

XPath Optimization

Basi di dati 2

Disheng Qiu

disheng.qiu@gmail.com

Luca Rossi

luca.rossi.917@gmail.com

What kind of optimization?

- **Logical optimization**
 - *Equivalence and containment* between XPath queries.
 - Query rewriting in order to gain efficiency
- **Physical optimization**
 - Smart usage of physical structures
 - Storage strategies tailored to enable effective query evaluation.

Physical optimization

- What if a single document is *too big* to fit on a single disk page?
 - We have to *fragment* it into multiple pieces.
 - The way we do it affects *query performances*.

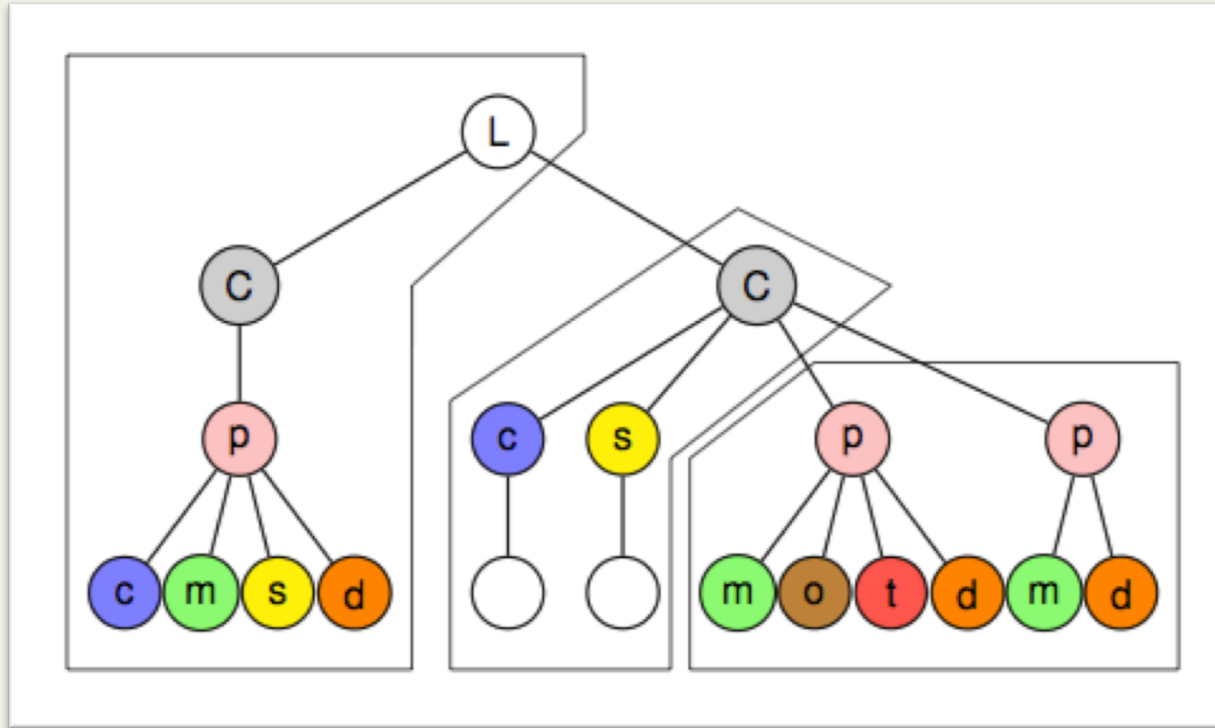
Fragmentation choices

- How could we fragment an XML document?
- Many choices:
 - Traversal-based (ex.: pre-order)
 - Store the document in **relations**. How?
 - Only **one relation** containing edges
 - A relation for each **element**
 - A relation for each **path**
- What are the **pros** and **cons**?
 - What factors should drive these decisions?

Questions that matter

- What are my **data access patterns**?
 - Do I need “wildcard” queries using * or // ?
 - Do I need **updates**?
 - Which are the most **frequent queries**?
 - I need to evaluate them efficiently
- What is the **structure** of my data?
 - What is the **depth** of the document tree?
 - How much **structured** is my data?

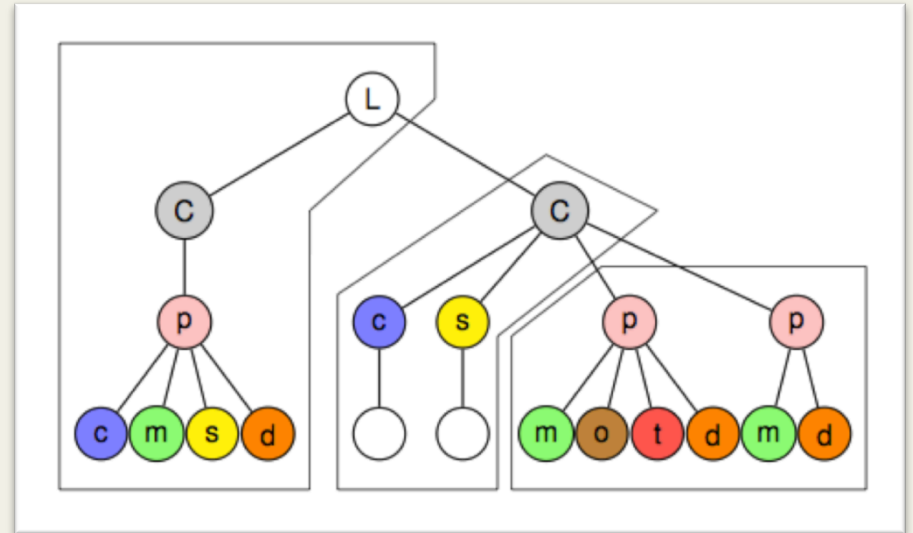
Traversal-based fragmentation



- A (pre-order) traversal groups as many nodes as possible within the current page
- When the page is full, a new page is used to store nodes encountered next

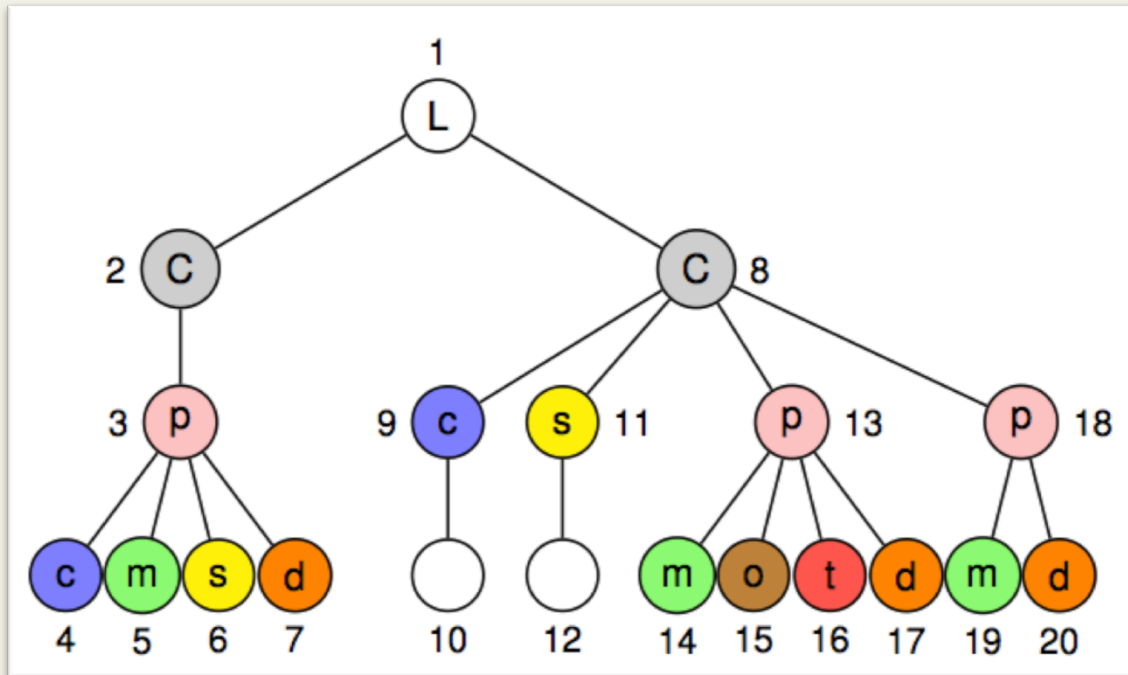
Traversal-based fragmentation

- Pros:
 - Space-effective
 - It works regardless of the data structure



- Cons:
 - Unpredictable partitioning...
 - ...leads to unpredictable performances
 - Updates are hard

Edge relation

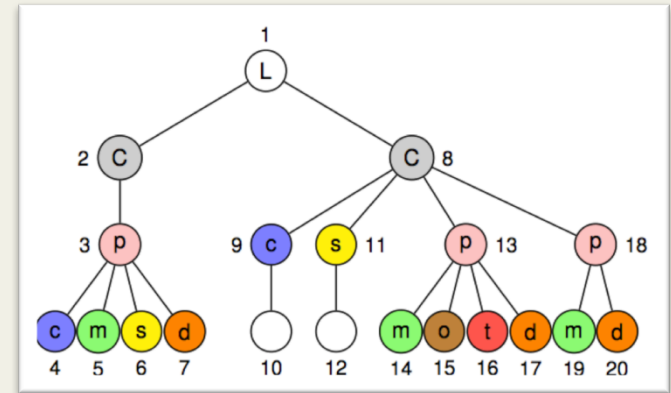


pid	cid	clabel
-	1	CD-library
1	2	CD
2	3	performance
3	4	composer
3	5	composition
3	6	soloist
3	7	date
1	8	CD
...

- Each node is given a unique identifier
- The document is stored as one relation representing a set of edges

Edge relation

- Pros:
 - Space-effective
 - Scalable (just add records)
- Cons:
 - Queries and updates may require **many joins**
 - One join for **each level** to be explored
 - /CD-library/CD/performance → 2 joins
 - /CD-library//composer → ???



pid	cid	clabel
-	1	CD-library
1	2	CD
2	3	performance
3	4	composer
3	5	composition
3	6	soloist
3	7	date
1	8	CD
...

Element-partitioned edge relations

CD-library	CD	performance	composer																								
<table><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>-</td><td>1</td></tr></tbody></table>	pid	cid	-	1	<table><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>1</td><td>2</td></tr><tr><td>1</td><td>8</td></tr></tbody></table>	pid	cid	1	2	1	8	<table><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>2</td><td>3</td></tr><tr><td>8</td><td>13</td></tr><tr><td>8</td><td>18</td></tr></tbody></table>	pid	cid	2	3	8	13	8	18	<table><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>3</td><td>4</td></tr><tr><td>8</td><td>9</td></tr></tbody></table>	pid	cid	3	4	8	9
pid	cid																										
-	1																										
pid	cid																										
1	2																										
1	8																										
pid	cid																										
2	3																										
8	13																										
8	18																										
pid	cid																										
3	4																										
8	9																										

- Same as before, but one edge relation for each element

Element-partitioned edge relations

- Pros:
 - Saves some **space**
 - Saves the I/O needed to find elements with the **same name** (now grouped)
- Cons:
 - Queries and updates still require **many joins**
 - **Less scalable** than before
 - A relation is added for each element name

performance		composer	
pid	cid	pid	cid
2	3	3	4
8	13	8	9
8	18		

CD-library		CD	
pid	cid	pid	cid
-	1	1	2
		1	8

Path-partitioned fragmentation

/CD-library:	<table border="1"><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>-</td><td>1</td></tr></tbody></table>	pid	cid	-	1	/CD-library/CD:	<table border="1"><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>1</td><td>2</td></tr><tr><td>1</td><td>8</td></tr></tbody></table>	pid	cid	1	2	1	8
pid	cid												
-	1												
pid	cid												
1	2												
1	8												
/CD-library/CD/composer:	<table border="1"><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>3</td><td>4</td></tr></tbody></table>	pid	cid	3	4								
pid	cid												
3	4												
/CD-library/CD/performance/composer:	<table border="1"><thead><tr><th>pid</th><th>cid</th></tr></thead><tbody><tr><td>8</td><td>9</td></tr></tbody></table>	pid	cid	8	9								
pid	cid												
8	9												

path
/CD-library
/CD-library/CD
/CD-library/CD/performance
/CD-library/CD/performance/composer
...

- One relation for each **distinct path** in the document
- Another relation containing all **unique paths**

Path-partitioned fragmentation

- Pros:
 - Fast query evaluation (saves some joins)
 - 1) Scan the paths relation to identify matching paths
 - 2) For each such path, scan the corresponding relation
 - Good for structured data
- Cons:
 - Requires a relation for each path
 - Bad match for unstructured data
 - Hard to scale

Logical optimization

Logical optimization

- Rewriting query to be minimal size
- To describe minimal query we have to define two concept:
 - When one query P is contained in query Q
 - When one query P is equals to the query Q

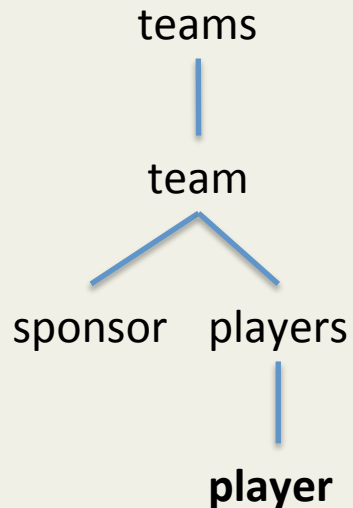
Logical optimization

- Rewriting query to be minimal size
- To describe minimal query we have to define two concept:
 - When one query P is contained in query Q
 - When one query P is equals to the query Q

//player contains(or equals to) /teams/team/players/player

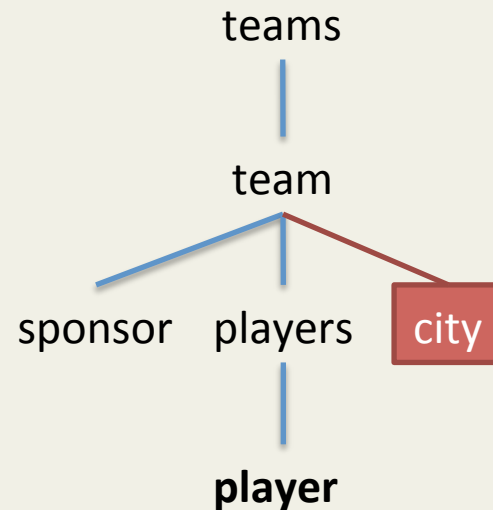
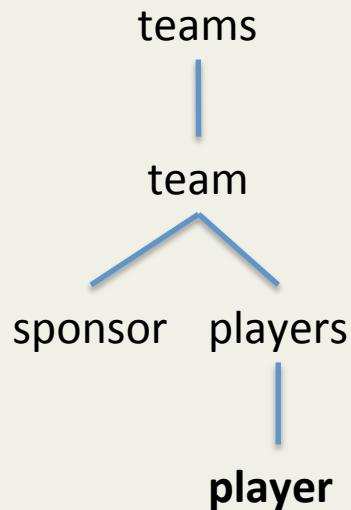
Tree patterns

- Given a Xpath query it is possible to define a tree pattern
- For `/teams/team[sponsor= "Roma"]/players/player`



Tree patterns

- Why tree patterns?
- It is easy to observe when two query are equals and when one query is contained in another



Logical optimization

- Some relations are always true
- Other relations are true only in specific settings (observing DTD)
- `/bookstore/book` contains(or equals) `/bookstore/book[isbn]` (always)
- If books always has isbn (REQUIRED on DTD) the relation is equal

Logical optimization

- Working with minimal size XPath means evaluating fewer “expressions” (and not useful)
- E.g. `//player[number][role][name][stats]`, if the player has always a number, a role, a name and stats it is useless computing `[x]`.
- Equivalent to `//player`

Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<studenti>
  <studente matricola="XX">
    <corsostudi>XX</corsostudi>
    <anagrafica>
      <cognome>XX</cognome>
      <nome>XX</nome>
    <eta>XX</eta>
    </anagrafica>
    <esami>
      <esame corso="XX">
        <data><gg>XX</gg><mm>XX</mm><aa>XX</aa></data>
        <voto>XX</voto>
      </esame>
    </esami>
  </studente>
  ...
</studenti>
```

Problems

1. **dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"**
2. cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"
3. matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"
4. età degli studenti che hanno sostenuto più di 5 esami
5. età media degli studenti di Ingegneria Informatica
6. l'ultimo esame passato dagli studenti di Ingegneria Informatica

Solutions

1. `//studente[.//esame/@corso = "Sistemi Informativi"]/anagrafica`

Problems

1. dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"
2. **cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"**
3. matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"
4. età degli studenti che hanno sostenuto più di 5 esami
5. età media degli studenti di Ingegneria Informatica
6. l'ultimo esame passato dagli studenti di Ingegneria Informatica

Solutions

1. `//studente[./esame/@corso = "Sistemi Informativi"]/anagrafica`
2. `//studente[./esame[voto>28 and @corso = "Analisi"]]/anagrafica/
cognome`

Problems

1. dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"
2. cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"
3. **matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"**
4. età degli studenti che hanno sostenuto più di 5 esami
5. età media degli studenti di Ingegneria Informatica
6. l'ultimo esame passato dagli studenti di Ingegneria Informatica

Solutions

1. `//studente[./esame/@corso = "Sistemi Informativi"]/anagrafica`
2. `//studente[./esame[voto>28 and @corso = "Analisi"]]/anagrafica/
cognome`
3. `//studente[anagrafica/eta < 20 and ./esame/@corso="Chimica"]/
@matricola`

Problems

1. dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"
2. cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"
3. matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"
4. **età degli studenti che hanno sostenuto più di 5 esami**
5. età media degli studenti di Ingegneria Informatica
6. l'ultimo esame passato dagli studenti di Ingegneria Informatica

Solutions

1. `//studente[./esame/@corso = "Sistemi Informativi"]/anagrafica`
2. `//studente[./esame[voto>28 and @corso = "Analisi"]]/anagrafica/
cognome`
3. `//studente[anagrafica/eta < 20 and ./esame/@corso="Chimica"]/
@matricola`
4. `//studente[count(./esame)>5]/anagrafica/eta`

Problems

1. dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"
2. cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"
3. matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"
4. età degli studenti che hanno sostenuto più di 5 esami
5. **età media degli studenti di Ingegneria Informatica**
6. l'ultimo esame passato dagli studenti di Ingegneria Informatica

Solutions

1. `//studente[./esame/@corso = "Sistemi Informativi"]/anagrafica`
2. `//studente[./esame[voto>28 and @corso = "Analisi"]]/anagrafica/
cognome`
3. `//studente[anagrafica/eta < 20 and ./esame/@corso="Chimica"]/
@matricola`
4. `//studente[count(./esame)>5]/anagrafica/eta`
5. `avg(//studente[corsostudi = "Ingegneria Informatica"]/anagrafica/eta)`

Problems

1. dati anagrafici degli studenti che hanno sostenuto l'esame di "Sistemi Informativi"
2. cognomi degli studenti che hanno preso più di 28 al corso di "Analisi"
3. matricola degli studenti che hanno meno di 20 anni e hanno sostenuto l'esame di "Chimica"
4. età degli studenti che hanno sostenuto più di 5 esami
5. età media degli studenti di Ingegneria Informatica
6. **l'ultimo esame passato dagli studenti di Ingegneria Informatica**

Solutions

1. `//studente[./esame/@corso = "Sistemi Informativi"]/anagrafica`
2. `//studente[./esame[voto>28 and @corso = "Analisi"]]/anagrafica/
cognome`
3. `//studente[anagrafica/eta < 20 and ./esame/@corso="Chimica"]/
@matricola`
4. `//studente[count(./esame)>5]/anagrafica/eta`
5. `avg(//studente[corsostudi = "Ingegneria Informatica"]/anagrafica/eta)`
6. `//studente[corsostudi="Ingegneria Informatica"]//esame[last()]`