Chapter 15
Introduction to Database Concepts
Differences Between Tables and Databases

• When we think of databases, we think of tables of information:
  – iTunes show the title, artist, running time on a row
  – Your car’s information is one line in the state’s database of automobile registrations
  – The U.S. is a row in the demography table for the World’s listing of country name, population, etc.
Comparing Tables

- These images show how the row of data is described using tags

| Canada  | 32805041 | 1.61 | 5 | 80.1 |

```xml
<demogData>
    <country>Canada</country>
    <population>32805041</population>
    <fertility>1.61</fertility>
    <infant>5</infant>
    <lifeExpct>80.1</lifeExpct>
</demogData>
```
The Database’s Advantage

- Metadata is the key advantage of databases over other approaches to recording data as tables
  - enables content search
- Two most important roles in defining metadata
  - **Identify the type of data**: each different type of value is given a unique tag
  - **Define the affinity of the data**: Tags enclose all data that is logically related
XML: A Language for Metadata Tags

- **XML** stands for the Extensible Markup Language
- It is a tagging scheme
- What makes XML easy and intuitive is that there are no standard tags to learn
- Tags are created as needed
  - This trait makes XML a *self-describing language*
XML: A Language for Metadata Tags

• There are a couple of rules:
  – Always match tags
  – Basically anything goes
• XML works well with browsers and Web-based applications
• XML must be written with a text editor to avoid unintentionally including the word processor’s tags (see Chapter 4)
## Rules for Writing XML

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Required first line</strong></td>
<td><code>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;&gt;</code> must appear as the first line of the file, starting in the first position.</td>
</tr>
<tr>
<td><strong>First tag</strong></td>
<td>The first tag encountered is the root element, and it must enclose all of the file’s content; it appears on the second or possibly third line.</td>
</tr>
<tr>
<td><strong>Closing tags</strong></td>
<td>All tags must be closed.</td>
</tr>
<tr>
<td><strong>Element naming</strong></td>
<td>Observe these rules:</td>
</tr>
<tr>
<td></td>
<td>- Names can contain letters, numbers, and underscore characters.</td>
</tr>
<tr>
<td></td>
<td>- Names must not start with a number or punctuation character.</td>
</tr>
<tr>
<td></td>
<td>- Names must not start with the letters xml (or XML, or Xml, etc.).</td>
</tr>
<tr>
<td></td>
<td>- Names cannot contain spaces.</td>
</tr>
<tr>
<td><strong>Case sensitivity</strong></td>
<td>Tags and attributes are case sensitive.</td>
</tr>
<tr>
<td><strong>Proper nesting</strong></td>
<td>All tags must be well nested.</td>
</tr>
<tr>
<td><strong>Attribute quoting</strong></td>
<td>All attribute values must be quoted; paired single quotes (apostrophes) or paired double quotes are okay; use one type inside the other if needed; no “curly” quotes of any type.</td>
</tr>
<tr>
<td><strong>White space</strong></td>
<td>White space is preserved and converted to a single space.</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>XML comments have the form <code>&lt;!-- This is a comment. --&gt;</code>.</td>
</tr>
</tbody>
</table>
XML

- As with HTML, the tag and its companion closing tag surround the data
- XML tag names cannot contain spaces
- Both UPPERCASE and lowercase are allowed
- XML is case sensitive
- Like HTML, XML doesn’t care about white space between tags
XML Example

• Scenario:
  – Create a database for the Windward Islands archipelago in the South Pacific
  – Plan what information will be stored
  – Develop those tags:

  <archipelago>
    <island>
      <iName>Tahiti</iName>
      <area>1048</area>
    </island>
    ...
  </archipelago>
XML

<?xml version = "1.0" encoding="UTF-8" ?>

• This required line is added at the beginning of the file
• Note the question marks.
• This line identifies the file as containing XML data representations
• The file also has standard UTF-8 encoded characters
Windward Islands XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<archipelago>
  <island><iName>Tahiti</iName> <area>1048</area></island>
  <island><iName>Moorea</iName> <area>130</area></island>
  <island><iName>Maiao</iName> <area>9.5</area></island>
  <island><iName>Mehetia</iName> <area>2.3</area></island>
  <island><iName>Tetiaroa</iName> <area>12.8</area></island>
</archipelago>
```

**Figure 15.1** XML file encoding data for the Windward Islands database. The first line states that the file contains XML tags.
Expanding the Use of XML

• To create a database of the two similar items (in this chapter, archipelagos), put both sets of information in the file
• As long as the two sets use the same tags for the common information, they can be combined
• Extra data is allowed and additional tags can be created (\texttt{<a\_name>} to identify which archipelago is being used)
Expanding the Use of XML

• Group sets of information by surrounding them with tags
• These tags are the root elements of the XML database
• A root element is the tag that encloses all content of the XML file
  – In Figure 15.1 the <archipelago> tag was the root element
<?xml version = "1.0" encoding="UTF-8" ?>
<geo_feature>

<archipelago>
  <a_name>Windward Islands</a_name>
  <island><iName>Tahiti</iName> <area>1048</area> </island>
  <island><iName>Moorea</iName> <area>130</area> </island>
  <island><iName>Maiao</iName> <area>9.5</area> </island>
  <island><iName>Mehetia</iName> <area>2.3</area> </island>
  <island><iName>Tetiaroa</iName> <area>12.8</area> </island>
</archipelago>

<archipelago>
  <a_name>Galapagos</a_name>
  <island><iName>Isabela</iName> <area>4588</area> <elev>1707</elev> </island>
  <island><iName>Fernandina</iName> <area>642</area> <elev>1494</elev> </island>
  <island><iName>Tower</iName> <area>14</area> <elev>76</elev> </island>
  <island><iName>Santa Cruz</iName> <area>986</area> <elev>846</elev> </island>
</archipelago>

</geo_feature>

Figure 15.2  XML file for the Geographic Features database.
Attributes in XML

• HTML tags can have attributes to give additional information
• Tags of XML also have attributes
  – They have a similar form
  – Must always be set inside simple quotation marks
  – Tag attribute values can be enclosed either in paired single or paired double quotes

<entry warnIfNone="Ain't there!">The user entered this data.</entry>
Attributes in XML

• Writing tag attributes is easy enough
• The rules for using quotes are straightforward
• Use attributes for additional metadata, not for actual content
Effective Design with XML Tags

• XML is a flexible way to encode metadata

• **Identification Rule**: Label Data with Tags Consistently
  – You choose the tags, but once you’ve decided you must always surround that kind of data with that tag
  – Keeps data together
Effective Design with XML Tags

- **Affinity Rule**: Group Data Referencing an Entity
  - Enclose in a pair of tags all tagged data referring to the same thing
  - Grouping it keeps it all together, but it also makes an association of the tagged data items as being related to each other
Effective Design with XML Tags

• **Collection Rule:** Group Instances
  – When you have several instances of the same kind of data, enclose them in tags
  – Keeps them together and implies that they are instances of the same type
The XML Tree

• The rules for producing XML encodings of information produce hierarchical descriptions
  – Can be thought of as trees
  – The hierarchy is a consequence of how the tags enclose one another and the data
Tables and Entities

• Lets set aside the tagging and XML and focus on database tables
• The XML tree on the next slide shows the root element to the left and the leaves (content) to the right
Figure 15.3 The XML displayed as a tree. The encoding from Figure 15.2 is shown with the root element (geo_feature) to the left and the leaves (content) shown to the right.
Database Tables

• Any group of things with common characteristics that specifically identify each one can be formed into a database table

• Contains a set of things with common attributes
Database Tables

<table>
<thead>
<tr>
<th>ID</th>
<th>IslandName</th>
<th>Area</th>
<th>Elev</th>
<th>Archipelago</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tahiti</td>
<td>1048</td>
<td></td>
<td>Windward Islands</td>
</tr>
<tr>
<td>2</td>
<td>Moorea</td>
<td>130</td>
<td></td>
<td>Windward Islands</td>
</tr>
<tr>
<td>3</td>
<td>Maiao</td>
<td>10</td>
<td></td>
<td>Windward Islands</td>
</tr>
<tr>
<td>4</td>
<td>Mehetia</td>
<td>2</td>
<td></td>
<td>Windward Islands</td>
</tr>
<tr>
<td>5</td>
<td>Tetaaroa</td>
<td>13</td>
<td></td>
<td>Windward Islands</td>
</tr>
<tr>
<td>6</td>
<td>Isabela</td>
<td>4588</td>
<td>1707</td>
<td>Galapagos Islands</td>
</tr>
<tr>
<td>7</td>
<td>Fernandina</td>
<td>642</td>
<td>1494</td>
<td>Galapagos Islands</td>
</tr>
<tr>
<td>8</td>
<td>Tower</td>
<td>14</td>
<td>76</td>
<td>Galapagos Islands</td>
</tr>
<tr>
<td>9</td>
<td>Santa Cruz</td>
<td>985</td>
<td>846</td>
<td>Galapagos Islands</td>
</tr>
</tbody>
</table>

**Figure 15.4** A database table instance (from Microsoft Access) representing the same information as shown in Figure 15.2.
Database Vocabulary

- **Entities**: rows of the database table
- **Attribute Name**: column heading
- **Entity Instance**: value in a row
- **Table Instance**: whole table
What to Notice

• Rows are all different
  – Two rows can have the same value for some attributes, but not all
• Even when we don’t know the data for an attribute value it is still a characteristic
• The rows can be in any order
• The columns can be in any order
What to Notice

• Rearranging the rows or columns will result in the same table
• If we add (or remove) rows, or change a value we create a new table instance
Properties of Entities

- A database table can be empty
  - It is a table with no rows
- An entity is anything defined by a specific set of attributes
- A table exists with a name and column headings
- Once entity instances have been specified, there will be rows
- Among the instances of any table is the “empty instance”
Every One is Different

• Amoebas are not entities, because they have no characteristics that allow us to tell them apart
• One-celled animals are entities
• In cases where it is difficult to process the information specifically identifying an entity, we might select an alternate encoding
• Entities are the data of databases
Relational Database Tables

- Tables are technically called relations, but we’ll continue to call them database tables.
- The rows must always be different, even after adding rows.
  - Be sure the table has all of the attributes (columns) needed to tell the entities apart.
  - You can always add a sequence number to guarantee that every row is different.
Keys

• By itself, repeated data in a column is not a problem.

• We are interested in columns in which all of the entries are always different, because they can be used to look up data:
  – such a column is called a candidate key
  – doesn't have to be just one column (it can be multiple columns together)
Keys

• **Primary Key**: candidate key that the computer and user agree will be used to locate entries during database operations
A Database Table’s Metadata

• It is possible to succinctly describe a database table with a database scheme or database schema
  – Attributes are listed, one per row
  – For each attribute, the user specifies its data type and whether or not it is the primary key
  – It is also customary to include a brief description

• The database scheme is the database table’s metadata
Computing with Tables

• To get information from database tables, we write a query describing what we want

• **Query**: command that tells the database system how to manipulate its tables to compute the answer
  – the answer will be in the form of another database table

• We need to know 6 operations
Project Operation

• *Project* (pronounced prōJECT) picks out and arranges columns from one database table to create a new, possibly “narrower”, table.
Select Operation

- The Select operation picks out rows according to specified criterion
Cross-Product Operation

• Combines two tables in a process like multiplication
  – For each row in the first table, we make a new row by appending a row form the second table
  – All combinations are in the result
**Cross-Product Operation**

- Because we pair all rows, a table with \( m \) rows crossed with a table with \( n \) rows will produce a table with \( m \times n \) rows.
- Using cross-product with other table operations is powerful:
  - Often follow with a select to choose wanted entities.
  - Then a project to narrow to wanted attributes.
**Union** Operation

- Combines two tables with compatible attributes (columns)
- The result has rows from both tables
- For any rows that are in both tables, only one copy is included in the result
**Difference Operation**

- The opposite of the Union operation
- D1 – D2 contains the rows of the D1 table that are not also in the D2 table
Join Operation

- Join is a combination of a Cross-Product, followed by a Select operation
- Takes two database tables, and an attribute from each one (D1.a1 and D2.a2)
Summarizing the Science

• Joining is optional
  – It is always possible to express what Join does using only Cross-Product and Select

• Five operations do the work
  – Given a set of database tables for entities, the operations of Project, Select, Cross-Product, Union and Difference are sufficient to create any database table derivable from them
• The Relationship Metadata
  – Notice that we used operations in queries
  – We have exploited the fact that data in one table was related to data in another table
  – To maximize the help the database system gives us, we need to tell the software about these relationship
  – a relationship is a property of two attributes saying that there is a connection between their data values
SQL: The Language of Databases

• The most widely used database language is SQL (Structured Query Language)
• The operations we call Project and Select are combined into one command called Select
  – uses WHERE to specify the formula
• Users INNER JOIN rather than just JOIN
Structure of a Database

• There are two forms of tables:
  – The *physical* database is stored on the disk drives of the computer system and is a permanent repository of the database
  – The *logical* database is known as the view of the database, created for users on-the-fly, customized for their needs
Physical Database

• The physical database is designed by database administrators
• Data must be accessed fast
• The physical database is set up to avoid redundancy (duplicate information)
  – There is a good chance that data stored in various places will not be updated
Logical Database

• The logical database shows users the view of the information they need and want
• It doesn’t exist permanently, but is created every time they need it
• The logical database is retrieved from the one copy stored in the physical database, and provided to the users as needed
Logical Database

- Creating a new copy each time is essential
  - If it were to be created once and then stored on the user’s computer, then there would be two copies of the information
- The other advantage of creating specialized versions of the database for each user is that different users want to see different information
Queries

• A **query** is a specification using the five operations and Join that define a table from other tables

• Queries are written in the standard database language **SQL** (Structured Query Language)

• **SQL** allows a new query to be run each time it is selected or opened