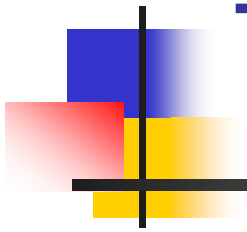
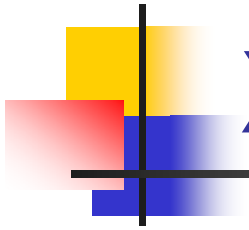


COMP4317: XML & Database
Tutorial 10: Xpath & SQL

Week 11

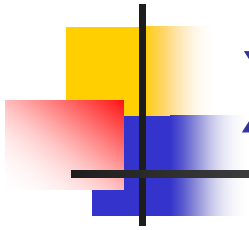
Thang Bui @ CSE.UNSW



XML to Database

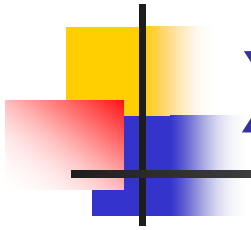
- Create tables

```
CREATE TABLE filename_tbl (
    pre INTEGER PRIMARY KEY,
    post INTEGER,
    level INTEGER,
    tag VARCHAR(256),
    text VARCHAR(1000)
);
CREATE TABLE filename_attr (
    pre INTEGER REFERENCES filename_tbl (PRE),
    attr VARCHAR(256),
    value VARCHAR(1000)
);
```



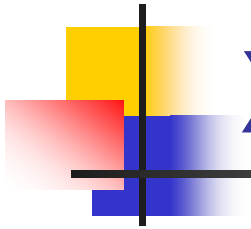
XML to Database

- Build the tree with pre,post
  - Start document
    - Insert into stack a document node `<0,?,0,null,null>`
    - `pre=post=level=0`
  - Start element
    - Make a node `<++pre, ?, ++level, tag, null>`
    - *Add to the parent's children*
    - Push to stack



XML to Database

- Text:
  - Make a node `<++pre,++post,(level+1),null, text>`
  - *Add to the parent's children*
- End element
  - Assign `++post` to node on the stack
  - `level--`
- End document
  - Assign `++post` to the document node



XML to Database

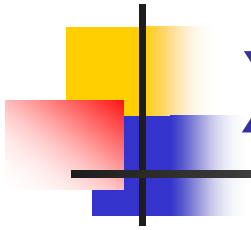
- Generate INSERT statements

```
INSERT INTO filename_tbl (pre, post, level, tag, text)
VALUES (n.pre, n.post, n.level, n.tag, n.text);

INSERT INTO filename_attr (pre, attr, value)
VALUES (x.pre, x.attr, x.value);
```

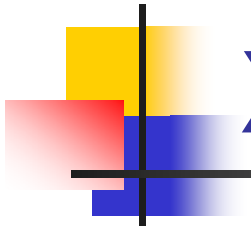
- Note:
  - NULL value in database is NULL, not "null“
- Remove the tables

```
DROP TABLE filename_tbl;
DROP TABLE filename_attr;
```



XPath to SQL

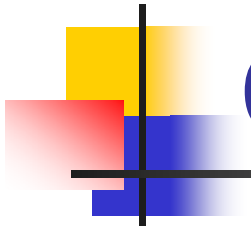
- First instance of the table r_0 for pre=0
- Each tag on XPath
  - A new instance of the table r_i
  - tagname is not "*" => r_i.tag="tagname"
  - r_i.pre > r_{i-1}.pre AND r_i.post < r_{i-1}.post
- Select the last r_i.pre



XPath to SQL

- Ex: //a//*//c
- SQL:

```
SELECT DISTINCT r4.pre
FROM table r1, table r2, table r3, table r4
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre>r2.pre AND r3.post<r2.post AND r3.tag!=""
  AND r4.pre>r3.pre AND r4.post<r3.post AND r4.tag="c"
ORDER BY r4.pre
```

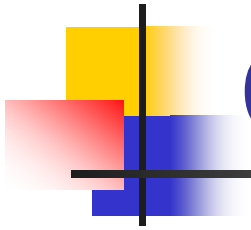


Children?

- Ex: /a
- SQL

Correct if there is only child!

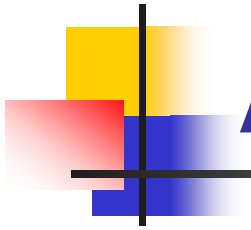
```
SELECT DISTINCT r2.pre
FROM table r1, table r2
WHERE r1.pre=0
  AND r2.pre=r1.pre+1 AND r2.post=r1.post-1
  AND r2.tag="a"
ORDER BY r2.pre
```



Children?

- Better use “level”
- Ex: //a/b
- SQL:

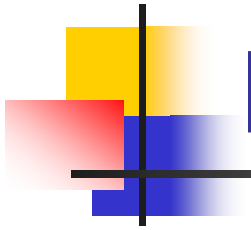
```
SELECT DISTINCT r3.pre
FROM table r1, table r2, table r3
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre>r2.pre AND r3.post<r2.post AND r3.tag="b"
                    AND r3.level=r2.level+1
ORDER BY r3.pre
```



Ancestor axis

- x is an ancestor of y: x.pre < y.pre AND x.post > y.post
- Ex: //a/ancestor::b
- SQL:

```
SELECT DISTINCT r3.pre
FROM table r1, table r2, table r3
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre<r2.pre AND r3.post>r2.post AND r3.tag="b"
ORDER BY r3.pre
```



Filters

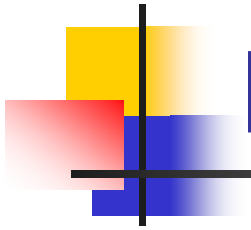
- Two options:
  - As normal table joint in the main statement
  - As a subquery in the WHERE clause

Filters – Main statement

- Ex: //a[.//b]
- SQL:

When there may be many “b” in subtree “a”

```
SELECT DISTINCT r2.pre
FROM table r1, table r2, table r3
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre>r2.pre AND r3.post<r2.post AND r3.tag="b"
ORDER BY r2.pre
```



Filters – Main statement

- Ex: //a[.//b]
- SQL:

```
SELECT DISTINCT r2.pre
FROM table r1, table r2, table r3
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre>r2.pre AND r3.post<r2.post AND r3.tag="b"
ORDER BY r2.pre
```

- How about: //a[not(.//b)] ?
  - NOT on assignment 5!

Filters – Subquery

- Ex: //a[.//b]
- SQL:

```
SELECT DISTINCT r2.pre
FROM table r1, table r2
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND EXISTS
      (SELECT sr1.pre FROM table sr1
       WHERE sr1.pre>r2.pre AND sr1.post<r2.post
                            AND sr1.tag="b")
ORDER BY r2.pre
```

Filters – Subquery

- Ex: //a[not(.//b)]
- SQL:

NOT on assignment 5!

```
SELECT DISTINCT r2.pre
FROM table r1, table r2
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND NOT EXISTS
      (SELECT sr1.pre FROM table sr1
       WHERE sr1.pre>r2.pre AND sr1.post<r2.post
                            AND sr1.tag="b")
ORDER BY r2.pre
```

Filters – Subquery

- Ex: //a[.//b][./c]
- SQL:

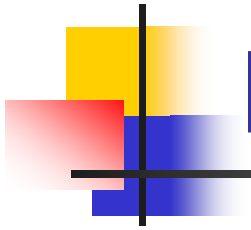
```
…
 AND EXISTS
     (SELECT sr11.pre FROM table sr11
      WHERE sr11.pre>r2.pre AND sr11.post<r2.post
                            AND sr11.tag="b")
 AND EXISTS
     (SELECT sr12.pre FROM table sr12
      WHERE sr12.pre>r2.pre AND sr12.post<r2.post
        AND sr12.level=r2.level+1 AND sr12.tag="c")
```

Filters – Subquery

- Ex: //a[//b]
- SQL:

Filters relate to root node

```
SELECT DISTINCT r2.pre
FROM table r1, table r2
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND EXISTS
      (SELECT sr1.pre FROM table sr1
       WHERE sr1.pre>r1.pre AND sr1.post<r1.post
                            AND sr1.tag="b")
ORDER BY r2.pre
```

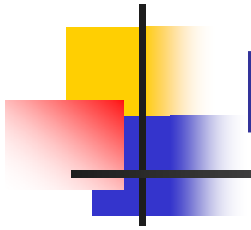


Filters - Attributes

- Ex: //a[@x='y']
- SQL:

Attribute table

```
SELECT DISTINCT r2.pre
FROM table r1, table r2, table_attr r3,
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND r3.pre=r2.pre AND r3.attr="x"
                    AND r3.value='y'
ORDER BY r2.pre
```

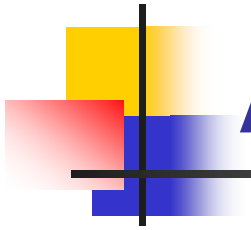


Filters - Attributes

- Ex: //a[@x='y']
- SQL:

Attribute table

```
SELECT DISTINCT r2.pre
FROM table r1, table r2
WHERE r1.pre=0
  AND r2.pre>r1.pre AND r2.post<r1.post AND r2.tag="a"
  AND EXISTS
      (SELECT sr1.pre FROM table_attr sr1
       WHERE sr1.pre=r2.pre AND sr1.attr="x"
                            AND sr1.value='y')
ORDER BY r2.pre
```



Assignment 5

- // / with tag, *, text()
- /ancestor::tag OR /ancestor::*
- [./] OR [.//] OR [/] OR [//]
  - With tag, *
- Bonus:
  - [@attr=value]