Windows Database Using Related Tables

This chapter examines techniques for establishing relationships among tables and extracting data from the tables.

Objectives (1 of 2)

- Explain the types of table relationships
- Display master-detail records
- Display a field from a second table using a lookup
- Create a search using a parameterized query and write a filter to retrieve specific data
- Assign data values to unbound controls
Objectives (2 of 2)

- Retrieve and display the parent row for a selected child row
- Retrieve and display an array of child rows for a selected parent row
- Create an application that displays from a many-to-many relationship
- Select the correct locations for handling and formatting data in a multitier application

Data Relationships

- Data stored in multiple related tables
- Primary table ➔ parent or master
- Second table ➔ child or detail
- The primary key of one table is included as a field in a related table to link the two tables
- The key included in the related table is called a foreign key

One-to-Many Relationships

- Most common type of relationship
- One record in the parent table relates to one or more records in the child table
  - Examples include:
    - Customer with multiple orders
    - Student with multiple courses
Many-to-Many Relationships

- Example: One author can write many books; one book can have many authors
- SQL Server and Access require a junction table to join two tables
- The junction table’s primary key consists of the foreign keys from both tables

One-to-One Relationships

- Least common relationship
- One record in the parent matches one record in the child table
- Usually the two table could be combined but may be kept separate for security reasons or the child table contains short-term information

Constraints

- A unique constraint specifies that no duplicate entries are allowed in a column
- Foreign-key constraints ensure that parent and child tables remain synchronized when records are deleted or changed

The referential integrity concept is enforced by the DBMS using foreign key constraints to ensure that changes and deletions in one table are made to all affected tables.
Creating a Dataset with More Than One Table

- The DataSet object can hold more than one table and the relationships between the tables.
- To generate a DataSet with multiple tables, check the boxes for both tables in the Data Source Configuration Wizard.

Viewing or Setting a Relationship

- A DataRelation object describes the relationship between the tables.
- Use the XML schema (.xsd) file for the dataset to create the relationship.

Specifying the Parent, Child, and Foreign Key

- When adding a relationship to a DataSet’s schema, it's important to get the parent/child relationship right.
- Determine which table is the “one side” and which is the “many side” in a one-to-many relationship.
- The “one side” should always be the parent and the “many side” should be the child.
Queries and Filters

- A **parameterized query** creates a new DataSet that holds the selected records.
  - **Filter** selects records from an existing DataSet.
  - Use a filter when you already have a complete DataSet.

Using a Parameterized Query

- SQL statement that retrieves selected record(s) based on a value that the user enters at run time.
  - The value supplied by the user is called a parameter which is the basis for the term **parameterized query**.

Selecting Specific Records

- If a DataSet is to contain only selected record(s), the SQL SELECT used by the TableAdapter can be modified.
  - Use a WHERE clause in an SQL query to specify which records to select, called the **criteria**.

```
SELECT Title, Author, ISBN FROM Books
WHERE Title = 'Romeo and Juliette'

SELECT Name, AmountDue FROM OverdueAccounts
WHERE AmountDue > 100

SELECT emp_id, lname, fname FROM employee
WHERE lname = 'Jones'
```
**Query Builder**

Enter the parameter in the Filter Box and the Where clause is created. The grid fills with the selected records.

---

**Writing a Filter**

- Rules for creating a filter are the same as for a WHERE clause of an SQL statement.
- Specify the field, a comparison operator (usually the equals sign), and the value to match.
  
  - "LastName = Jones"
  - "SalesAmount = 1000"
  - "Quantity > 0"

- Use single quotes to enclose the value to match when a string contains spaces.
  
  - "Title = "A Great Book"

---

**Useful Comparison Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
<td>&quot;Subject = 'Business'&quot;&lt;br&gt;&quot;Subject = &quot; &amp; subjectTextBox.Text &amp; &quot;]&quot;</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>&quot;Sales &gt; 1000&quot;&lt;br&gt;&quot;Sales &gt; &quot; &amp; salesTextBox.Text</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>&quot;Sales &lt; 1000&quot;&lt;br&gt;&quot;Sales &lt; &quot; &amp; salesTextBox.Text</td>
</tr>
<tr>
<td>Like</td>
<td>Pattern match</td>
<td>&quot;Subject Like ('B%')&quot; (SQL Server)&lt;br&gt;&quot;Subject Like 'B%%'&quot; (Access)</td>
</tr>
</tbody>
</table>
Binding a List at Run Time

- List usually appears empty until a selection is made
- Bind the control in code rather than at design time
- When the DataSource is set, the list fills with the data from the BindingSource

```csharp
Bind the list box with the filtered data.
 With employeeListBox
   .DataSource = BindingSource1
   .DisplayMember = "Name"
 End With
```

Completed Filter Exercise

- User selects a job description from the combo box
- The corresponding job_id is used to filter the data
- All matching records display in the list box

Unbound Data Fields

- Controls that do not have data bindings are called unbound controls
- Assign a field of data at run time, in code
- Must reference individual database fields from the dataset
Referring to Records and Fields

- Actual data values are held in **DataRow objects**
- Each Table object in a DataSet has a DataRows collection made up of DataRow objects
- The data values are held in the DataRow.Items collection

---

Referring to Fields

- Reference the DataRow Items collection

```csharp
firstNameString = employeeDataRow.Item(1).ToString
```

---

Working With Parent and Child Records

- DataRelation objects relate parent and child records
- Use the DataRelation to retrieve the parent of a child record or find an array of matching child row(s) for a given parent
- Remember the rule—the one side of a one-to-many relation is the parent; the many side is the child
Retrieving a Related Parent Row

From the selected employee name, the program retrieves the correct employee DataRow and the matching Jobs parent DataRow to display the job description.

Find the Parent Row for a Child Record

- Find the row in the child table (employee) that matches the combo box selection
- Get the parent row from the parent (jobs) table
- Retrieve specific field from the parent data row

```
Dim employeeDataRow As DataRow
employeeIDString = employeeComboBox.SelectedValue.ToString
employeeDataRow = PubsDataSet.employee.FindByemp_id(employeeIDString)

Dim jobDataRow As DataRow
jobDataRow = employeeDataRow("EmployeeToJobsRelation")

jobTitleTextBox.Text = jobDataRow!job_desc.ToString
```

Retrieving Related Child Rows

- Retrieve an array of related child rows
- The GetChildRows method is similar to retrieving a related parent row
- The difference is that the GetChildRows method returns an array of rows instead of a single row

```
The Complete Find Parent Program – p. 170
The Complete Find Child Program – p. 171-172
```
Many-to-Many Relationships

In an M:N relationship, two 1:M relationships must be set up. The junction table is the child table in each of the two 1:M relationships.

Multitier Considerations

- Be sure to think through the goals of OOP
- The presentation tier should provide only the user interface
- All user input and output formatting belongs in the form
- Data retrieval and any processing should be handled in other classes
- The user interface should be able to completely change without modifying the other classes
- Filtering or data retrieval method changes should not affect the interface