Бази от данни

Модел на полуструктурираните данни

Обзор на разгледаните модели

- E/R моделът борави с две основни понятия – множества от същности и връзки. Моделът има богата изразителна нотация и е предпочитан ако трябва да се прави дизайн на базата от данни.
- Релационният модел борави само с едно единствено понятие – релация. Той е предпочитан за реализиране на базата от данни, защото предоставя език за заявки, с които може да се манипулират данните. Езикът, който се използва е SQL
- В обектно-ориентирания модел има две основни понятия – класове и връзки между класовете, които се представят чрез следните свойства: атрибути, връзки и методи. Моделът се характеризира с мощна система за типизация, уникален идентификатор за всеки обект и възможност за полиморфизъм. Езикът, който се използва е ODL и OQL.
- Обектно-релационния модел съчетава предимствата на релационния модел – възможността за реализация на базата от данни и SQL, като допълва модела с възможността да се дефинират съставни типове и вгнездени релации. Езикът, който се използва е SQL (разширен).

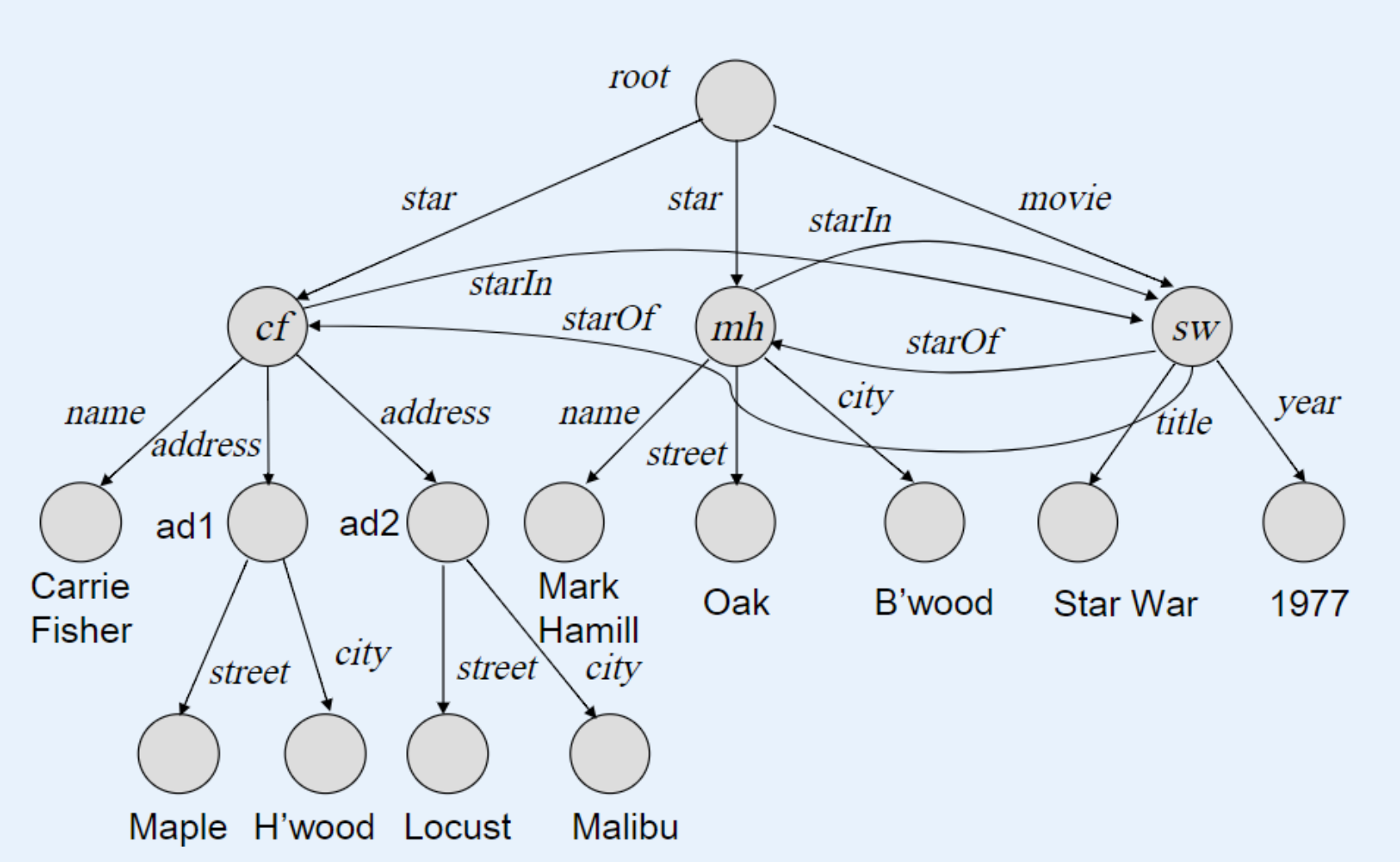
Въведение в модела

- Моделът на полуструктурираните данни има специална роля в СУБД:
  - Този модел е подходящ за интеграция на бази от данни, т.е. за описание на подобни данни, които се съдържат в две или повече бази от данни с различни схеми
  - Моделът се използва като модел за документация с нотации от типа на XML, които се използват при споделяне на информация чрез мрежата
- Моделът притежава по-голяма гъвкавост от досега разгледаните модели. Моделът на полуструктурираните данни изолзва граф за представяне на данните. Езикът, който се използва за представяне на графа е XML.

Модел на полуструктурираните данни

- При модела на полуструктурираните данни, самите данни носят информация за схемата на базата от данни. Това позволява схемата на базата от данни да варира във времето.
- Базата от полуструктурирани данни е набор от възли. Всеки възел е вътрешен възел или листо. Като всяко листо се асоциира с данни от атомарен тип.
- От всеки вътрешен възел излизат една или повече дъги
- Всяка дъга има етикет, който описва по какъв начин се свързват възлите по между си.
- Един от вътрешните възли се нарича корен и в него не влизат дъги.
- Всеки възел трябва да е достижим от корена. Не е задължително обаче, графът на базата от данни да е дърво.

Модел на полуструктурираните данни



Модел на полуструктурираните данни

- Коренът root е входна точка в базата от данни. Основните обекти, в случая звездите и филмите са представени чрез преките наследници на корена.
- Листата имат етикети, които описват данните.
- Три от вътрешните възли имат етикети – cf, sw, mh. Те представят съответно звездата Carrie Fisher, филмът Star Wars и звездата Mark Hamill. Етикетите на вътрешните възли не са част от модела, но придават по-голяма яснота в примера
- Етикетите на дъгите имат две роли. Нека имаме дъга, която излиза от възел N и влиза във възел M и има етикет L:
  - Ако възела N представлява обект, а възелът M представлява атрибут на обекта, то етикета L е името на този атрибут
  - Ако възлите N и M представят обекти, то етикета L е името на връзката от N към M

Интеграция на информацията чрез модела

- За разлика от другите модели, данните в полуструктурирания модел се само-описват, т.е. схемата на базата от данни е прикачена към самите данни.
- Във всеки възел (с изключение на корена) влизат дъги, чиито етикети описват връзките между отделните възли. Ако възела е листо (т.е. описание на атрибут) етикета на дъгата описва името на атрибута.
- Във всички други модели данните имат фиксирана схема, която е отделена от самите данни. Като от схемата се разбира, какви са връзките между отделните данни.
- Както отбелязахме, в полуструктурирания модел това не е така.
- Когато имаме голяма база от данни е задължително да използваме фиксирана схема (т.е. представените до момента модели на данни), но когато схемата на базата от данни е малка, тогава е удачно да се приложи модела на полуструктурираните данни (т.е. само-описващи се данни).

Интеграция на информацията чрез модела

- Едно от приложенията на гъвкавия полуструктуриран модел на данни е използването му при интеграция на информацията.
- Често възниква необходимост две или повече бази от данни да бъдат достъпни и разглеждани все едно са една база от данни. Например сливане на информация за служителите от различни клонове на една и съща компания.
- Ако схемите са еднакви, тогава лесно може да бъдат обединени данни, но това рядко се случва да е така. Възможно е да има разлики в типовете на различните атрибути, дори и имената на атрибутите в двете схеми да са едни и същи.
- Възможно е дори да се случи в единия клон на компанията да е използван релационен модел за представянето на данните, а в други клон обектно-ориентиран модел.
- Също така трябва да се вземе предвид, че такива бази от данни са използвани с години от приложения, което прави промяната на схемите на базата от данни практически невъзможна (при промяна това ще доведе до не работещи приложения). Този проблем е известен, като проблема на наследените бази от данни.

Интеграция на информацията чрез модела

- Едно възможно решение на проблема е чрез използване на модела на полуструктурираните данни.
- Мястото на този модел е в интерфейса, чрез който потребителя осъществява достъпа до наследените бази от данни.
- Самият интерфейс може въобще да не включва полуструктурирани данни, в този случай потребителят извършва заявки към интерфейса, който преобразува тези заявки към заявки за съответните наследени бази от данни.

XML и неговият модел на данни

- XML е нотация за маркиране на документи чрез етикети, подобна на HTML.
- За разлика от HTML, маркировката в XML се използва за описание на смисъла на информацията в документа
- Чрез XML може да бъде представен графа на базата от полуструктурирани данни в линейна форма.

Маркери в XML

- Маркерите в XML са: `<текст> </текст>` като винаги се използват по двойки отварящ маркер със затварящ маркер, в който се записва същия текст

- Маркировката винаги трябва да е вгнездена, например:

  `<tag1> .. <tag2>… </tag2> .. </tag1>`

- Но не може да бъде така: `<tag1> .. <tag2>… </tag1> .. </tag2>`

- XML документите могат да бъдат използвани в два режима:
  - **Добре структуриран XML** – при него могат да се използват произволни маркери. Този режим е много близък до полуструктурираните данни – структурата на XML документа не е предварително фиксирана
  - **Валиден XML** – при него се използва DTD (document type definition), където е описано какви маркери могат да се използват и как те могат да се вгнездят. Този режим е нещо средно между полуструктурираните данни и данните с фиксирана схема. Схемата в DTD е доста по-гъвкава от схемите в релационния и обектно-ориентирания модел.

Добре структуриран XML документ

- Минималното изискване за един добре структуриран XML-документ е наличие на декларация в началото на документа, която показва, че той е XML, и наличие на коренен маркер, който обгражда цялата останала част на документа.

- Общата структура на добре структуриран XML документ е:

```
<? Xml version = “1.0” encoding = “utf-8” standalone = “yes” ?>
<StarMovieData>
</StarMovieData>
```

Добре структуриран XML документ - Пример

```
<? Xml version = “1.0” encoding = “utf-8” standalone = “yes” ?>
<StarMovieData>
    <Star>
        <Name>Carrie Fishes</Name>
        <Address>
            <Street>123 Maple St.</Street><City>Hollywood</City>
        </Address>
        <Address>
            <Street>5 Locust Ln.</Street><City>Malibu</City>
        <Address>
    </Star>
    <Star>
        <Name>Mark Hamill</Name><Street>456 Oak Rd.</Street>
        <City>Brentwood</City>
    </Star>
    <Movie>
        <Title>Star Wars</title><Year>1977</Year>
    </Movie>
</StarMovieData>
```

Добре структуриран XML документ

- В XML документа по-горе не са представени връзките между звездите и филмите. Една възможност е да съхраняваме информация за всички филми, в които играе дадена звезда към тази част от XML документа в която се описва информацията за звездата.

- Този подход обаче води до излишество, тъй като цялата информация за всеки филм ще се повтаря за всеки актьор, който участва в него.

Валиден XML документ

- За да може XML документите да се обработват автоматично, нормално е те да се подчиняват на някаква схема – т.е. какви маркери могат да се използват, как трябва да са вгнездени те.
- Описанието на схемата на един XML документ е последователност от граматични правила – DTD
- Най-общо едно DTD изглежда по следния начин:

```
<!DOCTYPE   Stars [
   <!ELEMENT STARS (STAR*)>

]>
```

Валиден XML документ

- Зададеното име на коренния маркер, трябва да се използва във всеки XML документ, който се подчинява на това DTD
- Всеки елемент се описва с име, което представлява името на маркера, който ще бъде използван в XML документа
- Всеки елемент може да има компоненти, които се изброяват със запетаи след името на маркера – това са имена на маркери, които може да присъстват в XML документа
- Ако след името на елемент се използва списък от компоненти (#PCDATA), това означава, че в частта на XML документа съответна на този елемент няма маркери, а само текст.
- Имената на маркерите в XML са case-insensitive – не се прави разлика между малки и големи букви.

DTD схема пример

```
<!DOCTYPE   Stars [
   <!ELEMENT STARS (STAR*)>
   <!ELEMENT STAR(NAME,ADDRESS+,MOVIES)>
   <!ELEMENT NAME (#PCDATA)>
   <!ELEMENT ADDESS (STREET, CITY)>
   <!ELEMENT STREET (#PCDATA)>
   <!ELEMENT CITY (#PCDATA)>
   <!ELEMENT MOVIES (MOVIE*)>
   <!ELEMENT MOVIE (TITLE, YEAR)>
   <!ELEMENT TITLE (#PCDATA)>
   <!ELEMENT YEAR (#PCDATA)>
]>
```

DTD схема

- С всеки компонент може да се използва един от следните оператори, който се записва след името на компонента:
  - Оператор * - указва, че съответният компонент може да присъства 0 или повече пъти в XML документа
  - Оператор + - указва, че съответният компонент трябва да присъства 1 или повече пъти в XML документа
  - Оператор ? - указва, че съответният компонент трябва да не присъства или да присъства един път в XML документа
  - В списъка от компоненти на един елемент може да се използва и символът |. Неговата семантика е изключващо или т.е. в XML документа, трябва да присъства или компонент1 или компонент 2, но не и едновременно и двата компонента.
  - Например списъка от компоненти (#PCDATA | (STREET, CITY)) за елемента ADDRESS означава, че XML документа ще съдържа или текст или два маркера – street и city.

Валиден XML документ - пример

```
<STARS>
   <STAR><NAME> Carrie Fisher </NAME>
         <ADDRESS<STREET>123 Maple St.</STREET>
                 <CITY>Hollywood</CITY></ADDRESS>
         <ADDRESS><STREET>5 Locust Ln.</STREET>
                  <CITY>Malibu</CITY></ADDRESS>
         <MOVIES><MOVIE><TITLE>Star Wars</TITLE>
                       <YEAR>1977</YEAR></MOVIE>
                 <MOVIE><TITLE>Empire Strikes Back</TITLE>
                      <YEAR>1980</YEAR></MOVIE>
                 <MOVIE><TITLE>Return of the Jedi</TITLE>
                      <YEAR>1983</YEAR></MOVIE>
        </MOVIES>
   </STAR>
</STARS>
```

Валиден XML документ

- Ако един XML документ е валиден, т.е. свързан с DTD, това се указва по един от двата начина:
  - Включваме пълното описание на DTD в началото на XML документа
  - Указваме име на файл, който съдържа съответното DTD. В този случай приложението, което използва XML документа трябва да има достъп до файловата система, в която се съхранява DTD. Този вариант е удобен когато едно DTD се използва в много файлове.
  - И в двата случая параметъра STANDALONE в първия ред на XML документа трябва да има стойност “no”

- В първия случай пълното описание на DTD се осъществява непосредствено след първия ред на валидния XML документ.
- Във втория случай вторият ред на валидния XML документ трябва да има следният вид, например:

```
<?XML VERSION = “1.0” STANDALONE = “no”?>
<!DOCTYPE Stars SYSTEM “star.dtd”>
```

- Тук пълното описание на DTD е съдържание на съответния файл – т.е. Star.dtd е файлът, който съдържа описанието на DTD за звездите.

Атрибути в XML

- XML документите и полуструктурираните данни са доста силно свързани. Всеки XML документ може да се представи чрез граф на полуструктурирани данни.
- За целта за всяка двойка маркери <T> и </T> създаваме по един възел.
- Ако една двойка маркери - <S> и </S> е директно вгнездена в двойката маркери <T> и </T>, то от възела съответен на <T> и </T> прекарваме дъга с етикет S към възела съответен на <S> и </S>
- Ще въведем още един елемент от XML – атрибут на маркер.

Атрибути в XML

- Всеки отварящ маркер (таг), може да има атрибут, за които се задават стойности след името на маркера.
- Атрибутите могат да имат различни предназначения.
- Например да представят данни или да представят връзки.
- В случаите, когато се представят връзки може да има два възможни типове атрибути – ID и IDREF

Атрибути в XML - ID

- Ако един атрибут на елемент с име E има тип ID, това означава, че той ще приема уникални стойности в XML-документа, за всяка част от документа оградена с маркери <E> и </E>.

- В термините на полуструктурираните данни това означава, че този атрибут осигурява уникална идентификация на съответния възел.

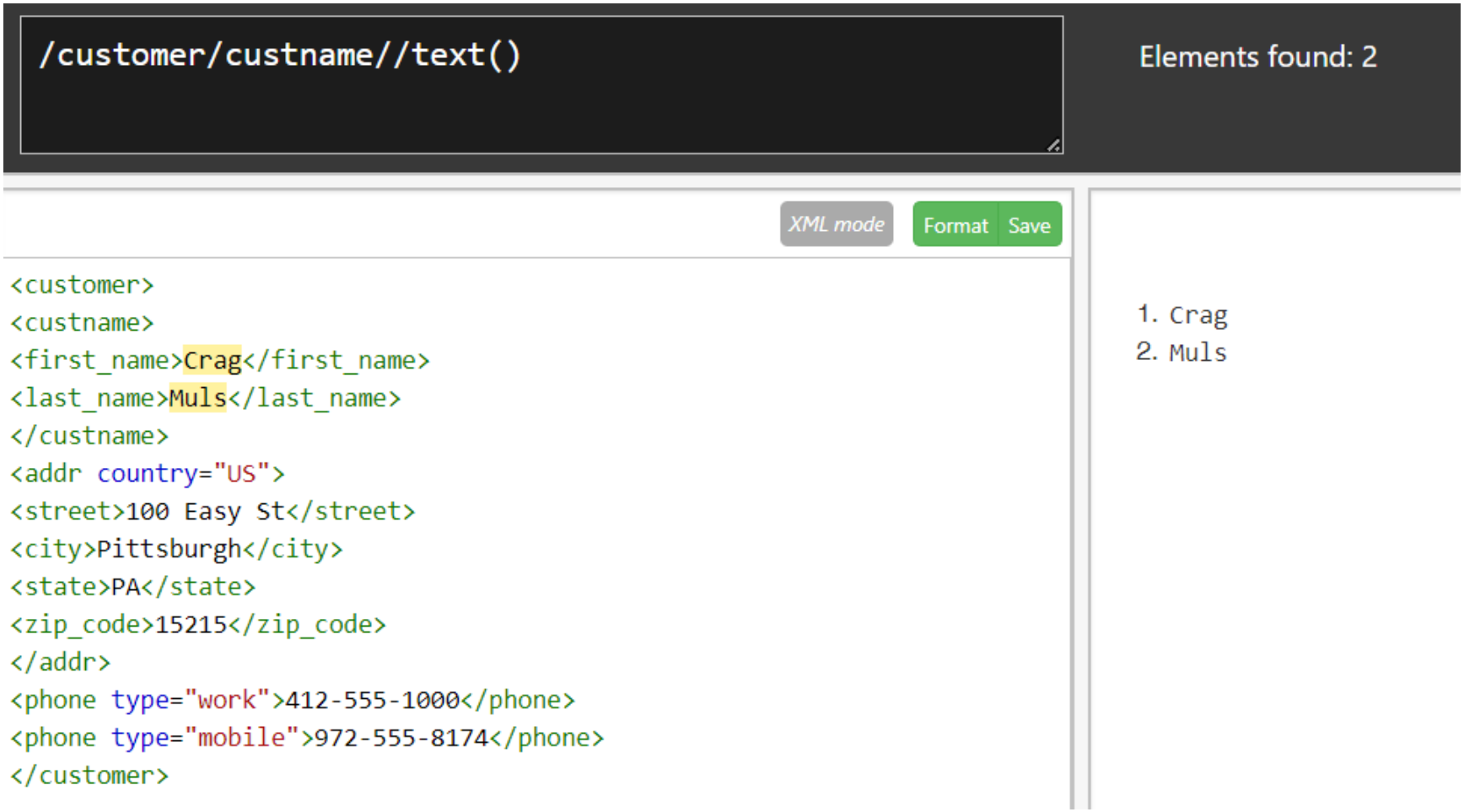
Атрибути в XML - IDREF

- Ако един атрибут на елемент с име E има тип IDREF, това означава, че той ще приема стойност на атрибути ID, асоциирани с други елементи в XML документа.
- В термините на полуструктурираните данни това може да се интерпретира по следния начин: от съответния възел излизат дъги към възлите, чиито ID присъстват в стойността на атрибута от тип IDREF на този възел, при това името на дъгата съвпада с името на атрибута от тип IDREF.
- Следващият пример илюстрира синтаксисът за деклариране на атрибутите в XML документа за представяне на връзките в полуструктурираните данни.

Атрибути в XML - Пример

```
<!DOCTYPE Stars-Movies[
    <!ELEMENT STARS-MOVIES(STAR*,MOVIE*>
    <!ELEMENT STAR(NAME,ADDRESS+)>
        <!ATTLIST STAR
            starId ID
            starredIn IDREFS>
    <!ELEMENT NAME (#PCDATA)>
    <!ELEMENT ADDRESS (STREET, CITY)>
    <!ELEMENT STREET(#PCDATA)>
    <!ELEMENT CITY (#PCDATA)>
    <!ELEMENT MOVIE (TITLE,YEAR)>
        <!ATTLIST MOVIE
            movieId ID
            starsOf IDREFS>
    <!ELEMENT TITLE (#PCDATA)>
    <!ELEMENT YEAR (#PCDATA)>
]>
```

XPath


/customer/custname//text()
Elements found: 2
XML mode
Format
Save
<customer>
<custname>
<first_name>Crag</first_name>
<last_name>Muls</last_name>
</custname>
<addr country="US">
<street>100 Easy St</street>
<city>Pittsburgh</city>
<state>PA</state>
<zip_code>15215</zip_code>
</addr>
<phone type="work">412-555-1000</phone>
<phone type="mobile">972-555-8174</phone>
</customer>
1. Crag
2. Muls

XQuery - BaseX