Today’s Topics

- Relational Databases
  - Reasons to have multiple tables
  - Relationships
- Normalization
  - Reasons
- Simple Queries

Thanks to Mary Jo Davidson and Curt White for much of the materials in this presentation!

A Relational Database

- Has multiple tables
- Has relationships defined between pairs of tables as follows:
  - One table has a **primary key (PK)** field defined
  - The other table has a **foreign key (FK)** field defined
  - Database associates records in these tables where PK=FK
### Primary Key
- Contents of this field are used to uniquely identify each record within a table
- You can choose a “natural key” – a piece of info that already exists in the data and is unique
  - Or
- Create a field named “ID” and set it to data type auto-number so that Access ensures it always has a unique value

### Foreign Key
- A field on one table that links to a field on another table...so the tables are related
- Foreign key values are entered by the user (not automatic), who must keep track of correct corresponding key
- The relationship between tables/keys is set up by the user via Access command

### Why Multiple Tables?
- Eliminates redundancy
- Saves storage space
- Makes adding data easier
- Allows for more secure access to only parts of the data
  - You can give users access to one table, but not another related table
Relationships allow us to

- Store data once (reduce errors)
- Reference data via other related data (save time)
- Use the data in a variety of ways (versatility)

Critical Point for relationships in Access

The data type of the primary and foreign key fields (in Access) must “coordinate”

- These coordinate
  - Number and auto-number
  - Number and number
  - Text and text
- These do not
  - Number and text

Database Design

- First step to creating a relational database – the creator (that’s you) should write down on paper:
  - Names of all the data (fields) that are to be stored in the database
  - Determine relationships (1:1, 1:M, M:M) between different sets of data
  - Split data into tables to achieve correct
Relationship Cardinality

- If two data entities, A and B, in a database are related in \(1:1\) Cardinality it means that for every item A, there is exactly one item B
  - Examples:
    - First Name and Last Name
    - Street Address and Zip Code

Relationship Cardinality

- If two data entities, A and B, in a database are related in \(1:M\) Cardinality (one-to-many) it means that for every item A, there can be multiple items B, but for every B there is only one A
  - Examples:
    - Person and Phone Numbers
    - Customer and Sales Transactions
    - City and Street Addresses

Relationship Cardinality

- If two data entities, A and B, in a database are related in \(M:M\) Cardinality (many-to-many) it means that for every item A, there multiple items B and for every B there can be multiple items A
  - Examples:
    - Students and Courses
    - Purchase Invoices and Product IDs
Normalization

- In a First Normal Form database, we
  - Eliminate repeating groups in our design that can occur due to 1:M or M:M relationships
  - Create separate tables for the data that would be repeated
  - Relate tables via a foreign key in repeating table

Normalization Example

Car Club (originally)
- Member ID (primary key)
- Member Name
- Member Address
- Member City
- Member State
- Member Zip
- Member Phone
- Dues Paid?
- National Member?
- these fields repeat 1 to n times
- Model of Car
- Make of Car
- Year of Car

Member Table
- Member ID (primary key)
- Name
- Address
- City
- State
- Zip
- Phone
- Dues Paid?
- National Member?
- Member Table
- Car Table
- Relationship
- Primary Key in Member Table?
- Foreign Key in Car Table?
- 1:M

Another Example

Student Records
- Student ID (primary key)
- Name
- Address
- City
- State
- Zip
- Phone
- these fields repeat 1 to n times
- Class Name
- Grade
- Number Credits
- Major
- Minor
- Degree Sought

Student Info
- Student ID
- Name
- Address
- City
- State
- Zip
- Phone
- Major
- Minor
- Degree Sought

Grades
- Student ID
- Class Name
- Grade
- Number Credits
- Student ID

Before

After
Steps to Design
in First Normal Form

- Review your design
- Identify multiple-occurrence situations
- Update your design
  - For each multiple occurrence situation - add a table that will contain one record for each occurrence of that multiple-occurrence situation
- Enter new table(s) to reflect your new design
- Establish relationships between tables

3 Reasons to Add Tables

- 1:M relationship
  - Solution: create separate table for repeated values
- M:M relationship
  - Solution: create intermediate table that links to both data tables (see example next)
- Multiple repeated values in a single column
  - Solution: create a Lookup Table that contains a single copy of each possible value and link it back to original data table (see example coming up)

M:M Relationship

- Many-to-many relationship
- Many entities are related to many entities and the reverse is also true
e.g.
- a Zoo animal could eat many types of food
  - and
- A food could be eaten by many different zoo animals
M:M Relationships

- You need an intermediary table between the two tables that would have a M:M relationship

Add a table between the existing tables

Add this table

Animals

<table>
<thead>
<tr>
<th>ID</th>
<th>Animal_ID</th>
<th>Food_ID</th>
</tr>
</thead>
</table>

Foods

| ID | ID |

Resident_Food

Add relationships between the tables

Resident

<table>
<thead>
<tr>
<th>ID</th>
<th>Resident_ID</th>
<th>Food_ID</th>
</tr>
</thead>
</table>

Food

| ID | ID |

Resident_Food
When the M:M relationship is supported......

- You can
  - Find all of the foods that each animal eats
  - AND
  - Find all of the animals that eat each food

Lookup Tables

- If your database has frequently repeated values in a single column,
- Then you can save storage space and reduce data entry time by entering each possible value once in a separate Lookup table.
- Store a primary key for each value in Lookup table.
- Replace old column in original table with a foreign key entry.

Lookup Table Example

<table>
<thead>
<tr>
<th>All Songs</th>
<th>Artist</th>
<th>Genre</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Elvis Costello</td>
<td>Alternative</td>
</tr>
<tr>
<td>Less Than Zero</td>
<td>Elvis Costello</td>
<td>Alternative</td>
</tr>
<tr>
<td>Let Them All Talk</td>
<td>Elvis Costello</td>
<td>Alternative</td>
</tr>
<tr>
<td>Wonderful Tonight</td>
<td>Eric Clapton</td>
<td>Rock</td>
</tr>
<tr>
<td>She Doesn't Care</td>
<td>Fine Young Cannibals</td>
<td>Rock</td>
</tr>
<tr>
<td>Don't Look Back</td>
<td>Fine Young Cannibals</td>
<td>Rock</td>
</tr>
</tbody>
</table>
Lookup Table Example

With Lookup Table...

<table>
<thead>
<tr>
<th>All Songs</th>
<th>Artist</th>
<th>Genre Lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alison</td>
<td>Elvis Costello</td>
<td>1</td>
</tr>
<tr>
<td>Less Than Zero</td>
<td>Elvis Costello</td>
<td>1</td>
</tr>
<tr>
<td>Let Them All Talk</td>
<td>Elvis Costello</td>
<td>1</td>
</tr>
<tr>
<td>Wonderful Tonight</td>
<td>Eric Clapton</td>
<td>2</td>
</tr>
<tr>
<td>She Drives Me Crazy</td>
<td>Fine Young Cannibals</td>
<td>2</td>
</tr>
<tr>
<td>Don't Look Back</td>
<td>Fine Young Cannibals</td>
<td>2</td>
</tr>
</tbody>
</table>

Relationship

Genre Lookup Table

<table>
<thead>
<tr>
<th>Genres</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td>2</td>
</tr>
<tr>
<td>Alternative</td>
<td>1</td>
</tr>
</tbody>
</table>

Seting up the Relationships in Access

Database Tools - Relationships
**Show Table – Add all of them**

**Close Show Table window**

**Edit Relationships | Create New**
Link Foreign Key to Primary Key

Create

Enforce Referential Integrity
Referential Integrity

- In order for a child record to be created, the parent record must already exist.
- A parent record cannot be deleted if it has child records.

Relationship is Set Up

We can now

- View all instructors and the classes each one teaches and vice versa.
- Use instructor info to query class_taught info.
- Develop forms to support data input for both tables combined.
Access Automatic Relationships

- If:
  - You create two tables, and
  - The first table has a primary key, and
  - The second table has a column with the same name and data type as the primary key in the first table
- Then: Access will automatically create a relationship between the two tables

Query

- Queries are Access “objects” that allow you to ask questions about the content of your database
- In Design View they allow you to specify any type of sort/filter over any combination of tables in your database
- In Datasheet View they create a structure that can be used like a Table, containing a result of the query

SQL – Structured Query Language

- SQL is an industry-standard textual command language used to build Queries.
- Access creates SQL commands behind the scenes when you create Queries.
- We won’t study SQL in this course.
Queries

- Find and display the data you specify
- Display only chosen fields
  - and/or
- Filter records that meet one or more criteria

Create Query

Creating Queries in Access

- Choose tables to be included
- Select fields to be displayed (and ordering of fields in results)
- Choose sort field, if desired
- In “Criteria” field, type filtering conditions for records to be displayed
### Specifying Criteria

- Criteria specifies conditions that must be met for included records
- If left blank, then all records are included
- Simplest criteria: just type a single value to match:
  - Example: **Criteria:** “Led Zeppelin” in Artist column matches all songs by Led Zeppelin.

### Specifying Criteria

- When typing text in criteria, put quotes ("") around them
  - Example: **Criteria:** “Led Zeppelin” matches that text.
- When typing dates, put “#” around them
  - Example: **Criteria:** #1/1/2009# matches that date

### Specifying Criteria

- You can also put relational operators before any value
  - Criteria: <10
    - Matches any number less than 10
  - Criteria: <"G"
    - Matches text that begins with A-F
  - Criteria: >#1/1/2009#
    - Matches dates before January 1, 2009
### Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or Equal to</td>
</tr>
<tr>
<td>=&gt;</td>
<td>Greater than or Equal to</td>
</tr>
</tbody>
</table>

### Specifying Criteria

- You can include wildcards in text matches:
  - `*` matches any number of any character
  - `G*` matches any word that begins with 'G'.
  - `?` matches any single character
  - `a?c` matches "abc" or "aac" or "aqc", etc.
  - 
  - `[...]` match any character within braces

### More on Queries

- **Logical OR**
  - choose based on either value x or value y
  - e.g. look for records in the state of Indiana or Illinois
    - Criteria: "IL" OR "IN"

- **Logical AND**
  - choose based on both value x and value y
  - e.g. look for records where state = "IL" and city = "Springfield" by keying
    - Criteria: "IL" in State column and "Springfield" in City column
Other Operators

- Between...and
  - **Criteria:** Between 1 and 5 matches any number between 1 and 5
- In (list)
  - **Criteria:** In ("IN", "IL", "WI") matches either "IN" or "IL" or "WI"
- Is Null
  - **Criteria:** Is Null matches empty elements

Let’s do some examples