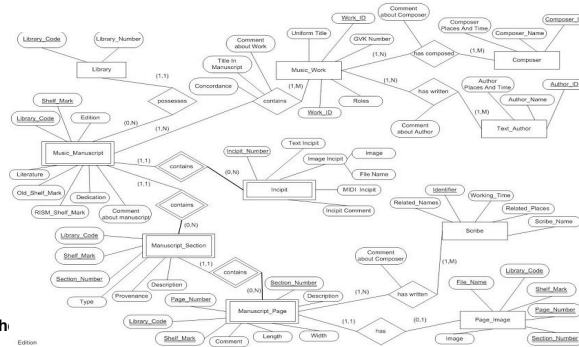


## Databases : Lecture 11: Entity/Relationship modelling Timothy G. Griffin Lent Term 2010



**Dr. Peter Chen**  
<http://bit.csc.lsu.edu/~ch>



[www.cl.cam.ac.uk/Teaching/current/Databases/](http://www.cl.cam.ac.uk/Teaching/current/Databases/)

1

## Conceptual Design

- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the integrity constraints (business rules) that hold?
- We can represent this information pictorially in E/R diagrams (and then map these to a relational schema later).

2

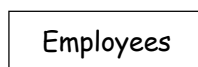
## E/R basics

- An **entity** is a real-world object that is distinguishable from other objects
- Each entity has **attributes** (with domains)
- A particular entity will have a value for each of its attributes
- An **entity type** defines a set of entities that have the same attributes
- An **entity set** is the collection of all entities of a particular entity type (at a particular point in time)

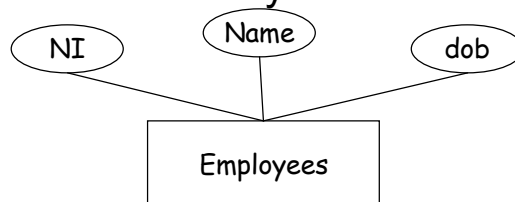
3

## Entities and attributes

- Entity types are drawn as rectangles, e.g.



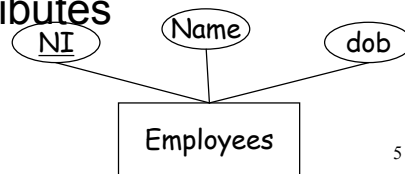
- Attributes are drawn as ovals, and attached to the entity sets with lines, e.g.



4

## Key attributes

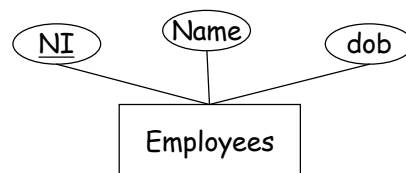
- A **key attribute** of an entity type is an attribute whose values are distinct for each entity
- Sometimes several attributes (a composite attribute) together form a key
  - NB: Such a composite should be **minimal**
- We underline key attributes



5

## Entity types to relations

- A (strong) entity type maps to a relation schema in the obvious way, e.g.



is mapped to the relation schema

$\text{Employees}(\overline{\text{NI}}:\tau_1, \text{Name}:\tau_2, \text{dob}:\tau_3)$

6

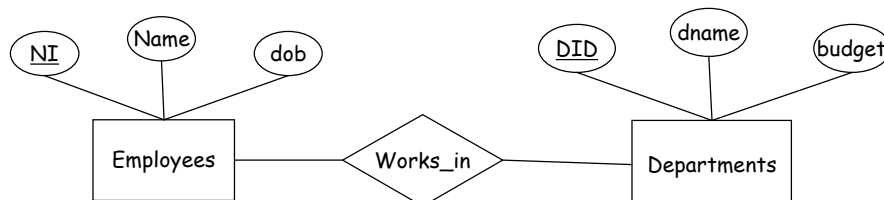
## Relationships

- A **relationship type** among two or more entity types defines a set of associations between entities from those types
  - Mathematically, relationship type R
$$R \subseteq E_1 \times \dots \times E_n.$$
- The set of instances of the relationship type is called the **relationship set**

7

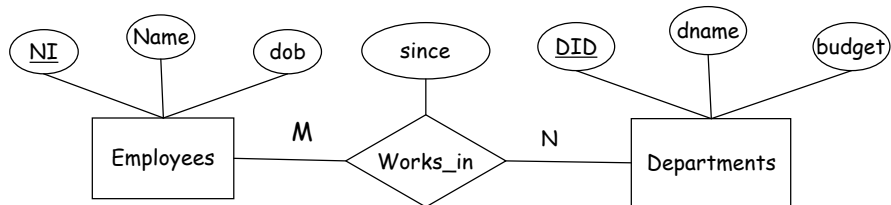
## Relationships in E/R

- Relationship types are represented by diamonds
- They connect the participating entity types with straight lines, e.g.



8

## Map to relation



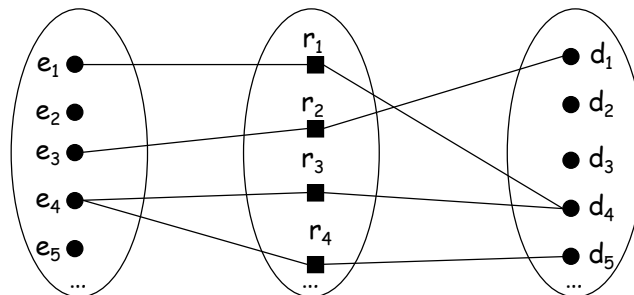
is mapped to the relation schema:

$\text{Works\_in}(\text{NI}:\tau_1, \text{DID}:\tau_2, \text{since}:\tau_3)$

9

## Relationship set diagrams

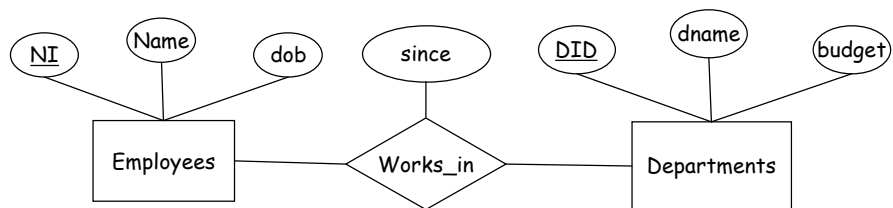
- Sometimes its useful to represent the relationship set diagrammatically



10

## Relationship attributes

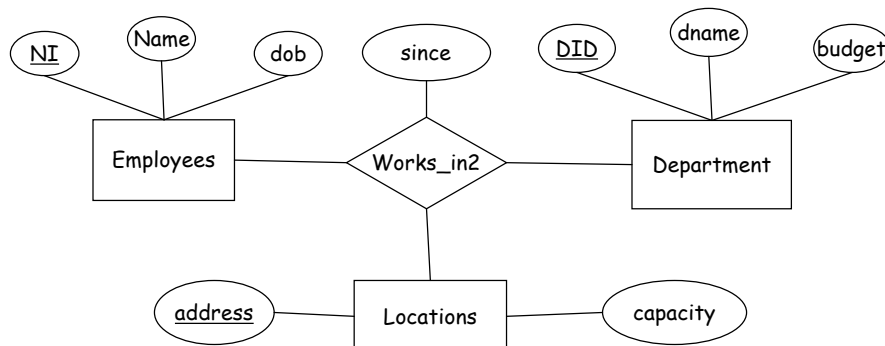
- Relationships can also have **attributes**
  - NB: A relationship must be uniquely determined by the entities, without reference to the relationship attributes



11

## N-ary relationships

- Although relatively rare, we can have n-ary relationships, e.g.



12

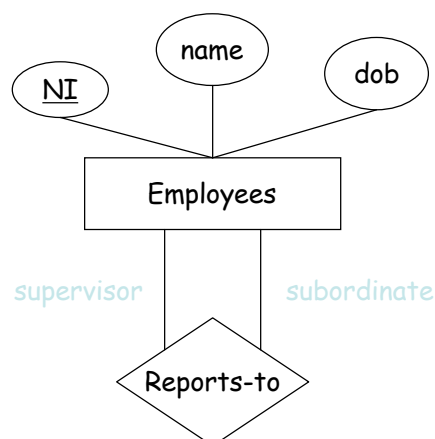
## Recursive relationships

- Each entity type in a relationship plays a particular **role**, which is associated with a **role name** (this is usually suppressed)
- An **recursive relationship** is when an entity type plays more than one role in the relationship type
- In this case the role name is required

13

## Recursive relationships in E/R

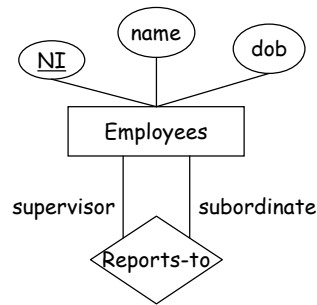
e.g.



14

## Recursive relationship sets

- Just pick appropriate field names! E.g.



is mapped to

`Reports_to(sup_NI:τ1, sub_NI: τ1)`

15

## Constraints on relationship types

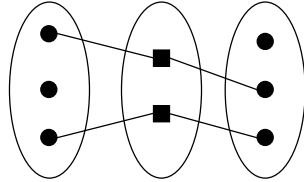
- For example:
  - An employee can work in many departments;  
a department can have many employees
  - In contrast, each department has at most one manager
- Thus we need to be able to specify the number of relationship instances that an entity can participate in.
- For binary relationships the possible ratios are: **1:1**, **1:N**, **N:1**, **M:N**

16

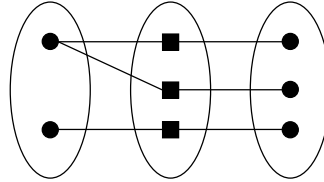


## Cardinality ratios

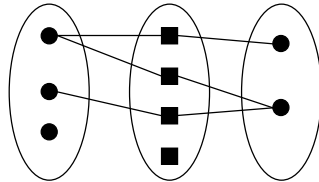
1:1



1:N



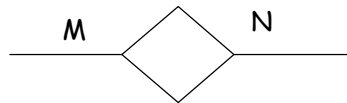
M:N



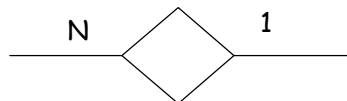
17

## Cardinality ratios in E/R

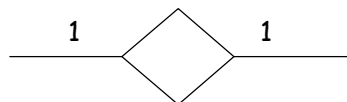
M:N



N:1



1:1



Note: Sometimes this is written using different arrowheads

18

## Participation constraints

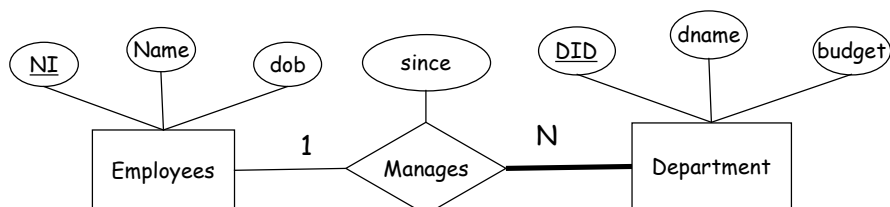
*Every department must have a manager*

- This is an example of a **participation constraint**
- The participation of an entity set, E, in a relationship set R is said to be **total** if every entity in E participates in at least one relationship in R. (If not its participation is said to be **partial**)

19

## Participation in E/R diagrams

- Total participation is displayed as a **bold line** between the entity type and the relationship
  - NB. Sometimes this is written as a double line



20

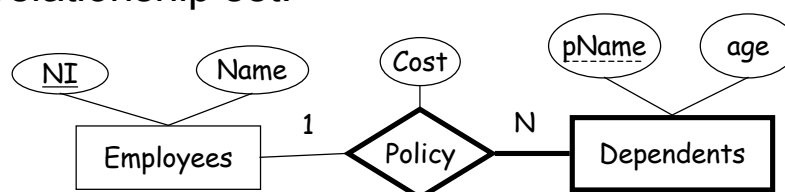
## Weak entity types

- An entity type may not have sufficient attributes to form a primary key
- Such an entity type is called a **weak entity type**
- A weak entity can only be identified uniquely by considering the primary key of another (**owner**) entity

21

## Weak entity types cont.

- Thus the owner and weak entity types must participate in a **1:N** relationship
- Weak entity set must have total participation in this **identifying** relationship set.



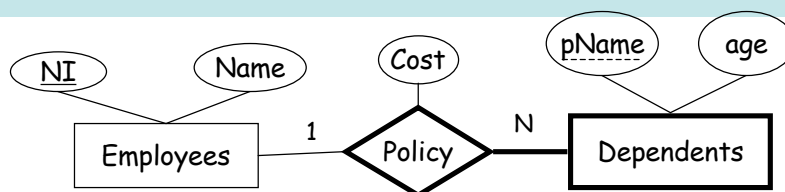
22

## Implementng Weak entity types

- Given a weak entity type,  $W$ , we generate a relation schema with fields consisting of the attributes of  $W$ , and the primary key attributes of the owner entity type
- For any relationship in which  $W$  appears we generate a relation schema which must take as the key for  $W$  all of its key attributes, including those from its owner set

23

## Example



is mapped to the following schema:

$\text{Dependents}(\underline{\text{pName}}:\tau_1, \text{NI}:\tau_2, \text{age}:\tau_3)$

$\text{Policy}(\underline{\text{pName}}:\tau_1, \text{NI}:\tau_2, \text{Cost}:\tau_4)$

Alternatively:

$\text{Policy}(\text{pName}:\tau_1, \text{NI}:\tau_2, \text{age}:\tau_3, \text{Cost}:\tau_4)$

24

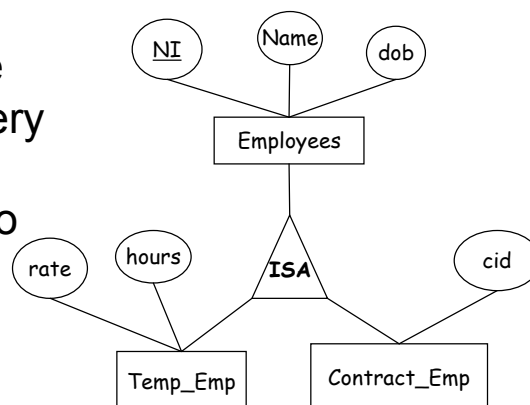
## Extended E/R modelling

- What we've seen so far is "classic" E/R
- Over the years a number of features have been added to the model and the modelling process
- These features include:
  - Sub- and super-classes
  - Specialisation
  - Generalisation
  - Categories
  - Higher/Lower-level entity sets
  - Attribute inheritance
  - Aggregation

25

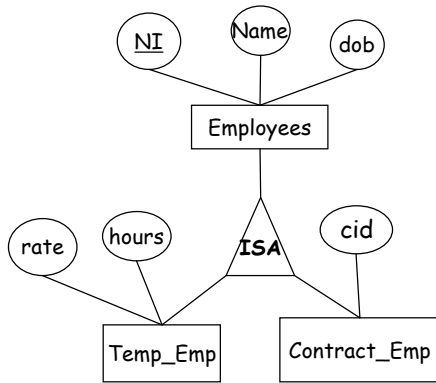
## ISA hierarchies

- We can devise **hierarchies** for our entity types
- If we declare **A ISA B**, every A entity is considered to be a B entity



26

## ISA Hierarchies



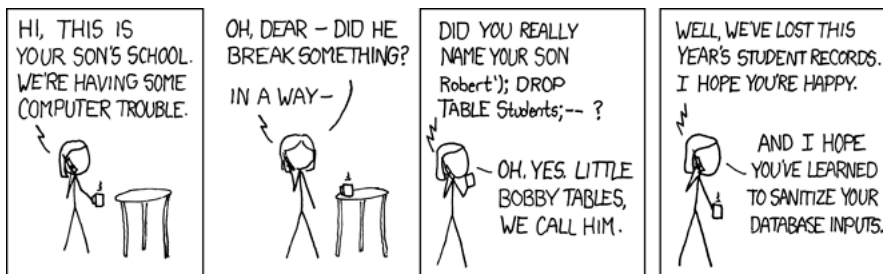
Two choices:

1. 3 relations  
(Employees, Temp\_Emp and Contract\_Emp)
2. 2 relations  
(Temp\_Emp and Contract\_Emp)

27

## Databases Lecture 12: Database Systems

**Timothy G. Griffin**  
Lent Term 2010



28

## What is a database system?

- A **database** is a large, integrated collection of data
- A database contains a model of something!
- A **database management system** (DBMS) is a software system designed to store, manage and facilitate access to the database

29

## What does a database system do?

- Manages Very Large Amounts of Data
- Supports **efficient** access to Very Large Amounts of Data
- Supports **concurrent** access to Very Large Amounts of Data
- Supports **secure, atomic** access to Very Large Amounts of Data

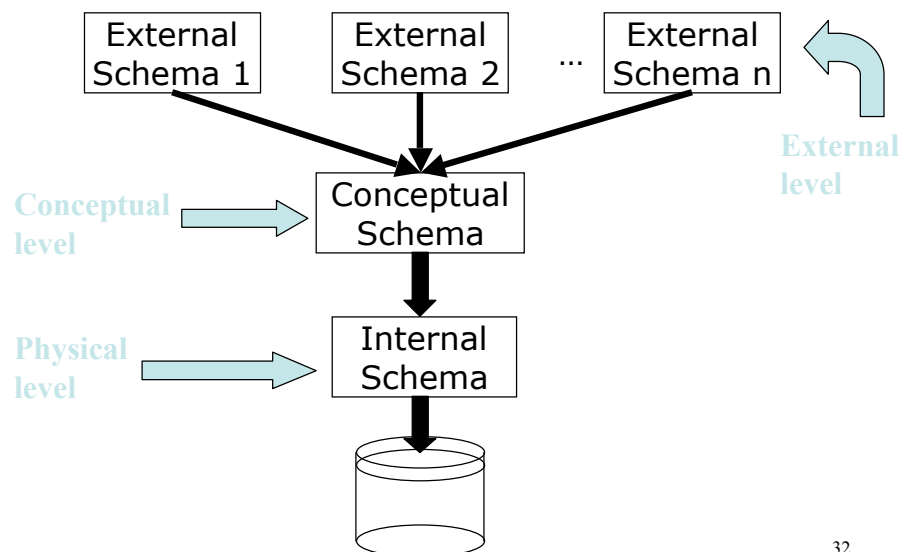
30

## Database system architecture

- It is common to describe databases in two ways
  - **The logical level:**
    - What users see, the program or query language interface, ...
  - **The physical level:**
    - How files are organised, what indexing mechanisms are used, ...
- It is traditional to split the logical level into two: overall database design (**conceptual**) and the views that various users get to see
- A **schema** is a description of a database

31

## Three-level architecture



32



## Logical and physical data independence

- **Data independence** is the ability to change the schema at one level of the database system without changing the schema at the next higher level
- **Logical data independence** is the capacity to change the conceptual schema without changing the user views
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema or user views

33

## Database Context

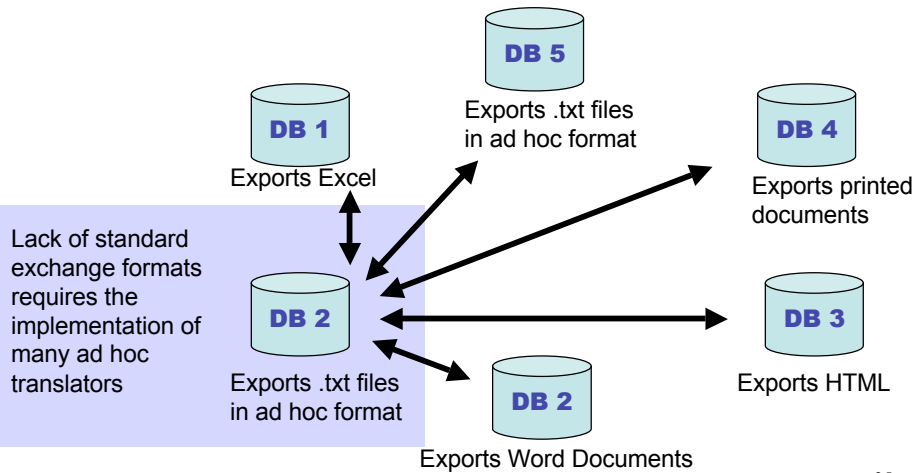
**Database systems are more and more likely to support features that “unlock” databases and allow them to easily interact in a larger context**

- Data-warehousing features
  - Data cube
- Inter-database exchange features
  - XML

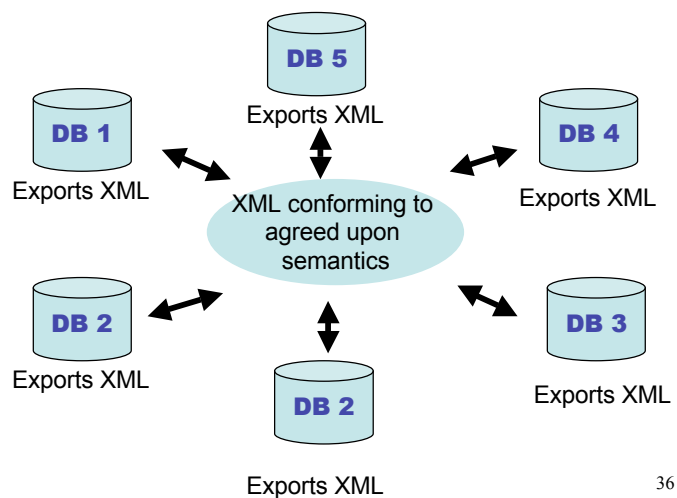
34

## The “Data Publishing” Problem

Need to share data without exposing internal details of your database.



## XML as a data exchange format



## XML and Databases

- **XML-enabled databases:**
  - Data stored in structured (usually relational) format.
  - XML primarily used as a **data exchange format**
  - Interfaces and SQL extensions provided to facilitate generation of XML and parsing of XML.
  - “Data-centric”
- **Native XML database:**
  - Allows direct storage and manipulation of XML data.
  - “Document-centric”

37

## What is XML?

- **Extensible Markup Language**
- **W3C proposal, Current version 1.0 (3<sup>rd</sup> ed.) February 2004**
- **Authors:**
  - **Tim Bray (Netscape)**
  - **Jean Paoli (Microsoft)**
  - **C.M. Sperberg-McQueen (W3C)**
  - **Eve Maler (Sun)**
  - **François Yergeau**

<http://www.w3.org/TR/REC-xml>

**XML has roots in HTML**

38

## HTML

- *Lingua-franca* for publishing hypertext on the web
- Designed to inform a web-browser both what information to render, **and** how it should be rendered
  - (Actually these shouldn't be mixed up)
- Easy to learn (Big win)
- Fixed tag set, rather odd syntax

39

## HTML: An example

```
<HTML>
Opening tag → <HEAD>
Text
(PCDATA) → <TITLE>
           Welcome to gmb's homepage
           </TITLE>
Closing tag → </HEAD>
           <BODY>
           <H1>Background info</H1>
           <IMG SRC="me-and-britney.jpg">
Attribute → I have a lot of great friends
(name and value)
           ...
           </BODY>
           </HTML>
```

40

## XML structure

- The fundamental construct is the **element**, which is essentially a pair of matching tags and the text between them, e.g.
  - `<name>Britney</name>` is an element
  - `<name>Victoria</nom>` is not an element
- XML documents must have **single** root element
- No fixed set of tags
- Elements can be properly nested, thus
  - `<name> ... <address> ... </address> ... </name>` ☺
  - `<name> ... <address> ... </name> ... </address>` ☹

41

## XML structure cont.

- We can represent various structures using nesting and repetition
- Tuple (Record):

```
<person>
  <name>Emma Bunton</name>
  <tel>020 8777 1234</tel>
  <email>baby@spicegirls.com</email>
</person>
```
- Lists:

```
<addresses>
  <person> ... </person>
  <person> ... </person>
  <person> ... </person> ...
</addresses>
```

42

## XML structure cont.

- Nesting can be used to avoid joins, e.g.

```
<bank>
  <cust><name>Britney Spears</name>
    <address>Florida</address>
  </cust> ...
  <acc>
    <accno>BS001</accno>
    <branch>Florida High Street</branch>
    <balance>10,000,000</balance>
  </acc> ...
  <saver>
    <sname>Britney Spears</sname>
    <saccno>BS001</saccno>
  </saver> ...
</bank>
```

43

## XML structure cont.

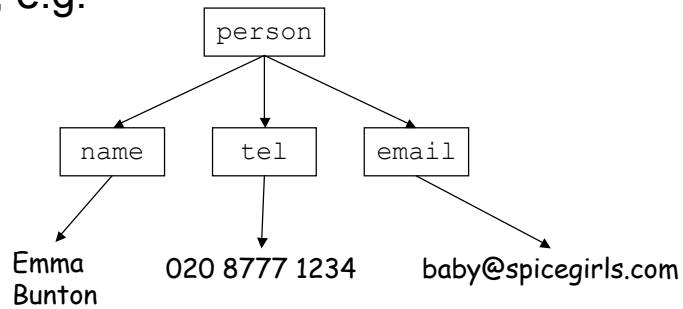
- Join avoiding:

```
<bank2>
  <cust>
    <name>Britney Spears</name>
    <address>Florida</address>
    <acc>
      <accno>BS001</accno>
      <branch>Florida High Street</branch>
      <balance>10,000,000</balance>
    </acc>
  </cust>
  ...
</bank2>
```

44

## XML and trees

- One can visualise XML documents as trees, e.g.



45

## Attributes

- In addition to elements we have **attributes**
- Attributes appear as `name=value` pairs in opening tags, e.g.
  - `<acc type="deposit"> ... </acc>`
  - `<acc type="saving" status="closed"> ... </acc>`
- (Aside: An element with no body can be abbreviated from `<foo></foo>` to `<foo/>`)

46

## DTDs

- XML documents can be created without any schema
- XML documents can contain a **document type definition (DTD)**, which is similar to a schema

47

## Example DTD

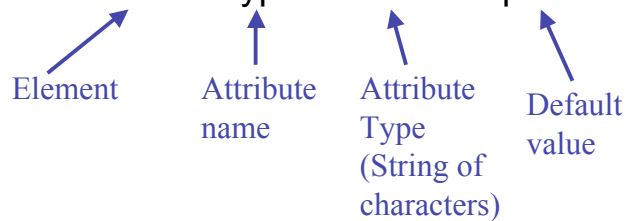
```
<!DOCTYPE bank [  
  <!ELEMENT bank ((acc|cust|saver)+)>  
  <!ELEMENT acc (accno branch balance)>  
  <!ELEMENT cust (name address)>  
  <!ELEMENT saver (sname saccno)>  
  <!ELEMENT accno (#PCDATA)>  
  <!ELEMENT branch (#PCDATA)>  
  <!ELEMENT balance (#PCDATA)>  
  <!ELEMENT name (#PCDATA)>  
  <!ELEMENT address (#PCDATA)>  
  <!ELEMENT sname (#PCDATA)>  
  <!ELEMENT saccno (#PCDATA)>  
>
```

48



## DTD details

- ‘|’ denotes alternative, ‘+’ denotes one or more, and ‘\*’ denotes zero or more
- ‘#PCDATA’ (Parsed Character Data) means any text!
- We can also specify attributes, e.g.
- `<!ATTLIST acc acctype CDATA "deposit">`



49

## Attributes

- An attribute of type **ID** provides a unique identifier for the element
- An attribute of type **IDREF** is a reference to an element
- Example:

```
<!ATTLIST account number ID #REQUIRED
           owners IDREFS #REQUIRED>

<account number="A001" owners="C001 C007">
...</account>
```

50

## Using DTDs

- DTDs are placed at the start of an XML document
- A document that conforms to its DTD is said to be **valid**
- Alternatively you can give a URL for a DTD, e.g.

```
<!DOCTYPE mybank SYSTEM
    "http://www.hsbc.com/mybank.dtd">
<mybank>
...
</mybank>
```

51

## Aside on DTDs

- Wouldn't it be better in ML?

```
datatype bank      = BANK of bankitem list
  and  bankitem = ACC of accno*branch*balance
      | CUST of name*address
      | SAVER of sname*saccno;

type accno  = string;
type branch = string;
type balance = string; (*could be int!*)
type name   = string;
type address = string;
type sname  = string;
type saccno = string;
```

52

## Schema

- You'll have noticed weaknesses with DTDs from a database schema point of view
  - Individual text elements and attributes can't be typed further
  - We don't need ordered sub-elements in database world
  - There is a lack of typing in IDs and IDREFs
- An effort to address these problems has led to a better schema language: **XML schema**

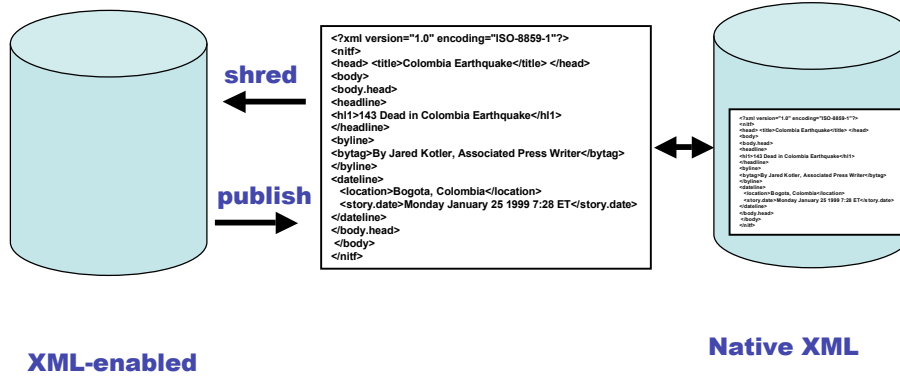
53

## Domain specific DTDs

- There are now lots of DTDs that have been agreed by groups, including
  - WML: Wireless markup language (WAP)
  - OFX: Open financial exchange
  - CML: Chemical markup language
  - AML: Astronomical markup language
  - MathML: Mathematics markup language
  - SMIL: Synchronised Multimedia Integration Language
  - ThML: Theological markup language ☺

54

## Native XML Databases



55

## Documents vs databases

- But this is a document, which is quite different from our world of databases

Document world	Database world
Lots of small documents	A few large databases
Static (normally)	Dynamic
Implicit structure	Explicit structure (schema)
Tagging	Records
Human friendly ☺	Machine friendly
Meta data: Author, title, date	Meta data: schema
Editing	Updating
Retrieval (IR)	Querying

56