Test Your Tech

A spreadsheet:
A. Only happens on laundry day.
B. Is covered with food during holiday meals.
C. Helps answer “what-if” questions.

Review

• Quiz on chapters 9 and 13
  ∗ Review the questions at the end of each chapter

Spreadsheets

Spreadsheets are a powerful abstraction for organizing data and computation

An Array of Cells

A spreadsheet is a 2-dimensional array of cells…it’s 3D with multiple cells
• The idea is that the rows or columns represent a common type of data
  • They will be operated on similarly, so that’s easy to do
  • Adding more data of the same type means adding more rows or columns
  • Often spreadsheets contain numbers, but text-only spreadsheets are useful, too

Looking for Similar Ideas

Spreadsheets are not so unusual
• The position (row/column) names the data, as with memory locations, variables, forms, etc.
• Operating on all elements of a column (or row) is an iteration, though not usually a WFI
• Setting a cell to a formula is an (unevaluated) assignment statement with cells as variables
• The formula is an expression
• Functions are built-in expressions

Think of spreadsheets as a handier interface for computing than JavaScript
Familiar Terminology

- column heading
- column name
- referenced cell
- formula
- cell
- row name
- row

Formulas

The data in a spreadsheet can be manipulated using formulas.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Common Name</td>
<td>Distance (km)</td>
<td>Body Length (m)</td>
</tr>
<tr>
<td>2</td>
<td>Swainson’s Hawk</td>
<td>13500</td>
<td>0.51</td>
</tr>
<tr>
<td>3</td>
<td>Willow Warbler</td>
<td>15000</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>Short-tailed Shy</td>
<td>12500</td>
<td>0.42</td>
</tr>
<tr>
<td>5</td>
<td>Long-Tailed Ski</td>
<td>18000</td>
<td>0.31</td>
</tr>
<tr>
<td>6</td>
<td>Arctic Tern</td>
<td>19000</td>
<td>0.15</td>
</tr>
</tbody>
</table>

The value in D2 (selected cell) is the value in B2 times 0.621...the result is shown but the cell has the formula.

Apply Formula Again

The data in a spreadsheet can be manipulated using formulas.

Notice the formula.

Filling Replicates Formulas

Fill is a spreadsheet shortcut for copy-and-paste.

• Grab the fill tab and pull in the direction to be pasted.

Relative & Absolute Addressing

• References to cells happen in two ways: Relative and Absolute (with $)
  • F2 relative column, relative row
  • F$2 relative column, absolute row
  • $F2 absolute column, relative row
  • $F$2 absolute column, absolute row

Relative references change when pasted/filled; absolute references do not!

A Powerful Translation

The graphic shows the equations in the cells with the translation:

• The row changes going down but the column doesn’t.
Creating a discount table is a case of using both relative and absolute references.

- Consider store credit of $1 per $10 spent.
- $3 store credit for every 2 CDs (1 earns $1).

<table>
<thead>
<tr>
<th>Spent</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10</td>
<td>$1.00</td>
<td>$4.00</td>
<td>$5.00</td>
<td>$7.00</td>
<td>$8.00</td>
<td>$11.00</td>
<td>$12.00</td>
<td>$13.00</td>
</tr>
<tr>
<td>$20</td>
<td>$3.00</td>
<td>$6.00</td>
<td>$8.00</td>
<td>$10.00</td>
<td>$11.00</td>
<td>$12.00</td>
<td>$13.00</td>
<td>$14.00</td>
</tr>
<tr>
<td>$30</td>
<td>$4.00</td>
<td>$6.00</td>
<td>$8.00</td>
<td>$10.00</td>
<td>$11.00</td>
<td>$12.00</td>
<td>$13.00</td>
<td>$15.00</td>
</tr>
<tr>
<td>$40</td>
<td>$5.00</td>
<td>$7.00</td>
<td>$9.00</td>
<td>$11.00</td>
<td>$12.00</td>
<td>$13.00</td>
<td>$15.00</td>
<td>$16.00</td>
</tr>
<tr>
<td>$50</td>
<td>$6.00</td>
<td>$8.00</td>
<td>$10.00</td>
<td>$12.00</td>
<td>$13.00</td>
<td>$14.00</td>
<td>$16.00</td>
<td>$17.00</td>
</tr>
<tr>
<td>$60</td>
<td>$7.00</td>
<td>$9.00</td>
<td>$11.00</td>
<td>$13.00</td>
<td>$14.00</td>
<td>$16.00</td>
<td>$18.00</td>
<td>$19.00</td>
</tr>
</tbody>
</table>

A cell is based on first column, top row data in that row and column. Must mix relative and absolute references.

Another handy property of fill is that it can make a series based on constants.

- Fill Sunday => Monday, Tuesday, Wed...
- Fill 22 Feb => 23 Feb, 24 Feb, 25 Feb...

More generally:
- Series fill will even count using a constant.
- Counting by odd sizes give 1st two items.

Crackers and cookies are:
- A. Bytes to share with friends.
- B. The best minor league baseball team of all time and their cheerleaders.
- C. Hackers who attempt to break a program (crackers) and data stored on your computer by a Web server (cookies).

Databases are collections of information; our study repeats a theme: Tell the computer the structure, and it can help you!
Why Study Databases?

Some of us want to compute, but all of us want information ...
- Much of the archived information is in tables
- Databases enhance applications, e.g. Web
- Once you know how to create databases, you can use them to personal advantage
- Databases introduce interesting ideas

How much of your information can be in a table?

Stone Age Databases

Before relational databases (the kind we study) there were only "flat files"
- Structural information was difficult to express
- All processing of information was "special cased" -- custom programs were needed
- Information repeated; difficult to combine
- Changes in format of one file means all programs that ever processed that file must be changed ... adding ZIP codes

E.F. Codd of IBM invented relational databases

Relational Databases

Information is stored in tables
- Tables store information about entities -- things or stuff ... keep entities of one kind
- Entities have characteristics called attributes
- Tables are tuples (rows or records) of attributes (columns or fields)
- Every row must be unique, identified by a key
- Relationships -- associations among the data values are stored

Table structure = schema
Table contents = instance

A Table

Tables have names, attributes, tuples

Example: Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Last</th>
<th>First</th>
<th>Hire</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doyle</td>
<td>Tom</td>
<td>01</td>
<td>500 30th Ave E</td>
</tr>
<tr>
<td>2</td>
<td>Fuller</td>
<td>Andrew</td>
<td>14 Aug 1982</td>
<td>205 W. Capital Wy</td>
</tr>
<tr>
<td>3</td>
<td>Wood</td>
<td>John</td>
<td>01 Apr 1983</td>
<td>722 Moore Biz Blvd</td>
</tr>
<tr>
<td>4</td>
<td>French</td>
<td>Martin</td>
<td>03 May 1983</td>
<td>4112 Old Redwood Rd</td>
</tr>
<tr>
<td>5</td>
<td>Buchanan</td>
<td>Teresa</td>
<td>17 Oct 1984</td>
<td>19 Walnut St</td>
</tr>
<tr>
<td>6</td>
<td>Sullivan</td>
<td>Owen</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Instance

Schema

Redundancy is Very Bad

Not every assembly of tables is a good database -- repeating data is bad
- Replicated data can differ in its different locations, e.g. multiple addresses can differ
- Inconsistent data is worse than no data
- Keep a single copy of any data, and if it is needed in multiple places, associate it with a key, and store key rather than the data

You can look it up

When looking for information, a single item might be the answer, but a table is more likely
- "Who is taking FIT100"? Table of students
- "Whose mile run time ≤ 4:00?" Runner table
- "Who won 2003 Grammy for 'Best New Artist'?" A table containing only a single row
- "Who is president of U.W.?" Empty Table

Queries to a DB (set of tables) produces tables
Tables From Tables

There are five fundamental operations on tables to create tables:

- **Select** -- pick rows from a table
- **Project** -- pick columns from a table
- **Union** -- combine two tables with like columns
- **Difference** -- remove one table from another
- **Product** -- create "all pairs" from two tables

Though not primitive "Join" is usually included

1. Select Operation

Select creates a table from the rows of another table meeting a criterion

**Select from Example On Hire < 1993**

<table>
<thead>
<tr>
<th>ID</th>
<th>Last</th>
<th>First</th>
<th>Hire</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Davis</td>
<td>Nancy</td>
<td>01 May 1992</td>
<td>1001 20th Ave E</td>
</tr>
<tr>
<td>2</td>
<td>Fuller</td>
<td>Andrew</td>
<td>14 Aug 1992</td>
<td>900 W. Capital Way</td>
</tr>
<tr>
<td>3</td>
<td>Wooster</td>
<td>Edward</td>
<td>01 Apr 1993</td>
<td>722 Main St.</td>
</tr>
<tr>
<td>4</td>
<td>Peacock</td>
<td>Margaret</td>
<td>03 May 1993</td>
<td>4110 Old Redmond Rd.</td>
</tr>
<tr>
<td>5</td>
<td>Buchanan</td>
<td>Steven</td>
<td>17 Oct 1994</td>
<td>13 Grand Blvd</td>
</tr>
<tr>
<td>6</td>
<td>Sullivan</td>
<td>Okan</td>
<td>12 Dec 1994</td>
<td>Country House</td>
</tr>
</tbody>
</table>

2. Project

Project creates a table from the columns of another table

**Project Last, First From Example**

<table>
<thead>
<tr>
<th>ID</th>
<th>Last</th>
<th>First</th>
<th>Hire</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Davis</td>
<td>Nancy</td>
<td>01 May 1992</td>
<td>1001 20th Ave E</td>
</tr>
<tr>
<td>2</td>
<td>Fuller</td>
<td>Andrew</td>
<td>14 Aug 1992</td>
<td>900 W. Capital Way</td>
</tr>
<tr>
<td>3</td>
<td>Wooster</td>
<td>Edward</td>
<td>01 Apr 1993</td>
<td>722 Main St.</td>
</tr>
<tr>
<td>4</td>
<td>Peacock</td>
<td>Margaret</td>
<td>03 May 1993</td>
<td>4110 Old Redmond Rd.</td>
</tr>
<tr>
<td>5</td>
<td>Buchanan</td>
<td>Steven</td>
<td>17 Oct 1994</td>
<td>13 Grand Blvd</td>
</tr>
<tr>
<td>6</td>
<td>Sullivan</td>
<td>Okan</td>
<td>12 Dec 1994</td>
<td>Country House</td>
</tr>
</tbody>
</table>

3. Union

Union (written like addition) combines two tables with same attributes

- **PoliticalUnits = States + Provinces**

4. Difference

Difference (written like subtraction) removes 1 table's rows from another

- **Eastern = States - WestCoast**

5. Product

Product (written like multiplication) combines columns and pairs all rows

**Colors = Blues x Reds**

There's divide, too, but forget it
Join

Join (written like a bow tie) combines rows (like \( \times \)) if common field matches.

\[
\text{Homes} = \text{States} \bowtie \text{Students}
\]

<table>
<thead>
<tr>
<th>States</th>
<th>Capital</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washington</td>
<td>Olympia</td>
<td>Mt. Ranier</td>
</tr>
<tr>
<td>Oregon</td>
<td>Salem</td>
<td>Columbia Lake</td>
</tr>
<tr>
<td>California</td>
<td>Sacramento</td>
<td>Golden Gate</td>
</tr>
<tr>
<td>Arizona</td>
<td>Phoenix</td>
<td>Good Carpenter</td>
</tr>
<tr>
<td>Nevada</td>
<td>Carson City</td>
<td>Las Vegas</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Students</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Last</td>
</tr>
<tr>
<td>John</td>
<td>Jones</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Smith</td>
</tr>
<tr>
<td>Brian</td>
<td>Tony</td>
</tr>
</tbody>
</table>

DB Operations

The five DB Operations can create any table from a given set of tables.

- All modern database systems are built on these relational operations.
- Join is not primitive, but can be built from 5.
- Join, select, and project are used most often.
- The operations are not usually used directly, but are used indirectly from other languages.

SQL, the DB language we learn, is built on basic 5.

Reflection

- Write for 5 minutes on this topic:
  - Compare and contrast spreadsheets and databases, and explain the reasons for choosing one or the other.