-1 +1

Question4: 1: (1) <option> Air Bag </option> (2) <color> blue </color> (3) <model> Voxy <special>
<color> blue </color> Tourig </special>
<color> red </color> </model> 5

2. <!DOCTYPE cars [4 1
<!ELEMENT cars (car)* >
<!ELEMENT car (maker*, year*, model*, option*, color*) > 5
<!ELEMENT maker >
<!ELEMENT year >
<!ELEMENT model (special(*) >
<!ELEMENT special (option*) >
<!ELEMENT option >
<!ELEMENT color >
<!ATTLIST model grade #CDATA #REQUIRED>

3

4. XML allows user to define new collections of tags that can be used to structure any type of data or document that user wishes to transmit.
However, XML does not prevent a user from designing tags that encode the display of the data in Web browser.

24

Information Organization Final Examination (January 25th, 2012)

Notice

- Write your name and student number to all the answer sheets.
- You can bring printed materials (slides, notes, memos, etc). But NO electronic device, such as cell phone, laptop, electronic dictionary, or calculator is allowed.
- You can write answers either in English or Japanese.

Question 1.

1. Suppose that user John executed the following SQL commands.

```
John:  CREATE TABLE lunch(name, price);
       REVOKE ALL PRIVILEGES ON lunch FROM PUBLIC;
       GRANT SELECT, INSERT ON lunch TO Tom WITH GRANT OPTION;
       GRANT SELECT, DELETE ON lunch TO Keiko;
```

Then, the following commands (1)-(4) were executed in this order, by the users indicated at the front of each command line.

```
(1) Tom:    GRANT SELECT ON lunch TO Hanako;
(2) Keiko:  INSERT INTO lunch(name, price) VALUES('bento','500');
(3) John:   REVOKE SELECT ON lunch FROM Tom;
(4) Hanako: SELECT * FROM lunch;
```

For each of (1)-(4), answer whether each command was granted or denied by the database. Also, describe reasons why.

2. What is the name of the access control model in which users can have privileges on objects and give privileges to other users?

Question 2.

Let us consider multi-granularity lock modes consisting of IS, IX, S, and X. Here IS is intension shared lock, IX is intension-exclusive lock, S is shared lock, and X is exclusive lock. Multi-granularity locking is applied on a hierarchy of nodes having containment relationships. Examples of nodes are table, index, and tuples.

1. To lock a tuple in X mode, what kinds of locks are necessary before locking the tuple? Also explain why.
2. To unlock a tuple locked in X mode, what kind of unlock order must be followed? Also explain why.

Suppose that transaction T1 uses the index to read only part of table R. For simplicity, assume that the index height is one. To access R, T1 uses the following locks: T1 gets an IS lock on R, gets an IS lock on an index page, and repeatedly gets an S lock on tuples of R.

3. Describe an appropriate lock schedule for transaction T2: T2 updates all the tuples of R.
4. Argue whether executions of T1 and T2 can be overlapped.
5. Describe an appropriate lock schedule for transaction T3: T3 reads a part of R using the index, and updates a few of them.
6. Argue whether executions of T1 and T3 can be overlapped.

Question 3.

Consider the following XML document d_1:

```
<lunch>
  <menu name="ramen" type="noodle">
    <price> 500  </price>
    <options>
      <option price="100"> large_noodle </option>
      <option price="200"> extra_meet </option>
    </options>
  </menu>
  <menu name="udon" type="noodle">
    <price>300</price>
    <options>
      <option price="150">large_noodle</option>
    </options>
  </menu>
</lunch>
```

1. Write the answers to the following XPath queries applied on d_1.

属性 [@__ = ' ']

A :: B 所有有B属性的A.

(a) //*[@price > "100"]/text()

(b) options/ancestor::*/price

(c) *[preceeding-sibling::*/attribute::price]

2. Write an XPath query which returns the following:

   (a) Find 'ramen' elements such that **all** of their 'option' descendants have 'price' larger than 100.

   (b) Find elements such that whose name is either 'menu' or 'option' and whose attribute value or text value is 'noodle'.

   (c) Find elements that have a sibling which has an attribute named 'special'.

3. Write a DTD e such that document d_1 is valid against e.

4. Write a DTD f such that document d_1 is also valid against f, and there is another document d_2 such that d_2 is valid against e but not valid against f. Show such DTD f and document d_2.

d_2 去掉 ATTLIST 的数.

d_2 去掉所有属性节点.

Question 4. Explain the following notions.

1. User authentication for web sites.
2. RW-, WR-, and WW-conflicts.
3. Optimistic concurrency control.

(1 *

Information Organization Final Examination (January 25th, 2012)

Notice

- Write your name and student number to all the answer sheets.
- You can bring printed materials (slides, notes, memos, etc). But NO electronic device, such as cell phone, laptop, electronic dictionary, or calculator is allowed.
- You can write answers either in English or Japanese.

Question 1. 25'

1. Suppose that user John executed the following SQL commands.

```
John:  CREATE TABLE lunch(name, price),
       REVOKE ALL PRIVILEGES ON lunch FROM PUBLIC;
       GRANT SELECT, INSERT ON lunch TO Tom WITH GRANT OPTION;
       GRANT SELECT, DELETE ON lunch TO Keiko;
```

Then, the following commands (1)-(4) were executed in this order, by the users indicated at the front of each command line.

```
(1) Tom:    GRANT SELECT ON lunch TO Hanako;
(2) Keiko:  INSERT INTO lunch(name, price) VALUES('bento','500');
(3) John:   REVOKE SELECT ON lunch FROM Tom;
(4) Hanako: SELECT * FROM lunch;
```

For each of (1)-(4), answer whether each command was granted or denied by the database. Also, describe reasons why.

2. What is the name of the access control model in which users can have privileges on objects and give privileges to other users?

Question 2. 30'

Let us consider multi-granularity lock modes consisting of IS, IX, S, and X. Here IS is intension shared lock, IX is intension-exclusive lock, S is shared lock, and X is exclusive lock. Multi-granularity locking is applied on a hierarchy of nodes having containment relationships. Examples of nodes are table, index, and tuples.

1. To lock a tuple in X mode, what kinds of locks are necessary before locking the tuple? Also explain why.
2. To unlock a tuple locked in X mode, what kind of unlock order must be followed? Also explain why.

Suppose that transaction T1 uses the index to read only part of table R. For simplicity, assume that the index height is one. To access R, T1 uses the following locks: T1 gets an IS lock on R, gets an IS lock on an index page, and repeatedly gets an S lock on tuples of R.

IS(R) IS(index) S(tuple) IX(R) X(tuples)

3. Describe an appropriate lock schedule for transaction T2: T2 updates all the tuples of R.
4. Argue whether executions of T1 and T2 can be overlapped.

X→S conflicts IX(R)

5. Describe an appropriate lock schedule for transaction T3: T3 reads a part of R using the index, and updates a few of them.
6. Argue whether executions of T1 and T3 can be overlapped.

SIX(R) SIX(index) S(tuples) X(tuples)

Question 3. Consider the following XML document d_1: 30'

```
<lunch>
  <menu name="ramen" type="noodle">
    <price> 500  </price>
    <options>
      <option price="100"> large_noodle </option>
      <option price="200"> extra_meet </option>
    </options>
  </menu>
  <menu name="udon" type="noodle">
    <price>300</price>
    <options>
      <option price="150">large_noodle</option>
    </options>
  </menu>
</lunch>
```

table R

index

/*[

/lunch/*[

[

1. Write the answers to the following XPath queries applied on d_1.

(a) //*[@price > "100"]/text()

(b) options/ancestor::*/price

(c) *[preceeding-sibling::*/attribute::price]

2. Write an XPath query which returns the following:

   (a) Find 'ramen' elements such that **all** of their 'option' descendants have 'price' larger than 100.

   (b) Find elements such that whose name is either 'menu' or 'option' and whose attribute value or text value is 'noodle'.

   (c) Find elements that have a sibling which has an attribute named 'special'.

3. Write a DTD e such that document d_1 is valid against e.
4. Write a DTD f such that document d_1 is also valid against f, and there is another document d_2 such that d_2 is valid against e but not valid against f. Show such DTD f and document d_2.

Question 4. Explain the following notions. 5'×3 = 15'

1. User authentication for web sites.
2. RW-, WR-, and WW-conflicts.
3. Optimistic concurrency control.

11 *

解　答　用　紙

年　月　日提出

科目/Subject	4 4 1 1 1 5 3 5 - 3	Score
Information Organization	氏名/Name 周游	
担当教員/Lecturer		

Q1.

1. (1) Granted. Because John granted Tom with the "WITH GRANT OPTION" on select on lunch, thus Tom is able to grant select on lunch to others

(2) Denied. John only gave SELECT and DELETE privilege to Keiko, so she is not allowed to insert.

(3) Granted. John is the creator of the table, he can revoke freely.

(4) Denied. Hanako received SELECT privilege from Tom, whose privilege is revoked in (3), this ~~makes~~ removes Hanako's (solely) privilege

2. Discretionary access control

25

Q2.

1. Must have IX or SIX on parent node. Conflicts will happen when other transactions try to read/write that parent node if there is no lock ~~held~~ held on ~~them~~ it.

2. From specific to general (i.e. bottom-up). Similar to the previous question, if we release the locks from up to bottom, problems will occur on ancestor nodes that are not locked.

1. Must have IX or SIX on parent node. Conflicts will happen when other transactions try to read/write that parent node if there is no lock held on it.
2. From specific to general (i.e. bottom-up). Similar to the previous question, if we release the locks from up to bottom, problems will occur on ancestor nodes that are not locked.
3. ~~T2 gets an IX on R then repeatedly gets an X lock on tuples of R and update the tuple~~ Get X on R
4. ~~Cannot overlap~~ Cannot overlap because lock conflicts on R.
5. T3 gets SIX on R and SIX on index, then repeatedly get an S on tuple and occasionally upgrades to X on some tuples.
6. Can overlap. The locks are not ~~conflict~~ conflict on parents.

SIX(R)
SIX(index)
S (tuples)
X(tuples)

Q3

1. (a) Large_noodle
extra_meat
Large_noodle

(b) no result
(not starting from "/")

(c) no result
(menu does not have preceding siblings)

2. (a) ~~/store~~ //* [~~text()~~ @name=ramen][/options/option[@price>100]]

(b)

(c) //* [~~pr~~ sibling::* /attribute::special]

次ページへつづく

前ページのつづき

3. e: <!ELEMENT lunch (menu*) >
<!ELEMENT menu (price, options+)>
<!ELEMENT price EMPTY>
<!ELEMENT options (option+)>
<!ELEMENT option EMPTY>
<!ATTLIST option price CDATA #REQUIRED>
d1 is valid against the previous DTD.

4. f: <!ELEMENT lunch ANY>
<!ATTLIST ~~lunch~~
<!ELEMENT menu ANY>
<!ATTLIST menu name CDATA #REQUIRED>
d1 valid against f.
d2 <lunch>
<menu>
<price>123</price>
<options>
<option ~~large</options~~ price="100">Large</option>
</options>
</menu>
</lunch>
d2 is valid against e but not f.

</options>
</menu>
</lunch>
d_2 is valid against e but not f.

Q4. 1. A website can ask user to use password authentication. This is done after SSL is used to establish a session key so that the transmission of password is secure. But this is still at risk if the user's information is leaked.
Digital signatures can also be used. the user encrypts the request using his private key and encrypt using websites's public key. The website use his private key and user's public key to get the original request. This makes it impossible to forge a fake request.

2. WR - conflicts : reading uncommited date. also called "dirty reads"
RW - conflicts : unrepeatable reads, two reads on the same object at different times return different results, thus called "unrepeatable"
WW - conflicts : overwriting uncommited data, two transactions wrote the data on the same object while the other contains uncommited data.

3. This is based on the assumption that conflicts are rare. Instead of using complex locking policies. we check for conflicts before transaction commits.

15

Information Organization
Exercises Part 2 (security and transaction, XML)

from the textbook by Ramakrishnan and Gehrke + α

S-1 What is the main idea behind discretionary access control? What is the idea behind mandatory access control? What are the relative merits of these two approaches?

S-2 What is a Trojan Horse attack and how can it compromise discretionary access control? Explain how mandatory access control protects against Trojan horse attacks.

S-3 Describe the privileges recognized in SQL. In particular, describe privileges regarding SELECT, INSERT, UPDATE, DELETE, and REFERENCES. For each privilege, indicate who acquires it automatically on a given table.

S-4 What is the difference between symmetric and public-key encryption? Give examples of well-known encryption algorithms of both kinds. What is the main weakness of symmetric encryption and how is this addressed in public key encryption?

S-5 Explain the need for each of the following limits in a statistical database system:

1. A maximum on the number of queries a user can pose.
2. A minimum on the number of tuples involved in answering a query.
3. A maximum on the intersection of two queries (i.e., on the number of tuples that both queries examine).

T-1 Define these terms: atomicity, consistency, isolation, durability, schedule, dirty read, unrepeatable read, serializable schedule, cascading abort.

T-2 Describe Strict 2PL.

T-3 What is the phantom problem? Can it occur in a database where the set of database objects is fixed and only the values of objects can be changed?

T-4 Consider the following actions taken by transaction T1 on database objects X and Y:

$$R(X), W(X), R(Y), W(Y)$$

1. Give an example of another transaction T2 that, if run concurrently to transaction T1 without some form of concurrency control, could interfere with T1.

T4

1) If the transaction T_2 performed W(Y) before T_1 performed R(Y), and then T_2 aborted, the value read by T_1 would be invalid and the abort would be cascaded to T_1

T_1	T_2
R(X)	
W(X)	W(Y)
R(Y)	
W(Y)	
	aborted

2) Strict 2PL would require T_2 to obtain an exclusive lock on Y before writing to it. This lock would have to be held until T_2 committed or aborted; this would block T_1 from reading Y until T_2 was finished, thus there would be no interference.

3) Denotes only 'safe' interleaving of transactions so there transactions are recoverable, avoid cascading aborts.

(2) is very simple and easy to implement. The lock manager only needs to provide a lookup for exclusive locks and an atomic locking mechanism (such as with a semaphore)

T-b. 1: Locking thrashing occurs when the database system reaches to a point where adding another new active transaction actually reduces throughput due to competition for locking

2. Explain how the use of Strict 2PL would prevent interference between the two transactions. Use symbol $s(X)$ for a shared-lock on X and $x(X)$ for an exclusive lock on X.

3. Strict 2PL is used in many database systems. Give two reasons for its popularity.

T-5 Consider a database with objects X and Y and assume that there are two transactions T1 and T2. Transaction T1 reads objects X and Y and then writes object X. Transaction T2 reads objects X and Y and then writes objects X and Y.

1. Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a write-read conflict.

2. Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a read-write conflict.

3. Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a write-write conflict.

4. For each of the three schedules, show that Strict 2PL disallows the schedule.

T-6 We call a transaction that only reads database object a read-only transaction, otherwise the transaction is called a read-write transaction. Give brief answers to the following questions:

1. What is lock thrashing and when does it occur?

2. What happens to the database system throughput if the number of read-write transactions is increased?

3. What happens to the database system throughput if the number of read-only transactions is increased?

4. Describe three ways of tuning your system to increase transaction throughput.

T-7 In a *recoverable* schedule, transactions commit only after all transactions whose changes they read commit. Below is an *unrecoverable* schedule:

$$T1 : R(A), T1 : W(A), T2 : R(A), T2 : W(A), T2 : R(B), T2 : W(B), T2 : Commit, T1 : Abort$$

$T2$ has read a value for A that should should never have been there (aborted transactions' efforts are not supposed to be visible to other transactions.) If $T2$ had not yet committed, we could deal with the situation by *cascading* the abort of $T1$ and also aborting $T2$; this process recursively aborts any transaction that read data written by $T2$, and so on. But $T2$ has already committed, and so we cannot undo its actions.

1.

T_1	T_2
R(X)	
	R(X)
W(X)	
	W(X)

来读被更改

not serializable

2.

T_1	T_2
W(X)	
	R(Y)
R(Y)	
	R(X)

not serializable

3.

T_1	T_2	T_3
R(X)		
	R(Y)	
	R(X)	W(X)
R(Y)		

not serializable

	serializable	conflict (不读未commit / abort)	view	recoverable	avoid	strict
1	(abort T_2) ✓ not decided	×	×	✓	✓	×
2	✓	✓	✓	not decide	×	×
3	✓	✓	✓	not decided	×	not decided
4 ~~T_1~~	(T_2, T_3 abort) not decided	×	not decided	not decided	×	×
5	✓	✓	✓	✓	✓	×
6	×	×	×	✓	✓	×
7	✓	✓	✓	✓ ○	✓ ○	✓ ✓
8	✓	×	×	×	×	×
9	✓	✓	✓	×	×	×
10	✓	✓	✓	✓	✓	✓
11	×	×	×	✓	✓	✓
12	×	×	○ ×	✓	×	×

4.

T_1	T_2	T_3
R(X)		
R(Y)		
W(X)	R(Y)	W(Y)
W(X)	R(Y)	

5.

T_1	T_2
R(X)	
	W(X)
W(X)	
	Abort
Commit	

6.

T_1	T_2
R(X)	
	W(X)
W(X)	
	commit
Commit	

7.

T_1	T_2
W(X)	
	R(X)
W(X)	
	Abort
Commit	

8.

T_1	T_2
W(X)	
	R(X)
W(X)	
	commit
Commit	

9.

T_1	T_2
W(X)	
	R(X)
W(X)	
	commit
Abort	

10.

T_1	T_2	T_3
	R(X)	
		W(X)
		commit
W(Y)		
Commit		
	R(Y)	
	W(Z)	
	commit	

11.

T_1	T_2	T_3
R(X)		
	W(X)	
	commit	
W(X)		
~~Commit~~		
		R(X)
		commit

If transactions read only the changes of committed transactions, not only is the schedule recoverable, but also aborting a transaction can be accomplished without cascading the abort to other transactions. Such a schedule is said to *avoid cascading aborts*.

A schedule is said to be *strict* if a value written by a transaction T is not read or overwritten by other transactions until T either aborts or commits. Strict schedules are recoverable, do not require cascading aborts, and actions of aborted transactions can be undone by restoring the original values of modified objects.

Now, consider the following classes of schedules: *serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts*, and *strict*. For each of the following schedules, state which of the preceeding classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly.

The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1. $T1 : R(X), T2 : R(X), T1 : W(X), T2 : W(X)$
2. $T1 : W(X), T2 : R(Y), T1 : R(Y), T2 : R(X)$
3. $T1 : R(X), T2 : R(Y), T3 : W(X), T2 : R(X), T1 : R(Y)$
4. $T1 : R(X), T1 : R(Y), T1 : W(X), T2 : R(Y), T3 : W(Y), T1 : W(X), T2 : R(Y)$
5. $T1 : R(X), T2 : W(X), T1 : W(X), T2 : Abort, T1 : Commit$
6. $T1 : R(X), T2 : W(X), T1 : W(X), T2 : Commit, T1 : Commit$
7. $T1 : W(X), T2 : R(X), T1 : W(X), T2 : Abort, T1 : Commit$
8. $T1 : W(X), T2 : R(X), T1 : W(X), T2 : Commit, T1 : Commit$
9. $T1 : W(X), T2 : R(X), T1 : W(X), T2 : Commit, T1 : Abort$
10. $T2 : R(X), T3 : W(X), T3 : Commit, T1 : W(Y), T1 : Commit, T2 : R(Y), T2 : W(Z), T2 : Commit$
11. $T1 : R(X), T2 : W(X), T2 : Commit, T1 : W(X), T1 : Commit, T3 : R(X), T3 : Commit$
12. $T1 : R(X), T2 : W(X), T1 : W(X), T3 : R(X), T1 : Commit, T2 : Commit, T3 : Commit$

T-8 Answer each of the following questions briefly. The questions are based on the following relational schema:

Emp(*eid*: integer, *ename*: string, *age*: integer, *salary*: real, *did*: integer)
Dept(*did*: integer, *dname*: string, *floor*: integer)

and on the following update command:

S ⇒ X

replace (salary = 1.1 * EMP.salary) where EMP.ename = 'Santa'

Salary = Emp.Salary − 100.

1. Give an example of a query that would conflict with this command (in a concurrency control sense) if both were run at the same time. Explain what could go wrong, and how locking tuples would solve the problem.

保持 consistency 提高 concurrency

2. Give an example of a query or a command that would conflict with this command, such that the conflict could not be resolved by just locking individual tuples or pages but requires index locking.

T_1	T_2
where	Insert (A)
Insert (B)	

⇒ result 1.1A, B

T_1: where, T_2: Insert (A), Insert B, ✗ 和 1.1A, 1.1B 步 A, B 都不同

Unit 7 P561 3. Explain what index locking is and how it resolves the preceding conflict.

should lock the index page containing the data entries with EMP.ename='Santa'.

课后 Unit 7 **T-9** Consider a database organized in terms of the following hierarchy of objects: The database itself is an object (D), and it contains two files ($F1$ and $F2$), each of which contains 1000 pages (P1...P1000 and P1001...P2000, respectively). Each page contains 100 records, and records are identified as $p : i$, where p is the page identifier and i is the slot of the record on that page.

Multiple-granularity locking is used, with S, X, IS, IX and SIX locks, and database-level, file-level, page-level and record-level locking. For each of the following operations, indicate the sequence of lock requests that must be generated by a transaction that wants to carry out (just) these operations:

D
F1 F2
page/row page 1001~2000
record record

1. Read record P1200:5. IS (D), IS (F_2), IS (P1200), S (P1200:5)
2. Read records P1200:98 through P1205:2. IS (D) IS (F_2), IS (P1200), S on 1201 through 1204, IS on P1025, S on P1200:98/99, S on P1205: 1/2.
3. Read all (records on all) pages in file $F1$. IS (D), S (F_1)
4. Read pages $P500$ through $P520$. IS (D), IS (F_1), S on P500 through P520
5. Read pages $P10$ through $P980$. IS (D), S on F_1.
6. Read all pages in $F1$ and (based on the values read) modify 10 pages. IS and IX on D, SIX on F1
7. Delete record $P1200 : 98$. (This is a blind write.) IX (D), IX (F_2), X (P1200)
8. Delete the first record from each page. (Again, these are blind writes.)
9. Delete all records. IX (D), X (F_1) and X (F_2) IX (D), X (F_1), X (F_2).

X-1 What is the main concepts of HTML? What are the short comings of HTML, and how does XML address them? unit 7. P226. 228

X-2 Explain the following terms and describe what they are used for: HTML, URL, XML, XPath, DTD, cookie, HTTP, HTTPS.

<? xml version='1.0'?>

well-formed ⇒ valid

X-3 Give an example of a DTD. Also give an XML document that is valid against the DTD but not well-formed, and vice versa.

Unit1, PPT, 28页

X-4 The following pages contain XPath language specification and tutorial:
http://www.w3.org/TR/xpath
http://www.w3schools.com/Xpath/

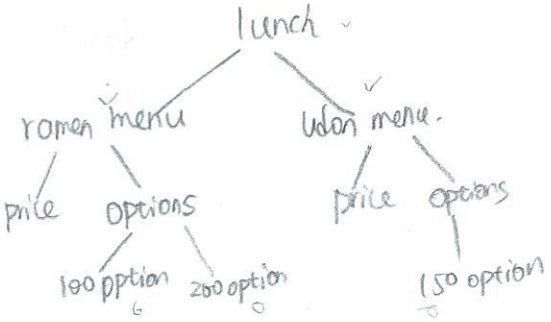
Construct a sample XML document, and write XPath query examples that use the XPath language constructs.

X-5

1. Design an algorithm for testing whether given XML documents are well-formed.
2. Design an algorithm for testing whether the DOM tree of a given well-formed XML document is valid against a given DTD. Try using nondeterministic finite automata (NFA).
3. Design an algorithm answering XPath queries over given XML documents. You may restrict the language constructs of XPath to a subset (for example, just considering a subset of document axis, and excluding predicates, etc).

X6 Consider the following XML document d_1:

```
<lunch>
  <menu name="ramen" type="noodle">
    <price> 500  </price>
    <options>
      <option price="100"> large_noodle </option>
      <option price="200"> extra_meet </option>
    </options>
  </menu>
  <menu name="udon" type="noodle">
    <price>300</price>
    <options>
      <option price="150">large_noodle</option>
    </options>
  </menu>
</lunch>
```



1. Write the answers to the following XPath queries applied on d_1.
   (a) //*[@price > "100"]/text()
   (b) options/ancestor::*/price
   (c) *[preceeding-sibling::*/attribute::price]
2. Write an XPath query which returns the following:

(a) extra_meet, large_noodle
(b) <price>500</price> <price>300</price>
(c) <option price="200">extra_meet</option>

(a) Find 'ramen' elements such that **all** of their 'option' descendants have 'price' larger than 100.

(b) Find elements such that whose name is either 'menu' or 'option' and whose attribute value or text value is 'noodle'.

(c) Find elements that have a sibling which has an attribute named 'special'.

3. Write a DTD e such that document d_1 is valid against e.

4. Write a DTD f such that document d_1 is also valid against f, and there is another document d_2 such that d_2 is valid against e but not valid against f. Show such DTD f and document d_2.

在DTD里面加(REQUIRED)

2. (a) *[@name()="ramen"][descendant::option[@price>100]]

(b) *[name()=menu/option][[attribute::*="noodle"]|[text()="noodle"]]

(c) *[sibling::*/attribute::special]

3. 去掉 REQUIRED.

4. d_2 无元素.