Key Observation behind this Work

Key observation:
- The occurrence of input values in the result of conjunctive queries follow certain regular patterns.

- Such patterns represent a fundamental property of queries

- ... and can be used to explain query computational complexity in various contexts.
Better understand and describe these occurrence patterns in query results.
Our Approach at a Glance

Ingredients:

- provenance polynomials of query results
  - ... to trace input values in the query result

- factorization of provenance polynomials guided by query structure
  - ... to get succinct, nested representations of the query result and its provenance polynomial

- new notion of *readability width* for conjunctive queries
  - ... to quantify how many times an input value is used in the (factorized) provenance polynomial of the query result
Provenance Polynomials
Annotational relational databases

- Annotate each tuple with elements from a commutative semiring. [GKT’07]
- Convenient generalisation of annotations in, e.g., incomplete databases, probabilistic databases, bag semantics, lineage in data warehousing.

Example of annotated database:

<table>
<thead>
<tr>
<th>Cust</th>
<th>ckey</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁</td>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>c₂</td>
<td>2</td>
<td>Dan</td>
</tr>
<tr>
<td>c₃</td>
<td>3</td>
<td>Li</td>
</tr>
<tr>
<td>c₄</td>
<td>4</td>
<td>Mo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ord</th>
<th>ckey</th>
<th>okey</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>o₁</td>
<td>1</td>
<td>1</td>
<td>1995</td>
</tr>
<tr>
<td>o₂</td>
<td>1</td>
<td>2</td>
<td>1996</td>
</tr>
<tr>
<td>o₃</td>
<td>2</td>
<td>3</td>
<td>1994</td>
</tr>
<tr>
<td>o₄</td>
<td>2</td>
<td>4</td>
<td>1993</td>
</tr>
<tr>
<td>o₅</td>
<td>3</td>
<td>5</td>
<td>1995</td>
</tr>
<tr>
<td>o₆</td>
<td>3</td>
<td>6</td>
<td>1996</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>okey</th>
<th>disc</th>
</tr>
</thead>
<tbody>
<tr>
<td>i₁</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>i₂</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>i₃</td>
<td>3</td>
<td>0.4</td>
</tr>
<tr>
<td>i₄</td>
<td>3</td>
<td>0.1</td>
</tr>
<tr>
<td>i₅</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>i₆</td>
<td>5</td>
<td>0.1</td>
</tr>
</tbody>
</table>

- Relation Cust uses annotations (or variables) $c₁, \ldots, c₄$.
- Relation Ord uses annotations (or variables) $o₁, \ldots, o₆$.
- Relation Item uses annotations (or variables) $i₁, \ldots, i₆$. 
Consider a join query \( Q = \text{Cust} \bowtie_{\text{ckey}} \text{Ord} \bowtie_{\text{okey}} \text{Item} \) on the three relations:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
\text{Q} & \text{ckey} & \text{name} & \text{okey} & \text{date} & \text{disc} \\
\hline
\text{c}_1 \cdot \text{o}_1 \cdot \text{i}_1 & 1 & \text{Joe} & 1 & 1995 & 0.1 \\
\text{c}_1 \cdot \text{o}_1 \cdot \text{i}_2 & 1 & \text{Joe} & 1 & 1995 & 0.2 \\
\text{c}_2 \cdot \text{o}_3 \cdot \text{i}_3 & 2 & \text{Dan} & 3 & 1994 & 0.4 \\
\text{c}_2 \cdot \text{o}_3 \cdot \text{i}_4 & 2 & \text{Dan} & 3 & 1994 & 0.1 \\
\text{c}_2 \cdot \text{o}_4 \cdot \text{i}_5 & 2 & \text{Dan} & 4 & 1993 & 0.4 \\
\text{c}_3 \cdot \text{o}_5 \cdot \text{i}_6 & 3 & \text{Li} & 5 & 1995 & 0.1 \\
\end{array}
\]

The annotation \( c_i \cdot o_j \cdot i_l \) of a result tuple \( t \) records its \textit{provenance}:

- \( t \) is the result of a join of input tuples annotated by \( c_i \text{ and } o_j \text{ and } i_l \).
- Conjunction expressed using the semiring operation (\( \cdot \)).
Annotated Relational Databases

Consider now the Boolean version $\pi_\emptyset(Q)$ of the join query $Q$:

<table>
<thead>
<tr>
<th>$Q$</th>
<th>ckey name okey date disc</th>
<th>$\pi_\emptyset(Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1 \cdot o_1 \cdot i_1$</td>
<td>1  Joe 1 1995 0.1</td>
<td>$c_1 \cdot o_1 \cdot i_1 +$</td>
</tr>
<tr>
<td>$c_1 \cdot o_1 \cdot i_2$</td>
<td>1  Joe 1 1995 0.2</td>
<td>$c_1 \cdot o_1 \cdot i_2 +$</td>
</tr>
<tr>
<td>$c_2 \cdot o_3 \cdot i_3$</td>
<td>2  Dan 3 1994 0.4</td>
<td>$c_2 \cdot o_3 \cdot i_3 +$</td>
</tr>
<tr>
<td>$c_2 \cdot o_3 \cdot i_4$</td>
<td>2  Dan 3 1994 0.1</td>
<td>$c_2 \cdot o_3 \cdot i_4 +$</td>
</tr>
<tr>
<td>$c_2 \cdot o_4 \cdot i_5$</td>
<td>2  Dan 4 1993 0.4</td>
<td>$c_2 \cdot o_4 \cdot i_5 +$</td>
</tr>
<tr>
<td>$c_3 \cdot o_5 \cdot i_6$</td>
<td>3  Li  5 1995 0.1</td>
<td>$c_3 \cdot o_5 \cdot i_6$</td>
</tr>
</tbody>
</table>

The annotation of $\pi_\emptyset(Q)$’s result (the nullary tuple) is:

$$c_1 \cdot o_1 \cdot i_1 + c_1 \cdot o_1 \cdot i_2 + c_2 \cdot o_3 \cdot i_3 + c_2 \cdot o_3 \cdot i_4 + c_2 \cdot o_4 \cdot i_5 + c_3 \cdot o_5 \cdot i_6$$

- There are 6 alternative derivations of the result.
- Disjunction expressed using the semiring operation $(+).$

Provenance polynomials of interest $= \text{Semiring annotations of query results.}$
Factorization and Readability of Query Provenance
Factorizing Provenance Polynomials

Consider again the previous provenance polynomial (we omit \((\cdot)\) operation):

\[
\psi_1 = c_1 o_1 i_1 + c_1 o_1 i_2 + c_2 o_3 i_3 + c_2 o_3 i_4 + c_2 o_4 i_5 + c_3 o_5 i_6
\]

We can factorize it as follows:

\[
\psi_2 = c_1 o_1 (i_1 + i_2) + c_2 (o_3 (i_3 + i_4) + o_4 i_5) + c_3 o_5 i_6.
\]

There are several \textit{algebraically equivalent} factorized representations due to
- distributivity of product over sum and
- commutativity of product and sum.
Readability of Provenance Polynomials

- A polynomial $\Phi$ is **read-$k$** if
  the maximum number of occurrences of any variable in $\Phi$ is $k$.

- The **readability** of $\Phi$ is the smallest number $k$ such that
  there is a read-$k$ polynomial equivalent to $\Phi$.

- Readability has been used for Boolean functions [Golumbic et al.'06].

- Example: $\psi_1$ is read-3 and $\psi_2$ is read-1. They are equivalent and have
  readability one.

  $$\psi_1 = c_1 o_1 i_1 + c_1 o_1 i_2 + c_2 o_3 i_3 + c_2 o_3 i_4 + c_2 o_4 i_5 + c_3 o_5 i_6.$$
  $$\psi_2 = c_1 o_1 (i_1 + i_2) + c_2 (o_3 (i_3 + i_4) + o_4 i_5) + c_3 o_5 i_6.$$

- Readability of $\Phi$ quantifies the succinctness of its factorization.
How to Factorize Query Provenance?

Our approach to define nesting structures of possible factorizations:

- They are statically derived from the query.
- They are independent of the database instance.

We call them **factorization trees** (or f-trees for short).
Factorization Trees of a Conjunctive Query

A factorization tree of a query $Q$ is a rooted unordered forest $\mathcal{T}$, where
- there is a one-to-one mapping between inner nodes in $\mathcal{T}$ and equivalence classes of attributes of $Q$, which do not contain any constants,
- there is a one-to-one mapping between leaf nodes in $\mathcal{T}$ and relations in $Q$,
- the attributes of each relation only appear in the ancestors of its leaf.

Example: Query $Q = \pi_\emptyset(\sigma_\phi(R \times S \times T \times U))$, with
- schemas $R(A_R, B_R, C)$, $S(A_S, B_S, D)$, $T(A_T, E_T)$, and $U(E_U, F)$,
- condition $\phi = (A_R = A_S = A_T, B_R = B_S, E_T = E_U)$.
Factorized Polynomials over Factorization Trees

\[ \sum_A \left[ \sum_B \left( \sum_C \sum_D S \right) \sum_E \left( T \sum_F U \right) \right] \]

foreach value \( a \in \text{Dom}_A \) do output sum of
  
  foreach value \( b \in \text{Dom}_B \) do output sum of
    
    foreach value \( c \in \text{Dom}_C \) do output sum of annotations of \( R \)-tuples \((a, b, c)\)
    \( \times \)
    
    foreach value \( d \in \text{Dom}_D \) do output sum of annotations of \( S \)-tuples \((a, b, d)\)
    \( \times \)
    
    foreach value \( e \in \text{Dom}_E \) do output sum of
      
      output sum of annotations of \( T \)-tuples \((a, e)\)
      \( \times \)
      
      foreach value \( f \in \text{Dom}_F \) do output sum of annotations of \( U \)-tuples \((e, f)\)
The read-6 provenance polynomial of a possible result to our previous query:

\[
\Phi = r_{111}s_{111}t_{12}u_{21} + r_{111}s_{111}t_{12}u_{22} + r_{111}s_{112}t_{12}u_{21} + r_{111}s_{112}t_{12}u_{22} + \\
r_{122}s_{121}t_{12}u_{21} + r_{122}s_{121}t_{12}u_{22} + r_{212}s_{211}t_{21}u_{11} + r_{212}s_{211}t_{22}u_{21} + r_{212}s_{211}t_{22}u_{22}.
\]

- The index of each annotation represents the tuple with that annotation.
- Thus, \( r_{111} \) is the annotation of the tuple \((1,1,1)\) in relation \(R\).
Factorized Polynomials over Factorization Trees

\[ \Phi = r_{111}s_{111}t_{12}u_{21} + r_{111}s_{111}t_{12}u_{22} + r_{111}s_{112}t_{12}u_{21} + r_{111}s_{112}t_{12}u_{22} + r_{122}s_{121}t_{12}u_{21} + r_{122}s_{121}t_{12}u_{22} + r_{212}s_{211}t_{21}u_{11} + r_{212}s_{211}t_{22}u_{21} + r_{212}s_{211}t_{22}u_{22}. \]

Over the above factorization tree, we obtain the equivalent read-2 polynomial:

\[ \Phi_1 = (r_{111}(s_{111} + s_{112}) + r_{122}s_{121})t_{12}(u_{21} + u_{22}) + r_{212}s_{211}(t_{21}u_{11} + t_{22}(u_{21} + u_{22})). \]
Readability Characterization of Conjunctive Queries

For any Boolean conjunctive query $Q$, there is a rational number $r(Q)$ such that:

- For any database $D$, the readability of the provenance of $Q(D)$ is at most $M \cdot |D|^{r(Q)}$, where $M$ is the max number of repeating relation symbols in $Q$.

- For any f-tree $T$ of $Q$ there exist arbitrarily large databases $D$ for which the factorized polynomial of $Q(D)$ over $T$ is at least read-$((|D|/|Q|)^{r(Q)}$.

Parameter $r(Q)$ is the **readability width** of $Q$.

Remarks:

- Trivial extension to non-Boolean conjunctive queries.
- We do not consider here query equivalence (modulo provenance polynomials).
Two Readability Dichotomies

1. Let $Q$ be a conjunctive query.
   - If $Q$ is *hierarchical*, then the readability of $Q(D)$ for any database $D$ is bounded by a constant.
   - If $Q$ is non-hierarchical, then for any f-tree $\mathcal{T}$ of $Q$ there exist arbitrarily large databases $D$ such that $\mathcal{T}(D)$ is read-$\Omega(|D|)$.

2. Let $Q$ be a conjunctive query without repeating relation symbols.
   - If $Q$ is hierarchical, then the readability of $Q(D)$ is 1 for any database $D$.
   - If $Q$ is non-hierarchical, then there exist arbitrarily large databases $D$ such that the readability of $Q(D)$ is $\Omega(\sqrt{|D|})$. 
What are these hierarchical queries?

A query is **hierarchical** if for any two equivalence classes of attributes in $Q$:
- either their sets of relation symbols are disjoint,
- or one is included in the other.

Examples:
- $Q = \pi_\emptyset(\text{Cust } \bowtie_{\text{ckey}} \text{Ord } \bowtie_{\text{okey},\text{ckey}} \text{Item})$ is not hierarchical.
  
  For $\text{rel(disc)}=\{\text{Item}\}$, $\text{rel(okey)}=\{\text{Ord}, \text{Item}\}$, $\text{rel(ckey)}=\{\text{Cust}, \text{Ord}\}$, we have $\text{rel(ckey)} \cap \text{rel(okey)} \neq \emptyset$ and $\text{rel(ckey)} \not\subseteq \text{rel(okey)}$ and $\text{rel(ckey)} \not\supseteq \text{rel(okey)}$.

- $Q$ becomes hierarchical if ckey is an attribute of Item, since:
  
  $\text{rel(disc)} \subseteq \text{rel(okey)} \subseteq \text{rel(ckey)}$.

![Diagram](attachment:diagram.png)
What are these hierarchical queries?

A query is **hierarchical** if for any two equivalence classes of attributes in $Q$:
- either their sets of relation symbols are disjoint,
- or one is included in the other.

Readability Width and Hierarchical Queries:
- All hierarchical queries have readability width 0.
- Readability width of a query $Q$ states how far $Q$ is from a hierarchical query.
The Hierarchical Property

Key to query characterisation in several contexts:

- In probabilistic databases, any tractable non-repeating conjunctive query is hierarchical; non-hierarchical queries are \( \#P \)-hard. [Suciu&Dalvi’07].

- In the finite cursor machine model of computation, any query that can be evaluated in one pass is hierarchical; non-hierarchical queries need more passes. [Grohe et al’07]
  ▶ Assumption: we are allowed to first sort the input relations.

- In the Massively Parallel computation model, any query that can be evaluated with one synchronisation step is hierarchical. [Suciu et al’11]
Thanks!
Definition: For a relation $R_i$ at a leaf, an ancestor node is non-relevant if it does not contain attributes of $R_i$. Let $NR$ be the set of nodes non-relevant to $R_i$.

Examples: The root node is not relevant to $U$ in the left factorization tree, and to $R$ and $S$ in the right factorization tree.
Bounds on the Readability of Factorized Representations

Consider:

- Any equi-join query \( Q = \sigma_\phi(R_1 \times \cdots \times R_n) \),
- A restriction of \( Q \) to \( NR \): \( Q_{NR} = \sigma_{\phi_{NR}}(\pi_{NR}R_1 \times \cdots \times \pi_{NR}R_n) \),
- Databases \( D \) and \( D_{NR} \) obtained by projecting \( D \) onto \( NR \).

The number of occurrences of the annotation for a tuple \( t \) in \( R_i \) in a factorized representation modelled on a factorization tree of \( \sigma_\phi(R_1 \times \cdots \times R_n) \) is:

\[
\left| \left| \pi_{NR}(\sigma_{S(R_i=\langle t \rangle)}\sigma_\phi(R_1 \times \cdots \times R_n) \right| \right|.
\]

**Upper bound**

- Further refinement: The number of occurrences is at most \( \left| \left| Q_{NR}(D_{NR}) \right| \right| \).
- Cover all attributes of \( Q_{NR} \) by \( k \) relations \( \Rightarrow \left| \left| Q_{NR}(D_{NR}) \right| \right| \leq |D|^k \).
- \( \Rightarrow \) minimum edge cover in the hypergraph of \( Q_{NR} \).

**Lower bound**

- Construct databases for which the number of occurrences is \( \left| \left| Q_{NR}(D_{NR}) \right| \right| \).
- Pick \( k \) attributes such that no two share a relation \( \Rightarrow \left| \left| Q_{NR}(D_{NR}) \right| \right| \geq |D|^k \).
- \( \Rightarrow \) maximum independent set in the hypergraph of \( Q_{NR} \).
Tightening the Bounds

Idea [Grohe&Marx’06]:
- Relax edge cover and independent set to their fractional (weighted) versions.
- They meet by LP duality
  - A fractional edge cover number can be an optimal solution to both the minimisation problem and its dual maximisation problem

For a query with equi-joins $Q$, the fractional edge cover number $\rho^*(Q)$ is an optimal solution to the linear program with variables $\{x_i\}_{i=1}^n$,

\[
\begin{align*}
\text{minimise} & \quad \sum_i x_i \\
\text{subject to} & \quad \sum_{i: R_i \in r(A)} x_i \geq 1 \quad \text{for all attributes } A, \text{ and} \\
& \quad x_i \geq 0 \quad \text{for all } i.
\end{align*}
\]

- Each $x_i$ represents one query relation (hyperedge in the hypergraph).
- For edge cover: $x_i$ can be either 0 or 1 and each node (=attribute) has to be covered by at least one edge.
- For fractional edge cover: $x_i \geq 0$ and each node can be covered by fractions of edges as long as the sum of all these fractions is above 1.
Special Case: Read-once Representations

Minimal number of occurrences of input annotations:

- \( NR = \emptyset \Rightarrow \) any annotation of \( R_i \) occurs at most once.
- If this holds for all relations, then all annotations occur at most once.
  - The readability of the representation is independent of the database size!
  - From the two factorization trees below, only the left one has this nice property.