Anne Surkey
Database Processing CIS 451/551
Dec 4, 2006
Assignment 6

1. DTD and XML Schema for the same Bibliography model.

DTD

```
<!DOCTYPE bib [
<!ELEMENT book (title, author+, publisher, keyword+)>
<!ELEMENT publisher (pub-name, pub-branch) >
<!ELEMENT title ( #PCDATA )>
<!ELEMENT author ( #PCDATA )>
<!ELEMENT keyword ( #PCDATA )>
<!ELEMENT pub-name( #PCDATA )>
<!ELEMENT pub-branch( #PCDATA )>
] >
```

XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="book">
  <xs:complexType name="BookType">
    <xs:sequence>
      <xs:element name="title" type="xs:string" />
        <xs:element name="author" minOccurs="1" maxOccurs="unbounded" type="xs:string" />
        <xs:element ref="publisher" />
        <xs:element name="keyword" minOccurs="1" maxOccurs="unbounded" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="publisher">
   <xs:complexType name="PubType">
     <xs:sequence>
       <xs:element name="pub-name" type="xs:string" />
       <xs:element name="pub-branch" type="xs:string" />
     </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

2.
r: t=40,000; b=800; t/b=50
s: t=30,000; b=2,000; t/b=15

r is the outer relation and s is the inner relation
nested loop join:.
worst case: 40,000x2000 + 800 = **80,000,800 block transfers**
40,000 + 800 = **40,800 seeks**
best case: 2,000 + 800 **= 2,800 block transfers and 2 seeks**

block-nested loop join: (assuming memory size M=3)
worst case: 2,000x800 + 800 = **1,600,800 block transfers**
2x800 = **1,600 seeks**
best case: 2,000 + 800 **= 2,800 block transfers and 2 seeks**

index-nested loop join:
30,000*(4+1) + 800 = **150,800 block transfers**
30,000*(4+1) + 800 = **150,800 seeks**

merge-join: (assuming memory size M=3, buffer block size of 1)
block transfer: (sorting phase) 800(2xlg[800/3] + 1) + 2000(2xlg[2000/3] + 1) + 800 + 2,000 = 60,000 + (merging phase) 800 + 2,000 = 2,800
TOTAL: 62,800 block transfers

Seeks: (sorting phase) 2x[800/3] + 800(2xlg[800/3] – 1) + 2000(2xlg[2000/3] - 1) + 800 + 2,000 = 56,268 + (merging phase) 800 + 2,000 = 2,800
TOTAL: 59,068 seeks

hash join: (assuming buffer block size of 1)
3(2,000 + 800) = **8,400 block transfers**
2(2,000 + 800) = **5,600 seeks**

3. Use the index on (branch-name, branch-city) first to retrieve each tuple for "downtown" and "< Brooklyn", then for each tuple check the assets individually.

4. The schedule is conflict serializable because there are no cycles in the precedence graph. Any of many additional edges that creates a cycle (such as <T5,T1>) will make it non-conflict serializable.

5. Yes, this schedule satisfies the timestamp ordering protocol because all the corresponding write operations are ordered according to the transaction timestamps. If, for example, the *write(B)* lines were swapped, then there would be a problem.