May 24, 2025

ISO/IEC JTC1/SC32/WG3

ANSI INCITS DM32

Database

Title: Implicit grouping of ALL non-aggregate columns

Author: Gerald Venzl

Project: SQL: 202X-CD

Status: Discussion Paper

1 Abstract

Several SQL implementations introduced the concept of GROUP BY ALL, which when specified, groups all columns in the SELECT statement that are not wrapped in aggregate functions. The claim is that this simplifies the syntax by allowing the column list to be maintained in a single location, the SELECT list, and prevents bugs by keeping the SELECT granularity aligned to the GROUP BY granularity by preventing duplication.

This paper aims to explore the idea of GROUP BY ALL and form a discussion of its benefits or downsides and potential introduction into the SQL standard.

2 GROUP BY ALL

The functionality itself is trivial as stated above. Instead of having to list all non-aggregation columns explicitly, the keyword `ALL` can be used in the GROUP BY clause, for example:

```
SELECT region, product, EXTRACT(YEAR FROM sale_date) AS sale_year,
SUM(amount) AS total_sales
 FROM sales
  GROUP BY ALL;
```

In the above example, the columns `region, product` do not need to be repeated in the GROUP BY clause. Instead, the user's focus can stay on the SELECT list and the desired report outputs. Furthermore, adding a new column (or removing an existing one) needs to only happen in one place, the SELECT list:

```
SELECT region, country, product, EXTRACT(YEAR FROM sale_date) AS
sale_year, SUM(amount) AS total_sales
 FROM sales
  GROUP BY ALL;
```

The benefit of GROUP BY ALL may become more apparent with longer SELECT lists, for example:

```
SELECT COUNT(*) AS total_employees, job_id, state, country, region,
SUM(salary) AS total_salary, AVG(salary) AS average_salary,
MIN(salary) AS min_salary, MAX(salary) AS max_salary
 FROM employees
  GROUP BY ALL;
```

The following SQL implementations support GROUP BY ALL:

- Databricks
- DuckDB
- Google Big Query
- Snowflake

Notably, none of the implementations state any downsides of using GROUP BY ALL.

2.1 Implicit GROUP BY

The exploration of GROUP BY ALL raises the question why a GROUP BY clause needs to be stated explicitly altogether. An alternative solution could be to transform queries with aggregate functions to queries with GROUP BY ALL, making the GROUP BY clause implicit. The standard already shows precedence of a similar concept in the `<having clause>` which states in Syntax Rule 1:

> Let HC be the <having clause>. Let TE be the <table expression> that immediately contains HC. **If TE does not immediately contain a <group by clause>, then "GROUP BY ()" is implicit**. Let T be the descriptor of the table defined by the <group by clause> GBC immediately contained in TE and let R be the result of GBC.

The author could find no concrete evidence supporting the need of a GROUP BY clause or the downsides of an implicit GROUP BY and invites members to share their opinion on the matter.

2.2 ORDER BY ALL

Some of the implementations mentioned above also support the concept of ORDER BY ALL which, when specified, orders the results based on the column positions of the SELECT list automatically. While not related to GROUP BY ALL, it is worth mentioning that the concept of automatic SELECT list inference has been adopted by some implementations for multiple clauses, for example:

```
SELECT region, product, EXTRACT(YEAR FROM sale_date) AS sale_year,
SUM(amount) AS total_sales
 FROM sales
  GROUP BY ALL
  ORDER BY ALL;
```

3 Conclusion

GROUP BY ALL simplifies the SQL syntax for the SQL writer, reader and maintainer, prevents duplication and common mistakes like missing an added SELECT list column in the GROUP BY clause. Readers of this paper are invited to share their opinions on this construct.

- End of paper -