youtube.com/@ryanbooz/playlists

Agenda

redgate

WORDLE

A DAILY WORD GAME

A few functions are only included with PostgreSQL >=14

Advent of Code Examples

Dec 07 Puzzle (2023)

- Listing of 5 playing cards by face value

- Bid value to use later in puzzle

- Order of cards in hand is important

Dec 07 Puzzle (2022)

- List of commands to move around file system

- Translate commands into disk hierarchy and file sizes within each directory

redgate

ETL vs ELT

redgate

Convert non-relational data into relational, tabular data.

redgate
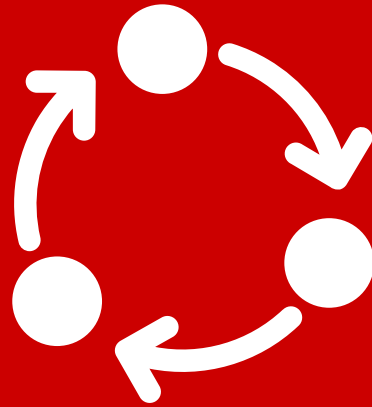
ETL vs ELT

Extract, Transform, Load

- External

- Not SQL focused

Extract, Load, Transform

- Internal

- SQL focused

redgate

Why Has ETL Been So Popular?

- External tools brought specialized functionality

- Databases didn't speak web languages well
  - ie. XML or JSON

- Specialized tools = specialized jobs

redgate

Keep processing close to the data for faster iteration

ELT in PostgreSQL

- Retain transactional consistency and control

- Plethora of functions to process and transform data

  - Regex

  - JSON

  - String

- Array and JSON data types often useful

redgate

Inserting Data

- Quickly dump data to tables with simple schema

- Post-process JSON, XML, strings, arrays, etc.

- Use COPY:
  - most supported method of getting data in quickly
  - CSV or custom delimiters

- Use code:
  - work in batches of rows to reduce transaction overhead

redgate

COPY vs \copy

- COPY is a PostgreSQL command, not SQL standard

- `COPY` requires files local to the server

- My examples primarily use `psql \copy` command

- `\copy` streams local files to `COPY's` `STDIN`

redgate

COPY Caution

- Requires correct column order, matching data types, and clean data (no conversion)

- Options like pgloader overcome some limitations

  - pre-checks on certain columns of data

redgate

https://bit.ly/ryan-booz-2023-talks
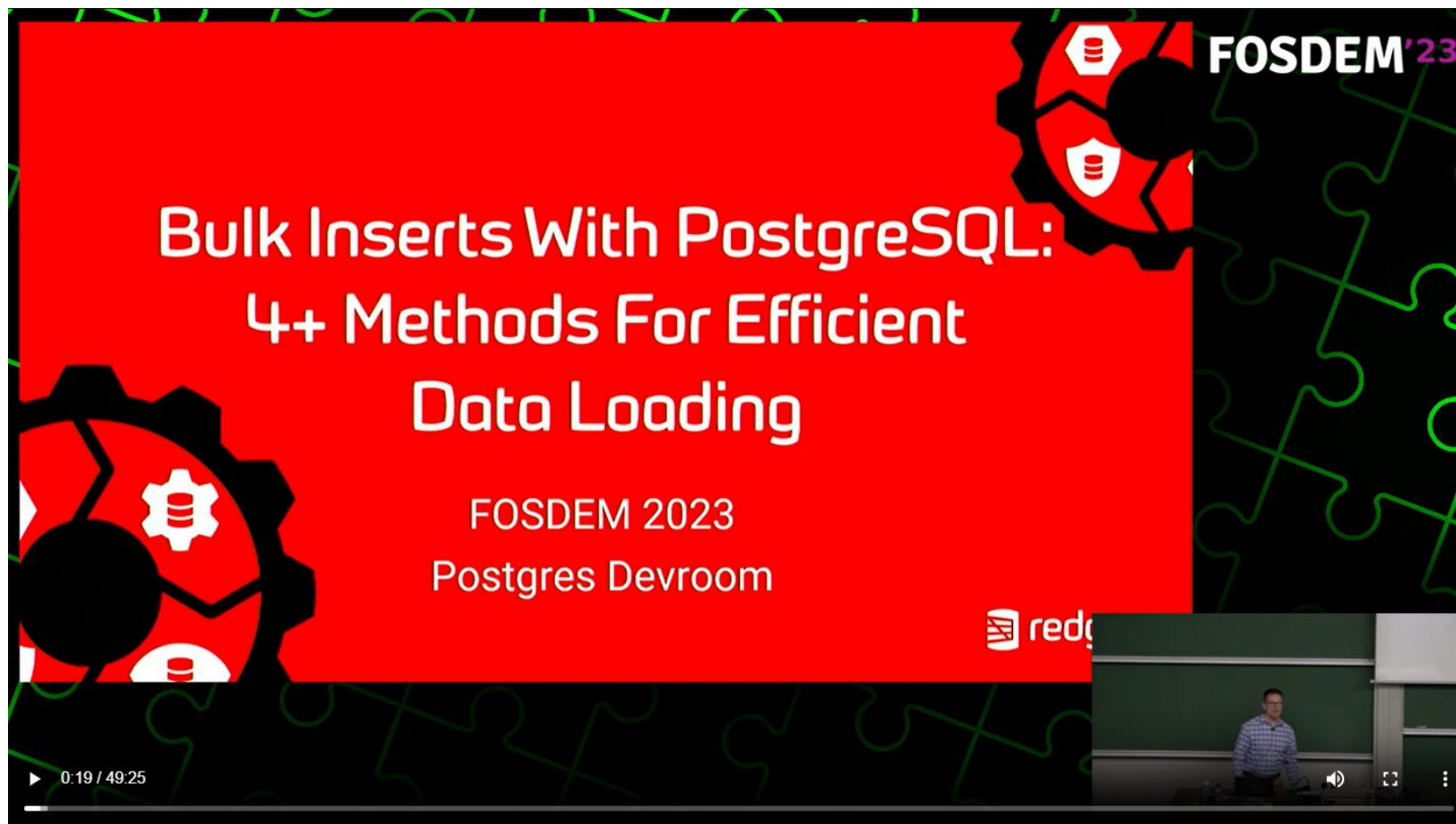
Data Import Rules – K.I.S.S

1 Create a generated ID for ordering later if needed

2 Add a timestamp column if it's time-series data

3 Pre-processes what makes sense, but don't go overboard

redgate

K.I.S.S. – Advent of Code

```sql
create table dec07 (
    id integer generated by default as identity,
    lines text
);

-- COPY the text into the appropriate columns
\COPY dec07 (lines) FROM input.txt NULL '';
```

redgate

K.I.S.S. – Wordle

```
{
    "data": {
        "author_id": "395950789",
        "created_at": "2022-01-15T03:09:22.000Z",
        "id": "1482188130191122123",
        "text": "Wordle 209
3/6\n\n\u2b1b\u2b1b\u2b1b\ud83d\udfe9\u2b1b\n\u2b1b\u2b1b\ud83d\udfe8\u2b1b\ud83d\udfe8\n\ud83d\udfe9\ud83d\udfe9\ud83d\udfe9\ud8
3d\udfe9\ud83d\udfe9"
    },
    "includes": {
        "users": [
            {
                "id": "395950789",
                "location": "Cali",
                "name": "Hall & Oates Enjoyer",
                "username": "wordlemaster",
                "verified": false
            }
        ]
    },
    "matching_rules": [
        {
            "id": "1482188147178053123",
            "tag": "wordle"
        }
    ]
}
```

K.I.S.S. – Wordle

```sql
CREATE TABLE tweets_raw(
    ts timestamptz NOT NULL,
    tweet_id bigint NOT NULL,
    tweet_raw JSONB NOT null,
);
```

K.I.S.S. – Wordle

```sql
CREATE TABLE wordle_tweet (
    ts timestamptz NOT NULL,
    created_at timestamptz NOT NULL,
    author_id bigint NOT NULL,
    author_handle TEXT NOT NULL,
    author_verified bool,
    author_location TEXT,
    tweet_id bigint NOT NULL,
    tweet TEXT NOT null,
    game int NULL,
    guess_total int null
);
```

redgate

Derived Table for Quick Prototyping

```sql
SELECT regexp_split_to_table($$32T3K 765
T55J5 684
KK677 28
KTJJT 220
QQQJA 483$$,'\n') lines;

lines      |
-----------+
32T3K 765|
T55J5 684|
KK677 28 |
KTJJT 220|
QQQJA 483|
```

Dollar Quoting

```sql
-- Dollar Quoting
SELECT $$32T3K 765
T55J5 684
KK677 28
KTJJT 220
QQQJA 483$$;

-- Useful with functions like format()
SELECT format($$This is demo is for %s!!$$, 'Scale 21x');
```

03/05
SQL/PostgreSQL Building Blocks

redgate

WITH ORDINALITY

- Set Returning Function in FROM clause

- Adds ordinal value of each emitted row

- Faster than ROW_NUMBER()
  - No need to rescan the entire table

- Retains order without an ORDER BY

redgate

CROSS JOIN

- For every row on the left table, iterate all rows on right table

- Output is the product of both sets

redgate

CROSS JOIN

```sql
SELECT * FROM
     generate_series(1,2) gs1
     CROSS JOIN generate_series(1,4) gs2;


gs1|gs2|
---+---+
  1|  1|
  1|  2|
  1|  3|
  1|  4|
  2|  1|
  2|  2|
  2|  3|
  2|  4|
```

redgate

CROSS JOIN LATERAL

- When right-hand table is an SRF the join is implicitly a LATERAL

- Functions can reference any column of output tables the left

- Allows chained queries to "reach back" to previous result sets for data

redgate

CROSS JOIN LATERAL

```sql
SELECT * FROM
      generate_series(1,2) gs1,
      generate_series(1,4) gs2;


gs1|gs2|
---+---+
  1|  1|
  1|  2|
  1|  3|
  1|  4|
  2|  1|
  2|  2|
  2|  3|
  2|  4|
```

reugate

SELECT T.a, CJ.b, CJ2.c FROM T

,

string_to_table(T.a, null) AS b

CJ,

string_to_array(CJ.a, null) AS c

CJ2

CROSS JOIN LATERAL

- Simplify SQL at a higher level by hiding calculations lower in the query

- Reorganize data by returning VALUES

redgate

CROSS JOIN LATERAL

```sql
SELECT id AS hand,
       t.* AS card,
       (split_part(lines,' ',2)) AS bid
FROM dec07, --> CROSS JOIN LATERAL
     string_to_table(split_part(lines,' ',1)) t;
```

redgate

CROSS JOIN LATERAL

```sql
SELECT id AS hand,
    t.card,
    t.position,
    bid
FROM dec07,
    string_to_table(split_part(lines,' ',1),null)
        WITH ORDINALITY t(card,position),
    split_part(lines,' ',2) bid;
```

redgate

CROSS JOIN LATERAL

```
...
select hm.step,
       hm.x, hm.y,
       h.x, h.y,
       t.x, t.y
from tmove tm
  join hmove hm on tm.step+1 = hm.step
cross join lateral
    (VALUES (tm.hx+hm.x, tm.hy+hm.y)) as h(x,y)
cross join lateral
    (VALUES (
        case when abs(h.y-tm.ty) = 2 then h.x
             when abs(tm.tx-h.x) <= 1 then tm.tx
             else tm.tx + hm.x end,
        case when abs(h.x-tm.tx) = 2 then h.y
             when abs(tm.ty-h.y) <= 1 then tm.ty
             else tm.ty + hm.y end
)) t(x,y)
...
```

redgate

CROSS JOIN LATERAL

```
...
select hm.step,
    hm.x, hm.y,
    h.x, h.y,
    t.x, t.y
from tmove tm
  join hmove hm on tm.step+1 = hm.step
cross join lateral
    (VALUES (tm.hx+hm.x, tm.hy+hm.y)) as h(x,y)
cross join lateral
    (VALUES (
        case when abs(h.y-tm.ty) = 2 then h.x
             when abs(tm.tx-h.x) <= 1 then tm.tx
             else tm.tx + hm.x end,
        case when abs(h.x-tm.tx) = 2 then h.y
             when abs(tm.ty-h.y) <= 1 then tm.ty
             else tm.ty + hm.y end
)) t(x,y)
...
```

reogate

04/05
Common Table Expression

redgate

Common Table Expression (CTE)

- WITH queries

- Reference the output of the query by a unique name

- Prior to Postgres 12 the CTE was materialized first
  - PG12+ planner attempts to in-line unless you add MATERIALIZED

redgate

Common Table Expression (CTE)

- Chain multiple CTEs together

- Reference any previous CTE

- Particularly helpful for query reuse (readability)

- Name output columns with parenthesis

redgate

```sql
SELECT * FROM (
        SELECT id AS hand,
        t.*,
        bid
        FROM dec07,
                string_to_table(split_part(lines,' ',1),null)
                        WITH ORDINALITY t(card,position),
                split_part(lines,' ',2) bid
) cb
JOIN (
        SELECT * FROM (VALUES  ('2',2),
                        ('3',3),
                        ('4',4),
                        ('5',5),
                        ('6',6),
                        ('7',7),
                        ('8',8),
                        ('9',9),
                        ('T',10),
                        ('J',11),
                        ('Q',12),
                        ('K',13),
                        ('A',14)) AS t(card,value)
) vals USING (card)
ORDER BY hand, position;
```

gate

```sql
WITH given_hand (hand,card,position,bid) AS (
SELECT id, t.*, bid
        FROM dec07,
        string_to_table(split_part(lines,' ',1),null)
        WITH ORDINALITY t(card,position),
        split_part(lines,' ',2) bid
),
card_converter (card,value) AS (
                VALUES  ('2',2),
                        ('3',3),
                        ('4',4),
                        ('5',5),
                        ('6',6),
                        ('7',7),
                        ('8',8),
                        ('9',9),
                        ('T',10),
                        ('J',11),
                        ('Q',12),
                        ('K',13),
                        ('A',14)
)
SELECT * FROM given_hand
        JOIN card_converter USING (card);
```

Readability is often cited as the primary benefit.
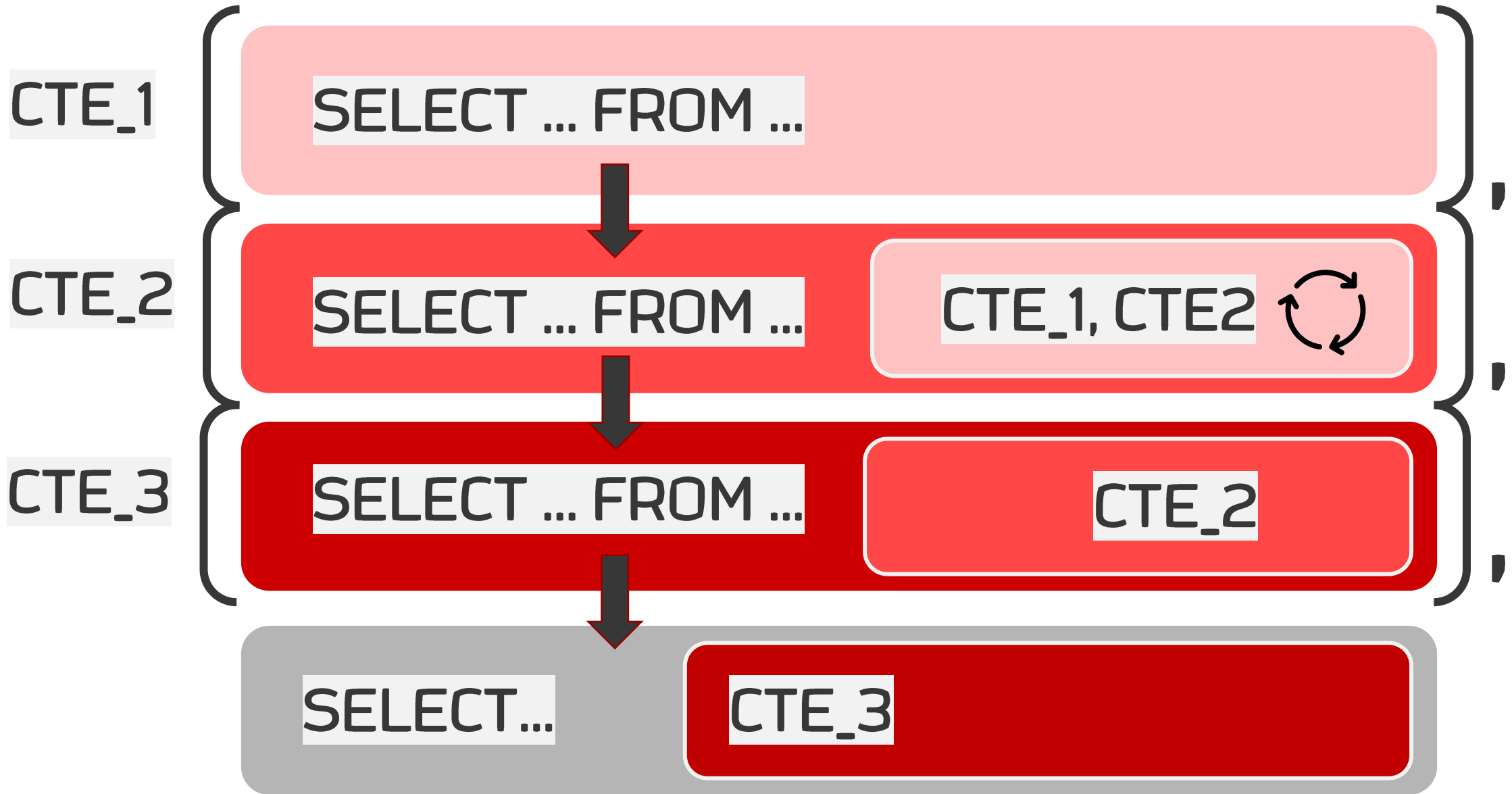
While true, increased complexity might be less performant

05/05
Recursive CTEs

Recursive CTEs

- The SQL language is declarative and batch-based

- Recursive CTEs = iterative processing using SQL

- Not otherwise be possible

- Recursive CTEs allow SQL to be Turing complete

redgate

WITH RECURSIVE

CTE_1
SELECT ... FROM ...

CTE_2
SELECT ... FROM ...   CTE_1, CTE2 ↻

CTE_3
SELECT ... FROM ...   CTE_2

SELECT...   CTE_3

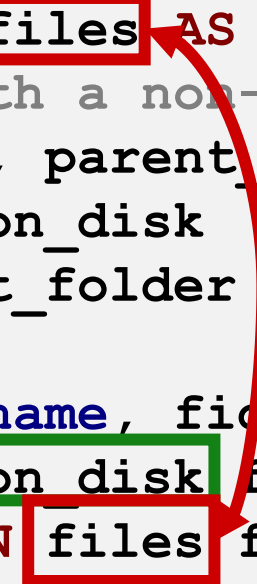| name	parent_folder	size
Folder_A		
Folder_A_1	Folder_A	
Folder_B	Folder_A	
Folder_A_2	Folder_A	
Folder_B_1	Folder_B	
File_A1.txt	Folder_A	1234
File_A2.txt	Folder_A	9876
File_B1.txt	Folder_B	4567

Recursive CTEs

```sql
WITH recursive files AS (
    -- start with a non-recursive, initial query
    SELECT name, parent_folder, SIZE FROM files_on_disk
    WHERE parent_folder IS NULL
)
SELECT * FROM files;
```

```
name        |parent_folder|size|
------------+-------------+----+
Folder_A    |             |    |
```

Recursive CTEs

```sql
WITH recursive files AS (
    -- start with a non-recursive, initial query
    SELECT name, parent_folder, SIZE
    FROM files_on_disk
    WHERE parent_folder IS NULL
UNION ALL
    SELECT fid.name, fid.parent_folder, fid.SIZE
    FROM files_on_disk fid
        INNER JOIN files f ON fid.parent_folder = f.name
)
SELECT * FROM files;
```
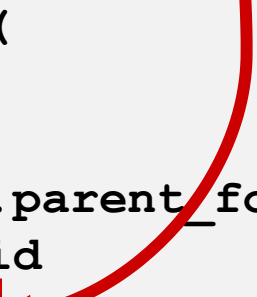
redgate

Recursive CTEs

```
name         |parent_folder|size|
-------------+-------------+----+
Folder_A     |             |    | <-- Initial query
Folder_A_1   |Folder_A     |    | <--|
Folder_B     |Folder_A     |    |    |
Folder_A_2   |Folder_A     |    |    |- Result of first join
File_A1.txt  |Folder_A     |1234|    |
File_A2.txt  |Folder_A     |6789| <--|
```

```sql
WITH recursive files AS (
    ...
UNION ALL
    SELECT fid.name, fid.parent_folder, fid.SIZE
    FROM files_on disk fid
        INNER JOIN files f ON fid.parent_folder = f.name
)
SELECT * FROM files;
```

ate

Recursive CTEs – Caution!

- Recursion continues until working table is empty

- Make sure there is an ending point (or add one!)

redgate

DEMO

redgate
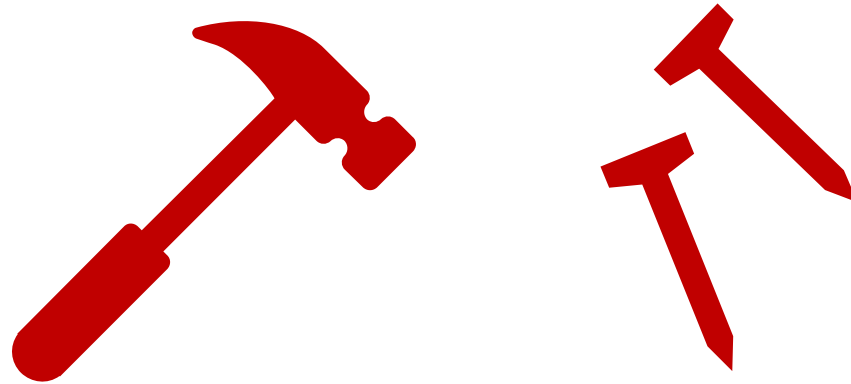
Explore More Functions

- FILTER clause

- string_to_array()

- regexp_matches()

- JSON functions

- WINDOW functions

- RANGE types

redgate

PostgreSQL Community

- Vik Fearing

- Feike Steenbergen

- David Kohn

- Sven Klemm

- John Pruitt

- Tobias Petry

- Bruce Momjain

- Andreas Scherbaum

- Ryan Lambert

- More, more, more...

redgate

What Questions do you have?

redgate

🎉 THANK YOU! 🎉

github.com/ryanbooz/presentations

redgate