Announcements

• Quiz will cover chapter 16 in Fluency
  * Nothing in QuickStart
• Read Chapter 17 for Wednesday
• Project 3
  * 3A due Friday before 11pm
  * 3B due Monday, March 17 before 11pm
A Table with a View (continued)

Primary keys, normalization, and SQL
Video

- Primary Keys
Fields (Attributes) and Primary Keys

• Primary Key (PK)
  * Field or attribute that uniquely identifies each entity (row)
### An example of a simple relational database

**Table name: PRODUCT**
Primary key: PROD_CODE
Foreign key: VEND_CODE

<table>
<thead>
<tr>
<th>PROD_CODE</th>
<th>PROD_DESCRIP</th>
<th>PROD_PRICE</th>
<th>PROD_ON_HAND</th>
<th>VEND_CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>001276-AB</td>
<td>Claw hammer</td>
<td>$12.95</td>
<td>23</td>
<td>232</td>
</tr>
<tr>
<td>103-21UUY</td>
<td>Huselitie chain saw, 16-in. bar</td>
<td>$189.99</td>
<td>4</td>
<td>235</td>
</tr>
<tr>
<td>QER-3425S</td>
<td>Sledge hammer, 16-lb. head</td>
<td>$18.63</td>
<td>6</td>
<td>231</td>
</tr>
<tr>
<td>SRE-657UC</td>
<td>Rat-tail filo</td>
<td>$2.00</td>
<td>15</td>
<td>232</td>
</tr>
<tr>
<td>ZZX3245Q</td>
<td>Steel tape, 12-ft. length</td>
<td>$6.79</td>
<td>8</td>
<td>235</td>
</tr>
</tbody>
</table>

**Database name:** Ch03_SaleCo

**Table name: VENDOR**
Primary key: VEND_CODE
Foreign key: none

<table>
<thead>
<tr>
<th>VEND_CODE</th>
<th>VEND_CONTACT</th>
<th>VEND_AREA_CODE</th>
<th>VEND_PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>230</td>
<td>Shelly K. Smithson</td>
<td>608</td>
<td>655-1234</td>
</tr>
<tr>
<td>231</td>
<td>James Johnson</td>
<td>615</td>
<td>123-4536</td>
</tr>
<tr>
<td>232</td>
<td>Annelise Crystall</td>
<td>508</td>
<td>224-2134</td>
</tr>
<tr>
<td>233</td>
<td>Candice Wallace</td>
<td>904</td>
<td>342-6567</td>
</tr>
<tr>
<td>234</td>
<td>Arthur Jones</td>
<td>615</td>
<td>123-3324</td>
</tr>
<tr>
<td>235</td>
<td>Henry Ortozo</td>
<td>615</td>
<td>899-3425</td>
</tr>
</tbody>
</table>
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
Video

- Redundancy and Normalization
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
• 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
<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 schemes

- 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

<table>
<thead>
<tr>
<th>Island</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>iName</strong></td>
</tr>
<tr>
<td><strong>area</strong></td>
</tr>
<tr>
<td><strong>elevation</strong></td>
</tr>
</tbody>
</table>

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

**Primary Key: iName**

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

- 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 tables
• The questions we ask of a database are answered with a whole new table, or view
Operations on Tables

- Five fundamental operations can be performed on tables:
  - Select
  - Project
  - Union
  - Difference
  - Product

- Join
### 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 row 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.
Join Operation

• Combines two tables, like the Product Operation, but doesn't necessarily produce all pairings

  * If the two tables each have fields with a common data type, the new table combines only the rows from the given tables that match on the fields

  * Syntax: \texttt{Table1} \texttt{Table2 On Match}
Animation

- 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:
  - Show users only what they need 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
Queries

• A query is a specification using the five operations and Join that create a view from other tables

• SQL (Structured Query Language)
  * Standard database language to write queries
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.
Connecting Database Tables by Relationships

- **Student and Home_Base tables**
  - The tables can have different security access restrictions based on their data
  - Other units can access Home_Base data without having access to more sensitive data in Student

- **Separate tables but not independent**
  - `Student_ID` connects (establishes a relationship) the two tables
    - Primary key in one, foreign key in the other
The Idea of Relationship

• 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, can not only find the address for each student (`Lives_At`), but 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 Join

* Match on the common field of Student_ID

Master_List = Student JOIN Home_Base
On Student.Student_ID = Home_Base.Student_ID

| Student_ID |
| First_Name |
| Middle_Name |
| Last_Name |
| Birthdate |
| On_Probation |
| Street_Address |
| City |
| State |
| Country |
| Postal_Code |

**Figure 16.18** Attributes of the Master_List table. Being created from Student and Home_Base allows Master_List to inherit its data types and key (Student_ID) from the component tables.
Practical Construction
Using QBE

• Query By Example
  * Given a template of a table we fill in what we want in the fields
**Figure 16.19** The Query By Example definition of the Master List table from MS Access.

**Figure 16.20** SQL query created from the Query By Example data in Figure 16.19.
• Storing the Dean's Data
  * Top_Scholar is information of interest only to the dean

**Figure 16.21** The Top_Scholar definition: (a) informal form, (b) in MS Access.
Creating a Dean's View

<table>
<thead>
<tr>
<th>Name</th>
<th>Source Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nickname</td>
<td>Top_Scholar</td>
<td>Used by the dean to seem &quot;chummy&quot;</td>
</tr>
<tr>
<td>First_Name</td>
<td>Student</td>
<td>Name information required because the dean forgets the person's actual name, being so chummy</td>
</tr>
<tr>
<td>Last_Name</td>
<td>Student</td>
<td></td>
</tr>
<tr>
<td>Birthdate</td>
<td>Student</td>
<td>Is student of &quot;drinking age&quot;?</td>
</tr>
<tr>
<td>City</td>
<td>Home_Base</td>
<td>Hometown (given by city, state) is important for small talk, but full address not needed by dean</td>
</tr>
<tr>
<td>State</td>
<td>Home_Base</td>
<td></td>
</tr>
<tr>
<td>Major</td>
<td>Student</td>
<td>Indicates what the student’s doing in college besides hanging out</td>
</tr>
<tr>
<td>GPA</td>
<td>Student</td>
<td>How's student doing grade-wise</td>
</tr>
<tr>
<td>Factoid</td>
<td>Top_Scholar</td>
<td>Data to remember student by</td>
</tr>
<tr>
<td>Summer_Plans</td>
<td>Top_Scholar</td>
<td>Or other conversation topic</td>
</tr>
</tbody>
</table>

*Figure 16.22* The Dean’s View fields showing their source in physical database tables.
Join Three Tables into One

• Join using **Top_Scholar**, **Student**, and **Home_Base** tables matching on the **Student_ID** attribute across all three tables

• Trim the Table
  * **Project** - retrieve certain columns

• Join-then-trim strategy
Software Creates Dean's View

Figure 16.23 The Query By Example definition of the Dean’s View table as expressed in Microsoft Access 2007.

Figure 16.24 SQL query created for the Dean’s View by the Query By Example data in Figure 16.22.
Structured Query Language

SQL
**SELECT**

- **SELECT** * FROM tablename;
  * Selects all fields from the table
- **SELECT** first_name, last_name, GPA
  FROM Students
  **WHERE** Student_ID = 0344567;
  * Selects first and last names, GPA for the student with ID of 0344567

<table>
<thead>
<tr>
<th>KEYWORD</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>Identifies columns to be displayed</td>
</tr>
<tr>
<td>FROM</td>
<td>Identifies tables hold the needed data</td>
</tr>
<tr>
<td>WHERE</td>
<td>Limits the number of rows to be returned</td>
</tr>
</tbody>
</table>
SELECT Examples

• `SELECT FName, LName FROM Student WHERE Major = "INFO";`
SELECT Examples

- Select records with or without empty fields
  
  * `SELECT LName
     FROM Student
     WHERE FName IS NULL;`
  
  * `SELECT LName
     FROM Student
     WHERE FName IS NOT NULL;`
SELECT Examples

• Sorting query results
  * `SELECT StudentID, LName FROM Student ORDER BY LName DESC;`
    • Descending order Z-A, 9-0
  * `SELECT StudentID, LName FROM Student ORDER BY LName ASC;`
    • Ascending order A-Z, 0-9
JOIN Examples

• **SELECT** Student.FName, Student.LName, Advisor.LName  
  **FROM** Student **INNER JOIN** Advisor  
  **ON** Student.AdvisorID = Advisor.AdvisorID;

  * Joins the Student and Advisor tables and displays first and last names of all students and each student’s advisor by last name.
Advanced Filtering

- Other ways to reduce the number of rows:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>=</td>
</tr>
<tr>
<td>Not equal</td>
<td>&lt;&gt;</td>
</tr>
<tr>
<td>Greater than</td>
<td>&gt;</td>
</tr>
<tr>
<td>Less than</td>
<td>&lt;</td>
</tr>
<tr>
<td>Greater than or equal to</td>
<td>&gt;=</td>
</tr>
<tr>
<td>Less than or equal to</td>
<td>&lt;=</td>
</tr>
</tbody>
</table>
Advanced Filtering Examples

- SELECT FName, LName
  FROM Advisor
  WHERE HireDate >= 1987;

- SELECT FName, LName
  FROM Student
  WHERE AdvisorID = 44232 AND Major = "INFO";
Aggregate Functions

- **sum, avg, max, min, count, etc.**
- **W3 Schools: SQL Tutorial**
  * Search for Aggregate Functions
  * [http://www.w3schools.com/sql/sql_groupby.asp](http://www.w3schools.com/sql/sql_groupby.asp)
Aggregate Functions

• What is the total number of INFO majors?
  * `SELECT COUNT(Student_ID) FROM Student WHERE Major = "INFO";`

• What is the average GPA of INFO majors?
  * `SELECT AVG(Grade) FROM Student WHERE Major = "INFO";`