CompSci 516
Database Systems

Lecture 25
Recursive Query Evaluation and Data Mining

Instructor: Sudeepa Roy
Recursive Query in Databases
Recursion!

http://xkcdsw.com/1105
A motivating example

*Example: find Bart’s ancestors*

“Ancestor” has a recursive definition

- $X$ is $Y$’s ancestor if
  - $X$ is $Y$’s parent, or
  - $X$ is $Z$’s ancestor and $Z$ is $Y$’s ancestor
Recursion in Databases

• Consider a graph $G(V, E)$. Can you find out all “ancestor” vertices that can reach “x” using Relational Algebra/Calculus?

• NO! – ANCESTOR cannot be defined using a constant-size union of select-project-join queries (conjunctive queries)

• No RA/RC expressions can express ANCESTOR or REACHABILITY (TRANSITIVE CLOSURE) (Aho-Ullman, 1979)

• A limitation of RA/RC in expressing recursive queries

• Solution: Use “Datalog” language and include recursion in SQL
  – A long discussion in the DB community on whether recursion should be supported
Recursion in SQL

• SQL2 had no recursion
  – You can find Bart’s parents, grandparents, great grandparents, etc.
    ```sql
    SELECT p1.parent AS grandparent
    FROM Parent p1, Parent p2
    WHERE p1.child = p2.parent
    AND p2.child = 'Bart';
    
    – But you cannot find all his ancestors with a single query
  
• SQL3 introduces recursion
  – WITH clause
  – Implemented in PostgreSQL (common table expressions)
  – SQL:1999 (SQL3) and later versions support “linear Datalog”
Ancestor query in SQL3

WITH RECURSIVE Ancestor(anc, desc) AS
(
    (SELECT parent, child FROM Parent)
    UNION
    (SELECT a1.anc, a2.desc
    FROM Ancestor a1, Ancestor a2
    WHERE a1.desc = a2.anc)
)
SELECT anc
FROM Ancestor
WHERE desc = 'Bart';

Define a relation recursively

base case

recursion step

Query using the relation defined in WITH clause
Finding ancestors

• WITH RECURSIVE
  Ancestor(anc, desc) AS
  ((SELECT parent, child FROM Parent)
   UNION
   (SELECT a1.anc, a2.desc
    FROM Ancestor a1, Ancestor a2
    WHERE a1.desc = a2.anc))

Continue until no more new tuples are generated – reaches a “fixpoint”

Q. Why should it stop after finite number of steps?
Fixed point of a query

• Fixed point of a function: value of $x$ such that $f(x) = x$
  – E.g. $x = 0, 2$ for $f(x) = x^2 - x$

• A query $q$ is just a function that maps an input table to an output table, so a fixed point of $q$ is a table $T$ such that $q(T) = T$

To compute fixed point of $q$

• Start with an empty table: $T \leftarrow \emptyset$
  – Initiate all tables to $\emptyset$ for multiple recursive relations

• Evaluate $q$ over $T$
  – In the $i$-th iteration, use *all* recursive tables from the previous $i$-1-th iteration
  – If the result is identical to $T$, stop; $T$ is a fixed point
  – Otherwise, let $T$ be the new result; repeat

• Starting from $\emptyset$ produces the unique minimal fixed point (assuming $q$ is monotone)
  – In the previous slide, think of the definition as Ancestor = $q$(Ancestor)
Linear recursion

• With linear recursion, a recursive definition can make only one reference to itself
• Non-linear
  – WITH RECURSIVE Ancestor(anc, desc) AS
    ((SELECT parent, child FROM Parent)
     UNION
     (SELECT a1.anc, a2.desc
      FROM Ancestor a1, Ancestor a2
      WHERE a1.desc = a2.anc))
• Linear
  – WITH RECURSIVE Ancestor(anc, desc) AS
    ((SELECT parent, child FROM Parent)
     UNION
     (SELECT anc, child
      FROM Ancestor, Parent
      WHERE desc = parent))
More on recursion

- Negation+recursion is tricky!
  - Need “stratification”

- Alternative Datalog format

- Ancestor(x, y) :- Parent(x, y)
- Ancestor(x, y) :- Ancestor(x, z), Parent(z, y)

Union: Two rules with the same “head”

Comma “,” = Join on the same variables
(A glimpse of) Data Mining
Optional Reading Material

1. [RG]: Chapter 26

2. “Fast Algorithms for Mining Association Rules”
   Agrawal and Srikant, VLDB 1994

25,426 citations on Google Scholar in November 2019!
• 23,863 in November 2018
• 23,038 in November 2017
• 20,610 in November 2016
• 19,496 in April 2016

One of the most cited papers in CS!

• Acknowledgement:
  The following slides have been prepared adapting the slides provided by the authors of [RG] and using several presentations from the internet.
Four Main Steps in KD and DM (KDD)

• **Data Selection**
  – Identify target subset of data and attributes of interest

• **Data Cleaning**
  – Remove noise and outliers, unify units, create new fields, use denormalization if needed

• **Data Mining**
  – extract interesting patterns

• **Evaluation**
  – present the patterns to the end users in a suitable form, e.g. through visualization
Several DM/KD (Research) Problems

- Discovery of causal rules
- Learning of logical definitions
- Fitting of functions to data
- Clustering
- Classification
- Inferring functional dependencies from data
- Finding “usefulness” or “interestingness” of a rule
Mining Association Rules

• Retailers collect and store massive amounts of sales data
  – transaction date and list of items

• Association rules:
  – e.g. 98% customers who purchase “tires” and “auto accessories” also get “automotive services” done
  – Customers who buy mustard and ketchup also buy burgers
  – Goal: find these rules from just transactional data (transaction id + list of items)
Applications

• Can be used for
  – marketing program and strategies
  – cross-marketing (mass e-mail, webpages)
  – catalog design
  – add-on sales
  – store layout
  – customer segmentation
Notations

• Items $I = \{i_1, i_2, \ldots, i_m\}$

• $D$: a set of transactions

• Each transaction $T \subseteq I$
  
  – has an identifier TID

• Association Rule
  
  – $X \rightarrow Y$ (not Functional Dependency!)
  
  – $X, Y \subseteq I$
  
  – $X \cap Y = \emptyset$
Confidence and Support

- **Association rule** \( X \rightarrow Y \)

- **Confidence** \( c = \frac{|\text{Tr. with } X \text{ and } Y|}{|\text{Tr. with } X|} \)
  - \( c\% \) of transactions in \( D \) that contain \( X \) also contain \( Y \)

- **Support** \( s = \frac{|\text{Tr. with } X \text{ and } Y|}{|\text{all Tr.}|} \)
  - \( s\% \) of transactions in \( D \) contain \( X \) and \( Y \).
### Support Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Cereal</th>
<th>Beer</th>
<th>Bread</th>
<th>Bananas</th>
<th>Milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

- **Support(Cereal)**
  - \( \frac{4}{8} = 0.5 \)
- **Support(Cereal \rightarrow Milk)**
  - \( \frac{3}{8} = 0.375 \)
# Confidence Example

<table>
<thead>
<tr>
<th>TID</th>
<th>Cereal</th>
<th>Beer</th>
<th>Bread</th>
<th>Bananas</th>
<th>Milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- **Confidence(Cereal → Milk)**
  - \( \frac{3}{4} = 0.75 \)
- **Confidence(Bananas → Bread)**
  - \( \frac{1}{3} = 0.33333... \)
X $\rightarrow$ Y is not a Functional Dependency

For functional dependencies

- F.D. = two tuples with the same value of X must have the same value of Y
  - $X \rightarrow Y \implies XZ \rightarrow Y$ (concatenation)
  - $X \rightarrow Y$, $Y \rightarrow Z \implies X \rightarrow Z$ (transitivity)

For association rules

- $X \rightarrow A$ does not mean $XY \rightarrow A$
  - May not have the minimum support
  - Assume one transaction \{AX\}

- $X \rightarrow A$ and $A \rightarrow Z$ do not mean $X \rightarrow Z$
  - May not have the minimum confidence
  - Assume two transactions \{XA\}, \{AZ\}
Problem Definition

• Input
  – a set of transactions $D$
    • Can be in any form – a file, relational table, etc.
  – min support (minsup)
  – min confidence (minconf)

• Goal: generate all association rules that have
  – support $\geq$ minsup and
  – confidence $\geq$ minconf
Decomposition into two subproblems

• 1. Apriori
  – for finding “large” itemsets with support $\geq \text{minsup}$
  – all other itemsets are “small”

• 2. Then use another algorithm to find rules $X \rightarrow Y$ such that
  – Both itemsets $X \cup Y$ and $X$ are large
  – $X \rightarrow Y$ has confidence $\geq \text{minconf}$

• Paper focuses on subproblem 1
  – if support is low, confidence may not say much
  – subproblem 2 in full version of the paper
• Q. Which itemset can possibly have larger support: ABCD or AB
  – i.e. when one is a subset of the other?

• Ans: AB
  – any subset of a large itemset must be large
  – So if AB is small, no need to investigate ABC, ABCD etc.
• Start with individual (singleton) items \{A\}, \{B\}, ...

• In subsequent passes, extend the “large itemsets” of the previous pass as “seed”

• Generate new potentially large itemsets (candidate itemsets)
  – E.g., if \{AB\} \{AC\} are two large itemsets of size 2, a candidate itemset for size 3 is \{ABC\} (different last item in the otherwise same sequence)

• Then count their actual support from the data

• At the end of the pass, determine which of the candidate itemsets are actually large
  – becomes seed for the next pass

• Continue until no new large itemsets are found
Announcements
Announcements (11/26, Tues)

• **Final: 12/14 (Sat), 2-5 pm, in class**
  – Closed book/notes
  – *Comprehensive*, but likely to have more emphasis on material after midterm

• **Please fill out course evaluations!**
  – Currently only 1/52 😊 (thanks to you who filled it out!)
  – A very important step toward improving the class for the future – the course is for you, so all feedback and suggestions much appreciated for future offerings!
  – Will take a few minutes on Dukehub but a huge favor to us – we need your help!
  – A small token of appreciation:
    • 90% (>= 46) filled by the deadline, +4 bonus points in midterm to the entire class
    • 75% (>= 39) filled by the deadline, +2 bonus points in midterm to the entire class
Announcements (11/26, Tues)

• Please fill out course evaluations!
  – Currently only 1/52 🙂 (thanks to you who filled it out!)
  – A very important step toward improving the class for the future – the course is for you, so all feedback and suggestions much appreciated for future offerings!
  – Will take a few minutes on Dukehub but a huge favor to us – we need your help!
  – A small token of appreciation:
    • 90% (>= 46) filled by the deadline, +4 bonus points in midterm to the entire class
    • 75% (>= 39) filled by the deadline, +2 bonus points in midterm to the entire class
Announcements (11/26, Tues)

• Project slides (3 only) and report due on 12/11 (Wed)
  – Google slide deck will be posted
  – Everyone knows what every other group worked on and their results!
  – Project grading will be relative
  – Feel free to add voiceover in notes/ audio (encouraged!)

• Offline 3-slide per project
  – Tentative: Motivation/Problem (1), Your contributions (2)
  – You present the current status of the project
    • problem, example, your approach, future work
  – Best to show plots/ screenshots/ results/ demo!
  – Try to show the most interesting observation/findings in 3 slides
Summary!
Take-Aways

• DBMS Basics

• DBMS Internals

• Overview of Research Areas

• Hands-on Experience in DB systems
DB Systems

• Traditional DBMS
  – PostGres, SQL

• Large-scale Data Processing Systems
  – Spark/Scala, AWS

• New DBMS/NOSQL
  – MongoDB

• In addition
  – XML, JSON, JDBC, Python/Java
DB Basics

• SQL
• RA/Logical Plans
• RC
• Recursion in SQL / Datalog
  – Why we needed each of these languages
• Normal Forms
DB Internals and Algorithms

- Storage
- Indexing
- Operator Algorithms
  - External Sort
  - Join Algorithms
- Cost-based Query Optimization
- Transactions
  - Concurrency Control
  - Recovery
Large-scale Processing and New Approaches

- Parallel DBMS
- Distributed DBMS
- Map Reduce
- NOSQL

- Hope some of you will further explore Database Systems/Data Management/Data Analysis/Big Data as a researcher or practitioner!