

fn match_per_group_predicate

Michael Shoemate

April 22, 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` (outdated¹).

1 Hoare Triple

Precondition

None

Function

```
1 def match_per_group_predicate(  
2     enumeration: Expr,  
3     partition_by: Vec[Expr],  
4     identifier: Expr,  
5     threshold: u32,  
6 ) -> Optional[Bound]:  
7     # reorderings of an enumeration are still enumerations  
8     if isinstance(enumeration, Expr.Function) and isinstance( #  
9         enumeration.options.collect_groups, ApplyOptions.GroupWise  
10    ):  
11         input = enumeration.input  
12         function = enumeration.function  
13  
14         # FunctionExprs that may reorder data  
15         if function == FunctionExpr.Reverse:  
16             is_reorder = True  
17         elif isinstance(function, FunctionExpr.Random):  
18             method = str(function.method)  
19             is_reorder = method == "shuffle"  
20         else:  
21             is_reorder = False  
22  
23         if is_reorder:  
24             enumeration = input[0]  
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
```

¹See new changes with `git diff f5bb719..12e5590c rust/src/transformations/make_stable_lazyframe/truncate/matching/mod.rs`

```

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. 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 `per_group` truncation predicate.

The `per_group` predicate is only unambiguously identified This check happens on line ??.

Line ?? checks that `partition_by` is a singleton of the `identifier`, meeting the conditions of the postcondition.

We now check whether the truncation predicate is well-defined, on lines ?? and ??.

Finally, line ?? extracts the grouping columns from the predicate.

This predicate corresponds to a `num_groups` truncation predicate, because the over expression groups by the `identifier` column, and within each group, a dense ranking is applied to unique combinations of the grouping columns. If the only rows kept are those corresponding to grouping keys assigned dense ranks less than `threshold`, then each user identifier will have at most `threshold` unique combinations of the grouping columns after filtering by the predicate.

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