OpenTelemetry in the Postgres Protocol

Tristan Partin

October 23, 2026

OpenTelemetry

The open standard for telemetry

Logs, metrics, traces

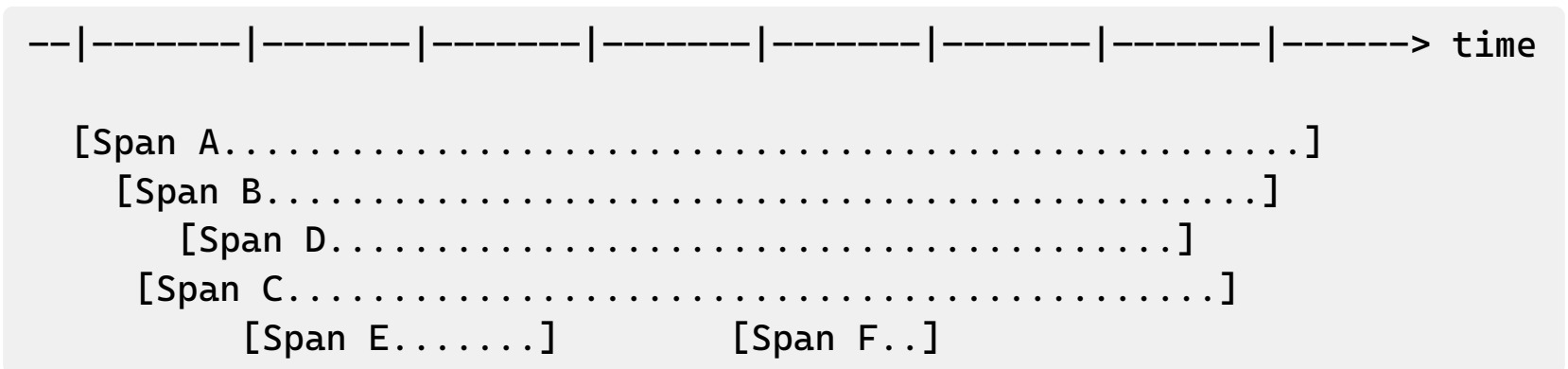
W3C Trace Context RFC: https://www.w3.org/TR/trace-context/

Spans

A span represents an operation within a transaction

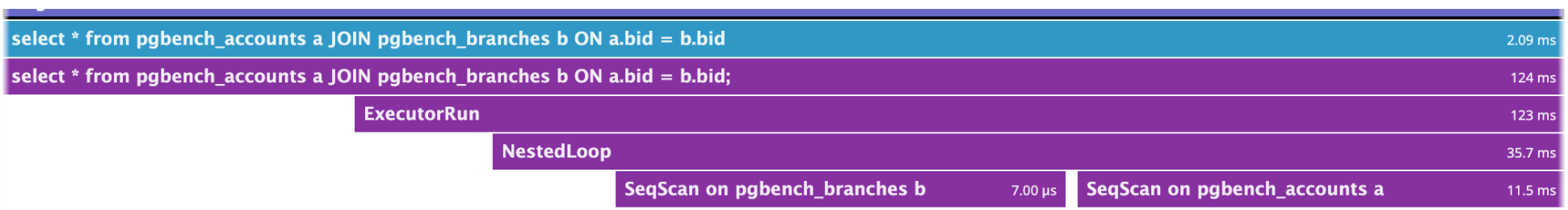
Traces

Directed acyclic graph of spans



pg_tracing

DataDog's effort to instrument PostgreSQL with OpenTelemetry tracing



https://github.com/DataDog/pg_tracing

Instrumenting Queries with Comments

```
/*traceparent='00-00000000000000000000000000000123-0000000000000123-01'*/ SELECT 1;
```

```
            trace_id             |    parent_id     |     span_id      |         span_start          |          span_end           |  span_type   | span_operation
---------------------------------+------------------+------------------+-----------------------------+-----------------------------+--------------+---------------
 00000000000000000000000000000123 | 0000000000000123 | 4268a4281c5316dd | 2024-03-19 13:46:43.97958+00  | 2024-03-19 13:46:43.980121+00 | Select query | SELECT $1;
 00000000000000000000000000000123 | 4268a4281c5316dd | 87cb96b6459880a0 | 2024-03-19 13:46:43.979642+00 | 2024-03-19 13:46:43.979978+00 | Planner      | Planner
 00000000000000000000000000000123 | 4268a4281c5316dd | f5994f9159d8e80d | 2024-03-19 13:46:43.980081+00 | 2024-03-19 13:46:43.980111+00 | Executor     | ExecutorRun
```

Instrumenting Queries with a GUC

```
BEGIN;
SET LOCAL pg_tracing.trace_context='traceparent=''00-00000000000000000000000000000005-0000000000000005-01''';
UPDATE pgbench_accounts SET abalance=1 where aid=1;
COMMIT;
```

```
select trace_id, span_start, span_end, span_type, span_operation from pg_tracing_consume_spans order by span_start;
```

```
            trace_id             |          span_start           |           span_end            |     span_type     |                      span_operation
---------------------------------+-------------------------------+-------------------------------+-------------------+-----------------------------------------------------------
 00000000000000000000000000000005 | 2024-07-05 14:24:55.305234+00 | 2024-07-05 14:24:55.305988+00 | Update query      | UPDATE pgbench_accounts SET abalance=$1 where aid=$2;
 00000000000000000000000000000005 | 2024-07-05 14:24:55.305266+00 | 2024-07-05 14:24:55.30552+00  | Planner           | Planner
 00000000000000000000000000000005 | 2024-07-05 14:24:55.305586+00 | 2024-07-05 14:24:55.305906+00 | ExecutorRun       | ExecutorRun
 00000000000000000000000000000005 | 2024-07-05 14:24:55.305591+00 | 2024-07-05 14:24:55.305903+00 | Update            | Update on pgbench_accounts
 00000000000000000000000000000005 | 2024-07-05 14:24:55.305593+00 | 2024-07-05 14:24:55.305806+00 | IndexScan         | IndexScan using pgbench_accounts_pkey on pgbench_accounts
 00000000000000000000000000000005 | 2024-07-05 14:24:55.649757+00 | 2024-07-05 14:24:55.649792+00 | Utility query     | COMMIT;
 00000000000000000000000000000005 | 2024-07-05 14:24:55.649787+00 | 2024-07-05 14:24:55.649792+00 | ProcessUtility    | ProcessUtility
 00000000000000000000000000000005 | 2024-07-05 14:24:55.649816+00 | 2024-07-05 14:24:55.650613+00 | TransactionCommit | TransactionCommit
```

A Problem

What happens between the time a query leaves the application and reaches the database?

A Problem

How much time did `pgbouncer` spend getting a connection from the pool?

Solution 1

`TraceContext` protocol message

```
struct TraceContext {
  const char *traceparent;
  const char *tracestate;
};
```

Solution 2

Query headers

Query Headers

```
org.postgresql/content-length: ...
io.opentelemetry/traceparent: 00-00000000000000000000000000000123-0000000000000123-01
io.opentelemetry/tracestate: rojo=00f067aa0ba902b7,congo=t61rcWkgMzE

SELECT 1;
```

pg_stat_activity

```
SELECT jsonb_pretty(query_headers) AS qh FROM pg_stat_activity WHERE pid = pg_backend_pid();
```

```
                                                                                                qh
-----------------------------------------------------------------------------------------------------------------------------------
 {
   "org.postgresql/content-length": "XXX",
   "io.opentelemetry/traceparent": "00-00000000000000000000000000000123-0000000000000123-01",
   "io.opentelemetry/tracestate": "rojo=00f067aa0ba902b7,congo=t61rcWkgMzE"
 }
```

Other Use Cases of Protocol Headers?

```
conn.execute(
  "SELECT * FROM pgbench_accounts",
  headers={"com.amazon.aws/rds/readonly": True}
)
```

```
org.postgresql/content-length: ...
com.amazon.aws/rds/readonly: true

SELECT 1;
```

Conclusion

If the general concept of OpenTelemetry in Postgres is interesting to you, let's get in touch. If you have a use case for protocol headers, let me know.