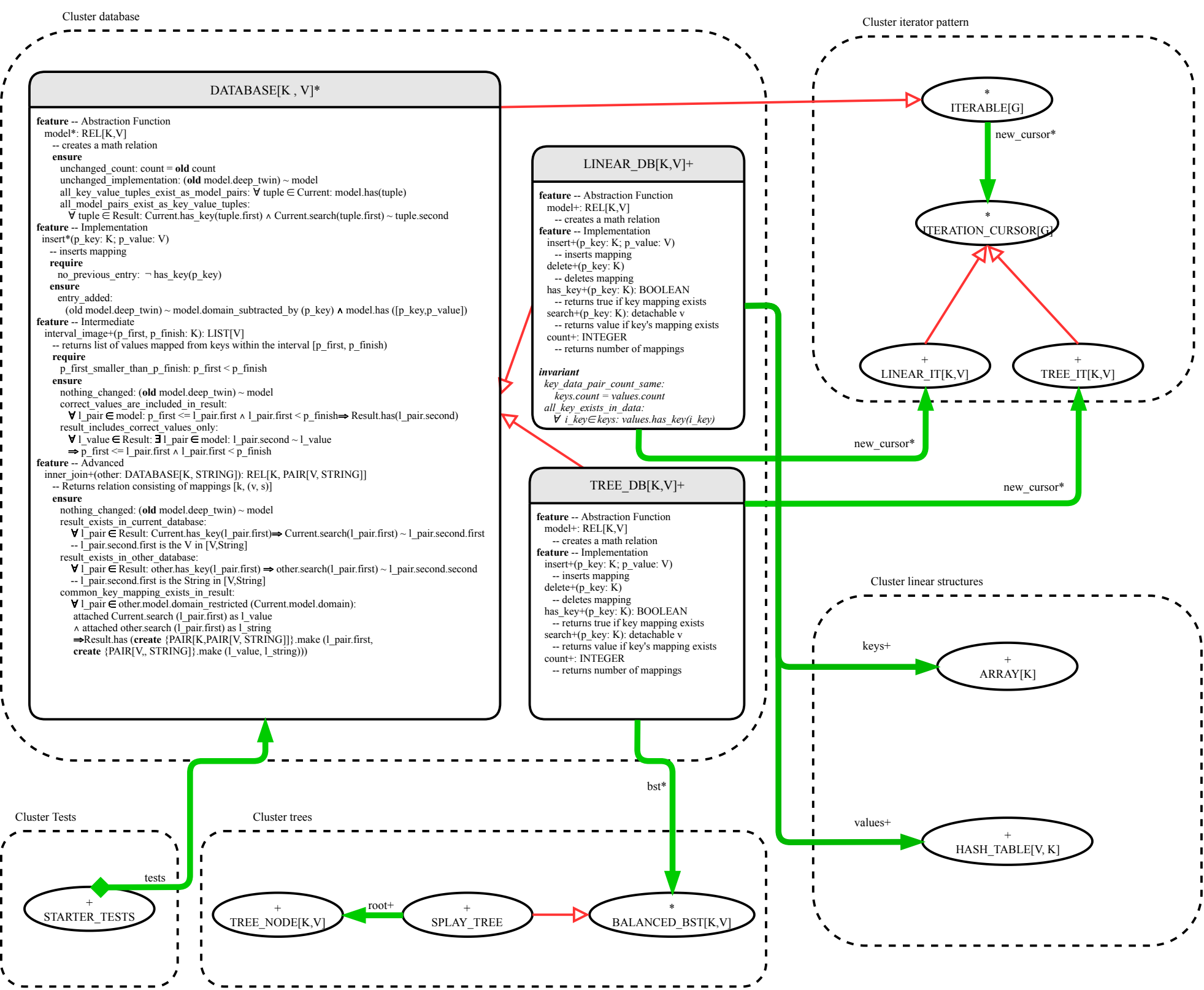
Cluster database
DATABASE[K , V]*
feature -- Abstraction Function
model*: REL[K,V]
-- creates a math relation
ensure
unchanged_count: count = old count
unchanged_implementation: (old model.deep_twin) ~ model
all_key_value_tuples_exist_as_model_pairs: ∀ tuple ∈ Current: model.has(tuple)
all_model_pairs_exist_as_key_value_tuples:
∀ tuple ∈ Result: Current.has_key(tuple.first) ∧ Current.search(tuple.first) ~ tuple.second
feature -- Implementation
insert*(p_key: K; p_value: V)
-- inserts mapping
require
no_previous_entry: ¬ has_key(p_key)
ensure
entry_added:
(old model.deep_twin) ~ model.domain_subtracted_by (p_key) ∧ model.has ([p_key,p_value])
feature -- Intermediate
interval_image+(p_first, p_finish: K): LIST[V]
-- returns list of values mapped from keys within the interval [p_first, p_finish)
require
p_first_smaller_than_p_finish: p_first < p_finish
ensure
nothing_changed: (old model.deep_twin) ~ model
correct_values_are_included_in_result:
∀ l_pair ∈ model: p_first <= l_pair.first ∧ l_pair.first < p_finish⇒ Result.has(l_pair.second)
result_includes_correct_values_only:
∀ l_value ∈ Result: ∃ l_pair ∈ model: l_pair.second ~ l_value
⇒ p_first <= l_pair.first ∧ l_pair.first < p_finish
feature -- Advanced
inner_join+(other: DATABASE[K, STRING]): REL[K, PAIR[V, STRING]]
-- Returns relation consisting of mappings [k, (v, s)]
ensure
nothing_changed: (old model.deep_twin) ~ model
result_exists_in_current_database:
∀ l_pair ∈ Result: Current.has_key(l_pair.first)⇒ Current.search(l_pair.first) ~ l_pair.second.first
-- l_pair.second.first is the V in [V,String]
result_exists_in_other_database:
∀ l_pair ∈ Result: other.has_key(l_pair.first) ⇒ other.search(l_pair.first) ~ l_pair.second.second
-- l_pair.second.first is the String in [V,String]
common_key_mapping_exists_in_result:
∀ l_pair ∈ other.model.domain_restricted (Current.model.domain):
attached Current.search (l_pair.first) as l_value
∧ attached other.search (l_pair.first) as l_string
⇒Result.has (create {PAIR[K,PAIR[V, STRING]]}.make (l_pair.first,
create {PAIR[V,, STRING]}.make (l_value, l_string)))
LINEAR_DB[K,V]+
feature -- Abstraction Function
model+: REL[K,V]
-- creates a math relation
feature -- Implementation
insert+(p_key: K; p_value: V)
-- inserts mapping
delete+(p_key: K)
-- deletes mapping
has_key+(p_key: K): BOOLEAN
-- returns true if key mapping exists
search+(p_key: K): detachable v
-- returns value if key's mapping exists
count+: INTEGER
-- returns number of mappings
invariant
key_data_pair_count_same:
keys.count = values.count
all_key_exists_in_data:
∀ i_key∈keys: values.has_key(i_key)
TREE_DB[K,V]+
feature -- Abstraction Function
model+: REL[K,V]
-- creates a math relation
feature -- Implementation
insert+(p_key: K; p_value: V)
-- inserts mapping
delete+(p_key: K)
-- deletes mapping
has_key+(p_key: K): BOOLEAN
-- returns true if key mapping exists
search+(p_key: K): detachable v
-- returns value if key's mapping exists
count+: INTEGER
-- returns number of mappings
Cluster iterator pattern
* ITERABLE[G]
new_cursor*
* ITERATION_CURSOR[G]
+ LINEAR_IT[K,V]
+ TREE_IT[K,V]
new_cursor*
new_cursor*
Cluster linear structures
keys+
+ ARRAY[K]
values+
+ HASH_TABLE[V, K]
bst*
Cluster Tests
tests
+ STARTER_TESTS
Cluster trees
+ TREE_NODE[K,V]
root+
+ SPLAY_TREE
* BALANCED_BST[K,V]