1 **SQL Server, SQL and the SqlDataSource**

CST272—ASP.NET

4 **Creating a New SQL Server Database**

- To create a new SQL Server database, from the "Project" menu select Add New Item…
- Alternately:
  1. Right-click the website name in "Solution Explorer
  2. Select the Add command from the "shortcut" menu
  3. Select the command New Item… from the sub-menu

5 **Add a SQL Server Database (Page 1)**

- In the "Add New Item" dialog window:
  1. In the "Installed Templates" pane, accept the default selection "Visual C#"
  2. Select "SQL Server Database" as the item type
     - Clicking "Data" on the left will restrict the search
  3. Enter "Name" for new database, e.g. "Inventory.mdf"
  4. Click the <Add> button

7 **Add a SQL Server Database (Page 2)**

- In the "Add New Item" dialog window (*con*.):
  5. In response to "You are attempting to add a special file type (database) to an ASP.NET Web site. In general, to use this type of item in your site, you should place it in the 'App_Data' folder. Do you want to place the file in the 'App_Data' folder?" in the dialog box, click <Yes> button

9 **Server Explorer (Page 1)**

- SQL Server database objects appear in the "Server Explorer" window near the upper-left of the Visual Studio IDE (just above the ToolBox)
- To view "Server Explorer" hover the mouse over its tab and click—slides out from side of IDE window
  - Click again and the window slides back and "hides"

10 **Server Explorer (Page 2)**

- Server Explorer uses the "Auto Hide" feature which is enabled when "pushpin" icon is facing left
  - Click the icon to push the pin "down" and disable Auto Hide so the window always is visible
- Many database developers prefer to keep Server Explorer hidden providing a larger work area

12 **SQL Server Objects**

- To "drill down" and view the list of the SQL Server database objects (including Tables) click the pointer symbol (▷) in front of the database name

14 **Create a Table**

- To create a new table right-click "Tables" in the list of SQL Server object types and select the Add New Table command from the "shortcut" menu

16 **Designing a Table**

- To design a table enter the information about each field's configuration:
  - Column name—the name given to the field
  - Data type—an attribute that specifies what type of data the field can hold (see next slide); to update property, click its value which expands the drop-down list and choose a value
  - Allow nulls—identifies fields that can be left blank when a record is saved
  - Default—a default field value for new records

17 **SQL Server Data Types (Page 1)**

- Each field in a SQL Server table must have a data type, some of which are:
  - char—fixed length (ending blanks are part of the length) ASCII (one byte per character) string storage
    - Format: char(*n*)
    - *n* is the length of the string between 1 to 4000 characters
  - nchar—fixed length Unicode (two bytes per character) string storage
    - Format: nchar(*n*)

18 **SQL Server Data Types (Page 2)**

- Each field in a SQL Server table must have a data type, some of which are (*con.*):
  - varchar—variable length ASCII string storage
    - Format: varchar(*n*)
  - nvarchar—variable length Unicode string storage
    - Format: nvarchar(*n*)

19 **SQL Server Data Types (Page 3)**

- Each field in a SQL Server table must have a data type, some of which are (*con.*):
  - int—an integer in the range -2,147,483,648 to 2,147,483,647
  - decimal—represents number with *fixed* (maximum) number of digits and decimal positions
  - money—represents monetary or currency (really a "decimal" data type)
  - bit—used to represent Boolean values (only stores values one (1) = TRUE and zero (0) = FALSE)

20 **The "Create Table" Window**

- The "Create Table" window has two panes:
  1. The "Design Window" at the top which is a GUI pane for setting properties for each table column
  2. The "T-SQL" pane at the bottom which is where the SQL "CREATE TABLE" command is built

- Updates to "Design Window" automatically update the SQL code
- The developer may type SQL code in this pane
- The table name *must* be entered in this pane

23 **The Primary Key**

- The primary key is the field (or in some cases two or more fields joined together) that uniquely identify records in a table
- To create a primary key for a table:
  1. Right-click the column name in the "Table Design" window for the field to be designated the primary key
  2. Select the Set Primary Key command from the "shortcut" menu

25 **Save the Table**

- To save the table design:
  - Click <Update> button found under the table name
  - In the "Preview Database Updates" window click <Update Database> button

28 **Refreshing Objects**

- When new objects such as a new table is created in a SQL Server database, they initially are not visible in the Server Explorer window
- Right-click on the folder that contains the new object and select command "Refresh" from shortcut menu

31 **The Identity Specification (Page 1)**

- SQL Server lets the developer create integer fields that *automatically* generate numbers *in sequence*:
  1. Click the "row select" box in "Table Design" window for field to be specified as the "Identity Specification" (usually the primary key)
  2. Scroll down to the "Identity Specification" property in the "Properties" window and click the pointer symbol (▷) to drill down to its properties
  3. Set (Is Identity) to "True"

32 **The Identity Specification (Page 2)**

- SQL Server lets the developer create integer fields that *automatically* generate numbers *in sequence* (*con*):
  4. Set "Identity Increment" which is the amount by which the sequence increments for each new record
  5. Set "Identity Seed" which is the starting value for the first record

34 **Opening the Table Definition**

- To modify the definition of a table in a SQL Server database:
  - Click the pointer symbol (▷) in front of "Tables" in the list of database objects to drill down and view the table object names
  - Right-click the table name and select "Open Table Definition" (or double-click the table name)

36 **Concatenated Primary Keys**

- Sometimes it take a combination of more that one field to uniquely identify each record in a table
- Known as a concatenated primary key
- To create the concatenated primary key:
  1. In the "Primary Key" column just left of the column Name, click and hold the mouse button to drag and select all fields that make up the primary key
  2. Right-click and select Set Primary Key command from the "shortcut" menu
  –
-

38 **Relationships (Page 1)**

- Relationships are the association between entities which must be enforced in the database
- In a one-to-many relationship between two tables, *one* entity in one table is related to *several* entities in another table (foreign key to a primary)

39 **Relationships (Page 2)**

- Example of the preceding:
  - No two records may have a duplicate value for the SupplierID (the primary key) in the "Supplier" table (the *one*-part of the relationship)
  - However in the "Product" table there may be multiple (the *many*-part of the relationship) identical values for the SupplierID (also known as the foreign key)
    - But the values in the "Product" table must match one of the SupplierID values in the "Supplier" table

40 **Relationships (Page 3)**

- To create and enforce relationships, always enter the "Relationship" function from the table that contains the foreign key (the *many*-part of the relationship)
  - In the example of Supplier ID's, this is the "Product" table, not the "Supplier" table
- The data type and field size of columns from both tables in the relationship must be the same
  - E.g. if the field in one table is data type int, the field in the other table must be the same

41 **Relationships (Page 4)**

- To create and enforce relationships (always create the relationships between tables before entering any data):
  1. In "Table Object" list right-click Foreign Keys
  2. Select the Add New Foreign Key command from the "shortcut" menu

43 **Relationships (Page 5)**

- To create and enforce relationships (*con*.)
  3. Scroll down in the "T-SQL" window to find the SQL code for the new foreign key

CONSTRAINT

4. Begin to update the constraint by giving it a name
   - A convention that often is employed uses the prefix "FK" followed by the name the two tables involved in the foreign key relationship

45 ☐ **Relationships (Page 6)**

- Create and enforce relationships (*con*.)

5. Following the FOREIGN KEY entry, type the name of the foreign key "ColumnName" from the list of column names in the table

47 ☐ **Relationships (Page 7)**

- Create and enforce relationships (*con*.)

6. Following the SPECIFICATION entry, type the name of the "ToTable" which is the table to which the foreign key links
7. Then key in the "ToTableColumn" which is the name of the primary key in the "ToTable"

49 ☐ **Re-Save the Tables Effected**

- Every time the design of a table is modified (including creating relationships) it must be saved
- Click <Update> button on the "Database" toolbar and then click <Update Database> button in the "Preview Database Updates" window

50 ☐ **Entering Table Data**

- To enter data into a SQL Server database table:

1. (If necessary) click pointer symbol (▷) in front of "Tables" in the list of database objects in "Server Explorer" to drill down and view the table object names
2. Right-click the table name and select the command Show Table Data from the shortcut menu
3. Key the data (click to another line to commit records)
4. Close the table; data is committed (saved) to the table automatically when you move to a new record, so it does not need to be saved manually

54 ☐ **Structured Query Language**

- SQL (pronounced see'-quel)—a relational database language developed by IBM in mid-1970's
- Represents any data as one or more tables
- Non-procedural language—lets DBMS (database management system) determine how the operation is executed
- Commands can be embedded into procedural language programs

55 ☐ **Structure Modification in SQL**

- Command line utilities used to create and modify the *structure* of relational database tables
- CREATE TABLE statement ...

  - Creates the structure for a new table in database
- ALTER TABLE statement …
  - Modifies the structure of one or more columns (fields) in an existing table
  - ALTER statements include ADD, DELETE and MODIFY

56 **The CREATE TABLE Statement**

- CREATE TABLE *tableName*
  (*columnList w/ dataType* …);
- CREATE TABLE Vendor
  (VendorNumber varchar(5),
  VendorName varchar(25),
  Address varchar(30),
  City varchar(15),
  State varchar(2),
  ZipCode varchar(5),
  Telephone varchar(12),
  Contact varchar(25),
  Fax varchar(12),
  TermDays int,
  TermPercent decimal(3,2),
  DateLastOrder date);

60 **Data Maintenance in SQL**

- An important feature of a DBMS (Database Management System) is to keep data *current*
- UPDATE/SET statement
  - Modifies contents (value) of one or more fields
- INSERT INTO statement
  - Adds a new rows (records)to a table
- DELETE [FROM] statement
  - Removes one or more records FROM a table

64 **Data Retrieval in SQL**

- *Reporting* both printed form AND on-line (monitor)
- Basic format of the SQL SELECT command:
  SELECT *columns*
  FROM *tables*
  WHERE *booleanExpression;*
- SELECT clause limits columns returned (required)
- FROM clause names tables from which the columns are selected (required)
- WHERE clause limits rows returned (optional)

65 **Basic SELECT Statement**

- The * is a wildcard that specifies *all* columns
- SELECT *
  FROM *tableName*;
- All rows returned since there is no WHERE clause

```
SELECT  *  FROM Vendor;
SELECT  *  FROM PurchaseOrder;
SELECT  *  FROM Payables;
SELECT  *  FROM Product;
```

68 **The ASP:SqlDataSource Web Control (Page 2)**

- The SqlDataSource requires that two properties be set (automatically set when clicking "smart tag" and selecting command "Configure Data Source"):
  - ConnectionString—specifies the database type, location and filename
    - Created in the "Web.config" file the first time the SqlDataSource is linked to SqlServer database in the "Configure Data Source" wizard
  - SelectCommand—set to an SQL SELECT query
    - Created by using the Configure the Select Statement window of the wizard

69 **Data Source Configuration: New Data Source**

- Switch to Design View and click the "smart tag" to access the Configure Data Source wizard:
  - On the Configure Data Source window click the <New Connection…> button
  - In the Add Connection window, select "Microsoft SQL Server Database File" as the Data Source
  - Click <Continue> and <Browse…> to find database
  - Select it and click the <Open> button
  - Choose Yes, save the connection as to store the connection string in the "Web.config" file
  - Click <Next> to go to next window of the wizard

73 **Data Source Configuration: Existing Connection String**

- Switch to Design View and click the "smart tag" to access the Configure Data Source wizard:
  - Select the Connection string associated with the desired database from the drop-down list of existing connection strings
  - Click <Next> to go to next window of the wizard

75 **Build the SQL Select Statement**

- The Configure the Select Statement step of wizard specifies how to create SELECT statement:
  - Specify columns from a table or view:
    - Select the table name and check (✓) the columns (fields) of data to be included in the SELECT query
    - Further define the statement by using the <WHERE> and <ORDER BY> buttons

- ➢ The SELECT statement is visible under the heading SELECT statement:
- ➢ Click <Next> and then <Finish> buttons

78 **The ASP:GridView Web Control (Page 1)**

- Binds to and displays data from a data source control in tabular view (rows and columns)
- To assign the data source:
  - Click the "smart tag" and select a data source (e.g. a SQLDataSource object) from Choose Data Source: drop-down list
    - ➢ This formats the columns to represent the fields of the selected query (the data source)

79 **The ASP:GridView Web Control (Page 2)**

- If the GridView control initially does not reformat to represent the template of the actual data from the SqlDataSource, click Refresh Schema command
  - The source code is updated automatically to include a <Columns> block and <asp:BoundField> tags which represent the data

81 **SELECT with *Column List***

- Limits which columns are returned in a query
- SELECT *columnNames* …
  FROM *tableName* ;

```
SELECT VendorName, City, State,  ZipCode
FROM Vendor;
SELECT PoNumber, VendorNumber, Subtotal, Tax, Shipping, DisCOUNT, PoTotal
FROM PurchaseOrder;
SELECT PoNumber, ProductNumber, Quantity
FROM Payables;
```

85 **The WHERE Clause**

- Base on a truth condition, limits *rows* returned
- SELECT *columnNames* …
  FROM *tableName*
  WHERE *booleanExpression*;
- Relational operators are …

```
=   is equal to
>   is greater than
<   is less than
>=      is greater than or equal to
<=      is less than or equal to
!=  is not equal to (or <>)
```

87 **WHERE is equal to (=)**

- SELECT *columnNames* …

FROM *tableName*
WHERE *columnName* = *criteria*;

- Character comparisons usually are *case sensitive*
- Those rows will be returned for records in which the relation condition is True

```
SELECT  *
   FROM Vendor
   WHERE VendorName = "ACME UTILITIES";
SELECT  *
   FROM Payables
   WHERE PoNumber = "10003";
```

88 **WHERE is greater than (or equal to) (> or >=)**

- SELECT *columnNames* ...
  FROM *tableName*
  WHERE *columnName* > *criteria*; (or >=)

```
SELECT  *
   FROM PurchaseOrder
   WHERE Subtotal > 38.15;
SELECT  *
   FROM PurchaseOrder
   WHERE Subtotal >= 38.15;
SELECT  *
   FROM Product
   WHERE Product >= "P";
```

89 **WHERE is less than (or equal to) (< or <=)**

- SELECT *columnNames* ...
  FROM *tableName*
  WHERE *columnName* < *criteria*; (or <=)

```
SELECT  *
   FROM Payables
   WHERE UnitPrice < 2;
SELECT  *
   FROM Payables
   WHERE UnitPrice <= 2;
SELECT  *
   FROM Product
   WHERE Product < "P";
```

90 **WHERE is not equal to (<>)**

- Returns all records *except* that of the criteria value
- SELECT *columnNames* ...
  FROM *tableName*

WHERE *columnName* <> *criteria*;

- SELECT *
  FROM Vendor
  WHERE State <> "NY";
  SELECT *
  FROM PurchaseOrder
  WHERE PoNumber <> "10001";

95 **WHERE with IN**

- SELECT *columnNames* …
  FROM *tableName*
  WHERE *columnName* IN (*criteriaList*);
- Specifies a comma-delimited *list of values* that evaluate as True

```
SELECT *
FROM PurchaseOrder
WHERE PoNumber IN ('10001', '10003');
SELECT *
FROM Product
WHERE Product IN ('PENCILS', 'PENS');
```

101 **ORDER BY Clause**

- Sorts on the named column (field)
- SELECT *columnNames* …
  FROM *tableName*
  ORDER BY *columnName*;

```
SELECT *
FROM Vendor
ORDER BY ZipCode;
SELECT *
FROM PurchaseOrder
ORDER BY VendorNumber;
```

102 **ORDER BY *Descending* order**

- Default without keyword DESC is ascending
- SELECT *columnNames* …
  FROM *tableName*
  ORDER BY *columnName* DESC;
- SELECT *
  FROM PurchaseOrder
  ORDER BY PoDate DESC;
  SELECT *
  FROM Payables
  ORDER BY UnitPrice DESC;

103 **ORDER BY *Multiple* Columns**

- Sorts by first field in list, then by second field within the first field, etc.
- Each field to be sorted in descending order requires a *separate* keyword DESC
- SELECT *columnNames* …
  FROM *tableName*
  ORDER BY *columnList* …;
- SELECT *
  FROM PurchaseOrder
  ORDER BY VendorNumber, PoNumber;

SELECT *
FROM Payables
ORDER BY PoNumber, Quantity DESC;

116 **SELECT with Join**

- A join operation links related fields from *more than one table*
- SELECT *columnList* […]
  FROM *tableNameList* …
  WHERE *primaryKey* = *foreignKey*;
- Column names that exist in more than a single table must be *prefixed* by the table name, e.g.
  – Vendor.VendorNumber
  – PurchaseOrder.VendorNumber

123 **SELECT with Join (No. 1)**

- SELECT *columnList* […]
  FROM *tableNameList* …
  WHERE *primaryKey* = *foreignKey*;

SELECT VendorName, PoNumber, PoTotal
FROM Vendor, PurchaseOrder
WHERE Vendor.VendorNumber = PurchaseOrder.VendorNumber;

125 **SELECT with INNER JOIN (No. 1)**

- The keywords INNER JOIN are a more professional version of a table join operation
- SELECT *columnList* […]
  FROM *tableName1* INNER JOIN
  tableName2 ON *primaryKey* = *foreignKey*;

SELECT VendorName, PoNumber, PoTotal
FROM Vendor INNER JOIN
PurchaseOrder ON Vendor.VendorNumber = PurchaseOrder.VendorNumber;

127 **Using "QueryBuilder" for the SQL Select Statement (Page 1)**

- Any alternative during the Configure the Select Statement step of the "Configure Data Source" wizard is to use the QueryBuilder

  - A GUI tool that uses drag-and-drop functionality to create the SELECT statement
- This tool especially is helpful when *joining* data from *multiple tables*

128 **Using "QueryBuilder" for the SQL Select Statement (Page 2)**

- To access the QueryBuilder:
  1. During the Configure the Select Statement step, select Specify a custom SQL statement or stored procedure:
  2. Click <Next> and then click the <QueryBuilder> button to access *intuitive* user interface for creating the SELECT statement
  3. Click <Add> to add tables to query (then <Close>)

130 **Using "QueryBuilder" for the SQL Select Statement (Page 3)**

- To access the QueryBuilder (*con.*):
  4. Check (✓) columns from the table or tables for the query; then click <OK>
  5. Finally click <Next> and then <Finish> buttons

136 **Additional Table Matching Conditions**

- If the join includes *more than two* tables, additional table matching conditions are specified in subsequent AND clauses
- SELECT *columnList* […]
  FROM *tableNameList* …
  WHERE *primaryKey1 = foreignKey1*
  AND *primaryKey2 = foreignKey2*
  […];

137 **SELECT with Join (No. 3)**

```
SELECT VendorName, Address, City, State, ZipCode, PurchaseOrder.PoNumber,
       Vendor.VendorNumber, PoDate, Product.ProductNumber, Product, UnitPrice,
       Quantity, UnitPrice * Quantity as Extension, Subtotal, Tax, Shipping, Discount,
       PoTotal
  FROM Vendor, PurchaseOrder, Payables, Product
  WHERE Vendor.VendorNumber = PurchaseOrder.VendorNumber
  AND PurchaseOrder.PoNumber = Payables.PoNumber
  AND Payables.ProductNumber = Product.ProductNumber;
```

138 **Additional INNER JOIN Table Matching Conditions**

- If an INNER JOIN includes *more than two* tables, specify additional INNER JOIN and ON clauses
  - Each INNER JOIN must follow the table definition to which it links
- SELECT *columnList* …
  FROM *tableName1* INNER JOIN
  tableName2 ON *primaryKey1 = foreignKey1* INNER JOIN
  tableName3 ON *primaryKey2 = foreignKey2* […];

139 **SELECT with INNER JOIN (No. 3)**

```
SELECT VendorName, Address, City, State, ZipCode, PurchaseOrder.PoNumber,
      Vendor.VendorNumber, PoDate, Product.ProductNumber, Product, UnitPrice,
      Quantity, UnitPrice * Quantity as Extension, Subtotal, Tax, Shipping, Discount,
      PoTotal
FROM Vendor INNER JOIN
      PurchaseOrder ON Vendor.VendorNumber = PurchaseOrder.VendorNumber
            INNER JOIN
      Payables ON PurchaseOrder.PoNumber = Payables.PoNumber INNER JOIN
      Product ON Payables.ProductNumber = Product.ProductNumber;
```