Much of the data people keep is not stored as text, but rather is organized in the form of tables. Knowing how that data is structured and becoming proficient at manipulating it is the key to finding things out.

**Displaying Information Informally**

- Tables are a common way to present information
- Informal tables must be regularized to be database tables ...
- A (relational) database is a set of tables -- tables describe entities
FIT 100 Terms of a Database Table

❖ Name the structural parts of database tables

A DB key is a field(s) that’s unique for each row

❖ Attributes have types -- Species is a string, Photo is jpg

<table>
<thead>
<tr>
<th>Attribute (Field)</th>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Butterfly</td>
</tr>
<tr>
<td>Genus</td>
<td>Butterfly</td>
</tr>
<tr>
<td>Species</td>
<td>Butterfly</td>
</tr>
<tr>
<td>LWS</td>
<td>Butterfly</td>
</tr>
<tr>
<td>GWS</td>
<td>Butterfly</td>
</tr>
<tr>
<td>Range</td>
<td>Butterfly</td>
</tr>
<tr>
<td>L_Diet</td>
<td>Butterfly</td>
</tr>
<tr>
<td>A_Diet</td>
<td>Butterfly</td>
</tr>
<tr>
<td>Photo</td>
<td>Butterfly</td>
</tr>
</tbody>
</table>

Tuple (Row) (Records)

- Comma
  - Butterfly: Polygonia satyrus
  - LWS: 1.75
  - GWS: 2.5
  - Range: North America
  - L_Diet: Nettles
  - A_Diet: Tree sap

- Carol's Fritillary
  - Butterfly: Speyeria carolae
  - LWS: 2.375
  - GWS: 3.375
  - Range: Clark County NV
  - L_Diet: Mountain Violet
  - A_Diet: Charleston Mountain Violet

- Behr's Metalmark
  - Butterfly: Apodemia virgulti
  - LWS: 0.75
  - GWS: 0.9375
  - Range: So. California, Baja
  - L_Diet: Buckwheat
  - A_Diet: Forest Nectar

FIT 100 Terminology

❖ The structure of a database is its database schema

❖ The schema specifies ...
  + The list of tables forming the database
  + For each table, the fields of its records
  + For each field, its properties, i.e. data type, key or not key, default value, etc.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Genus</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>Species</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>LWS</td>
<td>Double</td>
<td>The least wing span measurement</td>
</tr>
<tr>
<td>GWS</td>
<td>Double</td>
<td>The greatest wing span measurement</td>
</tr>
<tr>
<td>Range</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>L_Diet</td>
<td>String</td>
<td>Larval Diet</td>
</tr>
<tr>
<td>A_Diet</td>
<td>String</td>
<td>Adult Diet</td>
</tr>
<tr>
<td>Photo</td>
<td>JPEG</td>
<td></td>
</tr>
</tbody>
</table>

Primary Key – Genus:Species
Instances

- A database as the word is normally used, i.e. tables with specific contents, is known as a database *instance* (of a data base schema)
- There can be many instances of a single database schema

<table>
<thead>
<tr>
<th>Butterfly</th>
<th>Lilac-bordered Copper</th>
<th>Lycaena nivalis</th>
<th>1 - 1 3/8</th>
<th>Rockies &amp; west</th>
<th>Douglas knotweed</th>
<th>Flower nectars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug Copper</td>
<td>Lycaena epixanthe</td>
<td>7/8 - 1</td>
<td>Great Lakes &amp; east</td>
<td>Cranberries</td>
<td>Raindrops</td>
<td></td>
</tr>
<tr>
<td>Red Rim</td>
<td>Biblis hypania</td>
<td>2 - 3</td>
<td>Rio Grande Valley TX</td>
<td>Nectarburn and spurge</td>
<td>Rotting fruit</td>
<td></td>
</tr>
</tbody>
</table>
Avoiding redundancy … keep specialized tables

The Administration keeps permanent records

Others keep their own tables

These tables have no redundancy

Concept: Arrange tables to avoid redundancy, join tables to provide the data users want to see
Organizing Business Data

- Companies, universities, government agencies, etc. have many database applications in common
  - Employee records
  - Payroll records
  - Customers/clients/students records
  - Products/services listings
  - ...
- Databases, as described thus far, are completely adequate for representing and managing this data
- Database changes can be controlled
  - Changes can be made by “authorized” employees
  - Changes can be made periodically, in batches

Maintaining On-line Databases

- Many databases are only useful if they are on-line, i.e. can be changed interactively
  - Airline reservations
  - Credit card accounts, ATMs and other banking
  - Catalog merchandising
  - eCommerce
  - ...
- On-line changes are called transactions
- The “transactions concept” not only makes interactive DBs possible, it is a good metaphor for other database processing as well
Transactions ...

- A transaction is a single operation (reference or change) to a database usually involving only one or a few records
  - Credit card purchase
  - ATM withdrawal of cash
  - Flight reservation
  - …
- Many transactions are taking place at once, typically
- Keeping the DB “correct” is a problem!

VISA processed 110,086,395 transactions on December 14, 1998, a 1-day world’s record

Correct Database?

- Because transactions take place simultaneously, there is a possibility that two computers can be making changes to the same data at the same instant, possibly corrupting it …
- Consider two transactions
  - T1: Deposit $100 into Account #12345
  - T2: Withdraw $100 from Account #12345
- When transactions are over, the balance should be unchanged
- But what if the transactions take place “at once”?
Tale Of Two Transactions

- **T1**: Deposit $100
  - **T1.1**: Fetch balance for
  - **Acct #12345**: $500.00
  - **T1.2**: Balance = Balance + $100
  - **T1.3**: Set DB balance to $600
  - **T1.4**: End transaction

- **T2**: Withdraw $100
  - **T2.1**: Fetch balance for
  - **Acct #12345**: $500.00
  - **T2.2**: Balance >= $100?
  - **T2.3**: Yes, Bal = Bal - $100
  - **T2.4**: Set DB balance to $400
  - **T2.5**: End transaction

The depositor might be pleased, but the bank won’t be!

Correctness

- The DB System must assure that every DB change happens as if the transactions happened one-at-a-time
- The one-at-a-time protocol solves the “problem” with **T1** and **T2**:
  - **T1** applied first, then **T2**: $500 --> $600 --> $500
  - **T2** applied first, then **T1**: $500 -- $400 -- $500
- Transaction processing systems make sure such problems do not arise by “locking” the data (only one computer at a time can unlock the data)

“Two changes at once with unsavory results” can happen when multiple people work on the same document … use locking idea
Reliability

❖ What happens to the database when …
  + The power goes out
  + Someone spills coffee into the disk drives
  + The computer crashes with all the changes to the DB for the last three hours in its (volatile) RAM
  + A new employee accidentally deletes the payroll file before printing the pay checks?
  + A virus cleans off the corporate disks
  + A hacker infiltrates the enterprise and begins transferring funds to a Swiss bank account
  + A disgruntled employee deletes the retirement plans and stock option accounts for senior management
  + ...

These problems could have happened to you, too

Basic Mechanisms

❖ Several techniques preserve the integrity of the data

  + Error detection/correction in the hardware
  + Passwords and authentication assist in verifying that the person(s) making changes are legitimate
  + Validation … verify that changes to the DB are “plausible”
  + “Commitment” … keep record in a safe place of all changes to the database, and then when it has been verified, make the actual change effective; deletions never actually result in the data being removed

Backup copies of a DB must be made regularly, and kept off-site

Do you back up your information?
Redundancy is Very Very Very Good

- To protect against computer crashes, disk failures, loss of power, etc. duplicate the hardware, disks, power sources, etc.
- The duplicate systems can compare answers as a means of detecting errors
- RAID systems are arrays of disks that contain “hot spares” and special data encodings to recover from disk failures
- By keeping a snapshot of the database and a record of all of the transactions, it is possible in case of catastrophic disaster to reconstruct the database by applying all of the transactions to the old database