

WRITING RECURSIVE QUERIES

RETRIEVING HIERARCHY DATA FROM RELATIONAL TABLES

BEN LIS

POSTGRESCONF US 2019

IS THIS TALK RIGHT FOR YOU?

What we will cover

- Why we need recursive queries
- How they work
- How to write them

What you should know

- Basic knowledge of SQL through outer joins
- General programming concepts
- Slides and SQL:

<https://github.com/benjlis/talk-writing-recursive-queries>

ABOUT ME

- Data Engineer at LEI Smart
- Writes recursive queries on corporate hierarchy data
- Initially found recursive SQL a bit confusing
- Using PostgreSQL since 2015
- Started working with Oracle databases in 1988!
- Adjunct Associate Faculty in Applied Analytics at Columbia University
- Held various technical, product, and management roles on Wall Street
- [LinkedIn](#) and [Twitter](#)

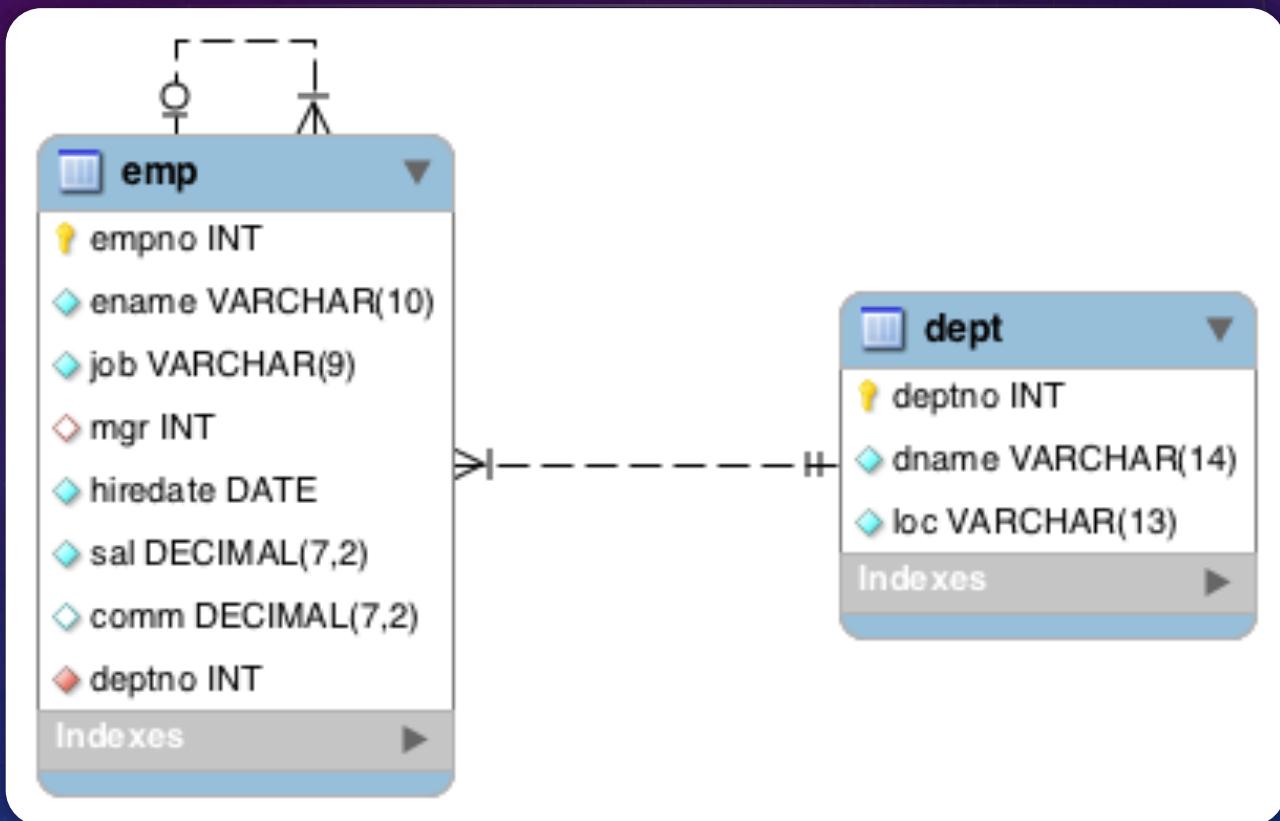
AGENDA

- Motivation
- Understanding Recursive SQL
- Writing Recursive Queries
- Next Steps
- Questions & Discussion

MOTIVATION

WHY WE NEED RECURSIVE SQL

HR DATABASE



hrex-create.sql

LET'S QUERY EMPLOYEES, THEIR BOSS & THE BOSS'S BOSS

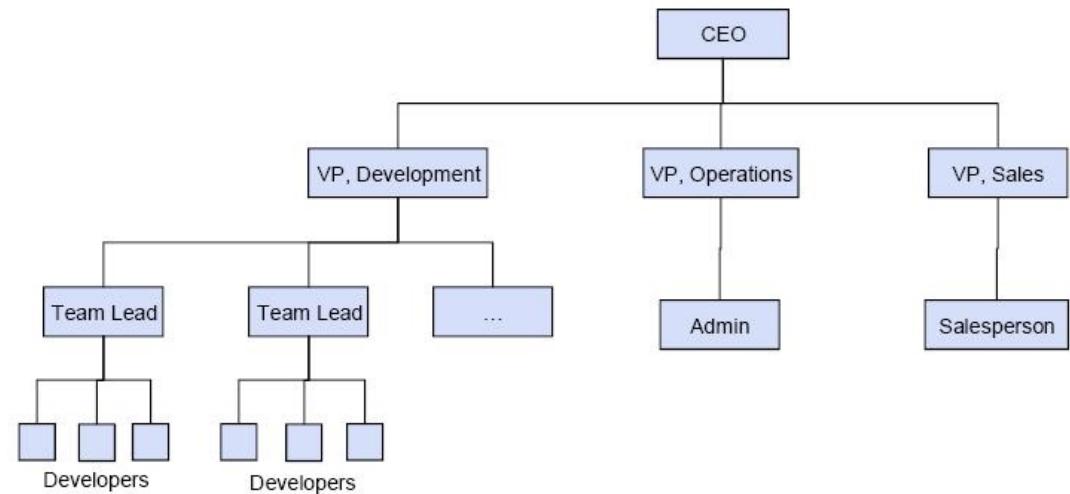
```
-- employees and their boss
select e.ename, e.job, b.ename boss
  from emp e left join emp b on (e.mgr = b.empno);

-- employees, their boss and boss's boss
select e.ename, e.job, b.ename boss, bb.ename bossboss
  from emp e left join emp b on (e.mgr = b.empno)
    left join emp bb on (b.mgr = bb.empno);
```

boss-queries.sql

LET'S QUERY EMPLOYEES' ENTIRE REPORTING LINE

HMM... HOW DO WE DO THAT?

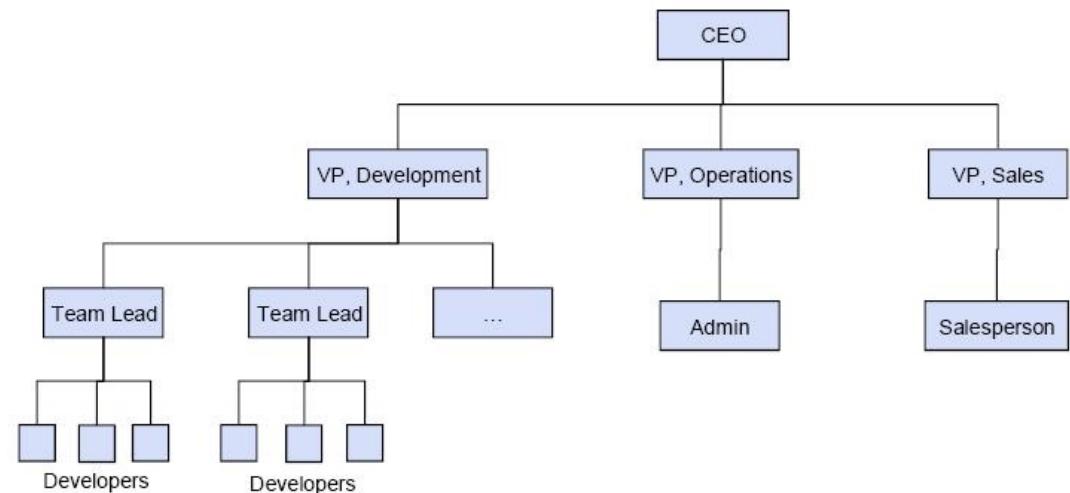


IDEA 1: EXTEND CURRENT APPROACH

“Add a few more JOINS”

Need to know the number of hierarchy levels in advance

Not realistic or robust!



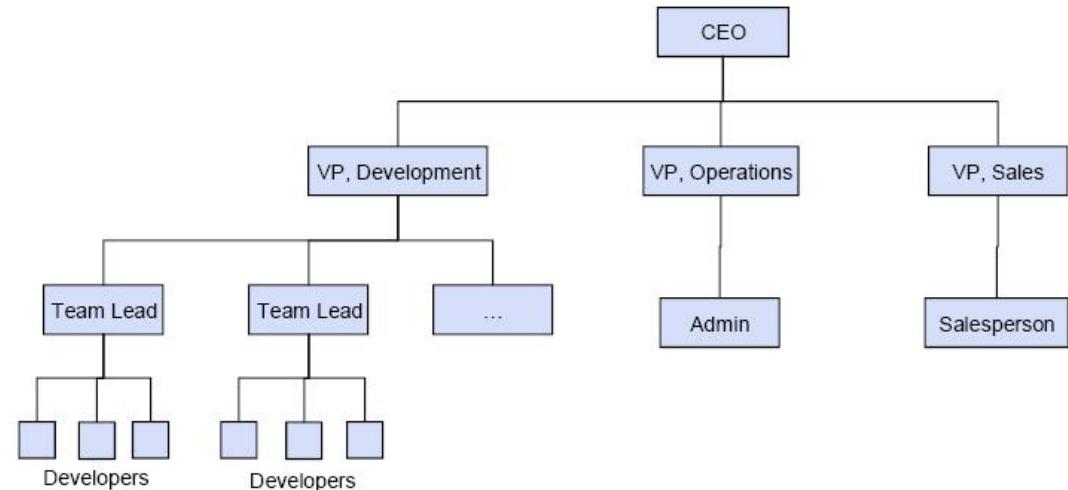
IDEA 2: WRITE A PROGRAM

Use Python, Java, PL/pgSQL, etc.

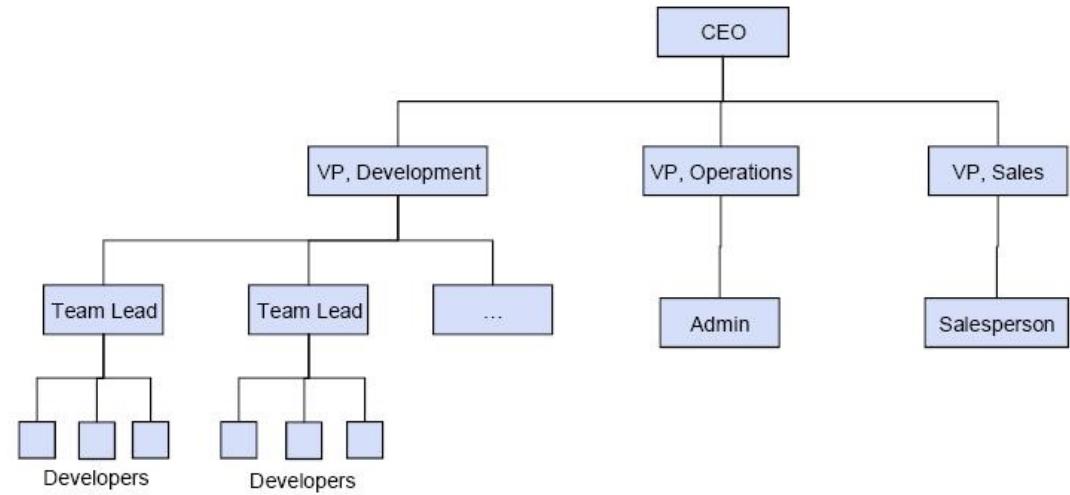
Query all employees

For each employee get their boss's record
and then their boss's boss's record, etc.
until you get a NULL

Complex and resource consumptive!



SHOULDN'T WE BE
ABLE TO WRITE
THIS QUERY IN
SQL?



WE SHOULD BE ABLE TO WRITE THIS QUERY IN SQL

- In the early days (decades) of SQL & relational databases we couldn't
- Writing a program (Idea 2) was the only solution
- Other names for the hierarchy problem:
 - Transitive Closure
 - Bill of Materials
 - Parts Explosion
- Good news: We can write this query in modern SQL using recursive SQL!

UNDERSTANDING RECURSIVE SQL

KEY IDEAS, CTE, SYNTAX, STRUCTURE & EVALUATION

RECURSIVE SQL SUPPORT



Part of ISO Standard SQL

Introduced in SQL:1999 (aka SQL 3)



Supported in

PostgreSQL

Oracle

SQL Server

DB2

KEY IDEAS

- **re·cur·sive /rə'kərsiv/**: a program or routine of which a part requires the application of the whole, so that its explicit interpretation requires in general many successive executions
- Self-referencing
 - Your Bosses = Your Manager + Your Manager's Bosses
- Recursive SQL is an iterative process
 - Think while loops
 - But SQL is declarative!!!!
- Tip: Put aside knowledge of recursion in programming
- Recursive SQL uses Common Table Expressions (CTEs)



COMMON TABLE EXPRESSION (CTE) EXAMPLE

```
with deptcomp(deptno, dname, empcnt, avgsal, totsal, totcomm, totcomp) as
  (select deptno, dname, count(emp), round(avg(sal), 2), sum(sal),
    coalesce(sum(comm), 0), sum(sal + coalesce(comm, 0)))
   from dept natural join emp
   group by deptno, dname)
select * from deptcomp where empcnt >= 5 and totcomm = 0;
```

cte.sql

- Primarily used to simplify queries
- Think of as inline view(s)

RECURSIVE CTE STRUCTURE

- Keyword: recursive
- Always a UNION or UNION ALL of
 - Non-recursive select
 - Recursive select

```
with recursive <cte-name>(<col-names>) as (
    <non-recursive select>
        union [all]
    <recursive select>
)
select ... from <cte-name> ...;
```

NON-RECURSIVE SELECT

- Evaluated 1st - where we start
- FROM clause:
 - Must not refer to <cte-name>
 - Refers to table containing hierarchy

```
with recursive <cte-name>(<col-names>) as (
    <non-recursive select>
        union [all]
    <recursive select>
)
select ... from <cte-name> ...;
```

RECURSIVE SELECT

- Evaluated 2..N times
- FROM clause:
 - Must reference <cte-name>
 - Join of the <cte-name> and table containing hierarchy
- N is a function of the data and the recursive select's where clause
- Think of DB engine dynamically adding correct # of joins to FROM

```
with recursive <cte-name>(<col-names>) as (
    <non-recursive select>
    union [all]
    <recursive select>
)
select ... from <cte-name> ...;
```

RECURSIVE CTE EVALUATION

QR = Query Results; WT = Working Table

Both initially empty

1. Execute <non-recursive select>
2. Add results to QR and WT
3. Repeat until WT is empty
 - a. Execute <recursive select> using WT data for <cte-name>
 - b. Add results to QR
 - c. Replace WT with results

```
with recursive <cte-name>(<col-names>) as (
    <non-recursive select>
    union [all]
    <recursive select>
)
select ... from <cte-name> ...;
```

WRITING RECURSIVE QUERIES

LET'S GET DOWN TO IT!

WHAT IS ADAMS REPORTING LINE?

```
with recursive rl(empno, ename, job, mgr) as
(select empno, ename, job, mgr from emp where ename = 'ADAMS'
 union all
 select b.empno, b.ename, b.job, b.mgr from rl r join emp b on r.mgr = b.empno)
select * from rl;
```

rl.sql

ALL EMPLOYEES FROM CEO TO ENTRY LEVEL

```
with recursive oc(empno, ename, job, mgr) as
(select empno, ename, job, mgr from emp where mgr is null
 union all
 select s.empno, s.ename, s.job, s.mgr from oc join emp s on oc.empno = s.mgr)
select * from oc;
```

oc.sql

INCLUDE A LEVEL

```
with recursive oc(level, empno, ename, job, mgr) as
(select 1, empno, ename, job, mgr
 from emp where mgr is null
 union all
select level+1, s.empno, s.ename, s.job, s.mgr
 from oc join emp s on oc.empno = s.mgr)
select * from oc;
```

oc-level.sql

INCLUDE A PATH

```
with recursive oc(level, empno, ename, job, mgr, rl) as
(select 1, empno, ename, job, mgr, array[ename]::text
 from emp where mgr is null
 union all
select level+1, s.empno, s.ename, s.job, s.mgr, oc.rl || array[s.ename]::text
 from oc join emp s on oc.empno = s.mgr)
select * from oc order by rl;
```

oc-path.sql

NEXT STEPS

RECOMMENDED READINGS & BEYOND HIERARCHY

RECOMMENDED READINGS

- [PostgreSQL documentation](#)
- [Fun with SQL: Recursive CTEs in Postgres – Craig Kerstiens](#)
- [Modern SQL in Open Source and Commercial Databases \(slides 45-60\) – Markus Winand](#)
- [Exporting a Hierarchy in JSON: with recursive queries - Dmitri Fontaine](#)
- [Joe Celko's Trees and Hierarchies in SQL for Smarties \(Book\)](#)

BEYOND HIERARCHY

- Hierarchy queries are the most common application of recursive SQL, but not the only application!
- Pretty wild recursive SQL queries:
 - [Solving the Traveling Salesman Problem with Postgres Recursive CTEs](#) - Periscope Data
 - [Mandelbrot Set](#)
 - [tic-tac-toe](#)
- My views:
 - Can doesn't mean should
 - Worth studying for learning advanced techniques

QUESTIONS