Title: GRU-058: GROUP BY ALL vs. subqueries
Date: 2026-02-27
Author: Peter Eisentraut
Status: change proposal

References:
[Foundation IWD] 9075_9IWD17-02-Foundation_2026-01-08.pdf
[BCN-046] "GROUP BY ALL"; Jan Michels; 2025-09-05
[W30-022r4] "GROUP BY <value expression>"; Fred Zemke; 2025-12-08
[W30-001] "Minutes of the BCN WG3 Meeting"; 2025-09-28

Abstract

Reconsider the interaction of GROUP BY ALL and subqueries.

1 Introduction

1.1 The problem

[BCN-046] introduced GROUP BY ALL (<group by shorthand>). To summarize, the effect of this clause is to expand to a GROUP BY list that contains the select list items that do not contain (for some value of "contain") aggregate functions. For example,

```
select a, sum(b) from t1 group by all;
```

expands to

```
select a, sum(b) from t1 group by a;
```

because the select list item "sum(b)" contains an aggregate function.

The original wording in paper [BCN-046] is:

> Let *NGBS* be the number of <derived column>s in *SL1* that do not contain an <aggregate function>.

This was amended on the fly (minutes in [W30-001]) to change "contain" to "directly contain", and so the current wording is:

> Let *NGBS* be the number of <derived column>s in *SL1* that do not directly contain an <aggregate function>.

At the time, this didn't actually matter because grouping by expressions, let alone expressions involving subqueries, was not allowed.

[W30-022r4] added support for grouping by <value expression>s, which includes subqueries. So now we can reconsider how this should work.

Consider the following setup:

```
create table t1 (a int, b int);
```

```
insert into t1 values (1, 10), (1, 11), (2, 20);
create table t2 (x int);
insert into t2 values (1);
```

Now consider the following query:

```
select a, (select sum(b) from t2) from t1 group by all;
```

Intuitively, this should be equivalent to the query from above:

```
select a, sum(b) from t1 group by all;
```

which is equivalent to

```
select a, sum(b) from t1 group by a;
```

and so one might expect this query to be equivalent to

```
select a, (select sum(b) from t2) from t1 group by a;
```

But because of the language "directly contains" — the second select list item does not "directly contain" an aggregate function — this query is actually equivalent to

```
select a, (select sum(b) from t2) from t1 group by a, (select sum(b) from t2);
```

which is not what we want.

But it would also not be sufficient to change back the "directly contains" to just "contains". Consider this query:

```
select a, (select sum(x) from t2) from t1 group by all;
```

The subquery is unrelated to the outer query and is essentially a constant, so this should be the same as

```
select a, (select sum(x) from t2) from t1 group by a, (select sum(x) from t2);
```

Therefore, we need to look for aggregate functions that refer to the query on the same level as the GROUP BY ALL.

This is what I'm proposing to implement here.

The following CD comment exists for this:

74.	P02-DEU-130		2-Minor Technical	*P02-07.13, <group by clause>*	After applying WG3:W30-022r4: Since grouping by subqueries is now allowed, review whether syntax rule 3) a) should still say "directly contain". Solution None provided with comment.

1.2 Solution sketch

The term to identify an aggregate function that applies to the query at a given level is *aggregation query*, defined for a <set function specification>. <set function specification> is defined as follows (subclause 6.9):

```
<set function specification> ::=
    [ <running or final> ] <aggregate function>
  | <grouping operation>

<running or final> ::=
    RUNNING | FINAL

<grouping operation> ::=
  GROUPING <left paren> <value expression>
      [ { <comma> <value expression> }... ] <right paren>
```

So roughly speaking, a <set function specification> is either an <aggregate function> or the special GROUPING(...) function.

This also means that

```
select a, grouping(a) from t1 group by all;
```

means

```
select a, grouping(a) from t1 group by a, grouping(a);
```

which seems surprising. (But grouping() is normally using in queries involving GROUPING SETS, so it does not appear together with GROUP BY ALL in practice.)

The other place where <aggregate function> is used is as part of <window function>. This would mean that window functions interact with GROUP BY ALL differently depending on whether the <window function type> is <aggregate function> or one of the other types.

We can fix both of these issues by using <set function specification> instead of <aggregate function>. And then we can refer to the aggregation query of the <set function specification> and restrict our consideration to those <set function specification>s whose aggregation query is the query that the GROUP BY ALL applies to.

2 Proposal for [Foundation IWD]

2.1 Changes to subclause 7.13, "<group by clause>"

1. Change syntax rule 3) a) as shown here:

> a) Let *NGBS* be the number of <derived column>s in *SL1* that do not ~~directly~~ contain a~~n <aggregate function>~~ <set function specification> whose aggregation query is *QS*.

3 Disposition of CD consultation comments

Close comment #74, P02-DEU-130.

4 Checklist

1.	Interactions with other concurrent proposals identified and editorial assistance	GRU-031?

	given	
2.	Concepts	n/a
3.	Access Rules	n/a
4.	Conformance Rules, including the Annex A for rules not in a Conformance Rules section and the "Implied feature relationships" table	n/a
5.	Check that any removed non-terminals, subclauses, defined terms, and the like are no longer referenced.	n/a
6.	BNF production split affecting immediate containment	n/a
7.	Lists of statements by category	n/a
8.	Collation coercibility determination for changes related to character strings	n/a
9.	Closing Possible Problems or Language Opportunity when a proposal resolves them	none
10.	Any new Possible Problems clearly identified	none
11.	Reserved and non-reserved keywords	n/a
12.	Implementation-defined and –dependent features clearly identified, and descriptions provided	n/a
13.	Information and Definition Schemata	n/a
14.	Incompatibilities Annex (for 2nd and subsequent editions of the document)	n/a
15.	Table of identifiers used by diagnostics statements	n/a
16.	Embedded SQL bindings and host language implications	n/a
17.	Dynamic SQL issues: including Dynamic descriptor areas	n/a
18.	CLI impact	none
19.	PSM impact	none
20.	MED impact	none
21.	OLB impact	none
22.	Schemata impact	none
23.	JRT impact	none
24.	XML impact	none
25.	MDA impact	none
26.	PGQ impact	none
27.	Impact on GQL for PGQ changes	n/a

— end of the paper —