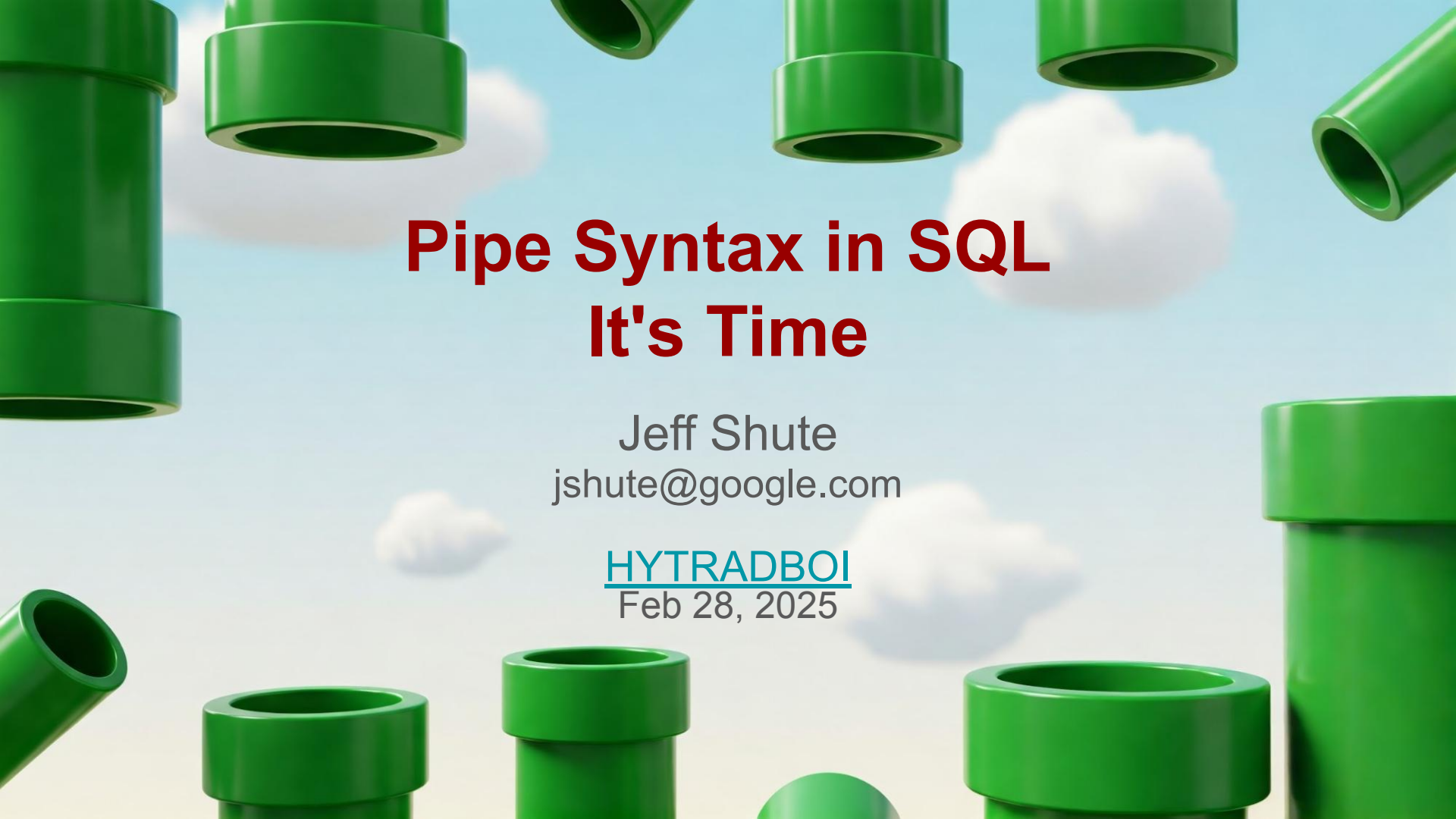


Pipe Syntax in SQL
It's Time

Jeff Shute
jshute@google.com

HYTRADBOI
Feb 28, 2025

SQL has problems?

“My biggest complaint ... is that the team never stopped to clean up SQL.

All the annoying features of the language have endured to this day.

SQL will be the COBOL of 2020, a language we are stuck with that everybody will complain about.”

— **Michael Stonebraker**
(Turing award winner)
Readings in Database Systems **(2015)**

The problem: SQL is way too hard to use!

- **For beginners**, SQL is **too hard to learn**.
- **For experts**, SQL is still awkward and annoying: **hard to write** and **hard to read**.

Many syntax problems:

- Operator ordering (`SELECT` ... `FROM` ... `WHERE` ... `GROUP BY` ...)
  - Rigid and arbitrary
  - Anything non-trivial *requires subqueries*
- "Inside-out" data flow, starting with tables or subqueries in the middle
- Verbose and repetitive:
  - List same columns in `SELECT`, `GROUP BY`, `ORDER BY`, and every subquery.
- Complex behavior (e.g., `SELECT` vs. `GROUP BY`)

It's been 50 years.
It's time to fix SQL.

It's been 50 years.
It's time to fix SQL.
Not replace it

What's great about SQL?

Declarative semantics

- Relational operators and composability

Ecosystem!

- Databases, query engines, and tools
- Familiar language, huge userbase
- Existing SQL code

We want to keep this!

- Migrations to new languages and tools are painful

"Piped Dataflow" in other languages

Works like Unix pipes:

- Collection of operators, chained together arbitrarily via "pipes"

In other modern query languages:

- Kusto (KQL), Splunk, PRQL, ...

In APIs:

- Python DataFrames, Flume / Beam, C# LINQ, ...

✅ Users generally find these easy to understand and use.

The solution: We can do piped data flow in SQL!

Apply pipe operators in any order, any number of times:

```
FROM Orders
|> SELECT o_orderpriority, o_orderdate AS date
|> SELECT *, EXTRACT(MONTH FROM date) AS month
|> WHERE month = 2
|> WHERE o_orderpriority = '1-URGENT'
|> ORDER BY date
|> LIMIT 20;
```

Query logic flows top to bottom.

But it's still declarative. Optimizers can reorder.

Why `|>`? Unfortunately, we already use `|` for bitwise OR.

Example with two-level aggregation (TPC-H query 13)

Standard SQL

```
SELECT c_count, COUNT(*) AS custdist
FROM
 (
   SELECT c_custkey, COUNT(o_orderkey) c_count
   FROM customer
   LEFT OUTER JOIN orders
       ON c_custkey = o_custkey
       AND o_comment NOT LIKE '%unusual%'
   GROUP BY c_custkey
 ) AS c_orders
GROUP BY c_count
ORDER BY custdist DESC;
```

Pipe syntax

```
FROM customer
|> LEFT OUTER JOIN orders
       ON c_custkey = o_custkey
       AND o_comment NOT LIKE '%unusual%'
|> AGGREGATE COUNT(o_orderkey) AS c_count
   GROUP BY c_custkey
|> AGGREGATE COUNT(*) AS custdist
   GROUP BY c_count
|> ORDER BY custdist DESC;
```

Pipe operators

Start a query

```
FROM ...      # Any standard FROM clause
```

Standard SQL clauses

```
|> WHERE <condition>
|> LIMIT <n> [OFFSET <n>]
|> ORDER BY <expr> [ASC|DESC], ...
|> [LEFT|...] JOIN <table> [ON / USING ...]
```

Choose columns

```
|> SELECT <expr> [[AS] alias], ...
|> EXTEND <expr> [[AS] alias], ...
|> SET <column> = <expr>, ...
|> RENAME <column> AS <name>, ...
|> DROP <column>, ...
```

Add an alias

```
|> AS <alias>
```

Aggregation

```
|> AGGREGATE <agg_expr> [[AS] alias], ...

|> AGGREGATE <agg_expr> [[AS] alias], ...
   GROUP BY <group_expr> [AS alias], ...
```

Call table-valued function

```
|> CALL tvf(args, ...)
```

Other operators

```
|> TABLESAMPLE <method> (args)
|> DISTINCT
|> PIVOT (...)
|> UNPIVOT (...)
```

(More details in BigQuery docs.)

Table-valued functions (TVFs) in pipe syntax

- **TVF call in standard syntax:**

```
SELECT *
FROM TVF( (<input_query>), args... )
```

- **Pipe form:**

```
<input_query>
|> CALL TVF(args...)
```

- **Now TVFs work like built-in pipe operators**

An example with TVFs (using BigQuery's ML TVFs)

Standard SQL

```
SELECT *
FROM
  ML.PREDICT(
    MODEL `proj.imdb_classifier`,
    (
      SELECT *
      FROM ML.PREDICT(
        MODEL `proj.nnlm_embedding`,
        (SELECT
           "Isabelle Huppert ..."
                AS embedding_input,
           7 AS reviewer_rating))
    )
  );
```

Pipe syntax

```
SELECT "Isabelle Huppert ..." AS embedding_input,
       7 AS reviewer_rating
|> CALL ML.PREDICT(MODEL `proj.nnlm_embedding`)
|> CALL ML.PREDICT(MODEL `proj.imdb_classifier`);
```

Interoperability

Pipe syntax can be mixed in anywhere a query works.

- **Mix pipe and non-pipe queries.**
  - In subqueries, VIEWs, WITH, etc
- **Add pipes on the end of any query.**
- **Use all the same tools.**

```
# Example mixing pipes and standard SQL
SELECT SUM(c_acctbal) balance
FROM (
  FROM Customer
  |> WHERE c_mktsegment = "BUILDING"
)
GROUP BY c_nationkey
|> WHERE balance > 0
|> AGGREGATE COUNT(*), SUM(balance);
```

Implementation in GoogleSQL (and OSS ZetaSQL)

- **GoogleSQL is a shared component**
  - Used in BigQuery, Spanner, F1, Procella, SQL Pipelines, ...
  - Implements all parsing and language analysis
  - Query engines consume resolved algebra, generate optimized plan
- **Enabling pipe syntax is easy!**
  - Just enable a flag
  - Query engines get the new syntax for free!

Evaluation

Nice idea, how's it working out?

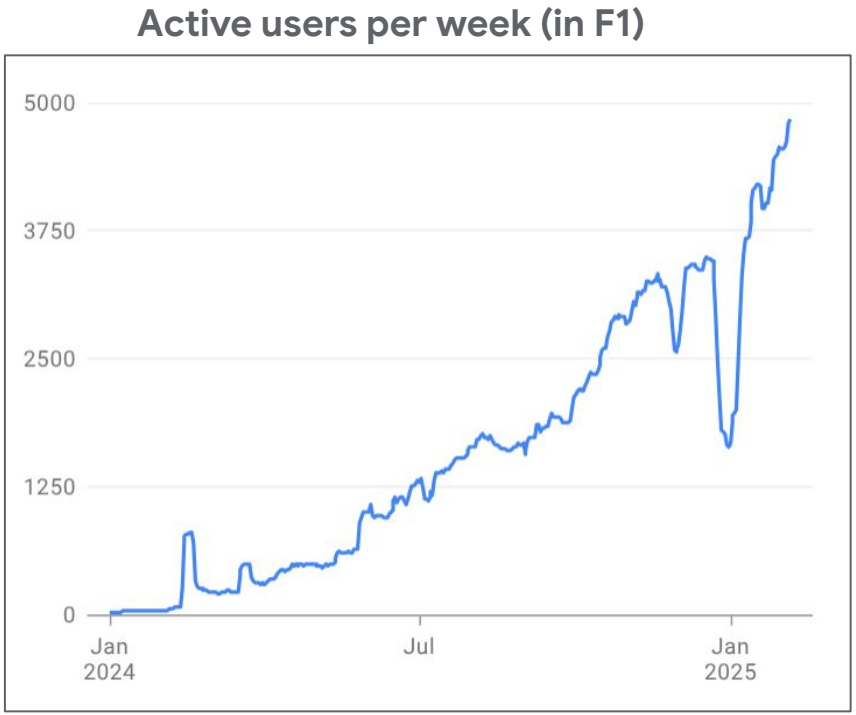
Who's it for? Everyone!

- **For SQL experts:**
  - Super easy to learn - 10 minutes, from a few examples
    - Same operators and syntax, just better structure
  - Then be immediately more productive writing and editing SQL

- **For SQL beginners, and SQL haters:**
  - Fixes many difficult or annoying parts, that confuse users or cause resistance

Usage at Google - First year

- Users see it, learn it quickly, want to use it.
- It's sticky, and spreads virally



What we're hearing from users

transformed writing SQL from something I dreaded to a pleasant experience.

Pipes is one of the **most exciting things** I have seen in a long time.

I'm starting to like SQL again!

Pipes is the **most useful feature added, perhaps ever**.

Holy cow. All my complaints about SQL just got addressed.

makes working on queries as a human _so_ much easier and understandable

It has been a **lifechanging experience** for me, making SQL _so_ much easier to write

SQL pipes are amazing and make the code so much more readable.

Conclusion

- **We can fix SQL's problems, without replacing SQL.**
  - Keep all the good things about SQL, including the ecosystem.
    - Same languages, same tools. **Just with better syntax.**
  - Easy, incremental adoption. No migrations.
- ***It's still SQL, but it's a better SQL.***

Give it a try!

- **Try it in BigQuery** (docs)
  - Open to all as of February!
- Try it in DataBricks / Spark (docs)
  - First version just released!

- Read the paper: SQL Has Problems. We Can Fix Them: Pipe Syntax In SQL (VLDB 2024)
- See OSS Zetasql
  - Query parser, analyzer, runnable reference implementation, etc.
- **For the community:** Support SQL pipe syntax in more systems?