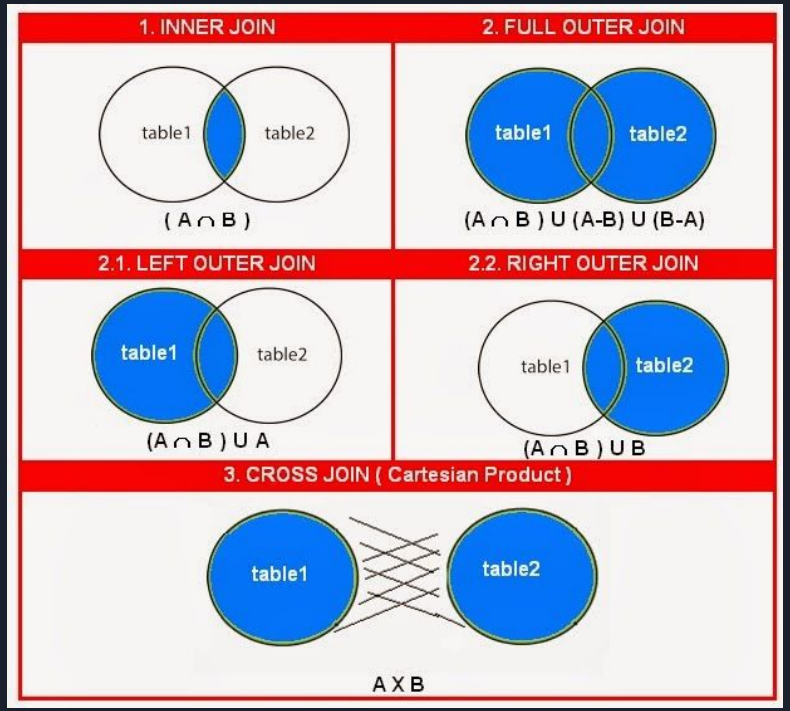
Lateral Joins

By: Femi Obiwumi

Recap/Intro



- Inner and Outer joins refer to which parts of each table will be represented in the joined set

- The lateral join effects how each table is generated
  - Allows for table2 to be generated using input from table1
- As such, the LATERAL keyword can be used on any type of join

What Does LATERAL Do?

- The LATERAL keyword is used to allow subqueries in a FROM statement to access columns in preceding subqueries.

A trivial example of `LATERAL` is

```
SELECT * FROM foo, LATERAL (SELECT * FROM bar WHERE bar.id = foo.bar_id) ss;
```

- Without the LATERAL, the sub-SELECT would be unable to access foo.bar_id.

JOIN LATERAL

- One common use for Lateral Joins is when a subquery uses columns from a different subquery to generate a result:

```
SELECT users.name,
       recc.name,
       recc.rank
FROM users CROSS JOIN LATERAL movie_rec(users.user_id) as recc(name, rank)
```

More uses for Lateral Join

Another common use for lateral joins is for generating top n features for each category:

```
SELECT geo.zipcode,
       geo.state,
       movie.name
FROM geo CROSS JOIN LATERAL
     (SELECT movie_name
      FROM streams
      WHERE geo.zipcode = streams.zipcode
      ORDER BY streams.county DESC limit 5) AS movie(name);
```

```
select d.id, to_json(array_agg(da.event_type)) activities
from developers d

join lateral (
  select event_type, developer_id
  from activities
  where developer_id=d.id
  limit 5
) da on d.id=da.developer_id

group by d.id
order by d.id;
```

```
 id |                  activities
----+-------------------------------------
  1 | ["fork","fork","fork","fork","push"]
  2 | ["pull","fork","fork"]
  3 | ["push","push","push","pull","push"]
  4 | ["push","pull","pull","pull"]
  5 | ["pull","pull","fork","push","pull"]
  6 | ["push","pull","pull","pull","push"]
  7 | ["pull","push","pull"]
  8 | ["pull","push","pull","fork","pull"]
  9 | ["pull"]
 10 | ["pull","pull","fork"]
```

Works Cited

https://www.youtube.com/watch?v=QQT8e9aewV0&t=103s

https://blog.crunchydata.com/blog/iterators-in-postgresql-with-lateral-joins

https://www.postgresql.org/docs/9.4/queries-table-expressions.html

https://heap.io/blog/postgresqls-powerful-new-join-type-lateral