Recursion in SQL

Basic recursive WITH statement
SQL is not a “Turing complete” language

- Simple, convenient, declarative
- Expressive enough for most database queries
- But basic SQL can’t express unbounded computations
Example 1: Ancestors

\texttt{ParentOf(\texttt{parent,child})}

- \textit{Find all of Mary’s ancestors}
  - parents
    - Sue
    - Bob
  - grandparents
    - Fred
    - Jane
  - Three instances of \texttt{ParentOf}
    - ↑

Basic SQL Recursion
Example 2: Company hierarchy

Employee (ID, salary) ➔
Manager (mID, eID) ➔
Project (name, mgrID)

➢ Find total salary cost of project ‘X’
Example 3: Airline flights

Flight(\texttt{orig}, \texttt{dest}, \texttt{airline}, \texttt{cost})

- Find cheapest way to fly from ‘A’ to ‘B’
With \( R_1 \) As (query-1),
\( R_2 \) As (query-2),
\( \ldots \)
\( R_n \) As (query-n)
(query involving \( R_1, \ldots, R_n \) (and other tables))
Basic SQL Recursion

SQL With Statement

With \( R_1(A_1, A_2, \ldots, A_m) \) As (query-1),
\( R_2 \) As (query-2),
\[ \cdots \]
\( R_n \) As (query-n)
<query involving \( R_1, \ldots, R_n \) (and other tables)>
Basic SQL Recursion

With Recursive

```
R1 As (query-1),
R2 As (query-2),
...
Rn As (query-n)
```

<query involving R1,...,Rn (and other tables)>
Basic SQL Recursion

SQL With Recursive Statement

With Recursive

\[ R \text{ As ( base query Union all recursive query ) } \]

<query involving \( R \) (and other tables)>