of constructors.
10	<i>col_name</i> in all three <i>Set_clause</i> functions should be <i>Col_name</i> .
27	<i>notGuaranteed</i> no longer relevant. No errors for when a user tries to access a table whose classification is not dominated by his clearance.
30	<i>Tab</i> is used in definition of <i>Reference</i> although it has not yet been defined.
33	<i>_lub_</i> for 'worths' should be of type <i>Worth</i> \rightarrow <i>Worth</i> \rightarrow <i>Worth</i> .
35	Type error in second definition of <i>_**_</i> (<i>a</i> is not a list; should be nil).
36	<i>projectTuples</i> should take a <i>Table_spec</i> not a <i>TableSpec</i> . Brackets missing around <i>visible spec</i> .
37	All references to <i>Action</i> should be to <i>Query</i> , except <i>action</i> and <i>guaranteed_action</i> which should be removed. Missing arguments from <i>update</i> .
38	<i>Action</i> no longer a syntactic function.
39	<i>set_func_all</i> and <i>set_func_distinct</i> call <i>Value</i> which has missing <i>v</i> argument.
41	Error in description of <i>Table_spec</i> .
45,46	In <i>Query delete</i> and <i>Query update</i> , argument to <i>changeSpec</i> has incorrect type.
51	The description of <i>Query update</i> implies that classification of a field can only be raised.

Table 2: Errors in 18/3/92 spec

4 ADDITIONS AND CHANGES TO ANNEX 2

4.1 Specification of 28/2/92

1. *Action* is no longer a sort.
2. Constructor *let* added; constructors *action* and *guaranteed_action* deleted.
3. Constructor *update* has extra clauses *groupby* and *having*.
4. Operators *definitely* and *possibly* added.
5. Errors *noSuchTable*, *noSuchDirectory*, *ambiguousEvaluate*, *ambiguousHaving* and *nullValue* have been added.
6. *tooTall* changed
7. *GroupedResult*, *Maybe* and *MaybeResult* added.
8. *make_data* now classified at user's clearance rather than bottom
9. Definition of *lookup* in place.
10. *check_where_complete* added.

11. *resultBool* and *resultClass* changed.
12. *engroup* and *eliminate* added (because of 'groupby' and 'having').
13. Significant changes made to syntactic clauses and functions because of new/different constructors and because of identified 'holes' in Annex 2 to the ITT [5] being plugged.

4.2 Specification of 18/3/92

Modifications on 28/2/92 version are marked by sidelines.

5 FURTHER ISSUES

There is a problem if a user tries to access a table whose classification is not dominated by his clearance; suggest an *accessDenied* error message.

It is not secure for a user to delete rows whose existence class is below his clearance. *deleteQuery* now insists that the user clearance be the same as the existence class.

6 PHASE TWO SPECIFICATION PROBLEMS

Many minor errors reported in the transformation specs of 06/01/93 (see file of replaced specs).

The following give rise to possible security violations:

6.1 Output Filter

Output filter specs faxed on 21st January incorrect.

6.1.1 Definitions

Function *filter_where* should discard rows where boolean is true.

6.1.2 Select Query Filter

Check query which determines whether or not a SELECT should proceed has been omitted.

6.2 Transformations

6.2.1 SELECTs in FROM-list

Insecurity reported on 16th February. Problem with

SSQL query

```
| SELECT * FROM
|     SELECT X FROM Y GROUPBY Z
```

With an outer select, if user is not cleared to see everything in the *GROUP BY*-column, then an additional TSQL check query is generated. Not the case with an inner select. *tuple_list_{make_outer}* checks clearance, *tuple_list_{make}* does not.

6.2.2 Column Existence

Column existence class 'forgotten' in transformations.

6.2.3 SELECTs in Expressions

Neither the checks on the groupby columns, nor the checks on the class of the where clause are made in the places where selects appear within an expression (EXISTS and single-value). This gives rise to the same sort of insecurities as with missing the *GROUP BY* checks for a *SELECT* in a *FROM*-list (see above).

6.2.4 Directory and Table Classes

These are not mentioned in the query transformations.

6.2.5 Ordering of Output Tables

This is not specified and is a possible source of insecurity if output tables are not put in some fixed order not dependent on information the client is not cleared to see.

6.2.6 *simplify_{ands}* and *simplify_{ors}*

The algorithms of *simplify_{ands}* and *simplify_{ors}* give rise to insecurities. In *simplify_{ands}*, if the expression evaluates to *true*, then the client knows that all the operands are *true* and the class is the l.u.b. of the classes of the operands, whereas if the expression evaluates to *false* then the client knows that some operand is *false* and the class is taken to be the g.l.b. of the classes of the operands which are *false*. Consider the case where the classes of the false operands are incomparable - the g.l.b is bottom. Replaced by l.u.b.s. Similar problem (and solution) for *simplify_{ors}*.

7 PROBLEMS DETECTED DURING PHASE 2 PROOFS

During the Phase 2 proof stage it was discovered necessary to make minor amendments to the *HideDerTableData* operation in version 1.16 of [2] and the value and table computation specifications in version 1.13 of [3]. (The versions mentioned are those of the drafts delivered to DRA just before the phase 2 proof work was begun in anger.)

7.1 *HideDerTableData*

Originally, *HideDerTableData* in [2] threw away not only rows whose existence the client is not cleared to know but also rows where the class of the *Where* clause is not dominated by the clearance of the user. This was intended to model not only the phase 1 hiding operation on the table, but also, mistakenly, the action of the output filter. However, the *Where* operation only throws away rows whose existence the client is not cleared to know (so that the output filter can determine whether or not a *MayNotBeComplete* message is appropriate). *HideDerTableData* has been modified to retain rows for which the user is not cleared to evaluate the *WHERE* clause. This gives stronger inductive hypotheses for the proofs about the value and table computations of [3], from which the desired facts about the results of the output filtering still follow.

This change is essentially concerned with the internal details of the formalisation and has no direct impact for the security of the system.

7.2 *Where*

The specification of *Where* in [3] was changed (a) so that rows for which the client is not cleared to evaluate the *WHERE* clause are retained and (b) so that rows whose existence the user is not cleared to know are removed at the outset. (a) was just an oversight in transcribing the effect of *tuple_listmake_outer*. (b) was found necessary to ensure that aggregate functions such as *COUNT(*)* inside the *WHERE* clause itself are not a source of covert channels. E.g. the output produced by a query such as:

SSQL query

```
|SELECT * FROM X WHERE COUNT(*) = 1
```

should only depend on rows whose existence the client is cleared to know.

7.3 *Group*

The specification of *Group* in [3] was changed so that the clearance returned does not depend on the result of the grouping if the client is not cleared to know the information required to partition the table into groups. If this is not done then the clearance is itself a possible covert channel. This covert channel would not arise in practice in an implementation which signals possible insecurity in inner selects by raising exceptions.

7.4 *Alltuples*

The specification of *Alltuples* in [3] was changed so that the clearance returned does not depend on the result of *Group* function if the client is not cleared to compute the operands of the *Alltuples* operation.

As with the change to *Group*, if this is not done then the clearance is itself a possible covert channel; moreover, the covert channel would not arise in practice in an implementation which signals possible insecurity in inner selects by raising exceptions.

7.5 *ExistsTuples/Single Value*

As formalised in version 1.13 of [3], the clearance returned by these value forms would itself be a possible covert channel. They have been modified to model more closely an implementation which detects immediately the situation where the client is not cleared to evaluate the operand table.

7.6 *SetFuncAllAnd/SetFuncAllOr/BinOpAnd/BinOpOr*

In version 1.13 of [3], the logical operators were formulated so that *AND*, for example, would return the l.u.b. of the clearances associated with *false* operands if there were any such operands. This make the clearance returned a possible covert channel. The *AND* operation is now defined to return the l.u.b. of the clearances associated with the *false* operands which the client is cleared to see (if there are any). This is secure and permits the desired use of forms such as:

SSQL Example

```
|      my_clearance DOM CLASS OF x AND x > 0
```

(In fact, the current formulation would still be secure, no matter what fixed means of combining the clearances were used instead of l.u.b., since only clearances associated with *false* value the client is cleared to know are combined.)

8 PHASE 2 ISSUES

8.1 Ordering of Tables

The formulation of [4] uses an ordering on the tables computed which is independent of the values held in the columns of the tables and so the ordering is not a potential covert channel. If in the actual implementation data-dependent ordering criteria are used then the system will be insecure unless steps are taken to ensure that the client is cleared to know the information required to order the table.

8.2 Modelling Issues

See section 3.5 of [4] for a discussion of modelling assumptions made in the phase 2 formal treatment.