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Proves soundness of get_margin in mod.rs at commit f5bb719 (outdated[1]).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 whose descriptors are no more restrictive than what is known in FrameDomain.

1 Hoare Triple

Precondition

Compiler-verified

- Argument domain is of type FrameDomain
- Argument grouping_column is of type BTreeSet<String>

Human-verified

None

Pseudocode

```
def get_margin(domain: FrameDomain, grouping_columns: set[str]) -> Margin:
    margin = domain.margins.get(
        grouping_columns, Margin.default()
    )  #

    subset_margins = {  #
        by: margin
        for by, margin in domain.margins.items()
        if by.issubset(grouping_columns)
    }

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

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

```

[1] See new changes with git diff f5bb719..4724f4c1 rust/src/domains/polars/frame/mod.rs

```
24     from math import prod
25 
26     all_mnps = {  #
27         by: m.max_num_partitions
28         for by, m in domain.margins.items()
29         if m.max_num_partitions is not None
30     }
31     mnps_covering = find_min_covering(grouping_columns, all_mnps)
32     if mnps_covering is not None:
33         margin.max_num_partitions = prod(mnps_covering.values())
34 
35     all_mips = {  #
36         by: m.max_influenced_partitions
37         for by, m in domain.margins.items()
38         if m.max_influenced_partitions is not None
39     }
40     mips_covering = find_min_covering(grouping_columns, all_mips)
41     if mips_covering is not None:
42         margin.max_influenced_partitions = prod(mips_covering.values())
43 
44     all_infos = (  #
45         m.public_info
46         for by, m in domain.margins.items()
47         if grouping_columns.issubset(by)
48     )
49     margin.public_info = max(all_infos, key={None: 0, "Keys": 1, "Lengths": 2}.get)
50 
51     return margin
```

Postcondition

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

2 Proofs

Proof. On line 4, `margin` is either a valid margin descriptor for `grouping_columns` 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 6, `subset_margins` is the subset of margins spanned by `grouping_columns`. Then 12 and 18 assign the smallest known descriptors over any margin spanning a subset of the grouping columns.

If `max_partition_length` or `max_partition_contributions` is known about a coarser data grouping (when grouped by fewer columns), then these descriptors still apply to a finer data grouping, as partition length or per-partition contributions can only decrease when more finely splitting data.

Therefore `max_partition_length` and `max_partition_contributions` remain valid after mutation.

Line 26 retrieves all known `max_num_partitions` descriptors. There are no manual preconditions to `find_min_covering`, therefore we claim the postcondition, that the output is a covering for `grouping_columns`.

The number of partitions can be no greater than the cardinality of the cartesian product of the partitions for each of the grouping keys. Therefore the code finds a set of `max_num_partitions` 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.

The same logic applies when updating the descriptor for `max_influenced_partitions`. Therefore `max_num_partitions` and `max_influenced_partitions` remain valid after mutation.

Finally, on 44, `all_infos` contains descriptors for grouping key invariants for any margin that includes `grouping_columns`. If partition keys and/or lengths are known for a finer partitioning, then they are also known for a coarser partitioning. Therefore `public_info` is updated to the most permissive descriptor for a partitining as fine or finer than `grouping_columns`.

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 `grouping_columns`. □