Gran Hopper Week ~ Dec. 6 – 12

Whereas Grace Murray Hopper, one of the first females in the field of computer science, engineered new programming languages (COBOL) and pioneered standards for computer systems which laid the foundation for many advancements in computer science; and Whereas the week of December 7, in honor of Grace Hopper's birthday, is designated as National Computer Science Education Week.

It's followed by some appropriate 'be it resolved' points about encouraging education and opportunities for females and underrepresented minorities in the field.
Rear Admiral Grace Hopper

- Computing pioneer (1906-1992)
- Invented the compiler and COBOL
Announcements

● This week
  ● Lab 11 and Project 3A due Wednesday at 10pm
  ● Best GoPosters survey due Friday at 10pm

● Finals week
  ● Lab 12 and Project 3B due Wednesday, Dec. 16 at 10pm
Announcements

- No final for this class
Announcements

● Read ch 17 for today
● Read the second half of ch 13 for Lab 12
Announcements

- Quiz this week in lab
  - Wednesday and Thursday
- Topics
  - Spreadsheets
  - XML
  - Privacy
  - Cookies
  - Identity Theft
● This week's GoPost discussion is posted
Dijkstra: Whether a computer can think is about as interesting as whether a submarine can swim.
Project 3

● Part A
  ● Build database tables
    ● Brief sample data
  ● Answer queries
    ● Get information out of the database
Part B

- Imagine a year has gone by.
  - The database has grown
  - *Download* the much-larger database
- Boat Club members had some accidents
Announcements

- **Project 3B**
  - You're called in to extend the database to make sure sailors are qualified to run the boats they rent
  - Build the queries
  - Answer the questions
    - Save database and queries, and upload to Collect It
    - Submit WebQ
**Important!**

- querySailorAge
  - current age:
    DateDiff("yyyy",[tableSailors]![birthdate],Now())
A Table with a View

Primary keys, normalization, and SQL

D.A. Clements
Video

- Relational databases and tables
Video

- Primary Keys (5 min.)
Primary/Foreign Key

- Controlled redundancy:
  - Stores relationship between tables
  - Database tables share common attributes only to enable the tables to be linked
  - True redundancy exists only when there is unnecessary duplication of attribute values
Problem Fields (Don’ts)

- **Calculated field** – can be computed by mathematical calculation or text concatenation
  - Waste of storage space (redundant),
  - No assurance the calculated value is updated when the user changes the input field(s)

- **Multipart field** – contains that should be two or more fields
  - Extra work when you want to analyze your data

- **Multivalue field** – multiple correct entries for the field
  - Create a separate subset table with each value in its own record.

- **Derived field** – contents of one or more fields absolutely predicts the contents of another
  - Should be dropped from the table
Redundancy and Normalization
(5 min.)
Entities

- Entity
  - Anything that can be identified by a fixed number of its characteristics (*attributes*)

- Attributes have
  - Names—field name, attribute, or column name
  - Values—the data stored in the table
Entities

- An entity defines a table
  - Name of the entity is the name of the table
  - Each attribute of that entity
    - The column heading is the attribute name
## Island table

<table>
<thead>
<tr>
<th>Island</th>
<th>Name</th>
<th>Area</th>
<th>Elevation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Isabela</td>
<td>4588</td>
<td>1707</td>
</tr>
<tr>
<td></td>
<td>Fernandina</td>
<td>642</td>
<td>1494</td>
</tr>
<tr>
<td></td>
<td>Tower</td>
<td>14</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>Santa Cruz</td>
<td>986</td>
<td>846</td>
</tr>
</tbody>
</table>

**Figure 16.4** A table instance for the island entity.
Properties of Entities

- A relational database table can be empty
- Instances Are Unordered
  - Order of the rows and columns does not matter in databases
  - Freedom to move the data is limited to exchanging entire rows or exchanging entire columns
Properties of Entities (cont'd)

- **Uniqueness**
  - No two rows can be the same
  - Two rows can have the same value for some attributes, just not all attributes
Properties Of Entities (cont'd)

● Atomic Data
  ● Not decomposable into any smaller parts
    ● Separate fields for street, city, state, postal code
  ● "Only atomic data" rule relaxed for certain types of data
    ● Dates, times, currency
Database schemas

- **Database schema** – way to define a table
  - Collection of table definitions that gives the name of the table, lists the attributes and their data types, and identifies the primary key

---

**Island**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iName</td>
<td>Text</td>
<td>Island Name</td>
</tr>
<tr>
<td>area</td>
<td>Number</td>
<td>Area in square kilometers</td>
</tr>
<tr>
<td>elevation</td>
<td>Number</td>
<td>Highest point on the island</td>
</tr>
</tbody>
</table>

Primary Key: iName

**Figure 16.5** Database table definition for an Island table.
Database Tables Summary

- Tables in databases have a structure that is specified by metadata.
- The structure is separate from its content.
- A table structures a set of entities.
  - Things that we can tell apart by their attributes.
- The entities of the table are represented as rows.
  - Rows and columns are unordered.
- Tables and fields should have names that describe their contents.
  - Fields must be atomic (indivisible).
  - One of more attributes define the primary key.
TABLE OPERATIONS
Operations on Tables

- A database is a collection of tables
- Main use of database is to look up information
  - Users specify what they want to know and the database software finds it
- We can perform operations on tables to produce new tables
- The questions we ask of a database are answered with a whole new table, or view
### Nations

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Capital</th>
<th>Lat</th>
<th>NS</th>
<th>Lon</th>
<th>EW</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ireland</td>
<td>IE</td>
<td>Dublin</td>
<td>52</td>
<td>N</td>
<td>7</td>
<td>W</td>
<td>History</td>
</tr>
<tr>
<td>Israel</td>
<td>IR</td>
<td>Jerusalem</td>
<td>32</td>
<td>N</td>
<td>35</td>
<td>E</td>
<td>History</td>
</tr>
<tr>
<td>Italy</td>
<td>IT</td>
<td>Rome</td>
<td>42</td>
<td>N</td>
<td>12</td>
<td>E</td>
<td>Art</td>
</tr>
<tr>
<td>Jamaica</td>
<td>JM</td>
<td>Kingston</td>
<td>18</td>
<td>N</td>
<td>77</td>
<td>W</td>
<td>Beach</td>
</tr>
<tr>
<td>Japan</td>
<td>JP</td>
<td>Tokyo</td>
<td>35</td>
<td>N</td>
<td>143</td>
<td>E</td>
<td>Kabuki</td>
</tr>
</tbody>
</table>

**Primary Key:** Name

---

**Figure 16.6** The Nations table definition and sample entries.
SELECT Operation

- Takes rows from one table to create a new table
  - Specify the table from which rows are to be taken, and the test for selection

  Syntax: `SELECT Test FROM Table`

- Test is applied to each rows of the table to determine if it should be included in result table
- Test uses attribute names, constants, and relational operators
- If the test is true for a given row, the row is included in the result table; otherwise it is ignored

  SELECT Interest='Beach' FROM Nations
<table>
<thead>
<tr>
<th>Name</th>
<th>Dom</th>
<th>Capital</th>
<th>Lat</th>
<th>NS</th>
<th>Lon</th>
<th>EW</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia</td>
<td>AU</td>
<td>Canberra</td>
<td>37</td>
<td>S</td>
<td>148</td>
<td>E</td>
<td>Beach</td>
</tr>
<tr>
<td>Bahamas</td>
<td>BS</td>
<td>Nassau</td>
<td>25</td>
<td>N</td>
<td>78</td>
<td>W</td>
<td>Beach</td>
</tr>
<tr>
<td>Barbados</td>
<td>BB</td>
<td>Bridgetown</td>
<td>13</td>
<td>N</td>
<td>59</td>
<td>W</td>
<td>Beach</td>
</tr>
<tr>
<td>Belize</td>
<td>BZ</td>
<td>Belmopan</td>
<td>17</td>
<td>N</td>
<td>89</td>
<td>W</td>
<td>Beach</td>
</tr>
<tr>
<td>Bermuda</td>
<td>BM</td>
<td>Hamilton</td>
<td>32</td>
<td>N</td>
<td>64</td>
<td>W</td>
<td>Beach</td>
</tr>
</tbody>
</table>

**Figure 16.7** Part of the table created by selecting countries with a Test for Interest equal to Beach.
A natural join
Physical and Logical Database

TABLES AND VIEWS
Structure of a Database

- Physical database and logical database
  - Physical database is the files, records in any order, no logical organization other than tables
  - Logical database is a view of database that shows only the rows and fields needed by the users
- Solves Information Overload:
  - Users see only what they need
  - Users see only what they have permission to see
Physical vs. Logical

Figure 16.15 Structure of a database system. The physical database is the permanent repository of the data; the logical database, or view of the database, is the form of the database the users see. The transformation is implemented by the query processor, and is based on queries that define the logical database tables from the physical database tables.
Physical Database

- Designed by database administrators
  - Fast to access
  - No redundancy/duplicating information
    - Multiple data can lead to inconsistent data
  - Backup copies in case of accidental data deletion or disk crash
Logical Database

- Creating specialized views of the data for different users' needs
  - Creating a new “result set” from the current data each time
    - Fresh
    - Accurate
Defining Physical Tables

- Database schemes (schema)
  - Metadata specification that describes the database design
Figure 16.16  Table declarations from Microsoft Access 2007: (a) Home_Base table declaration shown in the design view; and (b) students table declaration. Notice that the key is specified by the tiny key next to Student_ID in the first column.
A relationship is a correspondence between rows of one table and the rows of another table.

Because the key `Student_ID` is used in each table, we can not only find the address for each student (`Lives_At`), but we can also find the student for each address (`Home_Of`).

Relationship examples
Figure 16.17 The *Relationships* window from the Microsoft Access database system; the 1-to-1 *Lives_At* and *Home_Of* relationships are shown between *Home_Base* and *Students*. 
Defining Logical Tables

- Constructing a View Using \textit{Join}
  - Match on the common field of \texttt{Student\_ID}

\begin{verbatim}
Master\_List = Student JOIN Home\_Base
  On Student.Student\_ID = Home\_Base.Student\_ID
\end{verbatim}

\begin{tabular}{l}
  \texttt{Student\_ID} \\
  \texttt{First\_Name} \\
  \texttt{Middle\_Name} \\
  \texttt{Last\_Name} \\
  \texttt{Birthdate} \\
  \texttt{On\_Probation} \\
  \texttt{Street\_Address} \\
  \texttt{City} \\
  \texttt{State} \\
  \texttt{Country} \\
  \texttt{Postal\_Code}
\end{tabular}

\textbf{Figure 16.18} Attributes of the \texttt{Master\_List} table. Being created from \texttt{Student} and \texttt{Home\_Base} allows \texttt{Master\_List} to inherit its data types and key (\texttt{Student\_ID}) from the component tables.