CSCI-585
Exam II review

Lian Liu

Overview of Exam II

Possible topics:

- XML (20-30%)
- Query analysis and/or optimization (10-15%)
- Distributed databases (10-20%)
- OLAP and decision support (10-20%)

Overview of Exam II

Types of questions

- Same as Exam I:
  - T/F questions (5-10%)
  - Please describe…? What's the pros & cons of …? (10-20%)
  - Technical questions (50-70%)

XML

1. Consider the following XML DTD:

```
<!ELEMENT bib (entry*)>
<!ELEMENT entry (book | collection)>
<!ELEMENT book (author, title)>
<!ELEMENT collection (author, title,
publisher?, article+)>
<!ELEMENT article (author, title)>
<!ATTLIST entry (year CDATA #REQUIRED)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

(a) Does the following XML fragment conform to the DTD above? Why or why not?

```
<bib>
      <entry>
            <year>2008</year>
            <collection>
                  <author>A. Turing</author>
                  <title>Tests in CS</title>
            </collection>
      <book>
            <title>The Hunger Games</title>
            <author>Suzanne Collins</author>
      </book>
</bib>
```

XML

(a) Does the following XML fragment conform to the DTD above? Why or why not?

```
<bib>
      <entry>
            <year>2008</year>
            <collection>
                  <author>A. Turing</author>
                  <title>Tests in CS</title>
            </collection>
      <book>
            <title>The Hunger Games</title>
            <author>Suzanne Collins</author>
      </book>
</bib>
```

SOLUTION:

No because:
1) year is an attribute, not an element;
2) collection must have at least one article (but the publisher can be missing);
3) author must precede title in book;
4) there is no closing“entry" tag.

XML

1. Consider the following XML DTD:

```
<!ELEMENT bib (entry*)>
<!ELEMENT entry (book | collection)>
<!ELEMENT book (author, title)>
<!ELEMENT collection (author, title,
publisher?, article+)>
<!ELEMENT article (author, title)>
<!ATTLIST entry (year CDATA #REQUIRED)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

(b) Write an XPath expression for each of the following:

i. The names of all authors of books, collections or articles in collections.

ii. The names of authors of books that were published in 2012.

XML

1. Consider the following XML DTD:

```
<!ELEMENT bib (entry*)>
<!ELEMENT entry (book | collection)>
<!ELEMENT book (author, title)>
<!ELEMENT collection (author, title,
publisher?, article+)>
<!ELEMENT article (author, title)>
<!ATTLIST entry (year CDATA #REQUIRED)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

(b) Write an XPath expression for each of the following:

i. The names of all authors of books, collections or articles in collections.

SOLUTION:

//author

ii. The names of authors of books that were published in 2012.

SOLUTION:

/bib/entry[@year=2012]/book/author

XML

2. Assume that you are given an XML document called “doc. xml”. State in English what the following queries return.

(a)

```
<XML>
{
for $c in document(doc)//course[@prof=‘‘Susan Davidson’’]
let $g:= c//grade[text()= ‘‘A’’]
return
<SDCourse>
<course cnum= {$c/@cnum} />
<num>{count($g)}</num>
</SDCourse>
}
</XML>
```

(b)

```
<XML>
{
for $p in distinct-values(document(doc.xml)
//course/@prof)
return
<courses prof={$p}>
{
for $c in document(doc.xml)//course
where $c/@prof = $p
return <course>{$c/@cnum} </course>
}
</courses>
}
</XML>
```

XML

(a)

```
<XML>
{
for $c in document(doc)//course[@prof=‘‘Susan
Davidson’’]
let $g:= c//grade[text()= ‘‘A’’]
return
<SDCourse>
<course cnum= {$c/@cnum} />
<num>{count($g)}</num>
</SDCourse>
}
</XML>
```

SOLUTION:

For every course with professor ’Susan Davidson’, return the course (with cnum as an attribute) and number of A’s received as an element with tag num.

XML

(b)

```
<XML>
{
for $p in distinct-values(document(doc.xml)
//course/@prof)
return
<courses prof={$p}>
{
for $c in document(doc.xml)//course
where $c/@prof = $p
return <course>{$c/@cnum} </course>
}
</courses>
}
</XML>
```

SOLUTION:

For each professor, return an element with tag courses and professor as an attribute containing as subelements it all courses taught by that professor, represented as an element with tag course and value of cnum.

XML

3. Write down the XML Schema for the following XML document.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

Query optimization

1. Consider the following database tables at a Sailing club:

Sailors (<u>sid</u>, sname, rating, age)
Boats (<u>bid</u>, bname, color)
Reserves (<u>sid</u>, <u>bid</u>, day, rname)
Each table has some additional fields that we are not interested in.

a) For the following query:

```
SELECT sname
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid AND B.bid = R.bid AND B.
Color=’RED’
```

Draw the most basic query tree.

b) Use the heuristics for algebraic query optimization to transform/restructure the query-tree you generated in part a) above into a more efficient query-tree.

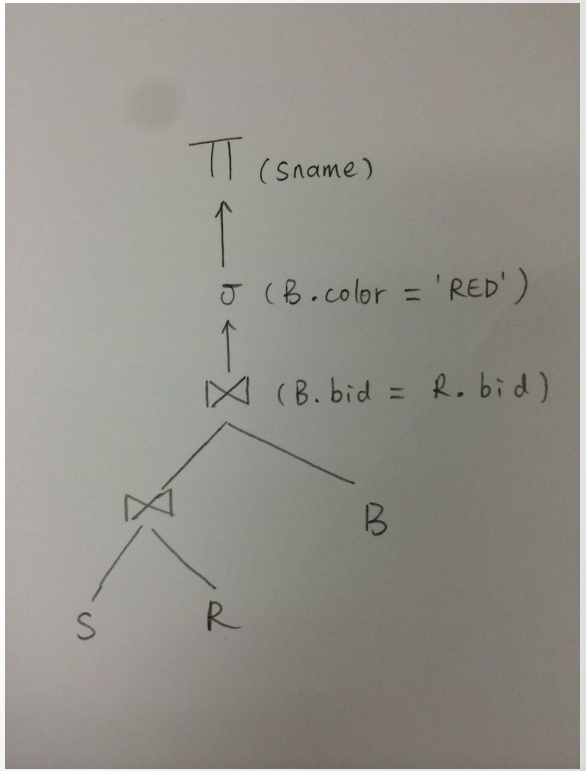
c) State any assumptions you are making.

Query optimization

a) For the following query:

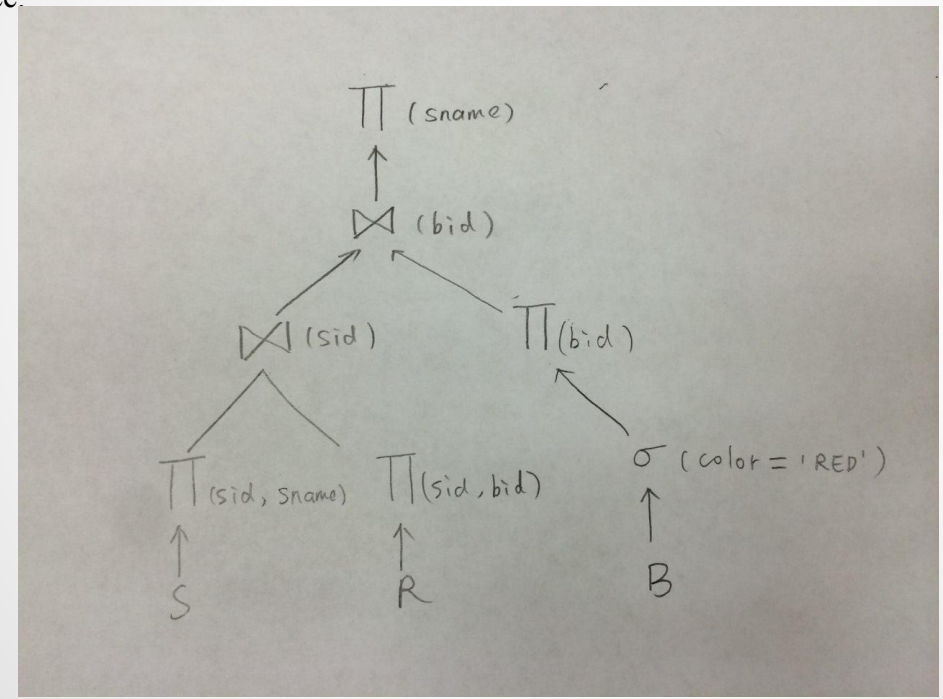
```
SELECT sname
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid AND B.bid = R.bid AND B.
Color=’RED’
```

Draw the most basic query tree.



Query optimization

b) Use the heuristics for algebraic query optimization to transform/restructure the query-tree you generated in part a) above into a more efficient query-tree.



Query optimization

c) State any assumptions you are making.

SOLUTION:

We assume not all boats are red.

OLAP

1. The following two dimensional OLAP cube contains sales forecasts (in millions)

	2010	2011	2012	2013	2014	2015	2016
LA	1	2	2	2	3	3	4
NY	1	2	2	2	2	3	3
Paris	1	1	1	2	2	2	3

a) Compute the corresponding prefix sum matrix.

b) Compute the results of the following range query using the prefix sum matrix. Find total sales in NY and Paris from 2012 thru 2015.

OLAP

a) Compute the corresponding prefix sum matrix.

SOLUTION:

	2010	2011	2012	2013	2014	2015	2016
LA	1	**3**	5	7	10	**13**	17
NY	2	6	10	14	19	25	32
Paris	3	**8**	13	19	26	**34**	44

b) Compute the results of the following range query using the prefix sum matrix. Find total sales in NY and Paris from 2012 thru 2015.

SOLUTION:

34 - 13 - 8 + 3 = 16

Distributed databases

1. Consider the following three relations with their keys underlined:

Movie (Movie_ID, Title, Director, Budget)
Theater (Theater_ID, Name, Location)
SHOW (Movie_ID, Theater_ID, StartDate, Duration)

Furthermore, assume the following two queries:

• Query 1 (q1):
select Movie.Movie_ID, Title, SHOW.Theater_ID, StartDate
from Movie, SHOW
where Movie.Movie_ID = SHOW.Movie_ID **and** Duration = 30

Suppose q1 is executed by an application that is located at sites S1 and S2, with frequencies 3 and 2, respectively.

• Query 2 (q2):
select Movie_ID, Director, budget
from Movie

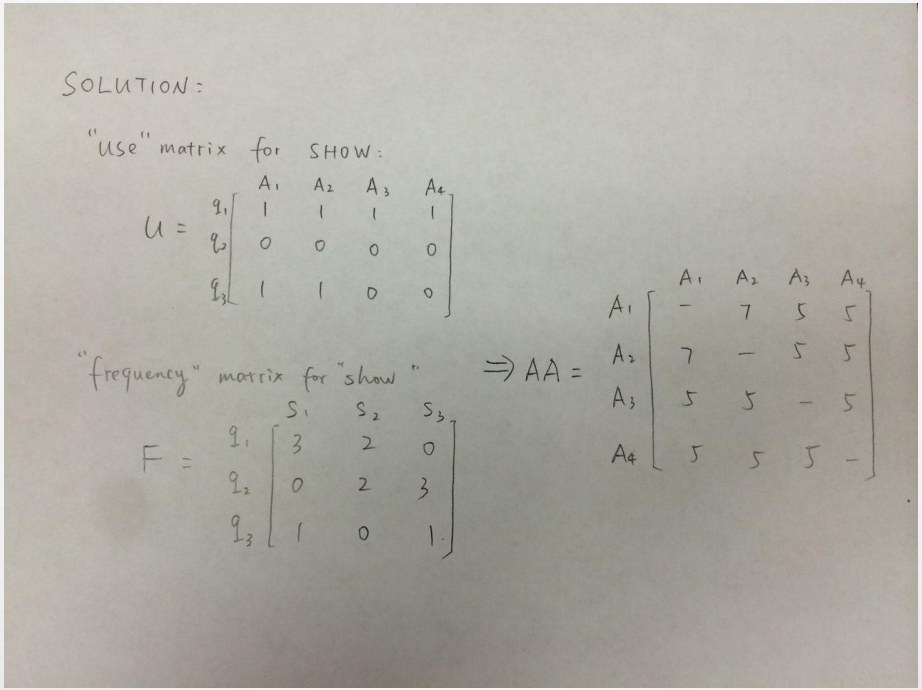
Suppose q2 is executed by an application that is located at sites S2 and S3, with frequencies 2 and 3, respectively.

• Query 3 (q3):
select MAX(budget)
from Movie, Theater, SHOW
where Movie.Movie_ID = SHOW.Movie_ID **and**
SHOW.Theater_ID = Theater.
Theater_ID **and**
Location = ‘Chicago’

Suppose q3 is executed by an application that is located at sites S1 and S3, with frequencies 1 and 1, respectively.

Distributed databases

a) Construct the affinity matrix AA containing all attributes of the relation SHOW.



SOLUTION:

"use" matrix for SHOW:

$$U = \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

"frequency" matrix for "show"

$$F = \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \end{array} \begin{array}{c} \begin{array}{ccc} S_1 & S_2 & S_3 \end{array} \\ \begin{bmatrix} 3 & 2 & 0 \\ 0 & 2 & 3 \\ 1 & 0 & 1 \end{bmatrix} \end{array}$$

$$\Rightarrow AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{bmatrix} - & 7 & 5 & 5 \\ 7 & - & 5 & 5 \\ 5 & 5 & - & 5 \\ 5 & 5 & 5 & - \end{bmatrix} \end{array}$$

Distributed databases

b) Use one of the approaches we have discussed in class to come up with a vertical fragmentation of the relation SHOW.