Matching XML Materialized Views

Principal Subproblems

- Selecting which views to materialize
- Deciding which view(s) can be used to answer a query and rewriting the query accordingly
- Keeping materialized views current in the presence of updates

Example Table, Query & View

- Relational table R(D) with XML column(s)

Year	Doc	Level	…
2009	△	gr	
2009	△	ug	

- Relational view V(D) (with XML columns)

Year	Course	Subj	Cnum	Hours	Lab	Level
2009	△	CS	348	3	No	ug
2009	△	CS	135	3	Yes	ug
	…					

where Course, Subj, Cnum, Hours, and Lab are extracted from Doc

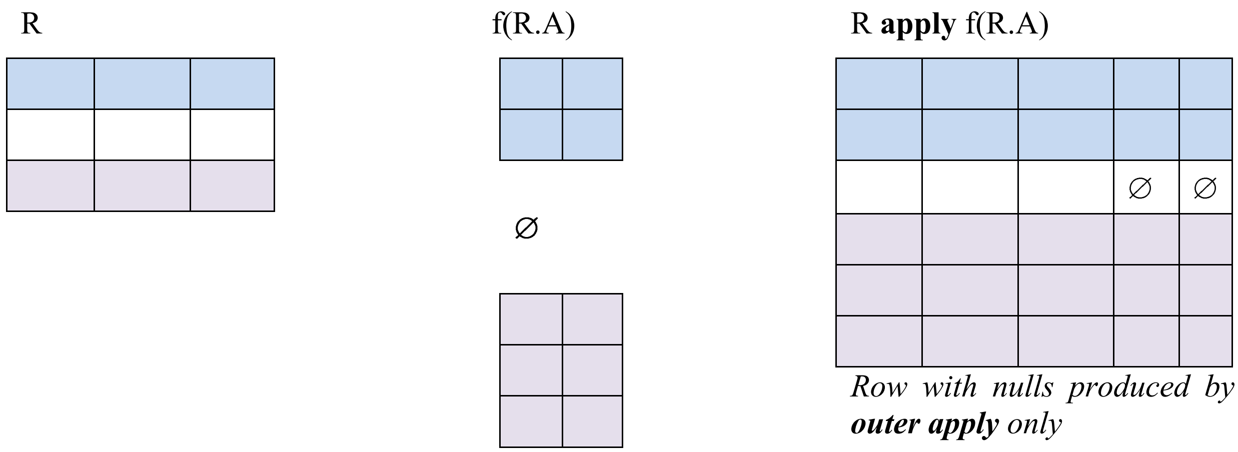
View Definition (SQLXML)

```
create materialized view V as
select year, level,
  crs.ordpath(), subj.ordpath(), hrs.ordpath(), lab.ordpath(),
  crs.value('@cnum', 'int'), subj.value('.', 'varchar(5)'),
  hrs.value('.', 'int'), lab.value('.', 'varchar(10)')
from T
  cross apply doc.nodes('cal/courses/course') as x1(crs)
  outer apply crs.nodes('subject') as x2(subj)
  outer apply crs.nodes('*//hours') as x3(hrs)
  outer apply crs.nodes('*//lab') as x4(lab)
where other = 'ug‘
```

Note: property functions (ordpath, value, nodes)

Apply operator

- To incorporate results of table-valued function (e.g., nodes):



*Row with nulls produced by **outer apply** only*

Effect of View Definition

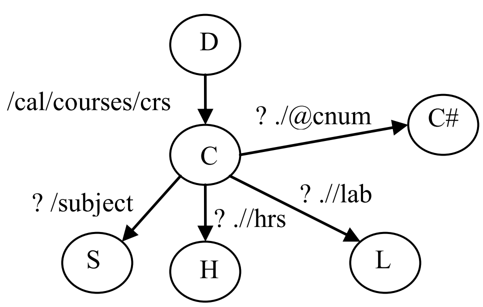
- For each year's undergraduate calendar, extract fields for year, level, subject, course number, hours, and lab.
    - For each of subject, course number, hours, and lab, store the value and (in a separate column) a reference (ordpath) to the subtree in the calendar text (doc).
- Given a query in terms of year, level, and doc, determine whether the query can be answered with data extracted in the defined view.
    - Need to compare XPath expressions
    - Need to account for cross vs. outer apply

Multi-Path Trees (MPTs)

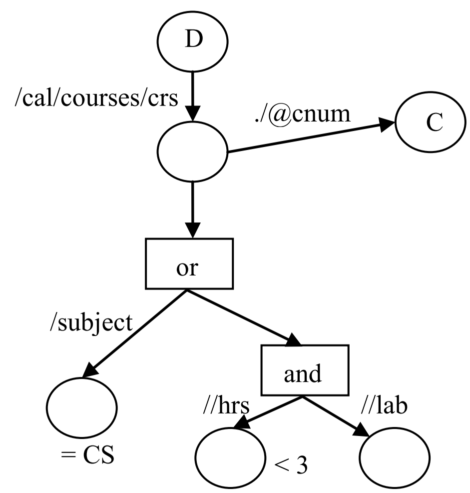
```
create materialized view V as
select year, level,
   crs.ordpath(),
   subj.ordpath(),
   hrs.ordpath(),
   lab.ordpath(),
   crs.value('@cnum', 'int'), subj.value('.', 'varchar(5)'),
   hrs.value('.', 'int'), lab.value('.', 'varchar(10)')
from T
   cross apply doc.nodes('cal/courses/course') as x1(crs)
   outer apply crs.nodes('subject') as x2(subj)
   outer apply crs.nodes('*//hours') as x3(hrs)
   outer apply crs.nodes('*//lab') as x4(lab)
where other = 'ug'
```

Can this view be used to answer the query: "Find all CS courses as well as all courses requiring labs lasting under 3 hours"?

View



Query



- Replace optional edges by trees representing disjunctions with nil
- Recursively expand paths into trees of nodes until each edge has a simple step as its label:
  - e.g., two nodes connected by /a/b becomes chain of three nodes connected by /a and /b
  - e.g., two nodes connected by a[b or c//d][e/f/g]/h becomes a tree of nodes
- Combine nodes connected by an edge labeled with self-axis

Normalization of MPTs

- Note some equivalences:
  A[B/C] ≡ A[B[C]]
  A[B and C] ≡ A[B][C]
  A[B or C][D] ≡ A[(B and D) or (C and D)]
  A[B/C or B/D] ≡ A[B[C or D]]
  But *not* A[B/C and B/D] ≡ A[B[C and D]]
  A[B or *nil* or *nil*] ≡ A[B or *nil*]
  A[B][*nil*] ≡ A[B]

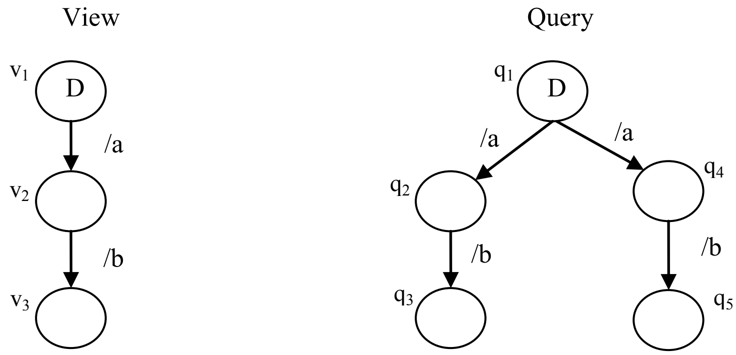
Therefore convert to Tree Disjunctive Normal Form:

- Push or nodes down over common prefixes
- Pull or nodes up so that they do not have siblings
- Eliminate and nodes unless they are roots of disjuncts and have at least two conjuncts
- Combine or – or sequences
- Combine and – and sequences
- Eliminate duplicate nil disjuncts
- Eliminate all nil conjuncts

Simple Chain Matching

Can we map V onto Q?

- Potentially exponentially many path matches
- But just considering the nodes is not expressive enough



- Node v_3 can match node q_3 only if node v_2 is mapped to node q_2
- Represent this by a *link*: (v_3, q_3, v_2, q_2)
- Similarly (v_3, q_5, v_2, q_4)

Link Set

Collection of 4-tuples (links) representing *possible* node matches *in context*

$H \subseteq \text{Nodes}(V) \times \text{Nodes}(Q) \times \text{Nodes}(V) \times \text{Nodes}(Q)$

A valid match $h \subseteq H$ must satisfy several properties (to be enumerated shortly)

Minimal cover for view V and query Q includes all possible valid matches and no subset does

Approach to using view V to answer query Q

- Find a minimal cover
- Choose a valid match
- Construct a query substitute

Conditions for a valid match h

1. Parent Compliance

$\forall(v , q , v_p , q_p) \in H$
Either v_p and q_p are the MPT roots, or $\exists v', q'$ such that $(v_p , q_p , v' , q') \in H$
(This is the situation we already noted.)

2. Root Compliance

Let v_0 and q_0 be the MPT roots; $v_0 \rightarrow q_0$ is implicit
These correspond to XML columns in R.
$\forall v \ \forall q \ \forall v_p \ \forall q_p \quad (v_0 , q , v_p , q_p) \notin H$ and $(v , q_0 , v_p , q_p) \notin H$

Let $k_1, \ldots, k_n$ be the key attributes of relations defined by V and by Q
$\forall D \ \ \prod_{k_1, \ldots, k_n} V(D) \supseteq \prod_{k_1, \ldots, k_n} Q(D)$
[Relational view matching, Goldstein & Larson, SIGMOD 2001]

3. Node-Test Compliance

$\forall(v , q , v_p , q_p) \in H$ such that v and q are data nodes (i.e., not and, or, or nil nodes)
NodeType(v) $\supseteq$ NodeType(q)
where T1 $\supseteq$ T2 if T1 evaluates to true whenever T2 evaluates to true
(e.g., for some schemas, addr $\supseteq$ cdn-addr)
$\forall t \quad t \supseteq t$
$\forall t \ \forall n \quad * \supseteq *{:}t \supseteq n{:}t \quad$ and $\quad * \supseteq n{:}* \supseteq n{:}t$

4. Restriction Compliance

$\forall(v , q , v_p , q_p) \in H$ such that v and q are data nodes (i.e., not and, or, or nil nodes)
Restriction(q) $\Rightarrow$ Restriction(v)
e.g., $< 3 \Rightarrow < 5$

- If no restriction, assume *true*
  Thus *always* satisfied if no restriction on view node, but
  if no restriction on query node, *cannot* have restriction on view node

5. Logic Compliance

- Every disjunct in Q must be mapped from somewhere in V
- At least one disjunct in V must be mapped to somewhere in Q
- Every conjunct in V must be mapped to somewhere in Q
- None of the conjuncts in Q need be mapped from anywhere in V
- a//b in V can be mapped to a/…/b or a…//b in Q as long as "…" does not include or nodes

Surprisingly complex

- Need to traverse MPTs for V and Q handling every combination of node pairs
- Previous work by Balmin et al. misses simple cases
  - A[B][C or D] → A[(B and C) or (B and D)] (addressed here by normalization)
  - A[B or C] → A[B or C] (they require one disjunct in V to map to all of Q)

Minimal Covers and Matches

- Consider earlier chain match
  $H_c = \{(v_2, q_2, v_1, q_1), (v_3, q_3, v_2, q_2), (v_2, q_4, v_1, q_1), (v_3, q_5, v_2, q_4)\}$
- Valid matches:
  $\{(v_2, q_2, v_1, q_1), (v_3, q_3, v_2, q_2)\}$
  $\{(v_2, q_4, v_1, q_1), (v_3, q_5, v_2, q_4)\}$

Match Extraction

- Valid match: partial function f from V to Q s.t.
  $L_f = \{(v, f(v), v_p, q_p) \in H_c\}$ satisfies parent and logic compliance
  (other compliances cannot be violated)
- Need to also have partial function from distinguished nodes in Q (corresponding to named components in query) back to V
  $m_f : Q_d \rightarrow V$ such that
  $m_f(q_i) = \text{nil if } f^{-1}(q_i) = \varnothing$
  $m_f(q_i) \in f^{-1}(q_i)$ otherwise
- Extracted match: (f, m_f)

Residual Predicate Filters

- Query rewrite needs to filter data from V
  - Conjuncts in Q might not be satisfied by all matched data in V
  - Disjuncts in V might not be matched in Q, so they must be filtered
  - More restricted types in Q need to be checked too

Continuations

- Some additional computation may be needed to determine value for an "answer" node in Q
  - For example, V → //d//b, with both d and b materialized, and Q → //d/p[b]
  - Need to determine p from either d or b.
- Use ordpath to reference underlying data

Other Correctives

- Unmapped projected nodes in the view indicate columns in V not used to answer query; thus need duplicate elimination
- Projected nodes in V could be mapped to nodes in Q that are not projected; thus need aggregation of view tuples to form single values in query result
- XQuery semantics could require sorting of tuples

Summary

- Materialized views can speed up XQuery
- Relational techniques must be augmented
- View matching requires careful understanding of XPath
- Optionality can be encoded by disjunction with nil
- Handling disjunction is surprisingly tricky
- Potentially exponentially many matches, but quadratic encoding available
- Query rewriting after matching requires careful study of SQL and XQuery

Further Work

- Extend to a broader class of XPath
    - Parent and ancestor axes straightforward
    - Sibling and following axes
    - Node-to-node comparisons
- Extracting MPTs from XQuery
    - Start with work of Arion et al. (2006)
- View selection for a given workload
- Updating XML materialized views

References

- P.-Å. Larson, G. Moerkotte, F.W. Tompa, and J. Zhou, "View Matching of Materialized XML Views," MSFT No. 323085.01, patent application filed June 28, 2008.
- A. Arion, V. Benzaken, I Manolescu, and Y Papakonstantinou. Structured materialized views for XML queries. In *Proc. Int. Conf. on Very Large Data Bases* (*VLDB*), 2007, pp. 87-98.
- A. Arion, V. Benzaken, I Manolescu, Y Papakonstantinou, and R. Vijay. Algebra-based identification of tree patterns in XQuery. in *Flexible Query-Answering Systems 2006*, pp.13-25.
- A. Balmin, F. Özcan, K. Beyer, R. Cochrane, and H. Pirahesh. A framework for using materialized XPath views in XML query processing. In *Proc. Int. Conf. on Very Large Data Bases* (*VLDB*), 2004, pp. 60-71.