Tutorial 9 (Solution)
SQL+XML

Classroom Exercise

Question 1

1. Consider the following relation:

Articles (ID, title, journal, issue, year, startpage, endpage, TR-ID)

It contains information on articles published in scientific journals. Each article has a unique ID, a title, and information on where to find it (name of journal, what issue, and on which pages). Also, if results of an article previously appeared in a "technical report" (TR), the ID of this technical report can be specified. We have the following information on the attributes:

• For each journal, an issue with a given number is published in a single year.

• The endpage of an article is never smaller than the startpage.

• There is never (part of) more than one article on a single page.

Consider the following six queries on Articles:

a. SELECT title FROM Articles WHERE year=2005;

b. SELECT title FROM Articles WHERE endpage=100;

c. SELECT title FROM Articles WHERE year>1995 AND year<2000;

d. SELECT title FROM Articles WHERE journal='JACM' AND issue=55;

e. SELECT title FROM Articles WHERE issue=55 AND journal='JACM';

f. SELECT title FROM Articles WHERE endpage-startpage>50;

Indicate which of the above queries would likely be faster (based on the knowledge you have from the course), if all of the following indexes were created.

• CREATE INDEX Idx1 ON Articles (year, startpage);

• CREATE INDEX Idx2 ON Articles (startpage, endpage);

• CREATE INDEX Idx3 ON Articles (journal, issue, year);

Question 1

Articles (ID, title, journal, issue, year, startpage, endpage, TR-ID)

Consider the following six queries on Articles:

a. SELECT title FROM Articles WHERE year=2005;

b. SELECT title FROM Articles WHERE endpage=100;

c. SELECT title FROM Articles WHERE year>1995 AND year<2000;

d. SELECT title FROM Articles WHERE journal='JACM' AND issue=55;

e. SELECT title FROM Articles WHERE issue=55 AND journal='JACM';

f. SELECT title FROM Articles WHERE endpage-startpage>50;

Indicate which of the above queries would likely be faster (based on the knowledge you have from the course), if all of the following indexes were created.

• CREATE INDEX Idx1 ON Articles (year, startpage);

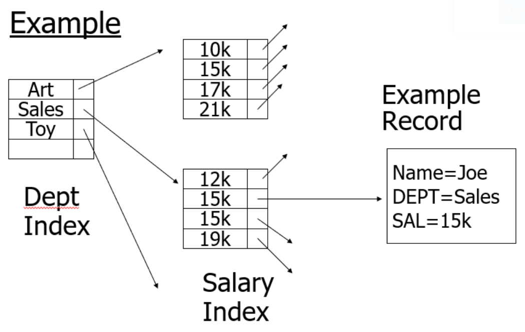
• CREATE INDEX Idx2 ON Articles (startpage, endpage);

• CREATE INDEX Idx3 ON Articles (journal, issue, year);

Solution:

#	1	2	3	4	5	6
Faster	X		X	X	X	
Same		X				X

Note:



[https://stackoverflow.com/questions/24315151/does-order-of-fields-of-multi-column-index-in-mysql-matter/24315527#24315527]

When discussing multi-column indexes, we can use an analogy to a telephone book. A telephone book is basically an index on last name, then first name. So the sort order is determined by which "column" is first. Searches fall into a few categories:

1. If you look up people whose last name is Smith, you can find them easily because the book is sorted by last name.
2. If you look up people whose first name is John, the telephone book doesn't help because the Johns are scattered throughout the book. You have to scan the whole telephone book to find them all.

3. If you look up people with a specific last name Smith and a specific first name John, the book helps because you find the Smiths sorted together, and within that group of Smiths, the Johns are also found in sorted order.

If you had a telephone book sorted by first name then by last name, the sorting of the book would assist you in the above cases #2 and #3, but not case #1.

So the order of columns in a multi-column index definitely matters. One type of query may need a certain column order for the index. If you have several types of queries, you might need several indexes to help them, with columns in different orders.

Question 2

2. Consider the following XML DTD for an employee database.

```
<!DOCTYPE emp [
<!ELEMENT emp (ename, children*, skills*)>
<!ELEMENT children (name, birthday)>
<!ELEMENT birthday (day, month, year)>
<!ELEMENT skills (type, exams+)>
<!ELEMENT exams (year, city)>
<!ELEMENT ename( #PCDATA )>
<!ELEMENT name( #PCDATA )>
<!ELEMENT day( #PCDATA )>
<!ELEMENT month( #PCDATA )>
<!ELEMENT year( #PCDATA )>
<!ELEMENT type( #PCDATA )>
<!ELEMENT city( #PCDATA )>
] >
```

Create a valid XML document that follows the rules of the above DTD.

Your document must illustrate all the rules of the DTD.

Question 2

```
<!DOCTYPE emp [
<!ELEMENT emp (ename, children*, skills*)>
<!ELEMENT children (name, birthday)>
<!ELEMENT birthday (day, month, year)>
<!ELEMENT skills (type, exams+)>
<!ELEMENT exams (year, city)>
<!ELEMENT ename( #PCDATA )>
<!ELEMENT name( #PCDATA )>
<!ELEMENT day( #PCDATA )>
<!ELEMENT month( #PCDATA )>
<!ELEMENT year( #PCDATA )>
<!ELEMENT type( #PCDATA )>
<!ELEMENT city( #PCDATA )>
]>
```

Create a valid XML document that follows the rules of the above DTD.

Your document must illustrate all the rules of the DTD.

```
<emp>
    <ename> John Jensen</ename>
    <children>
        <name> Peter Jensen</name>
        <birthday>
            <day>15</day>
            <month>August<month>
            <year>2017</year>
        </birthday>
    </children>
    <skills>
        <type>C++</type>
        <exams>
            <year>2018</year>
            <city>Singapore</city>
        </exams>
    </skills>
</emp>
```

<!ELEMENT *name content*>

- Element Types are composed of:
  - Subelements (identified by Name)
  - Attribute lists (identified by Name)
  - Selection of Subelemente (choice)
  - PCDATA **text that WILL be parsed by a parser**
- Quantifier for Subelements and Choice
  - "+" for at least 1
  - "*" for 0 or more
  - "?" for 0 or 1
  - Default: exactly 1
- EMPTY and ANY are special predefined Types

- "|": Declaring either/or Content
  <!ELEMENT note (to,from,header,(message|body))>

Solution:

Any specific example illustrating all the rules will do.

```
<emp>
      <ename> John Jensen</ename>
      <children>
            <name> Peter Jensen</name>
            <birthday>
                 <day>15</day>
                 <month>August<month>
                 <year>2017</year>
            </birthday>
      </children>
      <skills>
            <type>C++</type>
            <exams>
                  <year>2018</year>
                  <city>Singapore</city>
            </exams>
      </skills>
</emp>
```

Question 3

3. Consider the following relational database schema about students and courses (primary keys are underlined):

STUDENT(SID, NAME, EMAIL)

COURSE(CID, NAME, INSTRUCTOR, ROOM)

ENROLLS(SID, CID, GRADE)

In ENROLLS, SID and CID are foreign keys into STUDENT and COURSE, respectively. The following two constraints also hold: (a) each student is enrolled in some course, and (b) each course has at least one student enrolled.

(i) Design a DTD for exporting this data as an XML view to the chair's office. The chair wants to see the data grouped by courses and needs to have access to all the information in the database.

(ii) Give an example of a database instance for the relational schema and show the resulting document for the XML view according to the DTD that you have designed.

Question 3

STUDENT(SID, NAME, EMAIL)

COURSE(CID, NAME, INSTRUCTOR, ROOM)

ENROLLS(SID, CID, GRADE)

(i) Design a DTD for exporting this data as an XML view to the chair's office. The chair wants to see the data grouped by courses and needs to have access to all the information in the database.

Part (i)

```
<!DOCTYPE courses  [
<!ELEMENT  courses  (course*)>
<!ELEMENT  course  (name,  instructor,  room,  student+)>
<!ELEMENT  student  (name,  email,  grade)>
<!ATTLIST  course  cid  ID  #required>
<!ATTLIST  student  sid  CDATA  #required>
<!ELEMENT  name  (#PCDATA)>
<!ELEMENT  instructor  (#PCDATA)>
<!ELEMENT  room  (#PCDATA)>
<!ELEMENT  email  (#PCDATA)>
<!ELEMENT  grade  (#PCDATA)>
]>
```

<!ELEMENT *name content*>

<!ATTLIST element-name attribute-name attribute-type attribute-value>

The **attribute-type** can be one of the following:

Type	Description
CDATA	The value is character data
(*en1*|*en2*|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element

The **attribute-value** can be one of the following:

Value	Explanation
value	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is optional
#FIXED *value*	The attribute value is fixed

XML Elements vs. Attributes

```
<person sex="female">
 <firstname>Anna</firstname>
 <lastname>Smith</lastname>
</person>
```

```
<person>
 <sex>female</sex>
 <firstname>Anna</firstname>
 <lastname>Smith</lastname>
</person>
```

In the first example sex is an attribute. In the last, sex is an element. Both examples provide the same information.

There are no rules about when to use attributes and when to use elements. Attributes are handy in HTML. **In XML my advice is to avoid them. Use elements instead.**

Avoid XML Attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)

Attributes are difficult to read and maintain. Use elements for data. Use attributes for information that is not relevant to the data.

XML Attributes for Metadata

Sometimes ID references are assigned to elements. These IDs can be used to identify XML elements in much the same way as the ID attribute in HTML. This example demonstrates this:

```
<messages>
 <note id="501">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
 </note>
 <note id="502">
  <to>Jani</to>
  <from>Tove</from>
  <heading>Re: Reminder</heading>
  <body>I will not</body>
 </note>
</messages>
```

The ID above is just an identifier, to identify the different notes. It is not a part of the note itself.

What I'm trying to say here is that metadata (data about data) should be stored as attributes, and that data itself should be stored as elements.

Part (ii)

(ii) Give an example of a database instance for the relational schema and show the resulting document for the XML view according to the DTD that you have designed.

Student

sid	name	email
s01	Jim Halpert	j.halpert@cis.upenn.edu
s02	Dwight Schrute	schrutefarms@cis.upenn.edu
s03	Nelly Bertram	n.bertram@cis.upenn.edu

Course

cid	name	instructor	room
CIS550	DBMS	Dr. Davidson	SKIRAUD
CIS552	Advanced Programming	Dr. Weirich	TOWNE321
CIS500	Software Foundations	Dr. Pierce	MOORE216

Enrolls

sid	cid	grade
s01	CIS550	A
s01	CIS552	A
s02	CIS550	B+
s02	CIS500	B-
s03	CIS550	A-

```
<?xml version="1.0"?>
<!DOCTYPE courses SYSTEM "courses.dtd">
<courses>
  <course cid = "CIS550">
     <name>DBMS</name>
     <instructor>Dr. Davidson</instructor>
     <room>SKIRAUD</room>
     <student sid = "s01">
        <name>Jim Halpert</name>
        <email>j.halpert@cis.upenn.edu</email>
        <grade>A</grade>
     </student>
     <student sid = "s02">
        <name>Dwight Schrute</name>
        <email>schrutefarms@cis.upenn.edu</email>
        <grade>B+</grade>
     </student>
     <student sid = "s03">
        <name>Nelly Bertram</name>
        <email>n.bertram@cis.upenn.edu</email>
        <grade>A-</grade>
     </student>
  </course>
   …
</courses>
```

```
<!DOCTYPE courses [
<!ELEMENT courses (course*)>
<!ELEMENT course (name, instructor, room, student+)>
<!ELEMENT student (name, email, grade)>
<!ATTLIST course cid ID #required>
<!ATTLIST student sid CDATA #required>
<!ELEMENT name (#PCDATA)>
<!ELEMENT instructor (#PCDATA)>
<!ELEMENT room (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT grade (#PCDATA)>
]>
```