Measures in SQL



- Composable calculations are what is missing from SQL. We propose a new kind of column, called a measure, that attaches a calculation to a table.
- Like regular tables, tables with measures are composable and closed when used in queries.
- SQL-with-measures has the power,
 - conciseness
 - and reusability
 - of multidimensional languages but retains SQL semantics.
- Measure invocations can be expanded in place to simple, clear SQL.

SUM(cost) AS MEASURE sumCost,

AS MEASURE profitMargin,

ARRAY_AGG(prodName

ARRAY_AGG(custName

AT (ALL prodName)

AS MEASURE revenueGrowthYoY,

AS MEASURE top5Products

SUM(revenue) AS MEASURE sumRevenue,
(sumRevenue - sumCost) / sumRevenue

· sumRevenue AT (SET YEAR(orderDate)

ORDER BY sumRevenue DESC LIMIT 5)

ORDER BY sumRevenue DESC LIMIT 3)

AT (SET YEAR(orderDate)

SET productId MEMBER OF top5Products

AS MEASURE top3CustomersOfTop5Products

= CURRENT YEAR(orderDate) - 1))

= CURRENT YEAR(orderDate) - 1)



Measure calculations are holographic values defined as columns with AS MEASURE.

They are context sensitive expressions.

The context transformation operator AT modifies the evaluation context.

Syntax:

expression AT (contextModifier...)

contextModifier ::= WHERE predicate

ALL

ALL dimension

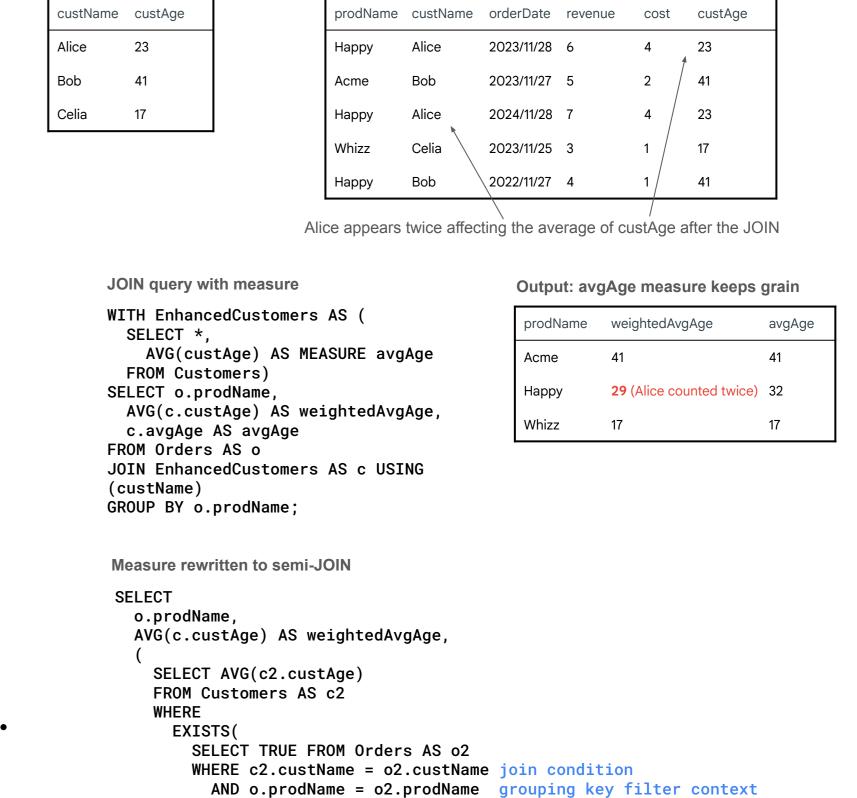
| SET dimension = [CURRENT] expression

| VISIBLE

A measure is locked to the grain of its defining table.

Joining another table does not introduce double-counting.

Measure invocations can still be rewritten into SQL without measures via a semi-join (WHERE EXISTS).



Implementing measures & context sensitive expressions as SQL rewrites

mple	Complexity	Query	Expanded query
	Simple measure can be inlined	SELECT prodName, avgRevenue FROM OrdersCube GROUP BY prodName	SELECT prodName, AVG(revenue) FROM orders GROUP BY prodName
	Join requires grain-locking	SELECT prodName, avgAge FROM OrdersCube GROUP BY prodName	SELECT o.prodName, AVG(ANY_VALUE(c.custAge) GROUP BY c.custName) FROM orders JOIN customers GROUP BY prodName → (something with GROUPING SETS)
	Period-over- period	SELECT prodName, avgAge - avgAge AT (SET year = CURRENT year - 1) FROM OrdersCube GROUP BY prodName	(something with window aggregates)
	Scalar subquery can accomplish anything	SELECT prodName, prodColor avgAge AT (ALL custState SET year = CURRENT year - 1) FROM OrdersCube	SELECT prodName, prodColor, (SELECT FROM orders WHERE <evaluation context="">) FROM orders GROUP BY prodName, prodColor</evaluation>

Topic ideas to chat about (more SQL innovations in the pipes!)

) AS avgAge

- Formal semantics for filter context transformations
- Using measures with LLM assistants
- Operators for managing grain
- Implementation strategies (efficient grain locking)
- Changing temporal grain for forecasts and timeseries
- Adding measures to nested structures typical in log files
- Sequential processing and measures



Julian Hyde @ApacheCalcite julianhyde@google.com



Conn

John Fremlin @johnfremlin google.com