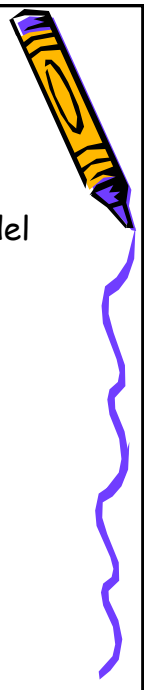
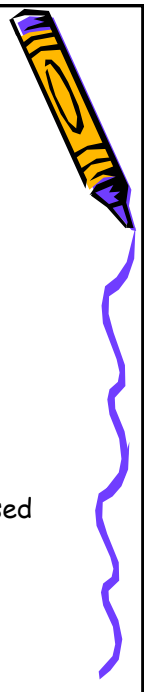
Ch 22 Modeling Persistent Data



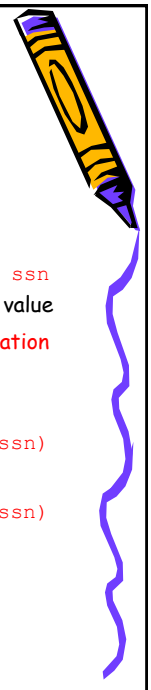
- Need way to organize data elements into a model
  - What are data elements?
    - Distinct pieces of information that describe a "thing"
      - Aka: a field, column, attribute, or variable
    - Collection of information that describe a "thing"
      - Aka: a record, text line, node/segment, or row
- Logical Data Modeling
  - Entity Relationship Diagram (ERD)
  - Fully-attributed Logical Data Model (LDM)
- Physical Data Modeling
  - PDM

Ch 22 Modeling Persistent Data (cont'd)



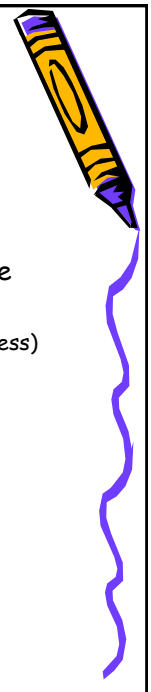
- Logical data modeling
  - Nouns become entities and attributes
  - Verbs/adverbs become relationships
  - Normalization
    - A process of steps to ensure your logical models
      - Reduce (or eliminate) redundancy
      - Accurately reflect the underlying mathematical principles
- Physical data modeling
  - Depends entirely on physical storage format to be used

Ch 22 Modeling Persistent Data (Normalization Notation)



- Notation
  - Functional dependency — `id → name, ssn`
    - e.g., name and ssn values are functionally dependent on the id value
  - Primary key — typically underlined in notation
    - Uniquely identify each entity instance
    - Functional dependencies help identify candidate keys
  - Relation — `student = (id, name, ssn)`
    - A textual way to describe an entity and its attributes
  - Tuple — `(id, name, ssn)`
    - A list of attributes

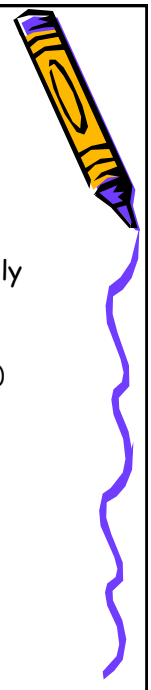
Ch 22 Modeling Persistent Data (Normalization: 1NF)



- First normal form (1NF)
  - Each attribute value is an atomic (or indivisible) value
  - Not in 1NF:

    Employee = (employeeID, employeeName, dependentName, address)
  - Why?
    - employeeName is not atomic (i.e., first middle last)
    - Address is not atomic (i.e., street, city, state zip)

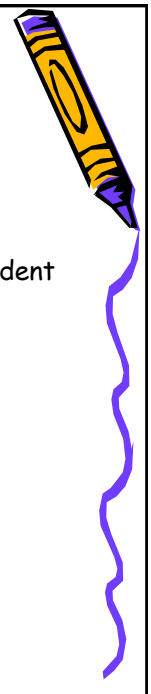
Ch 22 Modeling Persistent Data (Normalization: 2NF)



- Second normal form (2NF)
  - In 1NF and each non-key attribute is fully functionally dependent on the primary key (PK)
  - Not in 2NF:

    EmplProj = (ssn, projID, hours, emplID, projName, projLocation)
  - Why?
    - It is in 1NF, but
    - ssn → emplID
    - projID → projName, projLocation

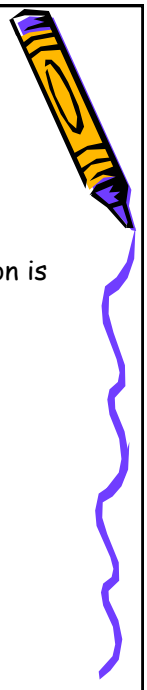
Ch 22 Modeling Persistent Data (Normalization: 3NF)



- Third normal form (3NF)
  - In 2NF and no non-PK attribute is transitively dependent on the PK
  - Not in 3NF:

    EmplDept = (ssn, birthDate, deptID, deptName, deptMgrSSN)
  - Why?
    - Is in 2NF, but
    - ssn → birthDate, deptID
    - deptID → deptName, deptMgrSSN

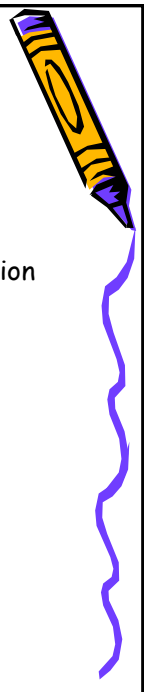
Ch 22 Modeling Persistent Data (Normalization: BCNF)



- Boyce-Codd normal form (BCNF)
  - Every non-trivial functional dependency in the relation is a dependency on a superkey
  - Not in BCNF:
    - For this, assume an employee can be on many projects

    EmplProj = (ssn, projID, projName, projHours)
  - Why?
    - projID → projName
    - But projID is not a superkey, need both ssn and projID

Ch 22 Modeling Persistent Data (Normalization: 4NF)



- Fourth normal form (4NF)
  - Every non-trivial multivalued dependency in the relation is a dependency on a superkey
  - Not in 4NF:

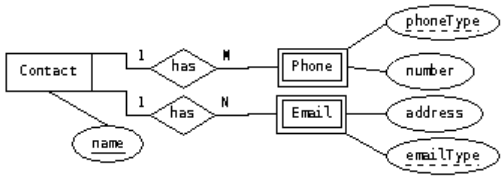
    LoanCust = (loanNumber, custID, custStreet, custCity)
  - Why?
    - loanNumber, custID → custStreet, custCity
    - custID → custStreet, custCity
    - i.e., custStreet, custCity values would be repeated for each customer loan

Ch 22 Modeling Persistent Data (In-Class Discussion)

- Small groups
  - Using your domain, develop an ERD
  - Review these
  - Using your domain, develop a LDM
  - Review these

Ch 23 XML Persistent Data Storage (Address Book Case Study)

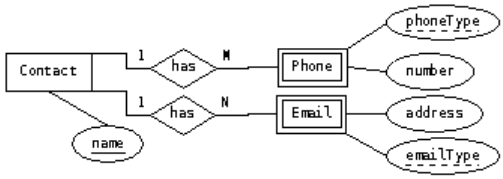
- Need to translate LDM into PDM
  - Contact = (<u>name</u>)
  - Phone = (<u>name</u>, <u>phoneType</u>, phoneNumber)
    - name, phoneType --> phoneNumber
  - Email = (<u>name</u>, <u>emailType</u>, emailAddress)
    - name, emailType --> emailAddress Functional dependencies for ABA



Ch 23 XML Persistent Data Storage (What is XML?)

- What is XML?
  - eXtensible Markup Language
  - A tag-based language used to describe data
  - What's a tag?
    - A programmer-defined data element embedded inside angled brackets
      - e.g., <student>
  - How are tags structured?
    - Each start tag has a matching end tag
      - e.g., <student> … </student> OR <student />
    - Data and other tags are between start and end tags
      - e.g., <student> <id>12345</id> … </student>
      - i.e., XML represents data using a tree (hierarchy) structure

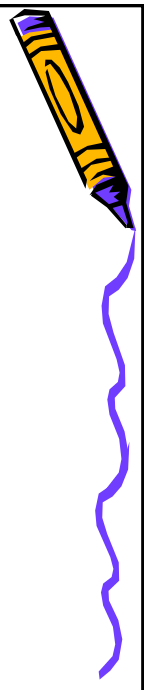
Ch 23 XML Persistent Data Storage (Translate LDM to XML)




```
<AddressBook>
  <Contact>
    <name></name>
    <Email>
      <emailType></emailType>
      <address></address>
    </Email>
    <Phone>
      <phoneType></phoneType>
      <number></number>
    </Phone>
  </Contact>
</AddressBook>
```

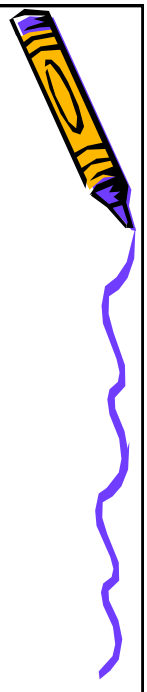
See:
ABA-template-A.xml
ABA-display-A1.xml
ABA-display-A2.xml

Ch 23 XML Persistent Data Storage (Choice of Primary Keys)



- LDM
  - Contact = (<u>name</u>)
  - Phone = (<u>name</u>, <u>phoneType</u>, phoneNumber)
    - name, phoneType --> phoneNumber
  - Email = (<u>name</u>, <u>emailType</u>, emailAddress)
    - name, emailType --> emailAddress
- Generally, bad idea to have application data be part of a primary key
  - Use a primary key value that is not tied to the data
  - Common approach:
    - Add id attribute (integer data type) as primary key
    - Each instance of data has a unique integer value/id

Ch 23 XML Persistent Data Storage (Choice of Primary Keys, cont'd)



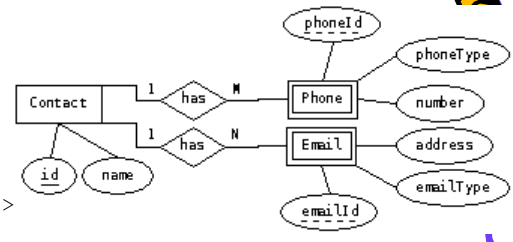
- Add id field to Contact entity
- Resulting LDM
  - Contact = (<u>id</u>, name)
    - id --> name
  - Phone = (<u>id</u>, <u>phoneType</u>, phoneNumber)
    - id, phoneType --> phoneNumber
  - Email = (<u>id</u>, <u>emailType</u>, emailAddress)
    - id, emailType --> emailAddress

Ch 23 XML Persistent Data Storage (Choice of Primary Keys, cont'd)

- Add id field to both Phone and Email entities
- Resulting LDM
  - Contact = (id, name)
    - id --> name
  - Phone = (id, phoneId, phoneType, phoneNumber)
    - id, phoneId --> phoneType, phoneNumber
  - Email = (id, emailId, emailType, emailAddress)
    - id, emailId --> emailType, emailAddress

Ch 23 XML Persistent Data Storage (Choice of Primary Keys, cont'd)

```
<AddressBook>
  <Contact>
    <id></id>
    <name></name>
    <Email>
      <emailId></emailId>
      <emailType></emailType>
      <address></address>
    </Email>
    <Phone>
      <phoneId></phoneId>
      <phoneType></phoneType>
      <number></number>
    </Phone>
  </Contact>
</AddressBook>
```



See:
ABA-template-B.xml
ABA-display-B1.xml
ABA-display-B2.xml
ABA-display-B3.xm;

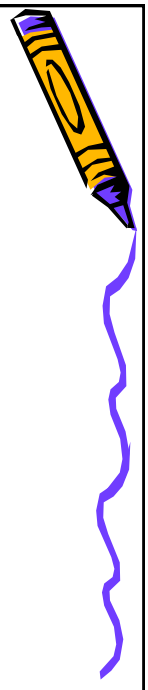
Ch 23 XML Persistent Data Storage (DOM)

- Text-based XML file
  - Parsed into a Document Object Model (DOM)
    - Use API from an XML language library
  - What's a DOM?
    - A tree of nodes that represent the hierarchy of tags expressed in the XML file
  - XML/DOM Examples (will discuss in a moment)
    - Display contents of DOM
      - displayXML.java
    - Change contents of DOM
      - changeXML.java

Ch 23 XML Persistent Data Storage (Create DOM using Java)

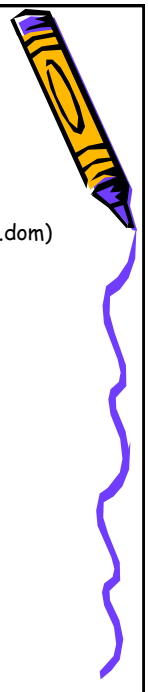
- To create a DOM from XML
  - DocumentBuilderFactory (javax.xml.parsers)
    - getInstance() method
  - DocumentBuilder (javax.xml.parsers)
    - parse(String filename) method
  - Document (org.w3c.dom)
    - a DOM object
  - XML/DOM Example (demo)
    - displayXML.java

Ch 23 XML Persistent Data Storage (DOM nodes)



- Details about DOM
  - A tree structure that has different types of nodes
    - Node — A generic node type
    - Element — A start tag translated into an Element node
    - Text — Data for a tag translated into a Text node

Ch 23 XML Persistent Data Storage (Java DOM nodes)

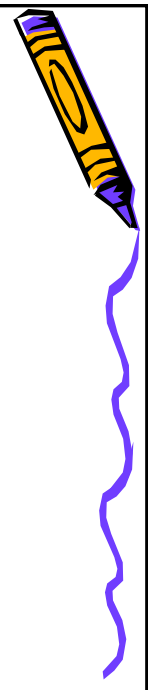


- Node (org.w3c.dom)
  - An interface in Java API
  - Has many subinterfaces in Java API, including
    - Attr
      - An attribute within an Element
    - Comment
      - Data within an XML comment
    - Document
      - The root of DOM tree
    - Element
      - Created for each start tag
    - Text
      - Data within a start/end tag

Ch 23 XML Persistent Data Storage (Save DOM using Java)

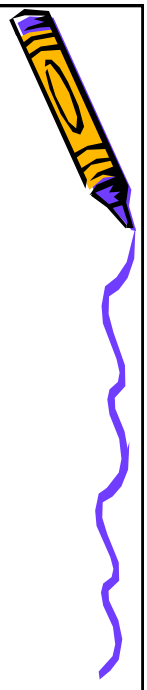
- Saving a DOM to XML
  - TransformerFactory (javax.xml.transform)
    - getInstance() method
  - Transformer (javax.xml.transform)
    - transform(DOMSource, StreamResult) method
  - DOMSource (javax.xml.transform.dom)
    - Contains a DOM object
  - StreamResult (javax.xml.transform.stream)
    - Typically contains a File or OutputStream
  - XML/DOM Example
    - changeXML.java (demo)

Ch 23 XML Persistent Data Storage (Java & DOM details)



- Details about Java and DOM
  - Look at the code!
  - XML/DOM Examples
    - Display contents of DOM
      - displayXML.java
    - Change contents of DOM
      - changeXML.java

Ch 24-OOD Case Study: XML



- Questions?