

*Project:* DRA FRONT END FILTER PROJECT

*Title:* Proof of Security (IIe)

*Ref:* DS/FMU/FEF/015

*Issue: Revision : 2.5*

*Date:* 5 December 2009

*Status:* Approved

*Type:* Specification

*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:* This document completes the formal proof of security for Phase 1 of the DRA front end filter project RSRE 1C/6130.

*Distribution:* HAT FEF File  
Simon Wiseman

## 0 DOCUMENT CONTROL

### 0.1 Contents List

<b>0</b>	<b>DOCUMENT CONTROL</b>	<b>2</b>
0.1	Contents List . . . . .	2
0.2	Document Cross References . . . . .	2
0.3	Changes History . . . . .	3
0.4	Changes Forecast . . . . .	3
<b>1</b>	<b>GENERAL</b>	<b>4</b>
1.1	Scope . . . . .	4
1.2	Introduction . . . . .	4
<b>2</b>	<b>PRELIMINARIES</b>	<b>4</b>
<b>3</b>	<i>Lemma1</i>	<b>5</b>
3.1	Relationship between <i>updateState</i> and <i>updateStateR</i> . . . . .	5
3.2	Proof that the Invariant on the State is Maintained . . . . .	7
3.2.1	<i>insertQuery</i> Lemma . . . . .	7
3.2.2	<i>deleteQuery</i> Lemma . . . . .	9
3.2.3	Update Lemmas . . . . .	11
3.2.4	<i>updateQuery</i> Lemma . . . . .	16
3.2.5	State Invariant Lemma . . . . .	19
3.3	Proof of <i>Lemma1</i> . . . . .	20
<b>4</b>	<b>PROOF OF SECURITY OF SSQL</b>	<b>20</b>
<b>5</b>	<b>CLOSING DOWN</b>	<b>20</b>
<b>6</b>	<b>THE THEORY fef015</b>	<b>21</b>
6.1	Parents . . . . .	21
6.2	Theorems . . . . .	21
<b>7</b>	<b>INDEX</b>	<b>23</b>

### 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/007. *Proof Strategy*. G.M. Prout, ICL Secure Systems, WIN01.
- [3] DS/FMU/FEF/009. *Proof of Security (I)*. G.M. Prout, ICL Secure Systems, WIN01.
- [4] DS/FMU/FEF/010. *Proof of Security (IIa)*. G.M. Prout, ICL Secure Systems, WIN01.
- [5] DS/FMU/FEF/011. *Proof of Security (IIb)*. G.M. Prout, ICL Secure Systems, WIN01.
- [6] DS/FMU/FEF/012. *Proof of Security (IIc)*. G.M. Prout, ICL Secure Systems, WIN01.
- [7] DS/FMU/FEF/013. *Proof of Security (IId)*. G.M. Prout, ICL Secure Systems, WIN01.

### 0.3 Changes History

**Issue** *Revision : 2.5 (5 December 2009)* Corrected L<sup>A</sup>T<sub>E</sub>X error.

**Issue 2.5** Removed dependency on ICL logo font

### 0.4 Changes Forecast

None.

# 1 GENERAL

## 1.1 Scope

This document completes the formal Phase 1 proof that the behaviour of the SSQL abstract machine is secure. It constitutes part of deliverable D6 of work package 1c, as given in section 7 of the Secure Database Technical Proposal, [1].

## 1.2 Introduction

This document is a proof script which provides a formal proof that the behaviour of the SSQL abstract machine is secure.

In the proof strategy document, [2], we state the *main\_thm*:

HOL output

```
|main_thm = ⊢ Lemma1 ∧ Lemma3 ∧ Lemma4 ∧ Lemma5 ⇒ behaviours SSQLam ∈ secure
```

This result is progressed by the proofs of *Lemma3*, *Lemma4* and *Lemma5* in the unwinding proof document [3] to give *main\_thm1*:

HOL output

```
|main_thm1 = ⊢ Lemma1 ⇒ behaviours SSQLam ∈ secure
```

*Lemma1* is the requirement on the critical components *hide* and *updateState*.

### Lemma1

```
| ⊢ hide ∈ secureHide ∧ (hide,updateState) ∈ secureUpdate
```

In [4], a formal proof of the first conjunct of *Lemma1* is given. In [5], formal proofs of the third and fourth conjuncts of the second conjunct of *Lemma1* are given. In [6] and [7], formal proofs of the first and second conjuncts of the second conjunct of *Lemma1* are given for *hideR* and *updateStateR*. In this document, we prove *Lemma1* by using the results from [4], [5], [6] and [7] together with a formal proof that *updateStateR* maintains the invariant on the representation state.

# 2 PRELIMINARIES

The following ProofPower instructions set up the new theory *fef015*.

SML

```
|open_theory "fef013";
| (force_delete_theory "fef015" handle _ => ());
|new_theory "fef015";
|push_merge_pcs["hol","wrk049","wrk049a","'pair1 "];
```

### 3 Lemma1

#### 3.1 Relationship between *updateState* and *updateStateR*

We first prove that if *updateStateR* maintains the invariant on the representation state, then *hide* and *updateState* satisfy the relation *secureUpdate*.

SML

```

| push_goal([],
|    $\ulcorner (\forall \text{ clear } s \ u \bullet \text{isState } s \Rightarrow \text{isState}(\text{Fst}(\text{updateStateR}(\text{clear}, u, s))))$ 
|    $\Rightarrow (\text{hide}, \text{updateState}) \in \text{secureUpdate}^\ulcorner$ ;
| a(REPEAT strip_tac THEN rewrite_tac[conjunct4, secureUpdate_def, hide_def]);
| a(LEMMA_T $\ulcorner (\forall c \ s_1 \ s_2 \ e$ 
|   •  $\text{absState}(\text{hideR}(c, \text{repState } s_1))$ 
|      $= \text{absState}(\text{hideR}(c, \text{repState } s_2))$ 
|      $\Rightarrow \text{Snd}(\text{updateState}(c, e, s_1)) = \text{Snd}(\text{updateState}(c, e, s_2))$ ) $\ulcorner$ 
|   rewrite_thm_tac);

```

SML

```

| (* *** Goal "1" *** *)
| a(REPEAT strip_tac);
| a(strip_asm_tac( $\forall\text{-elim}^\ulcorner s_1^\ulcorner \text{isState\_lemma}$ ));
| a(strip_asm_tac( $\forall\text{-elim}^\ulcorner s_2^\ulcorner \text{isState\_lemma}$ ));
| a(bc_tac[conjunct3, eq_sym_rule(all_ $\forall\text{-elim}$  hide_eq_lemma)]);
| a(fc_tac[hideR_lemma]);
| a(spec_nth_asm_tac 2  $\ulcorner c^\ulcorner$ );
| a(spec_nth_asm_tac 2  $\ulcorner c^\ulcorner$ );
| a(strip_asm_tac(list_ $\forall\text{-elim}^\ulcorner \text{hideR}(c, \text{repState } s_1)^\ulcorner, \ulcorner \text{hideR}(c, \text{repState } s_2)^\ulcorner \text{isState\_lemma2}$ ));

```

SML

```

| (* *** Goal "2" *** *)
| a(LEMMA_T $\ulcorner \forall c_1 \ c_2 \ s_1 \ s_2 \ e$ 
|   •  $\text{absState}(\text{hideR}(c_1, \text{repState } s_1))$ 
|      $= \text{absState}(\text{hideR}(c_1, \text{repState } s_2))$ 
|      $\wedge c_1 \text{ dominates } c_2$ 
|      $\Rightarrow \text{absState}$ 
|        $(\text{hideR}(c_1, \text{repState}(\text{Fst}(\text{updateState}(c_2, e, s_1))))))$ 
|      $= \text{absState}$ 
|        $(\text{hideR}(c_1, \text{repState}(\text{Fst}(\text{updateState}(c_2, e, s_2))))))$ ) $\ulcorner$ 
|   rewrite_thm_tac);

```

SML

```

(* *** Goal "2.1" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elimΓ s2⊃isState_lemma));
a(strip_asm_tac(∀_elimΓ s1⊃isState_lemma));
a(fc_tac[hideR_lemma]);
a(spec_nth_asm_tac 2 Γc1⊃);
a(spec_nth_asm_tac 2 Γc1⊃);
a(strip_asm_tac(list_∀_elimΓ[hideR (c1, repState s1)⊃,
  ΓhideR (c1, repState s2)⊃]isState_lemma2));
a(LEMMA_TΓe = (Fst e, Snd e)⊃ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], rewrite_tac[updateState_def]]);
a(list_spec_nth_asm_tac 10Γc2⊃,ΓrepState s1⊃,Γe⊃);
a(list_spec_nth_asm_tac 11Γc2⊃,ΓrepState s2⊃,Γe⊃);
a(fc_tac[conjunct2]);
a(list_spec_nth_asm_tac 1Γc2⊃,Γe⊃);
a(fc_tac[isState_lemma1] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.2" *** *)
a(REPEAT strip_tac);
a(strip_asm_tac(∀_elimΓ s⊃isState_lemma));
a(fc_tac[hideR_lemma]);
a(spec_nth_asm_tac 1 Γc2⊃);
a(DROP_NTH_ASM_T 4 ante_tac);
a(LEMMA_TΓe = (Fst e, Snd e)⊃ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[], rewrite_tac[updateState_def]]);
a(asm_fc_tac[]);
a(list_spec_nth_asm_tac 2Γc1⊃,Γe⊃);
a(fc_tac[hideR_lemma]);
a(spec_nth_asm_tac 1 Γc2⊃);
a(fc_tac[isState_lemma1] THEN asm_rewrite_tac[]);
a(⇒_tac THEN swap_nth_asm_concl_tac 1);
a(strip_asm_tac(list_∀_elimΓ[hideR (c2, repState s)⊃,
  ΓhideR (c2, Fst (updateStateR (c1, e, repState s))⊃]isState_lemma2));
a(fc_tac[conjunct1]);
val updateStateR_updateState_lemma = save_pop_thm"updateStateR_updateState_lemma";

```

HOL output

```

updateStateR_updateState_lemma =
| (∀ clear s u
| • isState s ⇒ isState (Fst (updateStateR (clear, u, s)))
| ⇒ (hide, updateState) ∈ secureUpdate

```

### 3.2 Proof that the Invariant on the State is Maintained

We now prove that the invariant on the representation state is maintained.

#### 3.2.1 *insertQuery* Lemma

SML

```

push_goal([],Γ∀ c i ns s ts • isState s ∧ tabExists c i s
  ⇒ isState(Fst(insertQuery(c,(i,ds),s,getTable i s)))Γ);
a(rewrite_tac[get_specΓisStateΓ,get_specΓStateSΓ,↔_def,∩_def,
  rewrite_rule[dom_def]tabExists_def,getTable_def]
  THEN REPEAT ∀_tac THEN strip_tac);
a(LIST_DROP_NTH_ASM_T[2,3](MAP_EVERY (fn _ => id_tac)));
a(strip_asm_tac(list_∀_elimΓsΓ,ΓFront iΓ,ΓyΓ]at_thm1));
a(rewrite_tac[insertQuery_def,changeSpec_def]);

```

SML

```

a(cases_tacΓ¬ Elms
  (Map
    (MkRow c
      o colDefaults
      c
      (Dir_tables
        (s @ Front i)
        @ Last i))
      ds)
    ⊆ RowSΓ
  THEN asm_rewrite_tac[]);
a(POP_ASM_T ante_tac THEN DROP_NTH_ASM_T 2 ante_tac
  THEN POP_ASM_T rewrite_thm_tac THEN REPEAT strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(DROP_NTH_ASM_T 5 ante_tac THEN
  rewrite_tac[↔_def,get_specΓIdeLΓ,get_specΓDirectorySΓ,∩_def,×_def,
  get_specΓUniverseΓ,rel_ext_clauses,get_specΓ$IPΓ] THEN REPEAT ⇒_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[↔_def,∩_def,⊕_single]
  THEN REPEAT strip_tac);

```

SML

```

(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_specΓMkDirectoryΓ]);
a(strip_asm_tac(list_∇_elim[ΓDir_tables yΓ,ΓLast iΓ,Γy'Γ]at_thm1));
a(DROP_NTH_ASM_T 6 ante_tac THEN POP_ASM_T rewrite_thm_tac THEN ⇒_tac);
a(DROP_NTH_ASM_T 4 ante_tac THEN
  rewrite_tac[↔_def,get_specΓIdeΓ,conv_rule(MAP_C let_conv)(get_specΓTableSpecΓ),
  ∩_def,×_def,get_specΓUniverseΓ,rel_ext_clauses,get_specΓ$PΓ]
  THEN REPEAT ⇒_tac);
a(rewrite_tac[⊕_single]THEN strip_tac THEN strip_tac THEN strip_tac);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[replaceRows_def,get_specΓMkTableSpecΓ]);
a(asm_fc_tac[] THEN REPEAT strip_tac);
a(fc_tac[∈l-∧_thm] THEN asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[∈l-elems_thm]));
a(fc_tac[⊆_def] THEN asm_fc_tac[]);

```

SML

```

(* *** Goal "1.1.2" *** *)
a(asm_fc_tac[] THEN asm_rewrite_tac[]);
(* *** Goal "1.2" *** *)
a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_specΓMkDirectoryΓ]);
a(DROP_NTH_ASM_T 2 ante_tac THEN rewrite_tac[functional_def,⊕_single]
  THEN REPEAT strip_tac THEN TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "1.3" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔_def,∩_def]));
(* *** Goal "1.4" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔_def,∩_def]));

```

SML

```

(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 4 ante_tac THEN rewrite_tac[functional_def,⊕_single]
  THEN REPEAT strip_tac THEN TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
val isState_insertQuery_lemma = save_pop_thm"isState_insertQuery_lemma";

```

HOL output

```

| isState_insertQuery_lemma =
| ⊢ ∀ c i ns s ts
|   • isState s ∧ tabExists c i s
|     ⇒ isState (Fst (insertQuery (c, (i, ds), s, getTable i s)))

```

**3.2.2 deleteQuery Lemma**

SML

```

| push_goal([], ⊢ ∀ c i ns s ts • isState s ∧ tabExists c i s
|   ⇒ isState(deleteQuery(c,(i,ns),s,getTable i s)) ⊢);
| a(rewrite_tac[get_spec ⊢ isState ⊢, get_spec ⊢ StateS ⊢, ↔_def, ⊃_def,
|   rewrite_rule[dom_def] tabExists_def, getTable_def]
|   THEN REPEAT ∀_tac THEN strip_tac);
| a(LIST_DROP_NTH_ASM_T [2,3] (MAP_EVERY (fn _ => id_tac)));
| a(strip_asm_tac(list_∀_elim[⊢ s ⊢, ⊢ Front i ⊢, ⊢ y ⊢] at_thm1));
| a(rewrite_tac[deleteQuery_def, changeSpec_def]);
| a(DROP_NTH_ASM_T 2 ante_tac THEN POP_ASM_T rewrite_thm_tac
|   THEN REPEAT strip_tac);

```

SML

```

| (* *** Goal "1" *** *)
| a(DROP_NTH_ASM_T 4 ante_tac THEN
|   rewrite_tac[↔_def, get_spec ⊢ IdeL ⊢, get_spec ⊢ DirectoryS ⊢, ⊃_def, ×_def,
|   get_spec ⊢ Universe ⊢, rel_ext_clauses, get_spec ⊢ $P ⊢] THEN REPEAT ⇒_tac);
| a(asm_fc_tac[]);
| a(POP_ASM_T ante_tac THEN rewrite_tac[↔_def, ⊃_def, ⊕_single]
|   THEN REPEAT strip_tac);

```

SML

```

| (* *** Goal "1.1" *** *)
| a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_spec ⊢ MkDirectory ⊢]);
| a(strip_asm_tac(list_∀_elim[⊢ Dir_tables y ⊢, ⊢ Last i ⊢, ⊢ y' ⊢] at_thm1));
| a(POP_ASM_T rewrite_thm_tac);
| a(DROP_NTH_ASM_T 3 ante_tac THEN
|   rewrite_tac[↔_def, get_spec ⊢ Ide ⊢, conv_rule(MAP_C let_conv)(get_spec ⊢ TableSpec ⊢),
|   ⊃_def, ×_def, get_spec ⊢ Universe ⊢, rel_ext_clauses, get_spec ⊢ $P ⊢]
|   THEN REPEAT ⇒_tac);
| a(rewrite_tac[⊕_single] THEN strip_tac THEN strip_tac THEN strip_tac);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[replaceRows_def,get_spec⌈MkTableSpec⌋]);
a(asm_fc_tac[] THEN REPEAT strip_tac);
a(fc_tac[∈l-extract_thm] THEN asm_fc_tac[]);
(* *** Goal "1.1.2" *** *)
a(asm_fc_tac[] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "1.2" *** *)
a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_spec⌈MkDirectory⌋]);
a(DROP_NTH_ASM_T 2 ante_tac THEN rewrite_tac[functional_def,⊕-single]
  THEN REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "1.3" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔-def,∩-def]));
(* *** Goal "1.4" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔-def,∩-def]));

```

SML

```

(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 3 ante_tac THEN rewrite_tac[functional_def,⊕-single]
  THEN REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
val isState_deleteQuery_lemma = save_pop_thm"isState_deleteQuery_lemma";

```

HOL output

```

isState_deleteQuery_lemma =
| ⊢ ∀ c i ns s ts
| • isState s ∧ tabExists c i s
| ⇒ isState (deleteQuery (c, (i, ns), s, getTable i s))

```

**3.2.3 Update Lemmas**

SML

```

push_goal([],Γ∀ r c tc u • r ∈ RowS ∧ isVal(updateRow c tc (u, r))
  ⇒ destVal (updateRow c tc (u, r)) ∈ RowSΓ);
a(REPEAT strip_tac);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[isVal_def]));
a(asm_rewrite_tac[destVal_def]);
a(POP_ASM_T ante_tac THEN rewrite_tac[updateRow_def]);
a(cases_tacΓ¬ u ∈ FunctionalΓ THEN
  cases_tacΓ((RelCombine u (R.data r)
    ; Graph (updateField c tc)
    ▷ {x|isError x}
    ; Graph destError
    = {}Γ
  THEN asm_rewrite_tac[¬giveError_eq_giveVal_thm,giveVal_eq_thm]);
a(⇒_T (rewrite_thm_tac o eq_sym_rule));
a(DROP_NTH_ASM_T 3 ante_tac THEN rewrite_tac[get_specΓRowSΓ
  THEN conv_tac(MAP_C let_conv)
  THEN PC_T1 "hol1"rewrite_tac[↔_def,×_def,get_specΓNumΓ,get_specΓDataSΓ,↔_def]);
a(rewrite_tac[get_specΓMkRowΓ,functional_def,⊕_thm,rel_combine_def]);
a(REPEAT strip_tac);

```

SML

```

(* *** Goal "1" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "2" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "3" *** *)
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "4" *** *)
a(DROP_NTH_ASM_T 11(strip_asm_tac o rewrite_rule[functional_def]));
a(asm_fc_tac[] THEN asm_fc_tac[]);
a(asm_rewrite_tac[]);
a(LEMMA_TΓz'' = (Fst z'',Snd z'')Γ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],id_tac]);
a(LEMMA_TΓz''' = (Fst z''',Snd z''')Γ pure_once_asm_rewrite_thm_tac
  THEN_LIST[rewrite_tac[],rewrite_tac[]]);
val isState_updateRow_lemma = save_pop_thm"isState_updateRow_lemma";

```

HOL output

```

isState_updateRow_lemma = ⊢ ∀ r c tc u
  • r ∈ RowS ∧ isVal (updateRow c tc (u, r))
  ⇒ destVal (updateRow c tc (u, r)) ∈ RowS

```

SML

```

push_goal([],Γ∀ r c t u •
  u ∈ Functional
  ∧ ((RelCombine
    ((revealRow c t)~ § u)
    (ListRel (TS_rows t))
    § Graph (updateRow c (TS_class t)))
    ▷ {x|isError x})
    § Graph destError
  = {}
  ∧ (∀ r • r ∈l (TS_rows t) ⇒ r ∈ RowS)
  ⇒ (r
    ∈l (RelList
      (ListRel (TS_rows t)
        ⊕ (RelCombine
          ((revealRow c t)~ § u)
          (ListRel (TS_rows t))
          § Graph (updateRow c (TS_class t)))
          § Graph destVal)) ⇒ r ∈ RowS)∇);

```

SML

```

a(rewrite_tac[revealRow_def] THEN REPEAT ∀_tac);
a(lemma_tacΓ∃ l • TS_rows t = l∇ THEN LIST
  [prove_∃_tac, POP_ASM_T rewrite_thm_tac]);
a(REV_LIST_INDUCTION_TΓl∇asm_tac);
(* *** Goal "1" *** *)
a(rewrite_tac[∈l_def, ⊕_null_thm2, rel_list_null_thm] THEN REPEAT strip_tac);

```

SML

```

(* *** Goal "2" *** *)
a  $\forall$ -tac;
a(cases_tac  $\lceil$  c dominates R_exist last  $\lrcorner$  THEN asm_rewrite_tac  $\lceil$ );
(* *** Goal "2.1" *** *)
a(REPEAT strip_tac);
a(fc_tac[rel_combine_null_lemma]);
a(lemma_tac  $\lceil$  ( $\forall$  r • r  $\in$  l  $\Rightarrow$  r  $\in$  RowS)  $\wedge$  last  $\in$  RowS  $\lrcorner$ );
(* *** Goal "2.1.1" *** *)
a(REPEAT strip_tac);
(* *** Goal "2.1.1.1" *** *)
a(spec_nth_asm_tac 4  $\lceil$  r'  $\lrcorner$ );
(* *** Goal "2.1.1.2" *** *)
a(spec_nth_asm_tac 3  $\lceil$  last  $\lrcorner$ );
(* *** Goal "2.1.2" *** *)
a(DROP_NTH_ASM_T 9 ante_tac THEN asm_rewrite_tac  $\lceil$ 
  THEN  $\Rightarrow$ -T asm_tac);
a(DROP_NTH_ASM_T 5 ante_tac);
a(cases_tac  $\lceil$   $\exists$  up • (#(Squash (Id (Dom (ListRel (l  $\hat{\wedge}$  [last])
   $\triangleright$  {r | c dominates R_exist r}))))), up)  $\in$  u  $\lrcorner$ );

```

SML

```

(* *** Goal "2.1.2.1" *** *)
a(strip_asm_tac(list_ $\forall$ -elim  $\lceil$  c  $\lrcorner$ ,  $\lceil$  last  $\lrcorner$ ,  $\lceil$  l  $\lrcorner$ ,  $\lceil$  up  $\lrcorner$ ,  $\lceil$  u  $\lrcorner$ ,  $\lceil$  t  $\lrcorner$   $\lceil$  conjunct1_lemma2)
  THEN POP_ASM_T rewrite_thm_tac);
a(REPEAT strip_tac);
(* *** Goal "2.1.2.1.1" *** *)
a(asm_fc_tac  $\lceil$ );
(* *** Goal "2.1.2.1.2" *** *)
a(lemma_tac  $\lceil$  isVal(updateRow c (TS_class t) (up, last))  $\lrcorner$ );
(* *** Goal "2.1.2.1.2.1" *** *)
a(strip_asm_tac( $\forall$ -elim  $\lceil$  (updateRow c (TS_class t) (up, last))  $\lrcorner$  val_or_error_type));
a(DROP_NTH_ASM_T 10 (strip_asm_tac o rewrite_rule[rel_ext_clauses, rel_combine_def]));
a(list_spec_nth_asm_tac 1  $\lceil$  # l + 1  $\lrcorner$ ,  $\lceil$  destError(updateRow c (TS_class t) (up, last))  $\lrcorner$ );
a(spec_nth_asm_tac 1  $\lceil$  (updateRow c (TS_class t) (up, last))  $\lrcorner$ );
a(POP_ASM_T(strip_asm_tac o rewrite_rule  $\lceil$  o  $\forall$ -elim  $\lceil$  (up, last)  $\lrcorner$ ));

```

SML

```

(* *** Goal "2.1.2.1.2.1.1" *** *)
a(spec_nth_asm_tac 1  $\lceil$  #(Squash (Id (Dom (ListRel ( $l \wedge [last]$ )
 $\triangleright \{r|c \text{ dominates } R\_exist\ r\}$ )))))) $\lceil$ );
a(POP_ASM_T ante_tac THEN asm_rewrite_tac[inv_rel_def]);
a(strip_asm_tac(list_ $\forall$ _elim[ $\lceil l \lceil$ , $\lceil last \lceil$ , $\lceil \{r|c \text{ dominates } R\_exist\ r\} \lceil$ ]size_ $\wedge$ _one_thm));
(* *** Goal "2.1.2.1.2.1.2" *** *)
a(POP_ASM_T(strip_asm_tac o rewrite_rule[list_rel_def,dot_dot_def,
length_\wedge_one_thm,nth_length_one_thm)));
(* *** Goal "2.1.2.1.2.2" *** *)
a(fc_tac[isState_updateRow_lemma] THEN asm_fc_tac[]);
a(asm_rewrite_tac[]);

```

SML

```

(* *** Goal "2.1.2.2" *** *)
a(LEMMA_T $\lceil$  RelCombine((Squash(Id(Dom
(ListRel( $l \wedge [last]$ )  $\triangleright \{r|c \text{ dominates } R\_exist\ r\}$ )))))) $\sim$  ; u)
(ListRel ( $l \wedge [last]$ ))
=
RelCombine((Squash(Id(Dom
(ListRel  $l \triangleright \{r|c \text{ dominates } R\_exist\ r\}$ )))))) $\sim$  ; u)
(ListRel  $l$ ) $\lceil$  rewrite_thm_tac);

```

SML

```

(* *** Goal "2.1.2.2.1" *** *)
a(asm_rewrite_tac[squash_ $\wedge$ _thm]);
a(LEMMA_T $\lceil$  {(#(Squash(Id (Dom(ListRel  $l \triangleright \{r|c$ 
dominates R_exist r))))
+ 1, #  $l + 1$ )} $\sim$ 
; u = {} $\lceil$  rewrite_thm_tac);
(* *** Goal "2.1.2.2.1.1" *** *)
a(rewrite_tac[rel_ext_clauses,inv_rel_def] THEN REPEAT strip_tac);
a(spec_nth_asm_tac 3  $\lceil y \lceil$ );
a(DROP_NTH_ASM_T 4(fn _ => id_tac));
a(strip_asm_tac(list_ $\forall$ _elim[ $\lceil l \lceil$ , $\lceil last \lceil$ , $\lceil \{r|c \text{ dominates } R\_exist\ r\} \lceil$ ]size_squash_plus1_thm));
a contr_tac;
a(DROP_NTH_ASM_T 5 ante_tac THEN DROP_NTH_ASM_T 3 ante_tac
THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2.2.1.2" *** *)
a(rewrite_tac[rel_combine_one_lemma]);

```

SML

```

(* *** Goal "2.1.2.2" *** *)
a(strip_asm_tac(list_∀_elim[⌈l⌉,⌈last⌉,⌈c⌉,⌈u⌉,⌈t⌉]conjunct1_lemma1));
a(asm_rewrite_tac[]);
a(REPEAT strip_tac THEN asm_fc_tac[]);
(* *** Goal "2.2" *** *)
a(REPEAT strip_tac);
a(fc_tac[rel_combine_null_lemma]);
a(lemma_tac[⌈(∀ r • r ∈ l ⇒ r ∈ RowS) ∧ last ∈ RowS⌉]);
(* *** Goal "2.2.1" *** *)
a(REPEAT strip_tac);
(* *** Goal "2.2.1.1" *** *)
a(spec_nth_asm_tac 4 ⌈r'⌉);
(* *** Goal "2.2.1.2" *** *)
a(spec_nth_asm_tac 3 ⌈last⌉);

```

SML

```

(* *** Goal "2.2.2" *** *)
a(DROP_NTH_ASM_T 9 ante_tac THEN asm_rewrite_tac[]
  THEN ⇒_T asm_tac);
a(DROP_NTH_ASM_T 5 ante_tac THEN asm_rewrite_tac
  [rel_combine_one_lemma,squash_∧_thm]);
a(strip_asm_tac(list_∀_elim[⌈l⌉,⌈last⌉,⌈c⌉,⌈u⌉,⌈t⌉]conjunct1_lemma1));
a(asm_rewrite_tac[]);
a(REPEAT strip_tac THEN asm_fc_tac[]);
val isState_updateRows_lemma = save_pop_thm"isState_updateRows_lemma";

```

HOL output

```

isState_updateRows_lemma = ⊢ ∀ r c t u
  • u ∈ Functional
    ∧ ((RelCombine ((revealRow c t)~ ; u) (ListRel (TS_rows t))
      ; Graph (updateRow c (TS_class t)))
      ▷ {x|isError x})
      ; Graph destError
    = {}
    ∧ (∀ r • r ∈l TS_rows t ⇒ r ∈ RowS)
⇒ r
  ∈l RelList
    (ListRel (TS_rows t)
      ⊕ (RelCombine
          ((revealRow c t)~ ; u)
          (ListRel (TS_rows t))
          ; Graph (updateRow c (TS_class t)))
        ; Graph destVal)
⇒ r ∈ RowS

```

**3.2.4 updateQuery Lemma**

SML

```

push_goal([], ⊢ ∀ c i us s ts • isState s ∧ tabExists c i s
  ⇒ isState(Fst(updateQuery(c,(i,us),s,getTable i s)))⊢);
a(rewrite_tac[get_spec⊢ isState⊢, get_spec⊢ StateS⊢, ↔_def, ⊃_def,
  rewrite_rule[dom_def] tabExists_def, getTable_def]
  THEN REPEAT ∀_tac THEN strip_tac);
a(LIST_DROP_NTH_ASM_T[2,3](MAP_EVERY (fn _ => id_tac)));
a(strip_asm_tac(list_∀_elim[⊢ s⊢, ⊢ Front i⊢, ⊢ y⊢] at_thm1));
a(rewrite_tac[conv_rule(MAP_C let_conv) updateQuery_def, changeSpec_def]);

```

SML

```

a(cases_tac⊢ ⊢ us ∈ Functional⊢ THEN asm_rewrite_tac[]);
a(cases_tac⊢ ⊢ Dom (⋃ (Ran us))
  ⊆ {n
    | ∃ c'
      • c'
        ∈ visibleCols
          c
            (Dir_tables y
              @ Last i)
          ∧ CS_posn c' = n}⊢
  THEN asm_rewrite_tac[]);

```

SML

```

a(cases_tac⌈((RelCombine ((revealRow c
                        (Dir_tables y
                          @ Last i))~
                        ; us)
      (ListRel
        (TS_rows
          (Dir_tables y
            @ Last i)))
      ; Graph
      (updateRow
        c
        (TS_class
          (Dir_tables y
            @ Last i))))
      ▷ {x|isError x})
      ; Graph destError
      = {}⌈
      THEN asm_rewrite_tac[]);

```

SML

```

a(POP_ASM_T ante_tac THEN POP_ASM_T ante_tac
  THEN DROP_NTH_ASM_T 3 ante_tac
  THEN DROP_NTH_ASM_T 2 rewrite_thm_tac
  THEN REPEAT strip_tac);
(* *** Goal "1" *** *)
a(DROP_NTH_ASM_T 7 ante_tac THEN
  rewrite_tac[↔_def,get_spec⌈IdeL⌋,get_spec⌈DirectoryS⌋,
  ∩_def,×_def,get_spec⌈Universe⌋,rel_ext_clauses,get_spec⌈$P⌋]
  THEN REPEAT ⇒_tac);
a(asm_fc_tac[]);
a(POP_ASM_T ante_tac THEN rewrite_tac[↔_def,∩_def,⊕_single]
  THEN REPEAT strip_tac);

```

SML

```

(* *** Goal "1.1" *** *)
a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_specΓMkDirectoryΓ]);
a(strip_asm_tac(list_∀_elimΓDir_tables yΓ,ΓLast iΓ,Γy'Γ)at_thm1));
a(DROP_NTH_ASM_T 7 ante_tac THEN DROP_NTH_ASM_T 6 ante_tac
  THEN POP_ASM_T rewrite_thm_tac THEN REPEAT ⇒_tac);
a(DROP_NTH_ASM_T 5 ante_tac THEN
  rewrite_tac[↔_def,get_specΓIdeΓ,conv_rule(MAP_C let_conv)(get_specΓTableSpecΓ),
  ∩_def,×_def,get_specΓUniverseΓ,rel_ext_clauses,get_specΓ$PΓ]
  THEN REPEAT ⇒_tac);
a(rewrite_tac[⊕_single]THEN strip_tac THEN strip_tac THEN strip_tac);

```

SML

```

(* *** Goal "1.1.1" *** *)
a(POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[replaceRows_def,get_specΓMkTableSpecΓ]);
a(list_spec_nth_asm_tac 2 [ΓLast iΓ,Γy'Γ]);
a(REPEAT strip_tac);
a(fc_tac[isState_updateRows_lemma] THEN asm_fc_tac[]);
a(POP_ASM_T (ante_tac o ∀_elimΓrΓ)THEN asm_rewrite_tac[]);
(* *** Goal "1.1.2" *** *)
a(asm_fc_tac[] THEN asm_rewrite_tac[]);

```

SML

```

(* *** Goal "1.2" *** *)
a(POP_ASM_T rewrite_thm_tac THEN rewrite_tac[get_specΓMkDirectoryΓ]);
a(DROP_NTH_ASM_T 2 ante_tac THEN rewrite_tac[functional_def,⊕_single]
  THEN REPEAT strip_tac THEN TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
(* *** Goal "1.3" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔_def,∩_def]));
(* *** Goal "1.4" *** *)
a(asm_fc_tac[]);
a(POP_ASM_T (strip_asm_tac o rewrite_rule[↔_def,∩_def]));

```

SML

```

(* *** Goal "2" *** *)
a(DROP_NTH_ASM_T 6 ante_tac THEN rewrite_tac[functional_def,⊕_single]
  THEN REPEAT strip_tac THEN TRY asm_rewrite_tac[]);
a(asm_fc_tac[] THEN asm_fc_tac[]);
val isState_updateQuery_lemma = save_pop_thm" isState_updateQuery_lemma";

```

HOL output

```

| isState_updateQuery_lemma =
| ⊢ ∀ c i ns s ts
|   • isState s ∧ tabExists c i s
|     ⇒ isState (deleteQuery (c, (i, ns), s, getTable i s))

```

**3.2.5 State Invariant Lemma**

SML

```

| push_goal([], ⌈∀ clear s • isState s ⇒ isState(Fst (updateStateR (clear, u, s)))⌈);
| a(REPEAT strip_tac);
| a(LEMMA_T ⌈u = (Fst u, Snd u)⌈ pure_once_asm_rewrite_thm_tac
|   THEN_LIST[rewrite_tac[], rewrite_tac[updateStateR_def] THEN REPEAT ∀_tac]);
| a(strip_asm_tac(∀_elim ⌈Fst u⌈ query_type) THEN asm_rewrite_tac[]
|   THEN cases_tac ⌈¬ Snd u = []⌈ THEN asm_rewrite_tac[]
|   THEN cases_tac ⌈tabExists clear (tabFromEffect (Fst u)) s⌈ THEN asm_rewrite_tac[]
|   THEN cases_tac ⌈¬ clear dominates TS_class (getTable (tabFromEffect (Fst u)) s)⌈
|   THEN asm_rewrite_tac[]);

```

SML

```

| (* 3 subgoals – Select automatically proven *)
| (* *** Goal "1" *** *)
| (* Insert *)
| a(DROP_NTH_ASM_T 7(strip_asm_tac o rewrite_rule[isInsert_def]));
| a(DROP_NTH_ASM_T 3 ante_tac THEN POP_ASM_T rewrite_thm_tac);
| a(rewrite_tac[destInsert_def, tabFromEffect_def] THEN ⇒_tac);
| a(fc_tac[isState_insertQuery_lemma] THEN asm_fc_tac[]);
| a(POP_ASM_T (ante_tac o ∀_elim ⌈Snd i⌈));
| a(rewrite_tac[]);

```

SML

```

| (* Delete *)
| a(DROP_NTH_ASM_T 7(strip_asm_tac o rewrite_rule[isInsert_def]));
| a(DROP_NTH_ASM_T 3 ante_tac THEN POP_ASM_T rewrite_thm_tac);
| a(rewrite_tac[destInsert_def, tabFromEffect_def] THEN ⇒_tac);
| a(fc_tac[isState_deleteQuery_lemma] THEN asm_fc_tac[]);
| a(POP_ASM_T (ante_tac o ∀_elim ⌈Snd d⌈));
| a(rewrite_tac[]);

```

SML

```

(* Update *)
a(DROP_NTH_ASM_T 7(strip_asm_tac o rewrite_rule[isInsert_def]));
a(DROP_NTH_ASM_T 3 ante_tac THEN POP_ASM_T rewrite_thm_tac);
a(rewrite_tac[destInsert_def,tabFromEffect_def] THEN =>_tac);
a(fc_tac[isState_updateQuery_lemma] THEN asm_fc_tac[]);
a(POP_ASM_T (ante_tac o  $\forall$ _elim $\ulcorner$ Snd u' $\urcorner$ ));
a(rewrite_tac[]);
val updateStateR_lemma = save_pop_thm"updateStateR_lemma";

```

HOL output

```

updateStateR_lemma =  $\vdash \forall$  clear s
  • isState s  $\Rightarrow$  isState (Fst (updateStateR (clear, u, s)))

```

### 3.3 Proof of Lemma1

We prove *Lemma1* using the results of section 3.1 and section 3.2 together with *secureHide\_lemma* from [4].

SML

```

val lemma1_thm =
  prove_rule[Lemma1,secureHide_lemma,
    rewrite_rule[updateStateR_lemma]updateStateR_updateState_lemma] $\ulcorner$ Lemma1 $\urcorner$ ;

```

HOL output

```

val lemma1_thm =  $\vdash$  Lemma1

```

## 4 PROOF OF SECURITY OF SSQL

We now give the main result *secureSSQL* that the behaviour of the SSQL abstract machine is secure.

SML

```

val secureSSQL = save_thm("secureSSQL",rewrite_rule[lemma1_thm]main_thm1);

```

HOL output

```

secureSSQL =  $\vdash$  behaviours SSQLam  $\in$  secure

```

## 5 CLOSING DOWN

The following *ProofPower* instruction restores the previous proof context.

SML

```

pop_pc();

```

## 6 THE THEORY fef015

### 6.1 Parents

*fef013*

### 6.2 Theorems

#### updateStateR\_updateState\_lemma

$$\begin{aligned} &\vdash (\forall \text{clear } s \ u \\ &\quad \bullet \text{isState } s \\ &\quad \Rightarrow \text{isState } (Fst \ (\text{updateStateR } (\text{clear}, u, s))) \\ &\quad \Rightarrow (\text{hide}, \text{updateState}) \in \text{secureUpdate} \end{aligned}$$

#### isState\_insertQuery\_lemma

$$\begin{aligned} &\vdash \forall c \ i \ ns \ s \ ts \\ &\quad \bullet \text{isState } s \wedge \text{tabExists } c \ i \ s \\ &\quad \Rightarrow \text{isState} \\ &\quad \quad (Fst \\ &\quad \quad \quad (\text{insertQuery} \\ &\quad \quad \quad \quad (c, (i, ds), s, \text{getTable } i \ s))) \end{aligned}$$

#### isState\_deleteQuery\_lemma

$$\begin{aligned} &\vdash \forall c \ i \ ns \ s \ ts \\ &\quad \bullet \text{isState } s \wedge \text{tabExists } c \ i \ s \\ &\quad \Rightarrow \text{isState} \\ &\quad \quad (\text{deleteQuery } (c, (i, ns), s, \text{getTable } i \ s)) \end{aligned}$$

#### isState\_updateRow\_lemma

$$\begin{aligned} &\vdash \forall r \ c \ tc \ u \\ &\quad \bullet r \in \text{RowS} \wedge \text{isVal } (\text{updateRow } c \ tc \ (u, r)) \\ &\quad \Rightarrow \text{destVal } (\text{updateRow } c \ tc \ (u, r)) \in \text{RowS} \end{aligned}$$

#### isState\_updateRows\_lemma

$$\begin{aligned} &\vdash \forall r \ c \ t \ u \\ &\quad \bullet u \in \text{Functional} \\ &\quad \wedge ((\text{RelCombine} \\ &\quad \quad (\text{revealRow } c \ t \ \sim \ \text{\% } u) \\ &\quad \quad (\text{ListRel } (TS\_rows \ t)) \\ &\quad \quad \text{\% } \text{Graph } (\text{updateRow } c \ (TS\_class \ t))) \\ &\quad \quad \triangleright \{x | \text{isError } x\}) \\ &\quad \quad \text{\% } \text{Graph } \text{destError} \\ &\quad \quad = \{\} \\ &\quad \wedge (\forall r \bullet r \in_l \text{TS\_rows } t \Rightarrow r \in \text{RowS}) \\ &\Rightarrow r \\ &\quad \in_l \text{RelList} \\ &\quad \quad (\text{ListRel } (TS\_rows \ t) \\ &\quad \quad \oplus (\text{RelCombine} \\ &\quad \quad \quad (\text{revealRow } c \ t \ \sim \ \text{\% } u) \\ &\quad \quad \quad (\text{ListRel } (TS\_rows \ t)) \\ &\quad \quad \quad \text{\% } \text{Graph } (\text{updateRow } c \ (TS\_class \ t))) \\ &\quad \quad \quad \text{\% } \text{Graph } \text{destVal}) \end{aligned}$$

$$\Rightarrow r \in RowS$$
**isState\_updateQuery\_lemma**

$$\vdash \forall c i us s ts$$

- $isState s \wedge tabExists c i s$

$$\Rightarrow isState$$

$$(Fst$$

$$(updateQuery$$

$$(c, (i, us), s, getTable i s)))$$
**updateStateR\_lemma**

$$\vdash \forall clear s$$

- $isState s$

$$\Rightarrow isState (Fst (updateStateR (clear, u, s)))$$
**secureSSQL**

$$\vdash behaviours SSQLam \in secure$$

## 7 INDEX

<i>fef015</i> .....	4
<i>isState_deleteQuery_lemma</i> .....	10
<i>isState_insertQuery_lemma</i> .....	8
<i>isState_updateQuery_lemma</i> .....	18
<i>isState_updateRows_lemma</i> .....	15
<i>isState_updateRow_lemma</i> .....	11
<i>lemma1_thm</i> .....	20
<i>secureSSQL</i> .....	20
<i>updateStateR_lemma</i> .....	20
<i>updateStateR_updateState_lemma</i> .....	6