



Advanced topics in Computer Science

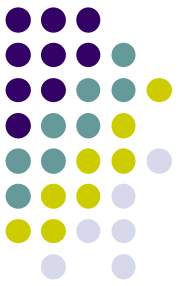
Jiaheng Lu

Department of Computer Science

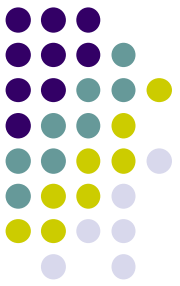
Renmin University of China

www.jiahenglu.net

Course



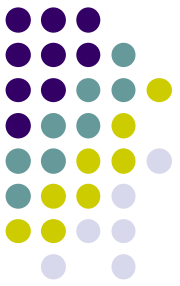
- Introduction to XML
 - concentrate on XML's uses for the web
 - many other uses!
- Three parts
 - XML standard
 - XML validation
 - XML transformations



XML Schema

- A more powerful way to describe the structure of XML documents.
- XML-Schema declarations are themselves XML documents.
 - They describe “elements” and the things doing the describing are also “elements.”

Structure of an XML-Schema Document



```
<? xml version = ... ?>
```

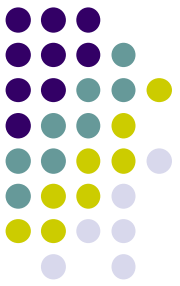
```
<xs:schema xmlns:xs =  
  "http://www.w3.org/2001/XMLSchema" >
```

```
</xs:schema>
```

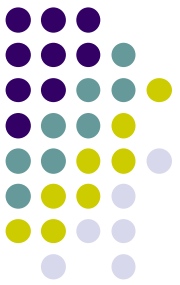
So uses of "xs" within the schema element refer to tags from this namespace.

Defines "xs" to be the *namespace* described in the URL shown. Any string in place of "xs" is OK.

The **xs:element** Element



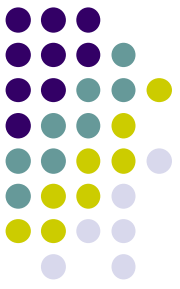
- Has attributes:
 1. **name** = the tag-name of the element being defined.
 2. **type** = the type of the element.
 - ◆ Could be an XML-Schema type, e.g., xs:string.
 - ◆ Or the name of a type defined in the document itself.



Example: xs:element

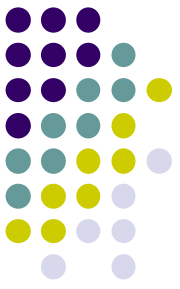
```
<xs:element name = "NAME"  
            type = "xs:string" />
```

- Describes elements such as
`<NAME>Joe's Bar</NAME>`



Complex Types

- To describe elements that consist of subelements, we use `xs:complexType`.
 - Attribute `name` gives a name to the type.
- Typical subelement of a complex type is `xs:sequence`, which itself has a sequence of `xs:element` subelements.
 - Use `minOccurs` and `maxOccurs` attributes to control the number of occurrences of an `xs:element`.

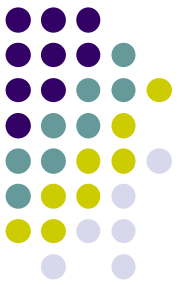


Example: a Type for Beers

```
<xs:complexType name = "beerType" >
  <xs:sequence>
    <xs:element name = "NAME"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "PRICE"
      type = "xs:float"
      minOccurs = "0" maxOccurs = "1" />
  </xs:sequence>
</xs:complexType>
```

Exactly one occurrence

Like ? in a DTD



An Element of Type beerType

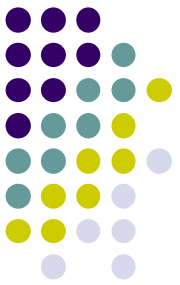
`<xxx>`

`<NAME>Bud</NAME>`

`<PRICE>2.50</PRICE>`

`</xxx>`

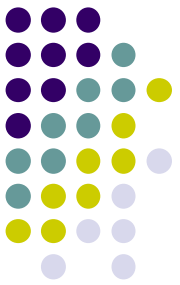
We don't know the name of the element of this type.



Example: a Type for Bars

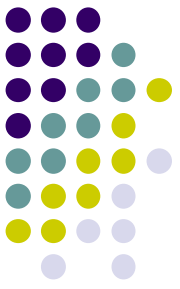
```
<xs:complexType name = "barType">
  <xs:sequence>
    <xs:element name = "NAME"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "BEER"
      type = "beerType"
      minOccurs = "0" maxOccurs =
        "unbounded" />
  </xs:sequence>
</xs:complexType>
```

Like * in a DTD



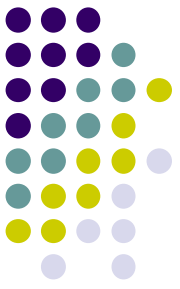
xs:attribute

- **xs:attribute** elements can be used within a complex type to indicate attributes of elements of that type.
- attributes of **xs:attribute**:
 - **name** and **type** as for **xs.element**.
 - **use** = "required" or "optional".



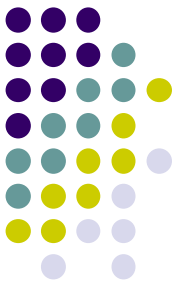
Example: xs:attribute

```
<xs:complexType name = "beerType" >
  <xs:attribute name = "name"
    type = "xs:string"
    use = "required" />
  <xs:attribute name = "price"
    type = "xs:float"
    use = "optional" />
</xs:complexType>
```



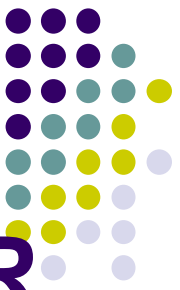
Restricted Simple Types

- `xs:simpleType` can describe enumerations and range-restricted base types.
- `name` is an attribute
- `xs:restriction` is a subelement.



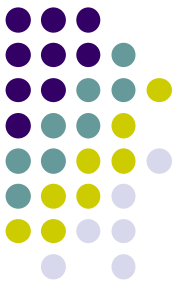
Restrictions

- Attribute **base** gives the simple type to be restricted, e.g., `xs:integer`.
- `xs:{min, max}{Inclusive, Exclusive}` are four attributes that can give a lower or upper bound on a numerical range.
- `xs:enumeration` is a subelement with attribute **value** that allows enumerated types.



Example: **license** Attribute for BAR

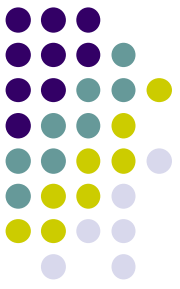
```
<xs:simpleType name = "license">
  <xs:restriction base = "xs:string">
    <xs:enumeration value = "Full" />
    <xs:enumeration value = "Beer only" />
    <xs:enumeration value = "Sushi" />
  </xs:restriction>
</xs:simpleType>
```



Example: Prices in Range [1,5)

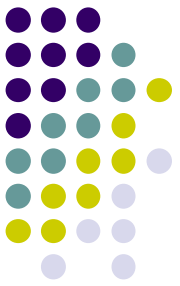
```
<xs:simpleType name = "price">  
  <xs:restriction  
    base = "xs:float"  
    minInclusive = "1.00"  
    maxExclusive = "5.00" />  
</xs:simpleType>
```

Keys in XML Schema



- An **xs:element** can have an **xs:key** subelement.
- **Meaning**: within this element, all subelements reached by a certain **selector** path will have unique values for a certain combination of **fields**.
- **Example**: within one BAR element, the **name** attribute of a BEER element is unique.

Example: Key



And @ indicates an attribute rather than a tag.

```
<xs:element name = "BAR" ... >  
  . . .
```

```
<xs:key name = "barKey">
```

```
  <xs:selector xpath = "BEER" />
```

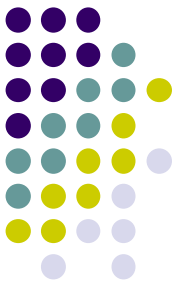
```
  <xs:field xpath = "@name" />
```

```
</xs:key>
```

```
  . . .
```

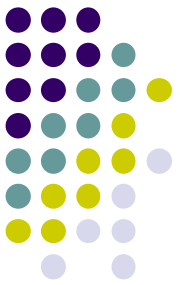
```
</xs:element>
```

XPath is a query language for XML. All we need to know here is that a path is a sequence of tags separated by /.



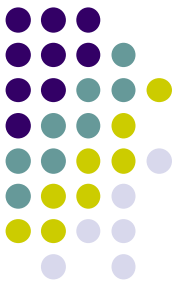
Foreign Keys

- An `xs:keyref` subelement within an `xs:element` says that within this element, certain values (defined by selector and field(s), as for keys) must appear as values of a certain key.



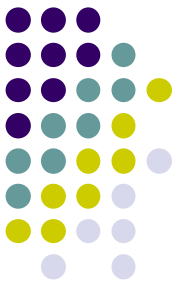
Example: Foreign Key

- Suppose that we have declared that subelement `NAME` of `BAR` is a key for `BARS`.
 - The name of the key is `barKey`.
- We wish to declare `DRINKER` elements that have `FREQ` subelements. An attribute `bar` of `FREQ` is a foreign key, referring to the `NAME` of a `BAR`.




Example: Foreign Key in XML Schema

```
<xs:element name = "DRINKERS"  
    . . .  
    <xs:keyref name = "barRef"  
        refers = "barKey"  
        <xs:selector xpath =  
            "DRINKER/FREQ" />  
        <xs:field xpath = "@bar" />  
    </xs:keyref>  
</xs:element>
```



Cloud computing



 usc interactive media

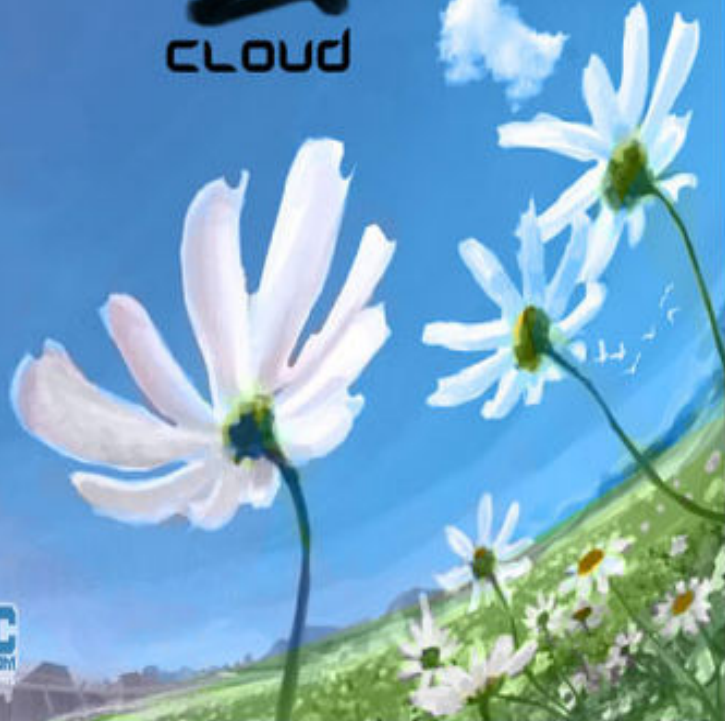
Have you ever wanted to fly among the clouds? To see the sky only where birds soar? Welcome to a dream of what could be. Welcome to Cloud.

High in the sky fly free; gather and create your own clouds, and use good to fight the darkness. The world of Cloud awaits you.

Created with the Electronic Arts Game Innovation Grant from the Division of Interactive Media at the University of Southern California School of Cinema and Television.

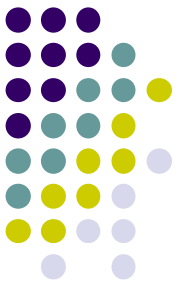
雲
CLOUD

雲
CLOUD



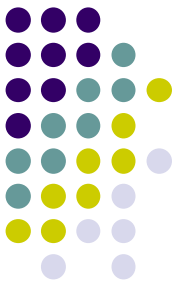
POWERED BY
BLASTHO

PC
CD-ROM
SYSTEM



Cloud Computing Definition

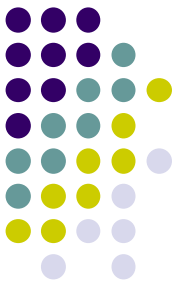
- **Cloud computing**
Distributed computing, Parallel computing, Grid computing a commercial realization.
- **Clouds**
Demand resources or services over Internet
scale and reliability of a data center.
- **Cloud computing**
dynamically scalable and virtualized resources over the Internet. Easy for users



Cloud Computing Definition

- ***Cloud computing** is on-demand access to virtualized IT resources that are housed outside of your own data center, shared by others, simple to use, paid for via subscription, and accessed over the Web.*

Character



- **Virtual.**

software, databases, Web servers, operating systems, storage and networking as virtual servers.

- **On demand.**

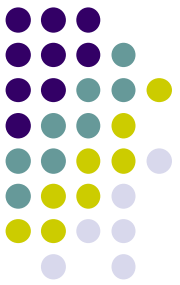
add and subtract processors, memory, network bandwidth, storage, and 32-bit or 64-bit architectures.

- **Subscription style.**

month-to-month deals

payable by credit card

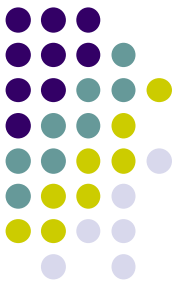
eg. Amazon charges in intervals of 10 cents per hour for EC2.



Character

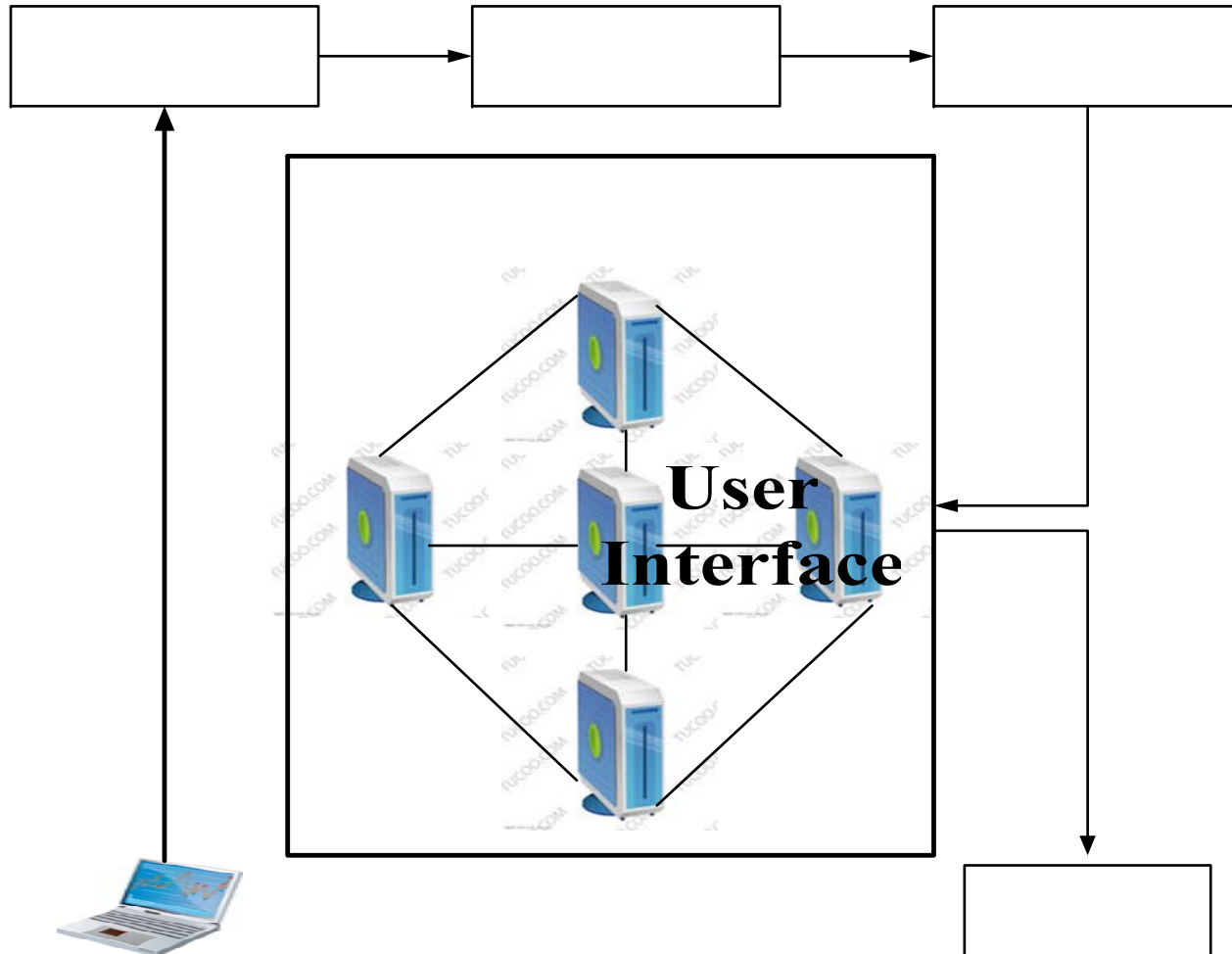
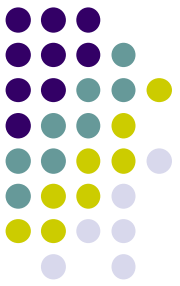
- Shared.
economies of scale
multitenant architecture
workloads balance
- Simple.
configure resources in a few minutes
using an interface
- Web based.
access over browser.
- Open, Standard.....

Types of Cloud Service



- Software as a Service (SaaS)
eg. Zoho office, Salesforce.com
- Platform as a Service
eg. Google App(Python Programming)
- Web Service
eg. Facebook, xiaonei
- Utility Computing
eg. Amazon.com, Sun, IBM
merged memory, I/O device, storage,
computing
- On-Demand Computing

The Architecture behind Cloud Computing System



Sy
Mana