**DataRow:**
A DataRow represent a row of data in data table. You add data to the data table using DataRow object. A DataRowCollection object represents a collection of data rows of a data table. You use DataTable's NewRow method to return a DataRow object of data table, add values to the data row and add a row to the data Table again by using DataRowCollection's Add method.

**Table 1: The Data Row Class properties**

| PROPERTY | DESCRIPITION |
|---|---|
| Item | Represents an item of a row |
| ItemArray | Represents all values in a row |
| RowState | Indicates the current state of a row |
| Table | Returns the DataTable to which this row is attached |

**Table 2: The Data Row class Methods**

| METHOD | DESCRIPTION |
|---|---|
| AcceptChanges | Comments all the changes made to this row |
| BeginEdit | Starts an edit operation on a row |
| CancelEdit | Cancels the current edit on a row |
| Delete | Deletes a DataRow |
| EndEdit | Ends the current edit on a row |
| GetChildRows | Returns child rows of a DataRow |
| GetParentRows | Returns parent rows of a DataRow |
| RejectsChanges | Rejects all the changes made since last AcceptChanges |

**DataColimn:**
The DataColumnCollection type returns a collection of columns that can be accessed through Columns property of the DataTable. The DataColumnCollection object represents a collection of columns attached to a data table. You add a data column to the DataColumnCollection using its Add method. The DataColumn object represents a column of a DataTable. For example, say you want to create a customers table that consists of three columns: ID, Address, and Name. You create three DataColumn objects and these columns to the DataColumnCollection using the DataTable.Column.Add method.

After creating a data table schema, the next step is to add data to the data table by using the DataRow object.

The DataColumn has some properties. These properties describe a column, such as its uniqueness, what kind of data you can store in that column, default value, caption, name, and so on. Below Table describes some of the DataColumn class members.

**Table: The DataColumn class properties**

| PROPERTY | DESCRIPITION |
|---|---|

| | |
|---|---|
| AllowDBNull | Both read and write, represent if column can store null values or not |
| AutoIncrement | Represent if column's value is auto increment or not |
| AutoIncrementSeed | Starting value of auto increment, applicable when AutoIncrement is true |
| AutoIncrementStep | Indicates the increment value |
| Caption | Caption of the column |
| ColumnMapping | Represent the MappingType of the column |
| ColumnName | Name of the column |
| DataType | Data type stored by the column |
| DefaultValue | Default value of the column |
| Expression | Represents the expression used to filter rows, calculate values, and so on |
| MaxLength | Represents maximum length of a text column |
| ReadOnly | Represents if a column is read-only or not |
| Unique | Indicates whether the values in a column must be unique or not |

**DataReader:**

A data reader provides an easy way for the programmer to read data from a database as if it were coming from a stream. The DataReader is the solution for forward streaming data through ADO.NET. The data reader is also called a firehose cursor or forward read-only cursor because it moves forward through the data. The data reader not only allows you to move forward through each record of database, but it also enables you to parse the data from each column. The DataReader class represents a data reader in ADO.NET.

Similar to other ADO.NET objects, each data provider has a data reader class for example; OleDbDataReader is the data reader class for OleDb data providers. Similarly, SqlDataReader and ODBC DataReader are data reader classes for Sql and ODBC data providers, respectively.

The IDataReader interface defines the functionally of a data reader and works as the base class for all data provider-specific data reader classes such as OleDataReader. SqlDataReader, and OdbcDataReader. Figure 5-36 shows some of the classes that implement IDbDataReader.

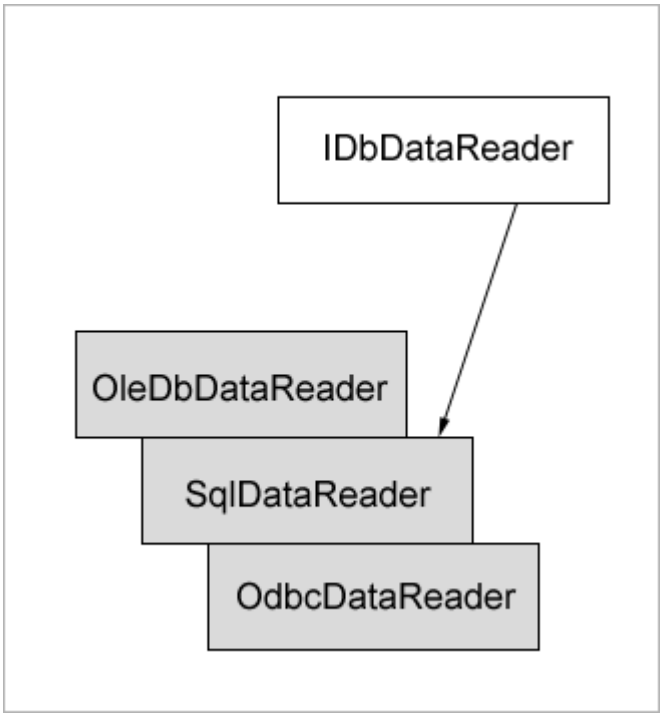


**Figure 5-36. Data Provider-specific classes implementing IdbDataReader**

**Initializing DataReader**

As you've seen in the previous examples, you call the ExecuteReader method of the Command object, which returns an instance of the DataReader. For example, use the following line of code:

```
SqlCommand cmd = new SqlCommand(SQL, conn);
// Call ExecuteReader to return a DataReader
SqlDataReader reader = cmd.ExecuteReader();
```

Once you're done with the data reader, call the Close method to close a data reader:

```
reader.Close();
```

**DataReader Properties and Methods**

Table 5-26 describes DataReader properties, and Table 5-27 describes DataReader methods.

**Table 5-26. The DataReader properties**

| PROPERTY | DESCRIPTION |
|---|---|

| | |
|---|---|
| Depth | Indicates the depth of nesting for row |
| FieldCount | Returns number of columns in a row |
| IsClosed | Indicates whether a data reader is closed |
| Item | Gets the value of a column in native format |
| RecordsAffected | Number of row affected after a transaction |

**Table 5-27. The DataReader methods**

| METHOD | DESCRIPTION |
|---|---|
| Close | Closes a DataRaeder object. |
| Read | Reads next record in the data reader. |
| NextResult | Advances the data reader to the next result during batch transactions. |
| Getxxx | There are dozens of Getxxx methods. These methods read a specific data type value from a column. For example. GetChar will return a column value as a character and GetString as a string. |