fn get_margin

Michael Shoemate

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

Proves soundness of `get_margin` in `mod.rs` at commit f5bb719 (outdated[1]).

`get_margin` returns a `Margin` for a given set of grouping columns (`by`) whose descriptors are no more restrictive than what is known in `FrameDomain`.

1 Hoare Triple

Precondition

Compiler-verified

Types matching pseudocode.

Human-verified

None

Pseudocode

```python
from math import prod

def get_margin(domain: FrameDomain, by: set[Expr]) -> Margin:
    margin = next(  #
        (m for m in domain.margins if m.by == by),
        Margin.by(by)
    )

    subset_margins = [  #
        margin
        for margin in domain.margins
        if margin.by.issubset(by)
    ]

    margin.max_length = min(  #
        m.max_length
        for m in subset_margins
        if m.max_length is not None
    )

    all_max_groups = [  #
        (m.by, m.max_groups)
        for m in domain.margins
        if m.max_groups is not None
    ]
```

[1]See new changes with `git diff f5bb719..bf9b9f8 rust/src/domains/polars/frame/mod.rs`

```
26     max_groups_covering = find_min_covering(grouping_columns, all_max_groups)
27     if max_groups_covering is not None:
28         margin.max_groups = prod(v for _, v in max_groups_covering)
29
30     all_invariants = (  #
31         m.invariant
32         for m in domain.margins
33         if by.issubset(m.by)
34     )
35     margin.invariant = max(all_invariants, key={None: 0, "Keys": 1, "Lengths": 2}.get)
36
37     return margin
```

Postcondition

Returns a `Margin` that describes properties of members of `domain` when grouped by `by`.

2 Proofs

Proof. On line 4, `margin` is either a valid margin descriptor for `by`, by the definition of `domain`, or it is the default margin, which is a valid margin descriptor for all potential datasets.

We now update descriptors based on other information available in `domain`.

On line 9, `subset_margins` is the subset of margins spanned by `by`. Then 15 assigns the smallest known descriptors over any margin spanning a subset of the grouping columns.

If `max_length` is known about a coarser data grouping (when grouped by fewer columns), then these descriptors still apply to a finer data grouping, as group length can only decrease when more finely splitting data. Therefore `max_length` remains valid after mutation.

Line 21 retrieves all known `max_groups` descriptors. There are no manual preconditions to `find_min_covering`, therefore we claim the postcondition, that the output is a covering for `by`.

The number of groups can be no greater than the cardinality of the cartesian product of the keys of each grouping column. Therefore the code finds a set of `max_groups` descriptors that covers the grouping columns, and then updates `margin` to the product of the covering.

On an aside, for utility, while the covering found may not be the smallest, the greedy algorithm will always choose a singleton cover if it is available, therefore this update to the descriptor cannot increase the descriptor.

Finally, on line 30, `all_invariants` contains invariants for any margin finer than `by`. If group keys and/or lengths are known for a finer grouping, then they are also valid for a coarser grouping. Therefore `invariants` is updated to the strongest invariant for any grouping as fine or finer than `by`.

Since the initial margin (4) was valid, and all updates have also been shown to be valid, `get_margin` returns a `Margin` that describes properties of members of `domain` when grouped by `by`. ☐