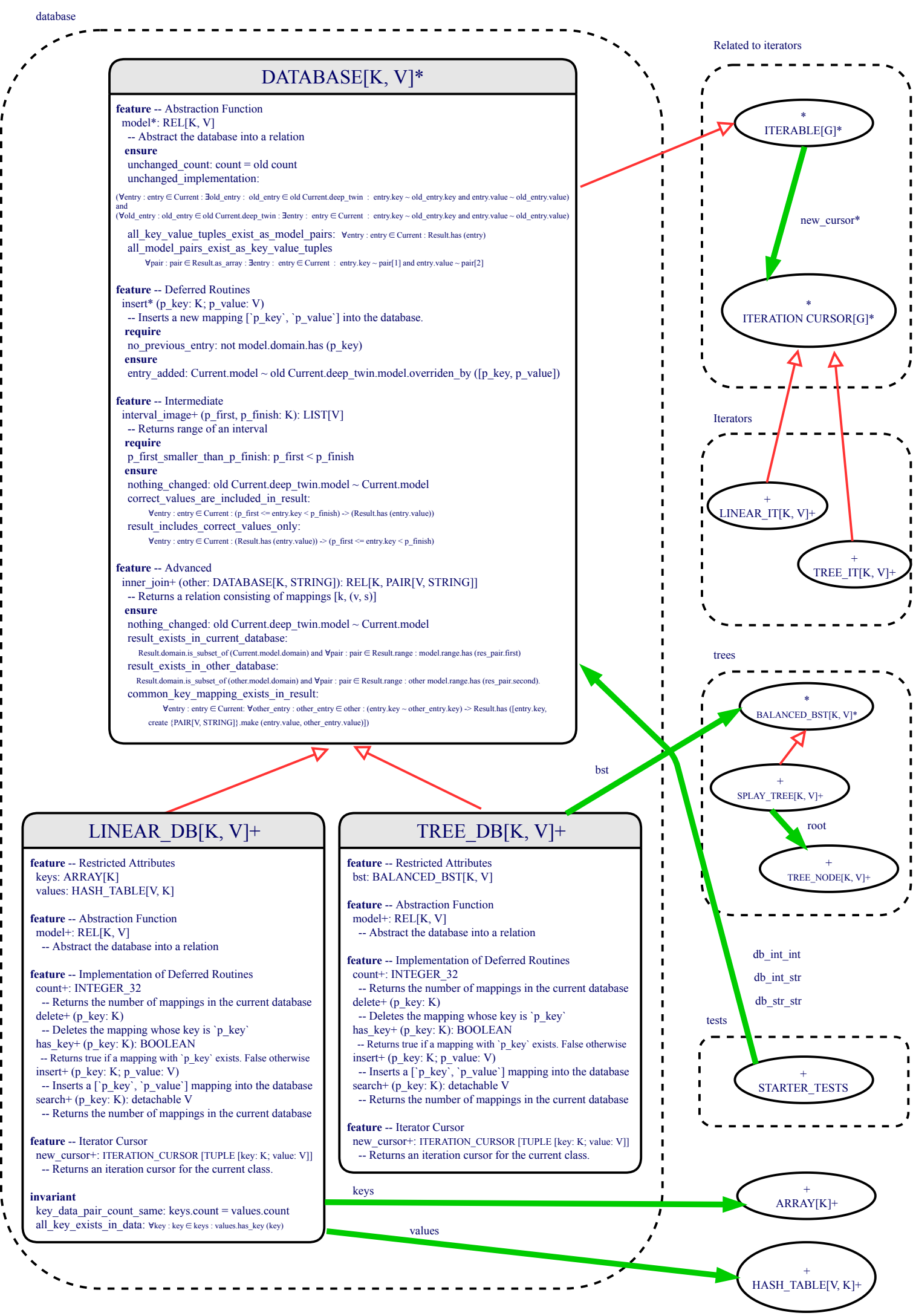
database
DATABASE[K, V]*
feature -- Abstraction Function
model*: REL[K, V]
-- Abstract the database into a relation
ensure
unchanged_count: count = old count
unchanged_implementation:
(∀entry : entry ∈ Current : ∃old_entry : old_entry ∈ old Current.deep_twin : entry.key ~ old_entry.key and entry.value ~ old_entry.value)
and
(∀old_entry : old_entry ∈ old Current.deep_twin : ∃entry : entry ∈ Current : entry.key ~ old_entry.key and entry.value ~ old_entry.value)
all_key_value_tuples_exist_as_model_pairs: ∀entry : entry ∈ Current : Result.has (entry)
all_model_pairs_exist_as_key_value_tuples
∀pair : pair ∈ Result.as_array : ∃entry : entry ∈ Current : entry.key ~ pair[1] and entry.value ~ pair[2]
feature -- Deferred Routines
insert* (p_key: K; p_value: V)
-- Inserts a new mapping [`p_key`, `p_value`] into the database.
require
no_previous_entry: not model.domain.has (p_key)
ensure
entry_added: Current.model ~ old Current.deep_twin.model.overriden_by ([p_key, p_value])
feature -- Intermediate
interval_image+ (p_first, p_finish: K): LIST[V]
-- Returns range of an interval
require
p_first_smaller_than_p_finish: p_first < p_finish
ensure
nothing_changed: old Current.deep_twin.model ~ Current.model
correct_values_are_included_in_result:
∀entry : entry ∈ Current : (p_first <= entry.key < p_finish) -> (Result.has (entry.value))
result_includes_correct_values_only:
∀entry : entry ∈ Current : (Result.has (entry.value)) -> (p_first <= entry.key < p_finish)
feature -- Advanced
inner_join+ (other: DATABASE[K, STRING]): REL[K, PAIR[V, STRING]]
-- Returns a relation consisting of mappings [k, (v, s)]
ensure
nothing_changed: old Current.deep_twin.model ~ Current.model
result_exists_in_current_database:
Result.domain.is_subset_of (Current.model.domain) and ∀pair : pair ∈ Result.range : model.range.has (res_pair.first)
result_exists_in_other_database:
Result.domain.is_subset_of (other.model.domain) and ∀pair : pair ∈ Result.range : other model.range.has (res_pair.second).
common_key_mapping_exists_in_result:
∀entry : entry ∈ Current: ∀other_entry : other_entry ∈ other : (entry.key ~ other_entry.key) -> Result.has ([entry.key,
create {PAIR[V, STRING]}.make (entry.value, other_entry.value)])
LINEAR_DB[K, V]+
feature -- Restricted Attributes
keys: ARRAY[K]
values: HASH_TABLE[V, K]
feature -- Abstraction Function
model+: REL[K, V]
-- Abstract the database into a relation
feature -- Implementation of Deferred Routines
count+: INTEGER_32
-- Returns the number of mappings in the current database
delete+ (p_key: K)
-- Deletes the mapping whose key is `p_key`
has_key+ (p_key: K): BOOLEAN
-- Returns true if a mapping with `p_key` exists. False otherwise
insert+ (p_key: K; p_value: V)
-- Inserts a [`p_key`, `p_value`] mapping into the database
search+ (p_key: K): detachable V
-- Returns the number of mappings in the current database
feature -- Iterator Cursor
new_cursor+: ITERATION_CURSOR [TUPLE [key: K; value: V]]
-- Returns an iteration cursor for the current class.
invariant
key_data_pair_count_same: keys.count = values.count
all_key_exists_in_data: ∀key : key ∈ keys : values.has_key (key)
TREE_DB[K, V]+
feature -- Restricted Attributes
bst: BALANCED_BST[K, V]
feature -- Abstraction Function
model+: REL[K, V]
-- Abstract the database into a relation
feature -- Implementation of Deferred Routines
count+: INTEGER_32
-- Returns the number of mappings in the current database
delete+ (p_key: K)
-- Deletes the mapping whose key is `p_key`
has_key+ (p_key: K): BOOLEAN
-- Returns true if a mapping with `p_key` exists. False otherwise
insert+ (p_key: K; p_value: V)
-- Inserts a [`p_key`, `p_value`] mapping into the database
search+ (p_key: K): detachable V
-- Returns the number of mappings in the current database
feature -- Iterator Cursor
new_cursor+: ITERATION_CURSOR [TUPLE [key: K; value: V]]
-- Returns an iteration cursor for the current class.
Related to iterators
* ITERABLE[G]*
new_cursor*
* ITERATION CURSOR[G]*
Iterators
+ LINEAR_IT[K, V]+
+ TREE_IT[K, V]+
trees
* BALANCED_BST[K, V]*
+ SPLAY_TREE[K, V]+
root
+ TREE_NODE[K, V]+
bst
db_int_int
db_int_str
db_str_str
tests
+ STARTER_TESTS
keys
+ ARRAY[K]+
values
+ HASH_TABLE[V, K]+