
Subject: Re: sqlite and dropdate / editdate etc
Posted by [jimlef](#) on Sat, 26 Sep 2020 05:43:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

I personally prefer epoch dates, but with that I was getting invalid info when assigning (automatically) to sqlctrls... Instead of editdate or dropdate showing a date, they'd show the integer value that was stored.

Perhaps that is a weakness of sqlite, but I'm just working things out as I go :) I know I still have a lot to learn as well...

Can always change back to ints (wrote (hacked together) a little c# util that understands epoch and the original datetime constructs for the conversions).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SQLite;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace convertdates
{
    public partial class Form1 : Form
    {
        DataTable dt = new DataTable();
        TransactionsTable transactionsTable = new TransactionsTable();
        ProductsTable productsTable = new ProductsTable();
        CustomersTable customersTable = new CustomersTable();

        string myConnection = "Data Source=sample.db;Version=3;";

        public Form1()
        {
            InitializeComponent();
            DateTime dt = new DateTime(1970, 1, 1);
            long sticks = dt.Ticks;
            label1.Text = sticks.ToString();
        }

        private string correctdate(long baddate)
        {
```

```

        DateTime dt = new DateTime(1970, 1, 1);
        long junk = baddate * 86400 * 10000000;
        long sticks = dt.Ticks;
        junk += sticks;
        DateTime retvalue = new DateTime(junk);
        // long datenum = dtp1.Value.Date.Ticks;
        // long dayssince1970 = ((baddate * 86400 * 10000000) - sticks) / 86400 / 10000000;
        return retvalue.ToShortDateString();
    }

    private void btnConvert_Click(object sender, EventArgs e)
    {
        dt = DisplayAllTransactions();
        foreach (DataRow row in dt.Rows)
        {
            transactionsTable.INVOICES_ID = int.Parse(row[0].ToString());
            transactionsTable.TRANSACTIONDATE = transactionsTable.DATEPAID =
correctdate(long.Parse(row[1].ToString()));
            UpdateTransactions(transactionsTable);
        }
        dt = SelectProducts();
        foreach (DataRow row in dt.Rows)
        {
            productsTable.PROD_ID = int.Parse(row[0].ToString());
            productsTable.DATEPURCHASED = correctdate(long.Parse(row[1].ToString()));
            UpdateProducts(productsTable);
        }
    }

    public DataTable DisplayAllTransactions()
    {
        //SQLiteConnection First
        SQLiteConnection conn = new SQLiteConnection(myConnection);

        //Create a Data Table to hold the datafrom database temporarily
        DataTable dt = new DataTable();

        try
        {
            //Write the SQL Query to Display all Transactions
            string sql = "SELECT INVOICES_ID, TRANSACTIONDATE, DATEPAID FROM
INVOICES";

            //SQLiteCommand to Execute Query
            SQLiteCommand cmd = new SQLiteCommand(sql, conn);

            //SQLiteDataAdapter to Hold the data from database
            SQLiteDataAdapter adapter = new SQLiteDataAdapter(cmd);

```

```

        //Open DAtabase Connection
        conn.Open();

        adapter.Fill(dt);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        if (ConnectionState.Open == conn.State) conn.Close();
    }

    return dt;
}
#region Method to Update transaction in Database
bool UpdateTransactions(TransactionsTable p)
{
    //create a boolean variable and set its initial value to false
    bool isSuccess = false;

    //Create SQL Connection for DAtabase
    SQLiteConnection conn = new SQLiteConnection(myConnection);

    try
    {
        //SQL Query to Update Data in dAtabase
        String sql = "UPDATE INVOICES SET TRANSACTIONDATE=@transactionDate,
DATEPAID=@datePaid WHERE INVOICES_ID=@id";

        //Create SQL Cmmmand to pass the value to query
        SQLiteCommand cmd = new SQLiteCommand(sql, conn);
        //Passing the values using parameters and cmd
        cmd.Parameters.AddWithValue("@transactionDate", p.TRANSACTIONDATE);
        cmd.Parameters.AddWithValue("@datePaid", p.DATEPAID);
        cmd.Parameters.AddWithValue("@id", p.INVOICES_ID);

        //Open the Database connection
        conn.Open();

        //Create Int Variable to check if the query is executed successfully or not
        int rows = cmd.ExecuteNonQuery();

        //if the query is executed successfully then the value of rows will be greater than 0 else
        it will be less than zero
        if (rows > 0)

```

```

        {
            //Query ExecutedSuccessfully
            isSuccess = true;
        }
        else
        {
            //Failed to Execute Query
            isSuccess = false;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        conn.Close();
    }

    return isSuccess;
}
#endregion
#region Select method for Product Module
public DataTable SelectProducts()
{
    //Create Sql Connection to connect Databaes
    SQLiteConnection conn = new SQLiteConnection(myConnection);

    //DataTale to hold the data from database
    DataTable dt = new DataTable();

    try
    {
        //Writing the Query to Select all the products from database
        String sql = "SELECT PROD_ID, DATEPURCHASED FROM PRODUCTS";

        //Creating SQL Command to Execute Query
        SQLiteCommand cmd = new SQLiteCommand(sql, conn);

        //SQL Data Adapter to hold the value from database temporarily
        SQLiteDataAdapter adapter = new SQLiteDataAdapter(cmd);

        //Open DAtabase Connection
        conn.Open();

        adapter.Fill(dt);
    }
    catch (Exception ex)

```

```

    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        conn.Close();
    }

    return dt;
}
#endregion
#region Method to Update Product in Database
bool UpdateProducts(ProductsTable p)
{
    //create a boolean variable and set its initial value to false
    bool isSuccess = false;

    //Create SQL Connection for Database
    SQLiteConnection conn = new SQLiteConnection(myConnection);

    try
    {
        //SQL Query to Update Data in dAtabase
        String sql = "UPDATE PRODUCTS SET DATEPURCHASED=@datepurchased
WHERE PROD_ID=@id";

        //Create SQL Cmmmand to pass the value to query
        SQLiteCommand cmd = new SQLiteCommand(sql, conn);
        //Passing the values using parameters and cmd
        cmd.Parameters.AddWithValue("@datepurchased", p.DATEPURCHASED);
        cmd.Parameters.AddWithValue("@id", p.PROD_ID);

        //Open the Database connection
        conn.Open();

        //Create Int Variable to check if the query is executed successfully or not
        int rows = cmd.ExecuteNonQuery();

        //if the query is executed successfully then the value of rows will be greater than 0 else
        it will be less than zero
        if (rows > 0)
        {
            //Query ExecutedSuccessfully
            isSuccess = true;
        }
        else
        {
            //Failed to Execute Query

```

```

        isSuccess = false;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
finally
{
    conn.Close();
}

return isSuccess;
}
#endregion

private void btnSelectDB_Click(object sender, EventArgs e)
{
    string ProgramDataDir =
Path.Combine(Environment.ExpandEnvironmentVariables("%userprofile%"), "Documents") + "\\";
    Directory.CreateDirectory(ProgramDataDir);
    ofDialog.DefaultExt = ".db";
    ofDialog.CheckFileExists = true;
    ofDialog.InitialDirectory = ProgramDataDir;
    if (ofDialog.ShowDialog() == DialogResult.OK)
    {
        myConnection = "Data Source=" + ofDialog.FileName + ";Version=3;";
    }
}

public DataTable SelectCustomers()
{
    //SQL Connection for Database Connection
    SQLiteConnection conn = new SQLiteConnection(myConnection);

    //DataTble to hold the value from database and return it
    DataTable dt = new DataTable();

    try
    {
        //Write SQL Query t Select all the DAta from dAtabase
        string sql = "SELECT * FROM CUSTOMERS";

        //Creating SQL Command to execute Query
        SQLiteCommand cmd = new SQLiteCommand(sql, conn);

        //Creting SQL Data Adapter to Store Data From Database Temporarily
        SQLiteDataAdapter adapter = new SQLiteDataAdapter(cmd);

```

```

        //Open Database Connection
        conn.Open();
        //Passign the value from SQL Data Adapter to DAta table
        adapter.Fill(dt);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        conn.Close();
    }

    return dt;
}
bool UpdateCustomers(CustomersTable dc)
{
    //SQL Connection for Database Connection
    SQLiteConnection conn = new SQLiteConnection(myConnection);
    //Create Boolean variable and set its default value to false
    bool isSuccess = false;

    try
    {
        //SQL Query to update data in database
        string sql = "UPDATE CUSTOMERS SET CUSTNAME=@name, EMAIL=@email,
CONTACT=@contact, ADDRESS=@address WHERE CUST_ID=@id";
        //Create SQL Command to pass the value in sql
        SQLiteCommand cmd = new SQLiteCommand(sql, conn);

        //Passing the values through parameters
        cmd.Parameters.AddWithValue("@name", dc.CUSTNAME);
        cmd.Parameters.AddWithValue("@email", dc.EMAIL);
        cmd.Parameters.AddWithValue("@contact", dc.CONTACT);
        cmd.Parameters.AddWithValue("@address", dc.ADDRESS);
        cmd.Parameters.AddWithValue("@id", dc.CUST_ID);

        //open the Database Connection
        conn.Open();

        //Int varialbe to check if the query executed successfully or not
        int rows = cmd.ExecuteNonQuery();
        if (rows > 0)
        {
            //Query Executed Successfully
            isSuccess = true;
        }
    }
}

```

```

        else
        {
            //Failed to Execute Query
            isSuccess = false;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        conn.Close();
    }

    return isSuccess;
}

public string GenerateNumber()
{
    Random random = new Random();
    string r = "";
    int i;
    for (i = 1; i < 11; i++)
    {
        r += random.Next(0, 9).ToString();
    }
    return r;
}

private void btnGeneralize_Click(object sender, EventArgs e)
{
    var rand = new Random();
    string[] firstNames = new string[] { "Jim", "Mary", "Catherine", "John", "Mark", "Luke",
"Lucy", "Abi", "Roger", "Ruth" };
    string[] lastNames = new string[] { "Jones", "Smith", "Johnson", "Yancovich",
"Steemburgin", "Carter", "Reagan", "Trump", "Jefferson", "Franklin" };
    dt = SelectCustomers();
    foreach (DataRow row in dt.Rows)
    {
        string CustFirstName = firstNames[rand.Next(10)];
        string CustLastName = lastNames[rand.Next(10)];
        string Email = CustFirstName + "." + CustLastName + "@gmail.com";
        string Contact = GenerateNumber();
        customersTable.CUST_ID = int.Parse(row[0].ToString());
        customersTable.CUSTNAME = CustFirstName + " " + CustLastName;
        customersTable.CONTACT = Contact;
        customersTable.EMAIL = Email;
        customersTable.ADDRESS = "0 Main Way";
        UpdateCustomers(customersTable);
    }
}

```

```

    }

}

private void dtp1_ValueChanged(object sender, EventArgs e)
{
    DateTime dt = new DateTime(1970, 1, 1);
    long sticks = dt.Ticks;
    long datenum = dtp1.Value.Date.Ticks;
    long dayssince1970 = (datenum - sticks) / 86400 / 10000000;
    label1.Text = dayssince1970.ToString();
    Clipboard.SetText(label1.Text);
}

}
class ProductsTable
{
    //Getters and Setters for Product Module
    public int PROD_ID { get; set; }
    public string PRODNAME { get; set; }
    public string PRODDescription { get; set; }
    public string DATEPURCHASED { get; set; }
    public decimal COST { get; set; }
    public int INVOICEID { get; set; }

}
public enum status
{
    notpaid = 0,
    voided,
    partialpayment,
    paidinfull
};
class TransactionsTable
{
    public int INVOICES_ID { get; set; }
    public int INVOICENUMBER { get; set; }
    public int CUSTOMERID { get; set; }
    public string TRANSACTIONDATE { get; set; }
    // public long due_date { get; set; }
    public string TERMS { get; set; }
    public decimal NONTAXABLESUB { get; set; }
    public decimal TAXABLESUB { get; set; }
    public decimal TAX { get; set; }
    public decimal GRANDTOTAL { get; set; }
    public decimal AMTPAID { get; set; }
    public string DATEPAID { get; set; }
    public int STATUS { get; set; }
}

```

```
}  
class CustomersTable  
{  
    public int CUST_ID { get; set; }  
    public string CUSTNAME { get; set; }  
    public string EMAIL { get; set; }  
    public string CONTACT { get; set; }  
    public string ADDRESS { get; set; }  
    public string CITY { get; set; }  
    public string STATE { get; set; }  
    public string ZIP { get; set; }  
    public int TAXABLE { get; set; }  
  
}  
}
```
