fn match_per_group_predicate

Michael Shoemate

October 25, 2025

This proof resides in **"contrib"** because it has not completed the vetting process.

Proves soundness of `match_per_group_predicate` in `mod.rs` at commit `f5bb719` (outdated[1]).

1 Hoare Triple

Precondition

Compiler Verified

Types matching pseudocode.

Precondition

None

Function

```python
def match_per_group_predicate(
    enumeration: Expr,
    partition_by: Vec[Expr],
    identifier: Expr,
    threshold: u32,
) -> Optional[Bound]:
    # reorderings of an enumeration are still enumerations
    if isinstance(enumeration, Expr.Function) and isinstance( #
        enumeration.options.collect_groups, ApplyOptions.GroupWise
    ):
        input = enumeration.input
        function = enumeration.function

        # FunctionExprs that may reorder data
        if function == FunctionExpr.Reverse:
            is_reorder = True
        elif isinstance(function, FunctionExpr.Random):
            method = str(function.method)
            is_reorder = method == "shuffle"
        else:
            is_reorder = False

        if is_reorder:
            enumeration = input[0]
```

[1] See new changes with `git diff f5bb719..3138a44 rust/src/transformations/make_stable_lazyframe/truncate/matching/mod.rs`

```
25
26      elif isinstance(enumeration, Expr.SortBy):
27          for key in enumeration.by:
28              check_infallible(key, Resize.Ban)
29          enumeration = enumeration.expr#
30
31      # in Rust, the != results in a boolean comparison, not a "ne" expression
32      if enumeration != int_range(lit(0), len(partition_by), 1, DataType.Int64): #
33          return None
34
35      # we now know this is a per group predicate,
36      # and can raise more informative error messages
37
38      # check if the function is limiting partition contributions
39      ids, by = partition(lambda expr: expr == identifier, partition_by) #
40
41      if not ids:
42          raise "failed to find identifier column in per_group predicate condition"
43
44      return Bound(by=by, per_group=threshold, num_groups=None) #
```

Postcondition

Theorem 1.1 (Postcondition)**.** If `enumeration` is an enumeration of rows, and `partition_by` includes `identifier`, then returns a `threshold` bound on per-group contribution, when grouped by the non-identifier columns in `partition_by`.

Proof. Reorderings of an enumeration, like reversal, shuffling and sorting are also valid enumerations, so lines 8-29 ignore these reorderings and re-assign the enumeration to the input of the reordering.

Due to the ambiguity between matching predicates that bound `num_groups` or `per_group`, an error is only raised if the predicate is unambiguously a `num_groups` truncation predicate. This check happens on line 32.

Line 39 splits the partition by expressions into two sets, one containing the `identifier` and the other containing the grouping columns.

This predicate corresponds to a `per_group` truncation predicate, because the over expression groups by both the `identifier` column and any expressions in `by`, and within each group, an enumeration is applied to count the number of rows. If the only rows kept are those whose enumeration is less than `threshold`, then each user identifier will have at most `threshold` rows when their records are grouped by `by`. Reorderings of the enumeration, like reversal, shuffling and sorting can be used to choose which records from each individual are kept.

Therefore the bound on user contributions constructed on line 44 is valid. □