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)

Keys – Primary & Foreign

- 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

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

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

Figure 16.4 A table instance for the island entity.
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

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>
<th>Island Name</th>
<th>Area (in square kilometers)</th>
<th>Elevation (highest point on the island)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isname</td>
<td>Text</td>
<td>Integer</td>
<td>Integer</td>
</tr>
<tr>
<td>area</td>
<td>Number</td>
<td>Number</td>
<td>Number</td>
</tr>
<tr>
<td>elevation</td>
<td>Number</td>
<td>Number</td>
<td>Number</td>
</tr>
</tbody>
</table>

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 defines the primary key

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

## 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

  ```sql
  SELECT Interest='Beach' FROM Nations
  ```

## 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: `Table1 x Table2 On Match`

Animation

- A natural join
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 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

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)

Relationships in Practice

- 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

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
Practical Construction Using QBE

• Query By Example
  * Given a template of a table we fill in what we want in the fields

The Dean's View

• Storing the Dean's Data
  * Top_Scholar is information of interest only to the dean

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

Creating a Dean's View

• Join Three Tables into One

Software Creates Dean's View
Structured Query Language

SQL

**SELECT**

- **SELECT** FROM tablename;
  - Selects all fields from the table
- **SELECT** first_name, last_name, GPA FROM Students
  - 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** FName FROM Student WHERE FName IS NULL;
- **SELECT** FName FROM Student WHERE FName IS NOT NULL;

**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

**SELECT Examples**

- **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

- Sort query results
  - **SELECT** StudentID, LName FROM Student ORDER BY LName DESC;

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

• `SELECT COUNT(Student ID)
  FROM Student
  WHERE Major = "INFO";`

• `SELECT AVG(Grade)
  FROM Student
  WHERE Major = "INFO";`