

EECS 3311

FALL 2018

Rafael Lugassy

rafaell

note

description: "A DATABASE ADT mapping from keys to two kinds of values"

author: "Jackie Wang and You"

date: "\$Date\$"

revision: "\$Revision\$"

class interface

DATABASE [V1, V2, K]

create

make

feature -- Commands

add_record (v1: V1; v2: V2; k: K)

-- Add a new record into current database.

require

non_existing_key: not Current.exists (k)

ensure

record_added: across

1 |..| Current.count as i

some

keys.at (i.item) ~ k and values_1.at (i.item) ~ v1 and values_2.at

(i.item) ~ v2

end

remove_record (k: K)

-- Remove a record from current database.

require

existing_key: across

Current as tuple

some

k ~ tuple.item.item (1)

end

ensure

database_count_decremented: Current.count = old Current.count - 1

key_removed: across

Current as tuple

all

k /~ tuple.item.item (1)

end

feature -- Constructor

make

-- Initialize an empty database.

ensure

```

empty_database: across
    Current as c
    all
        False
    end
object_equality_for_keys: keys.object_comparison
object_equality_for_values_1: values_1.object_comparison
object_equality_for_values_2: values_2.object_comparison

feature -- Queries

count: INTEGER_32
    -- Number of records in database.
    ensure
        correct_result: Result = keys.count and Result = values_1.count and Result =
values_2.count

exists (k: K): BOOLEAN
    -- Does key 'k' exist in the database?
    ensure
        correct_result: across
            old Current as tuple
            some
                (attached {K} tuple.item.item (1) as k1 implies k1 ~ k) and
attached {K} tuple.item.item (1)
            end
        end

get_keys (v1: V1; v2: V2): ITERABLE [K]
    -- Keys that are associated with values 'v1' and 'v2'.
    -- Your Task
    ensure
        result_contains_correct_keys_only: across
            Result as k
            all
                across
                    Current as tuple
                    all
                        k ~ tuple.item.item (1) implies (v1 ~ tuple.item.item (2)
and v2 ~ tuple.item.item (3))
                    end
                end
            end
        correct_keys_are_in_result: across
            1 |..| Current.count as i
            all
                (v1 ~ values_1.at (i.item) and v2 ~ values_2.at (i.item)) implies
across
                    Result as k

```

```

                                some
                                k.item ~ keys.at (i.item)
                                end
                        end
                end

feature -- alternative iteration cursor
-- Your Task

        another_cursor: RECORD_ITERATION_CURSOR [V1, V2, K]

feature -- feature(s) required by ITERABLE
-- Your Task

        new_cursor: TUPLE_ITERATION_CURSOR [K, V1, V2]
                -- Fresh cursor associated with current structure

invariant
        unique_keys: across
                1 |..| keys.count as i
                all
                        across
                                1 |..| keys.count as j
                                all
                                        i.item /= j.item implies keys.at (i.item) /~ keys.at (j.item)
                                end
                        end
        implementation_constraint: values_1.lower = 1
        consistent_keys_values_counts: keys.count = values_1.count and keys.count = values_2.count
        consistent_imp_adt_counts: keys.count = count

end -- class DATABASE

```

note

description: "Summary description for {RECORD}."

author: "Jackie Wang"

date: "\$Date\$"

revision: "\$Revision\$"

class interface

RECORD [V1, V2, K]

create

make

feature -- Attributes (Do not modify this section)

key: K

value_1: V1

value_2: V2

feature -- Commands (Do not modify this section)

make (v1: V1; v2: V2; k: K)

feature -- Equality

is_equal (other: like Current): BOOLEAN

-- Is `other` attached to an object considered

-- equal to current object?

end -- class RECORD

note

description: "Summary description for {RECORD_ITERATION_CURSOR}."
author: ""
date: "\$Date\$"
revision: "\$Revision\$"

class interface

RECORD_ITERATION_CURSOR [V1, V2, K]

create

make

feature -- Access

item: RECORD [V1, V2, K]
-- Item at current cursor position.

feature -- Cursor movement

forth
-- Move to next position

feature

make (values_1: ARRAY [V1]; values_2: LINKED_LIST [V2]; keys: LINKED_LIST [K])

feature -- Status report

after: BOOLEAN
-- Are there no more items to iterate over?

end -- class RECORD_ITERATION_CURSOR

note

description: "Summary description for {TUPLE_ITERATION_CURSOR}."
author: ""
date: "\$Date\$"
revision: "\$Revision\$"

class interface

TUPLE_ITERATION_CURSOR [K, V1, V2]

create

make

feature -- Access

item: TUPLE [K, V1, V2]
-- Item at current cursor position.

feature -- Cursor movement

forth
-- Move to next position

feature

make (values_1: ARRAY [V1]; values_2: LINKED_LIST [V2]; keys: LINKED_LIST [K])

feature -- Status report

after: BOOLEAN
-- Are there no more items to iterate over?

end -- class TUPLE_ITERATION_CURSOR

