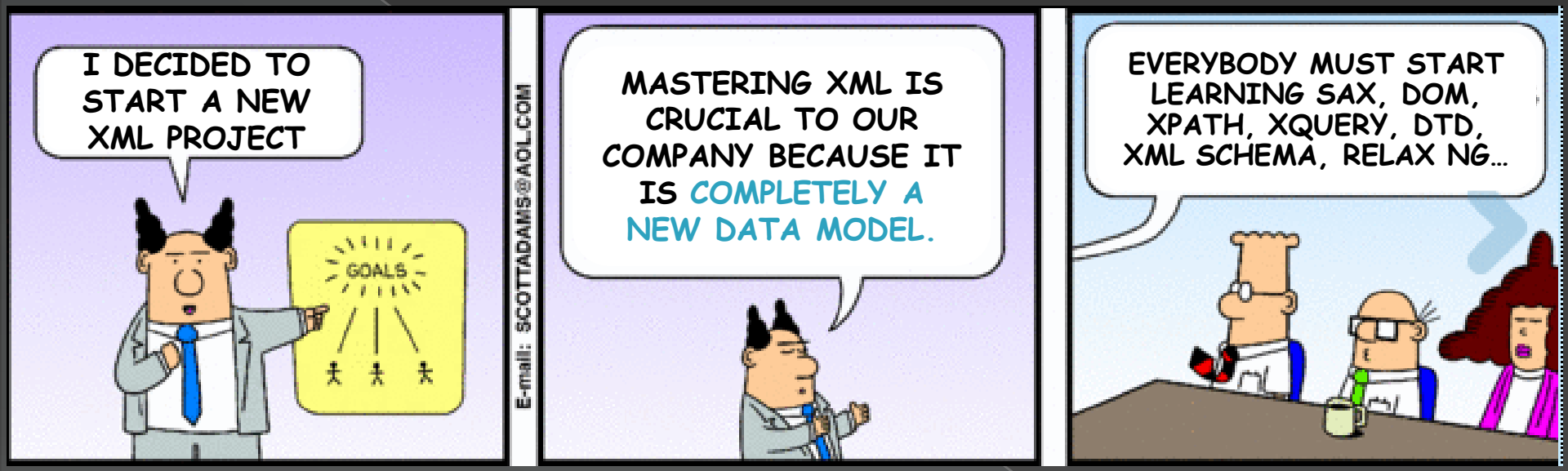
Relational-Style XML Query

Taro L. Saito **Shinichi Morishita**
University of Tokyo

June 10th, SIGMOD 2008
Vancouver, Canada

http://www.xerial.org/

If Your Manager Says …



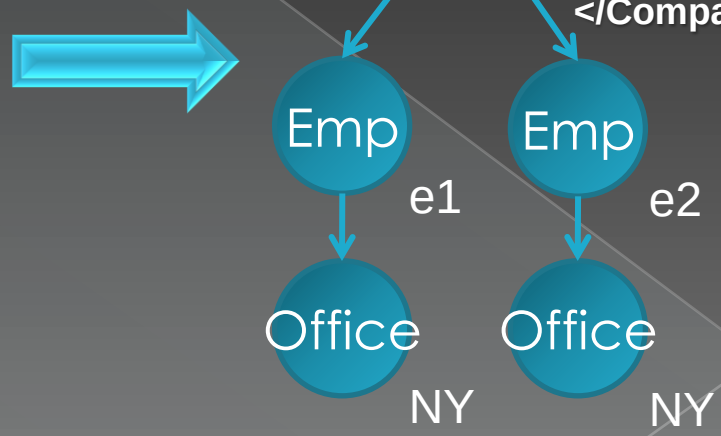
- It's a kind of tragedy…

Migration to XML Database

- Benefits of using XML:
  - XML is a portable text-data format
  - Tree-structured XML can reduce redundancy of relational data.

Company	Employee	Office
1	e1	NY
1	e2	NY

Relational Data




```
<Company value="1">
 <Emp value="e1">
    <Office>NY</Office>
 </Emp>
 <Emp value="e2">
    <Office>NY</Office>
 </Emp>
</Company>
```

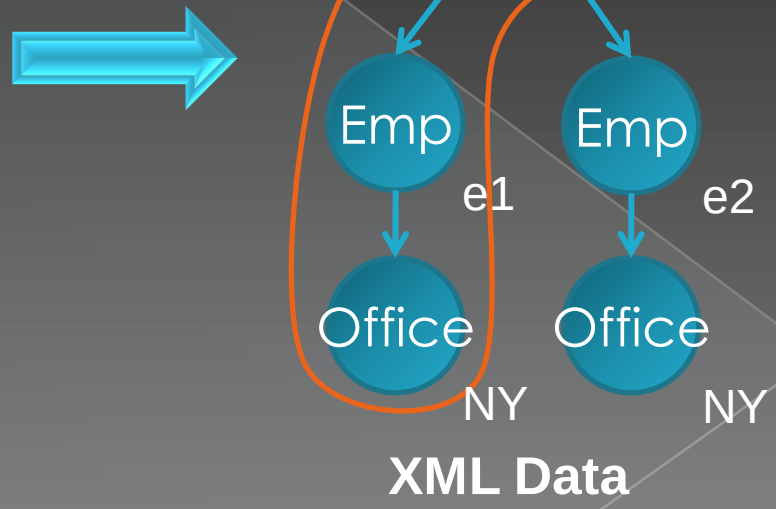
XML Data

Problem

- Querying relational data translated into XML
- Q: Retrieve a node tuple (Co, Emp, Office) from the XML data
  - e.g. XPath, a path expression query /Co/Emp/Office

Co	Emp	Office
1	e1	NY
1	e2	NY

Relational Data



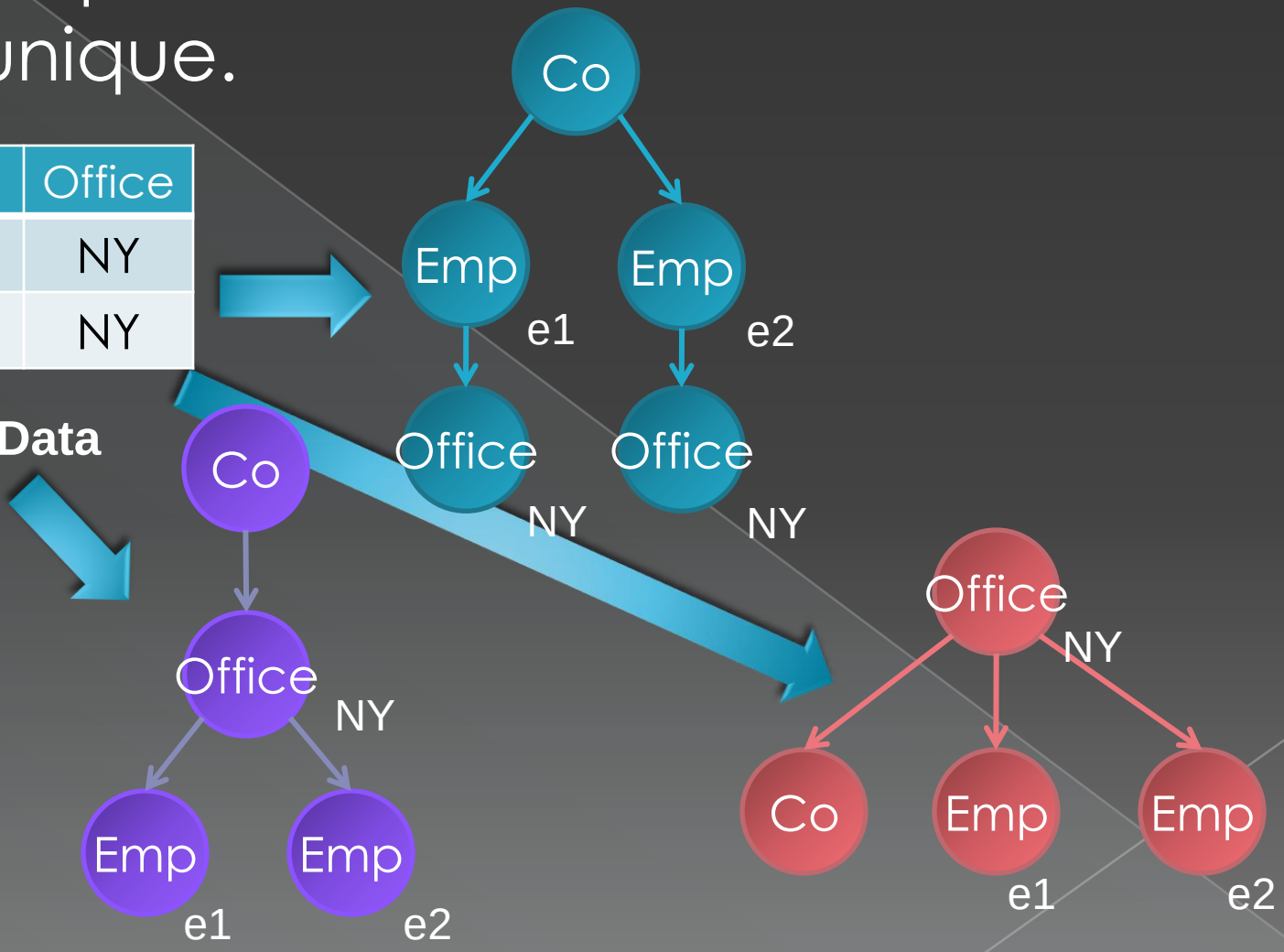
XML Data

A Pitfall: Structural Variations

- Tree-representation of relational data is not unique.

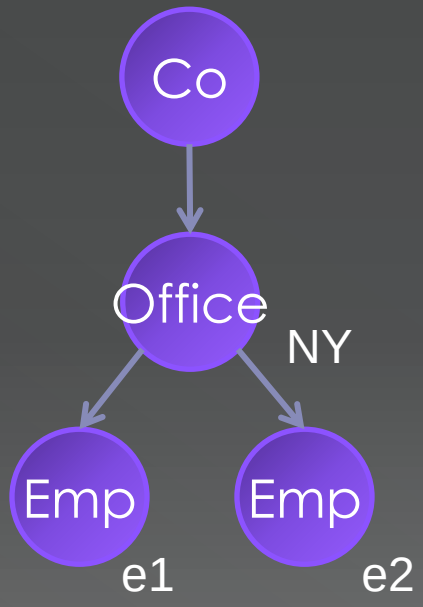
Co	Emp	Office
1	e1	NY
1	e2	NY

Relational Data

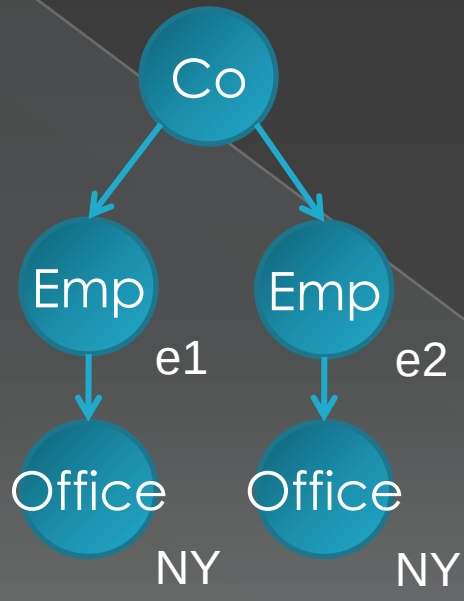


Inconvenience of XPath Query

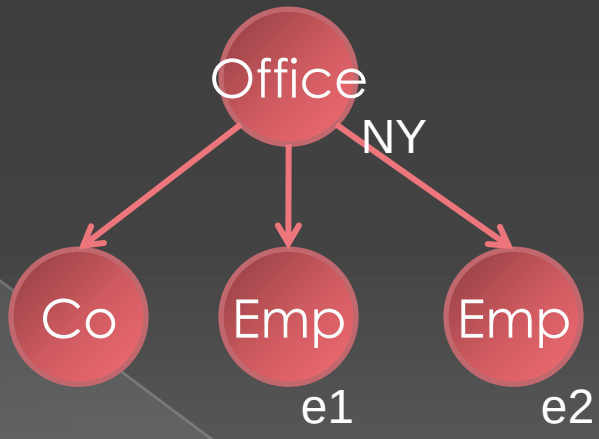
- User must know the entire XML structures to produce correct path queries.



/Co/Office/Emp



/Co/Emp[Office]



/Office[Co]/Emp

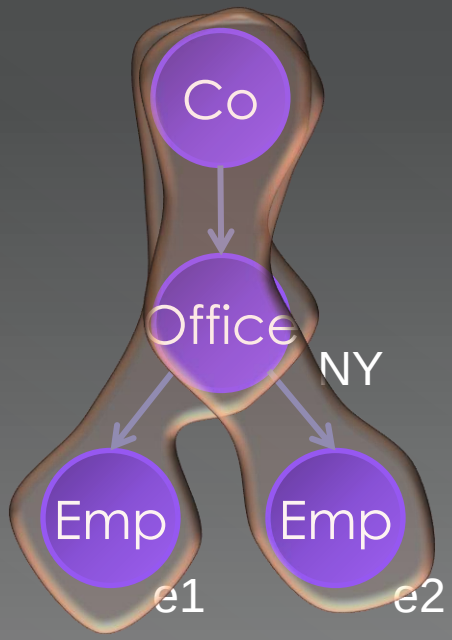
[X] : twig node to test

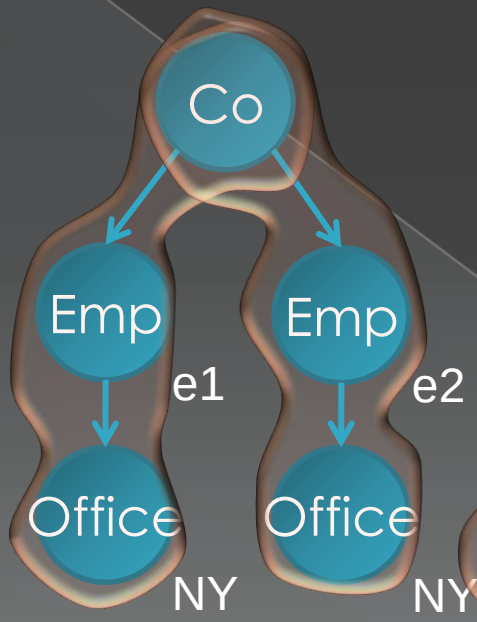
Relation in XML

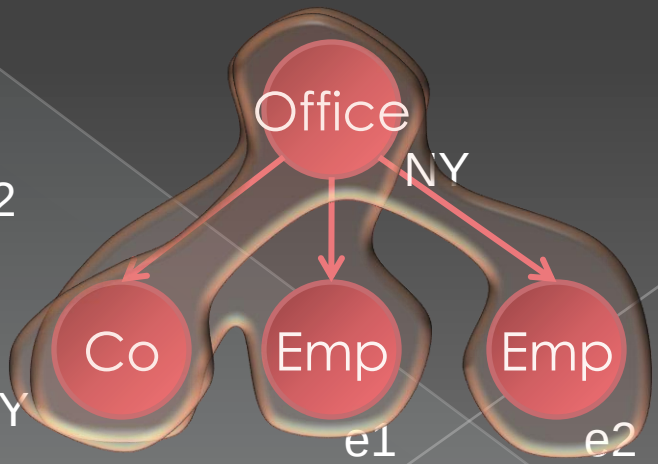
- A key observation:
  - Relation is simply embedded in XML

Co	Emp	Office
1	e1	NY
1	e2	NY

Relational Data



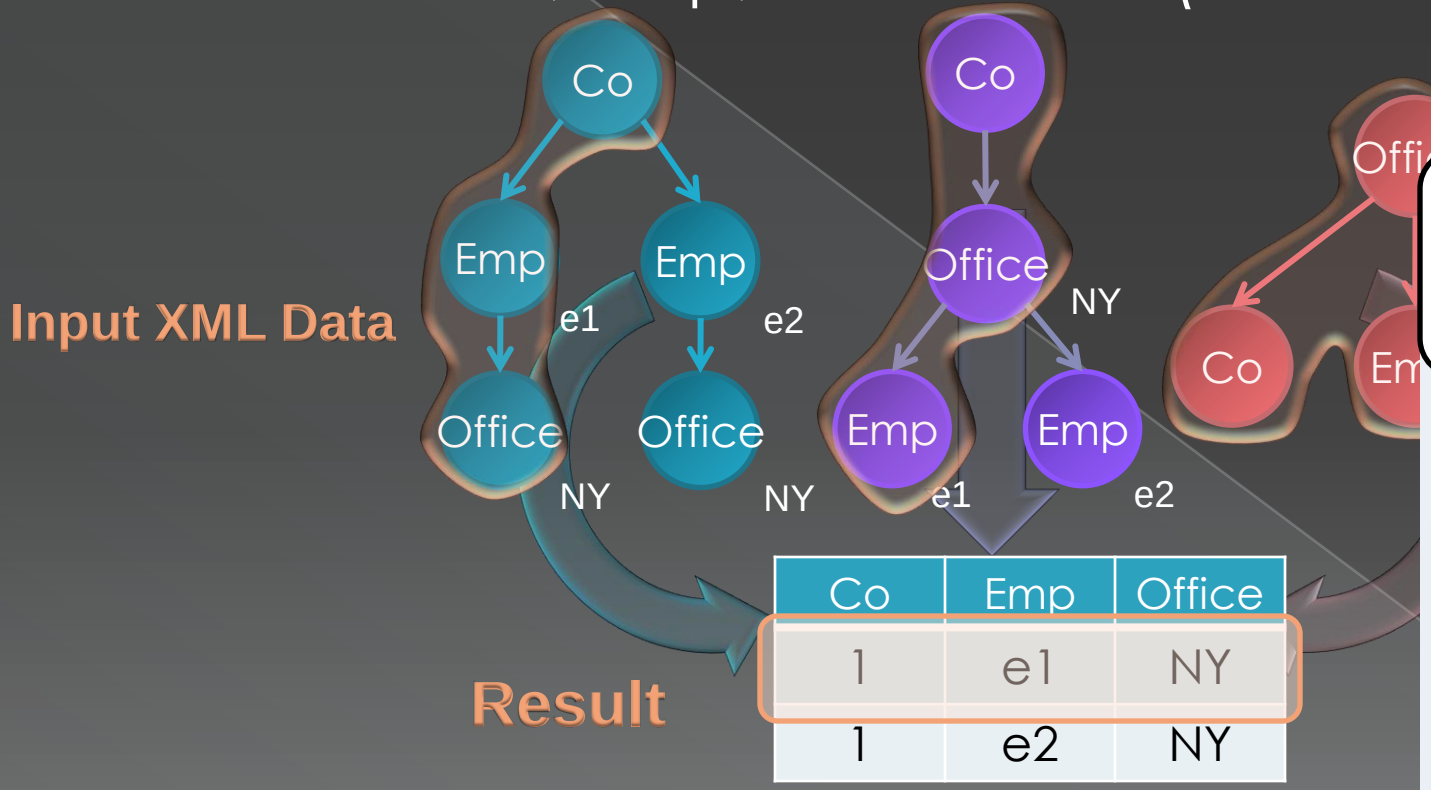


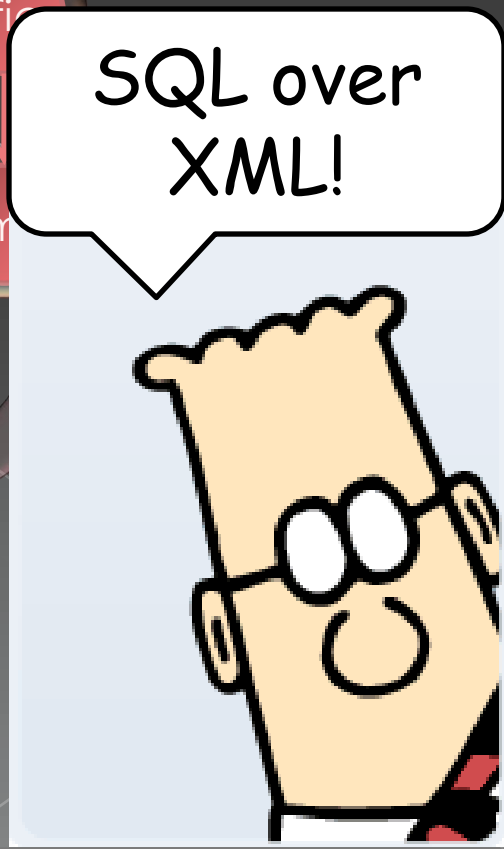


To Retrieve Relations in XML…

Relational-Style XML Query

- Query relations in XML
  - with an SQL-like syntax
- **SELECT** Co, Emp, Office **from** (XML Data)

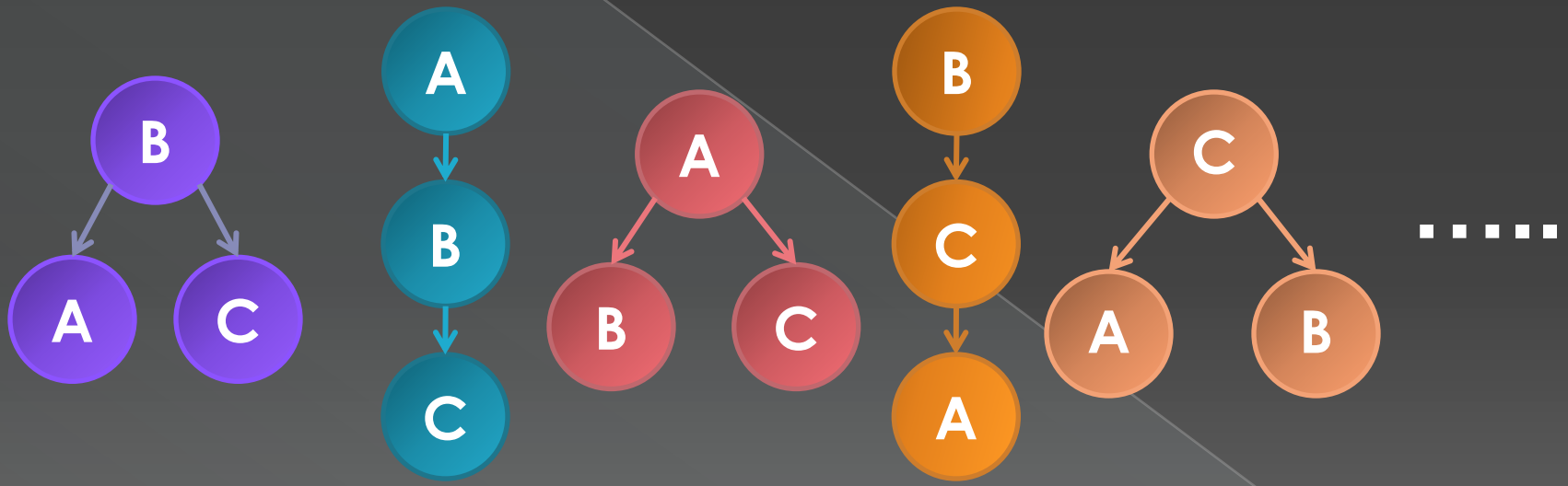




- The query statement is stable for variously structured XML data

Problem Definition

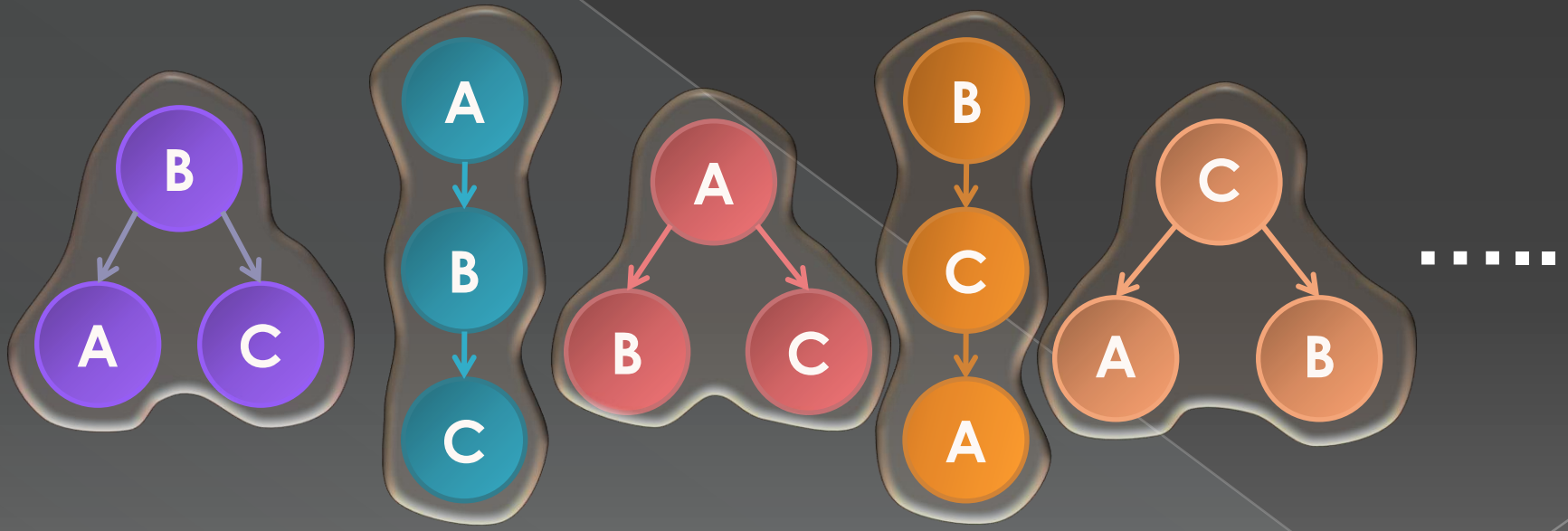
- Convert an SQL query, **SELECT A, B, C**, into an XML structure query.
  - There can be many structural variations of (A, B, C)



- For *N* nodes, there exists N^{N-1} structural variations.

Amoeba

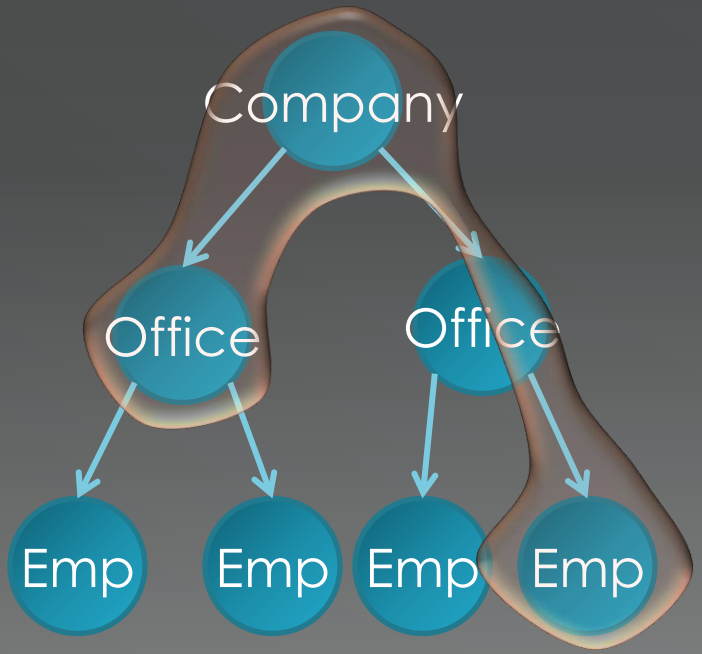
- A node tuple (A, B, C) is an amoeba iff one of the A, B and C is a common ancestor of the others.

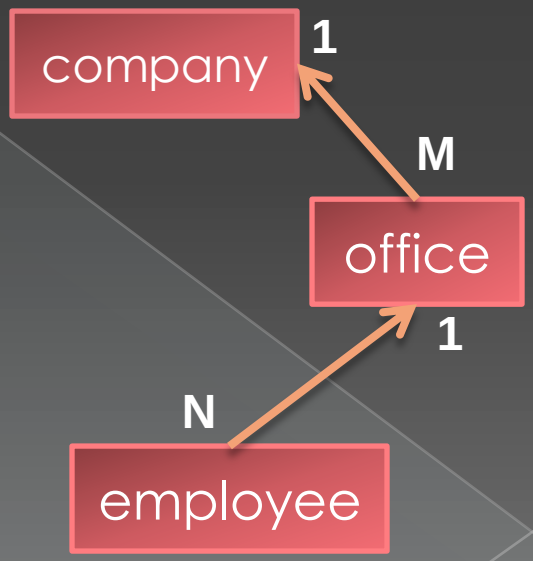


- Amoeba join retrieves all amoeba structures in the XML data.

Hidden Semantics in XML

- Some amoeba structure may not form a relation.
  - Why this structure is not allowed?
- Because there are functional dependencies (FD) implied in the XML structure.

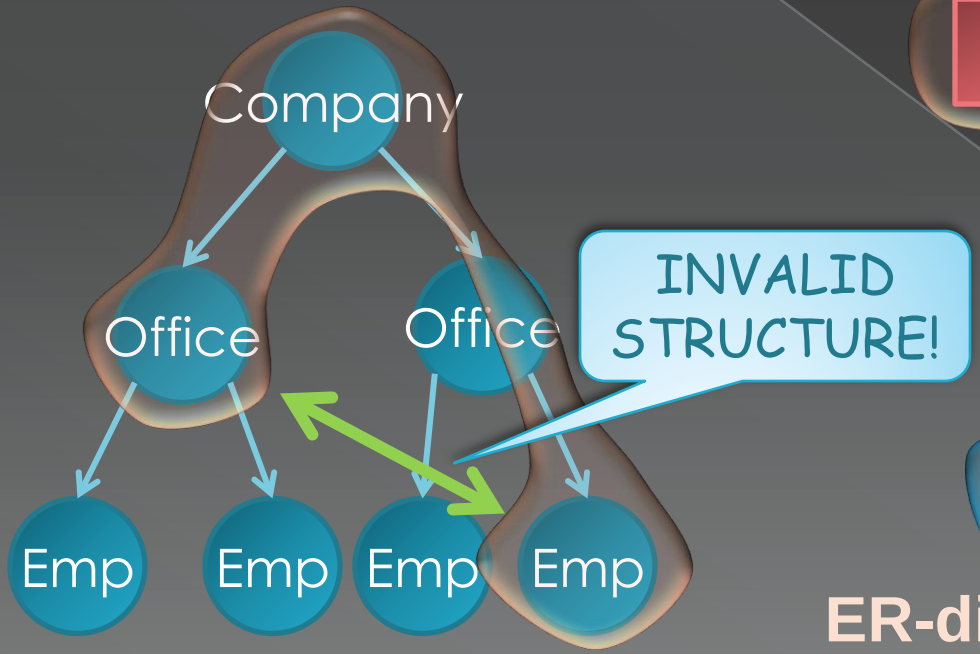


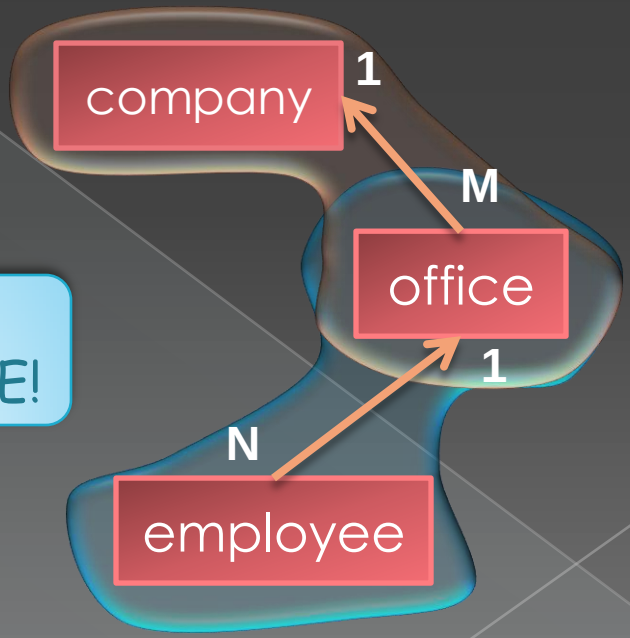


ER-diagram (Data Model)

Functional Dependencies (FD)

- FD: *X* -> *Y* (From a given *X*, *Y* is uniquely determined)
  - **employee-> office** (Each employee belongs to an office)
  - **office -> company** (Each office belongs to a company)
- Relation in XML must have an amoeba structure corresponding to each FD.

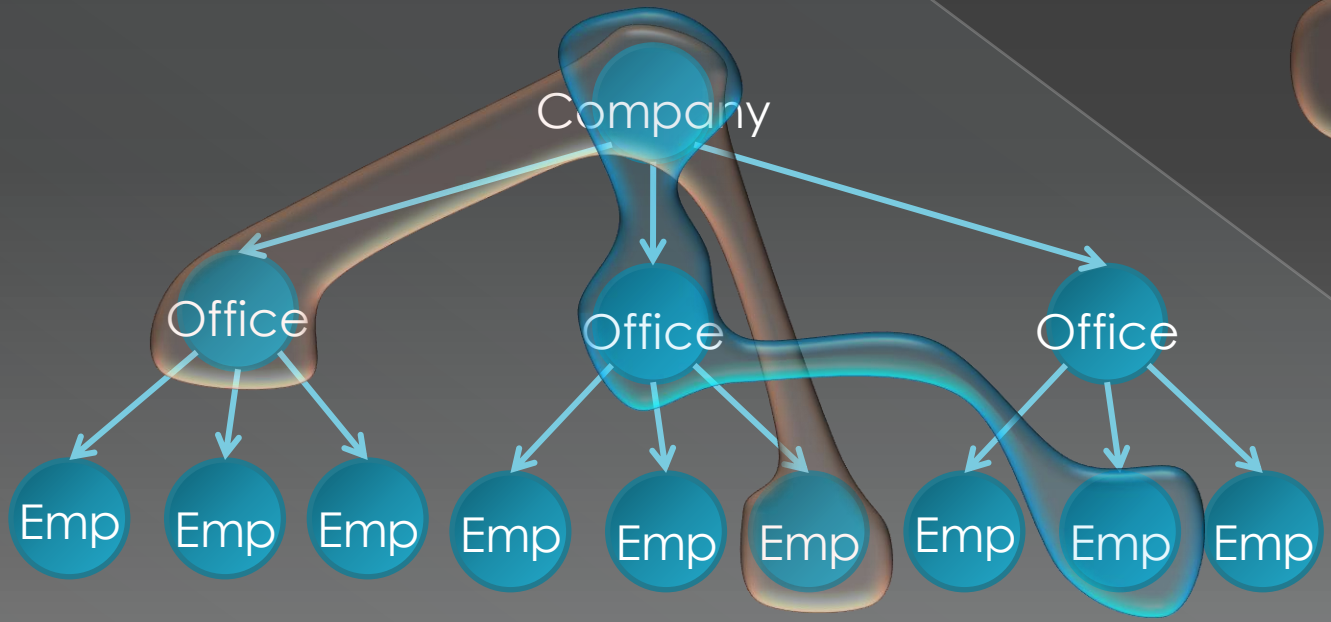


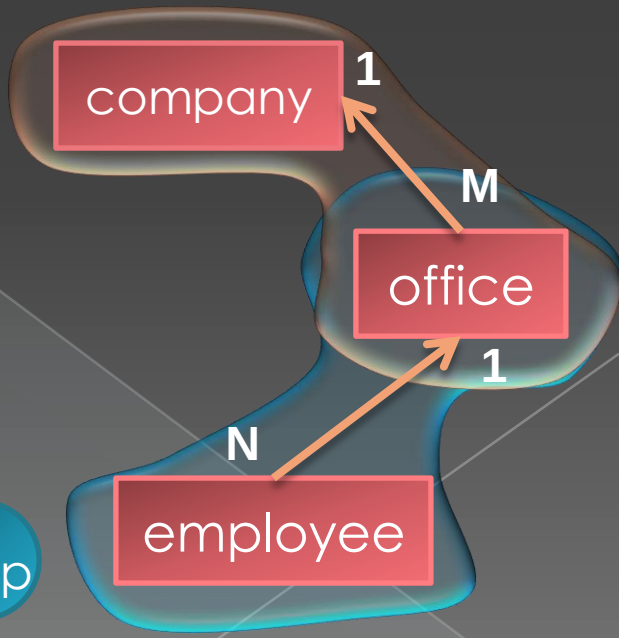


ER-diagram (Data Model)

If FDs are ignored….

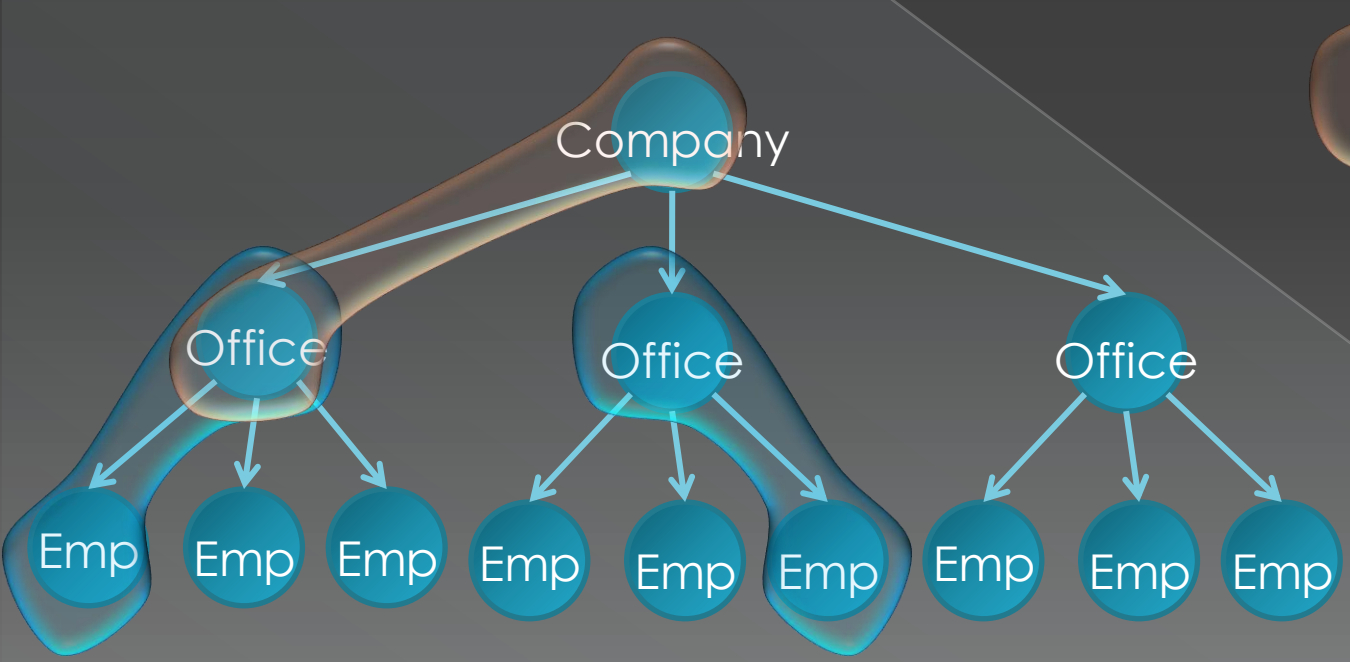
- The company has M offices, and each office has N employees:
- # of (company, office, employee) tuples:
  - When M = 100, N = 5 100 x (100 x 5) = 50,000
- While, # of correct answers is only M * N = 500

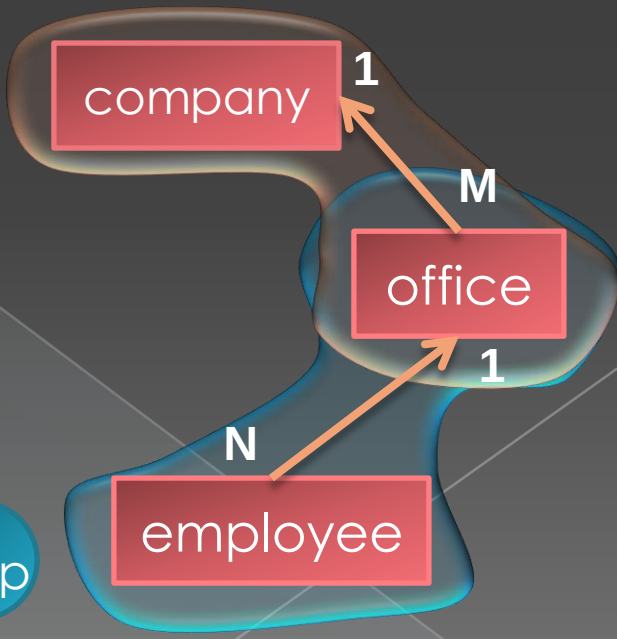




FD-Aware Amoeba Join

- FDs: Emp -> Office, Office -> Company
- Bottom-up construction of query results
  1. Amoeba Join (Employee, Office)
  2. Amoeba Join (Office, Company)

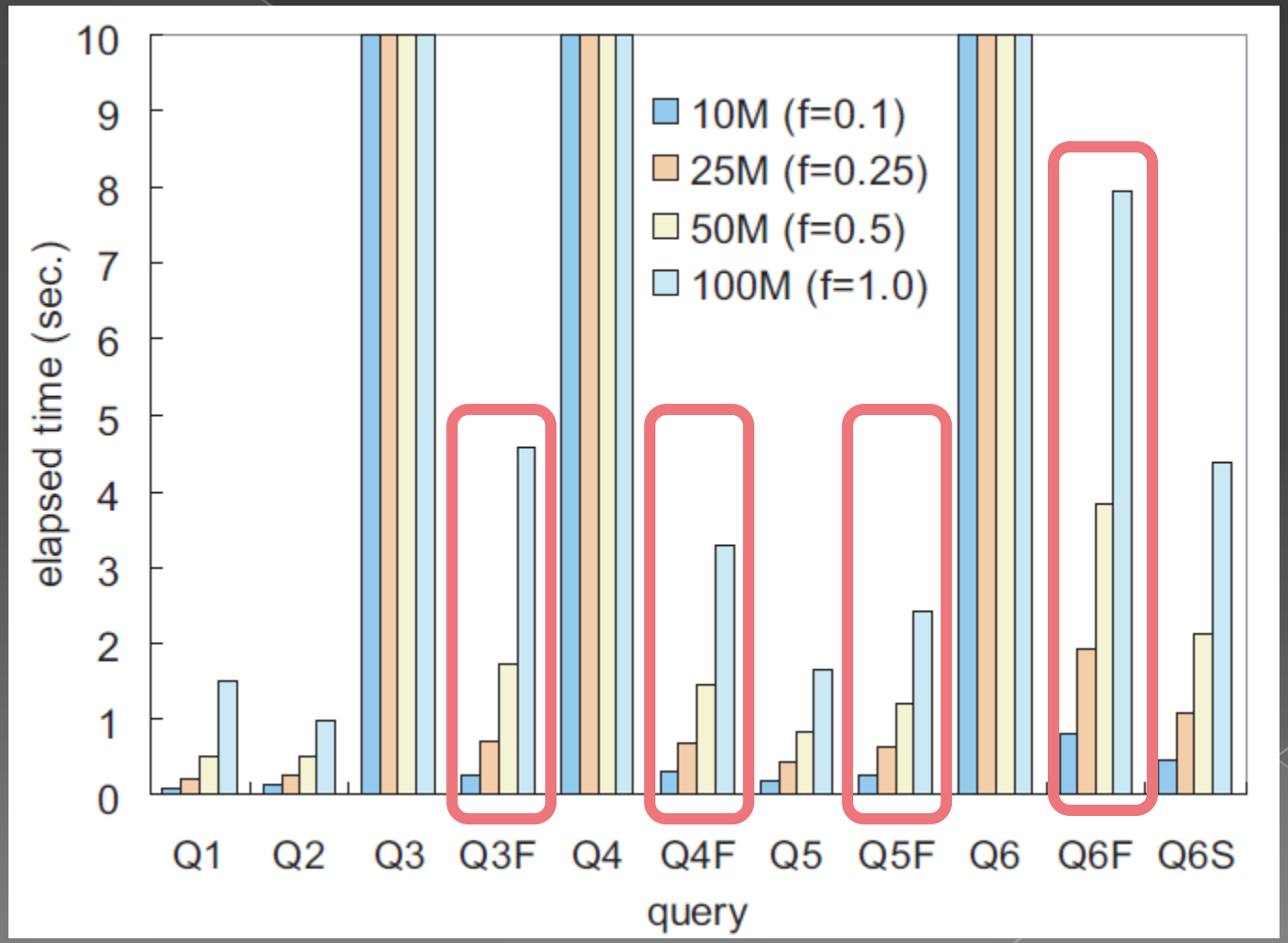




- FD-aware amoeba join avoids invalid XML structures.

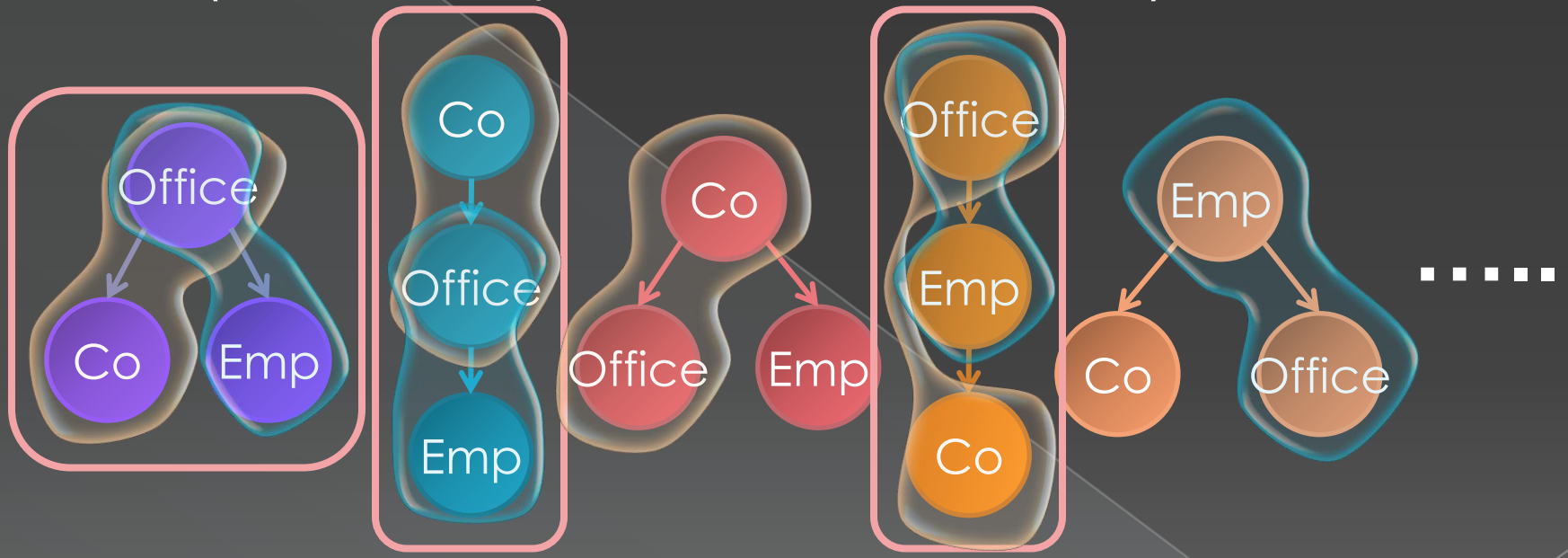
Query Performance

- FD-aware amoeba join scales well
  - For various sizes of XML data



Query Translation using FDs

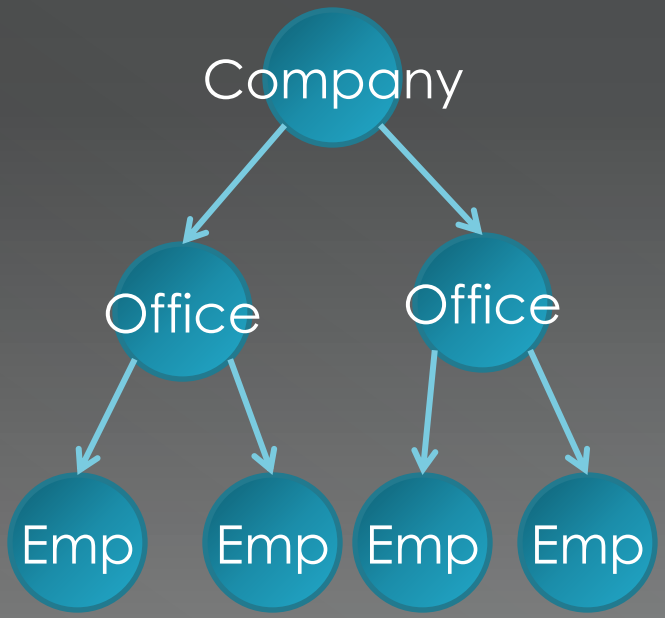
- Relational query into XML query
  - SELECT Co, Office, Emp
    - (with FDs: Emp -> Office, Office -> Co)

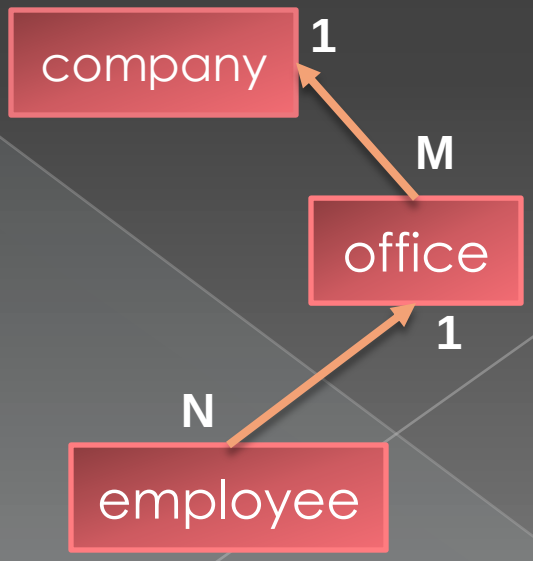


- XML structures of interest are automatically determined from a relation and functional dependencies

Detecting FDs

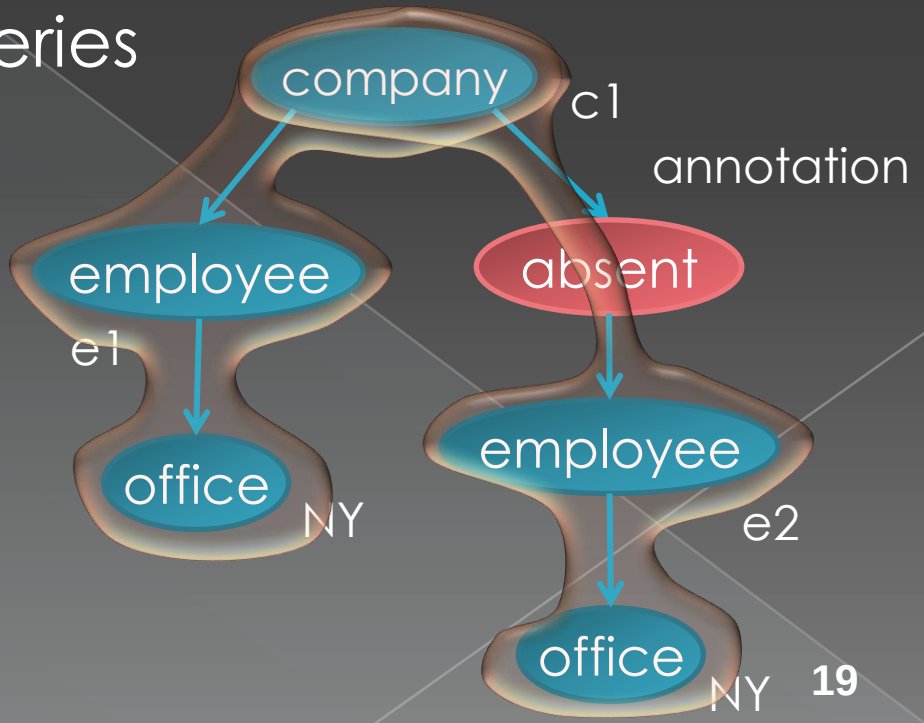
- A type of FDs required to determine XML structures to query is one-to-many (or one-to-one) relationships:
  - FD: Emp -> Office
    - Each employee belongs to an office
    - An office may have several employees (one-to-many)
- We can observe these relationships by counting node occurrences or directory from the ER-diagram.





Think in Relational-Style

- First, consider
  - XML := Relations + their annotations
- Steps
  - 1. Detect relational part from XML data
  - 2. Detect one-to-many(one) relationships (FDs)
  - 3. Write relational queries
    - SELECT Co, Emp, Office



- Note:
  - It is also possible to include annotations in query statements.

Summary of Our Contributions

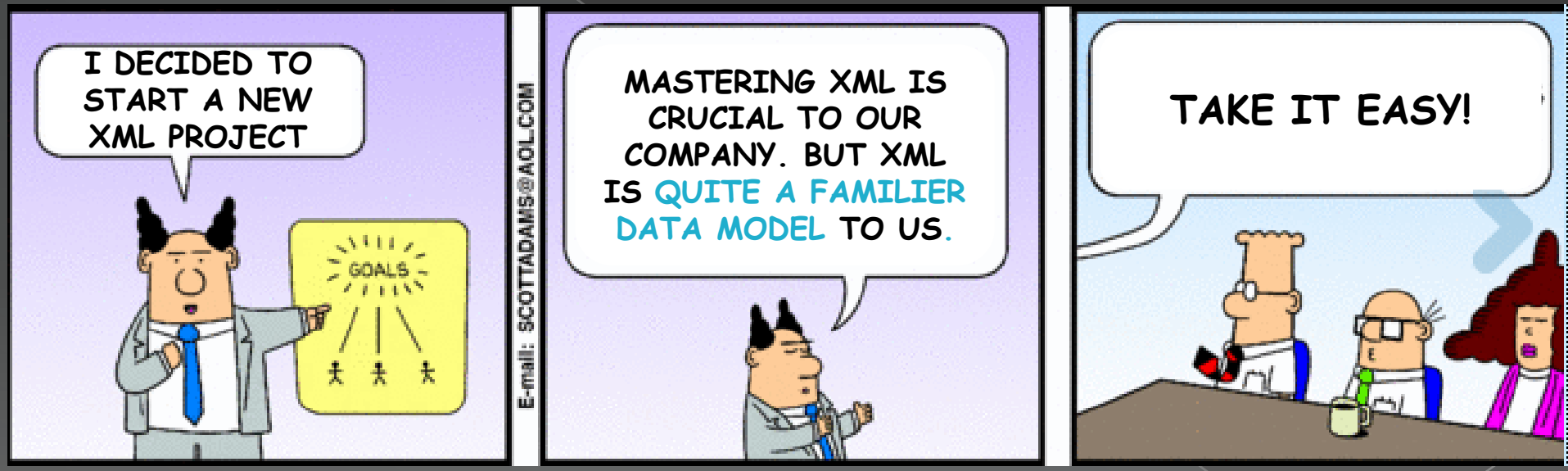
- Relation in XML
  - Defined using amoeba structure and FDs
- Relational-Style XML Query
  - Retrieves relations in XML with a SQL-like query syntax (SQL over XML)
  - Allows structural variations of XML data
- Departure from path expression queries
  - Target XML structures are automatically determined.

Applications of Relational-Style

- (see the paper for details)
- XML Algebra
  - Based on relational-semantics
    - selection, projection, etc.
- Keys for XML
  - A key is a special-case of FDs
- Database integration
- Schema evolution
- Managing relational data enhanced with XML syntax
- A lot more…

Conclusions

- **"It's Just SQL"**
  - A large number of XML data and queries are still relational.



- Before going deep into the XML world,

  Think in Relational-Style!!!