# Bulk inserts with PostgreSQL

## Four methods for efficient data loading

**Ryan Booz, Developer Advocate**

December 1, 2021

# Agenda

01 Death by a thousand INSERT's

02 4 methods for bulk loading data

03 Demo

04 SDK Frameworks

# 01
# Death by a thousand INSERTs

# How often do you load a lot of data into PostgreSQL?

01011

10011

{JSON}

C,S,V

00001

11001

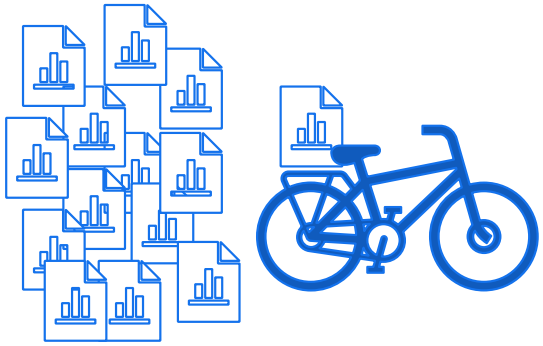
10110

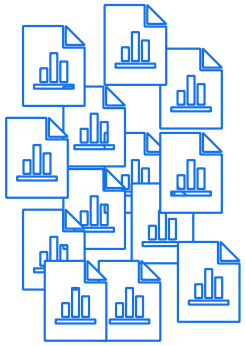
```
# "reader" = 1,000 rows
for row in reader:
            cur.execute(
            "INSERT INTO test_insert VALUES (%s, %s, %s)",
            row
        )
```

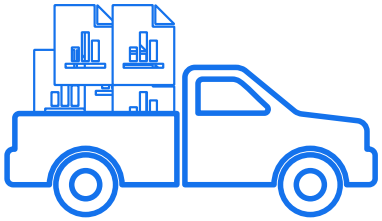
SLOW!

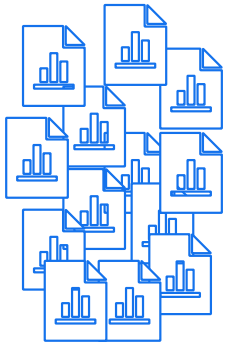
## INSERTs are slow because...

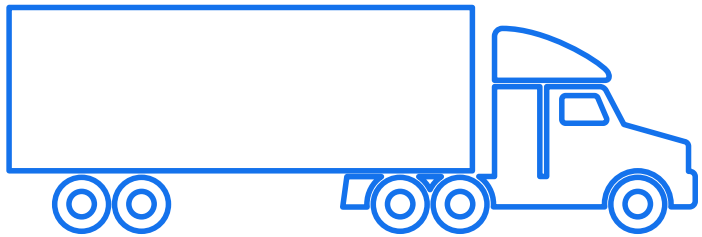
- ...every statement incurs overhead
  - Network
  - Parsing
  - Planning
  - Locks
  - Execution
  - Response
- Even local application --> DB has overhead

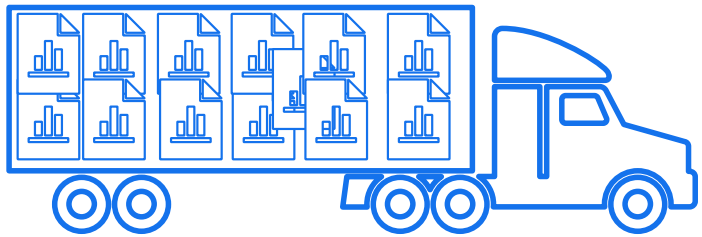


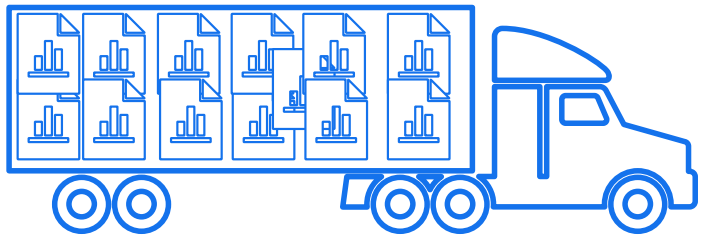


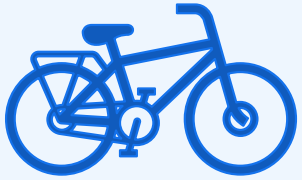


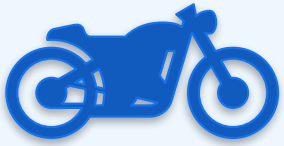


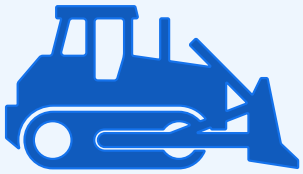












# 02

# Four methods for bulk loading data

# Multi-valued INSERT

## Multi-valued INSERT pseudocode

```
INSERT INTO table VALUES (1,2,3)
   ,(4,5,6)
   ,(7,8,9)
   ,(10,11,12)
   ,...;
```

```
# "reader" = 1,000 rows
  sql = "INSERT INTO test_insert VALUES
  next(reader) # Skip the header row.
  for row in reader:
      batch_count += 1
      sql += "('{}', {}, {}),".format(*row)

      if batch_count == 500:
         cur.execute(sql[:-1])
         batch_count = 0
         sql = "INSERT INTO test_insert VALUES "
```

# Multi-valued INSERT

- Little extra programming effort
- Supported regardless of the driver or language (even dynamic pl/pgsql)
- Moderately faster
- Generally requires batching subsets of rows

# ARRAY values INSERT

# ARRAY values INSERT pseudocode

```
INSERT INTO test_insert(time,tempc,cpu)
  SELECT * FROM unnest(
       $1::timestamptz[],
       $2::integer[],
       $3::float8[]
     ) a(a,b,c)
  ON CONFLICT DO NOTHING;
```

```
# converting column data to list
    date = data['time'].tolist()
    tempc = data['tempc'].tolist()
    cpu = data['cpu'].tolist()

    i=0
    batch_size=2000
    batch_end=batch_size
    total_length = len(date)
    while batch_end < total_length:
        cur.execute(
           "INSERT INTO test_insert SELECT * FROM
           unnest(%s::timestamptz[],%s::int[],%s::double precision[])
            a(t,v,s)
           ON CONFLICT DO nothing;",
          (date[i:batch_end],tempc[i:batch_end],cpu[i:batch_end]))
        i=batch_end
        batch_end+=batch_size
        conn.commit()
```

# ARRAY values INSERT

- Can Faster than multi-valued INSERT in some cases
- Avoids 65535 parameter limit 😀
- YMMV with language support for PostgreSQL arrays
- Caution: May not handle custom types correctly

COPY

# COPY*

- Preferred, optimized tool for PostgreSQL bulk load
- Reads from files or STDIN
- Paths are local to PostgreSQL server
- Can also pull data out to a file
- Not in the SQL standard

*Not **`psql \copy`** (but closely related)

# COPY Limitations

- Single transaction
- Single threaded
- No progress feedback prior to PG14
  - **`pg_stat_progress_copy`** view
  - In **`psql`**, run every second: `SELECT * FROM pg_stat_progress_copy \watch 1`
- No failure tolerance - stops on first error
  - Failed import takes space and leaves rows inaccessible 😕
- Minimal format configuration

**COPY does one job extremely well: *importing/ exporting data fast!***

# Unlogged Tables

# UNLOGGED Tables - Caution!

- Data inserted into UNLOGGED tables is not written to the Write-ahead Log (WAL)
- Eliminates some of that INSERT overhead
- Data is not crash safe
- Not accessible on replication servers (requires WAL)
- Available with CREATE and ALTER table

## Why use UNLOGGED tables?

- You data process can accept the risk of loss for increased INSERT
- Staging tables for ETL processes
- Intermittent, repeatable work (easy to redo)

# 03

# Demo

# 04

# SDK Frameworks

# What to look for in language SDKs

- Specific COPY support (including BINARY COPY format)
- Multi-valued and batching functions
- Is auto-commit enabled by default?
- Avoid parameterized query formatter
  - Use the ARRAY trick if the SDK only uses parameterized queries

# What questions do you have for me?

# Thanks!

Me: Ryan Booz

Twitter: @ryanbooz

Slack: slack.timescale.com