using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

namespace Jarloo.BulkUpsert

{

    internal class Program

    {

        private static void Main(string[] args)

        {

            List<Price> prices = new List<Price>();

            Random rnd = new Random();

            //Create some unique rows for INSERT

            for (int i = 0; i < 10000; i++)

            {

                prices.Add(new Price {Symbol = "AAPL", Value = rnd.NextDouble(), Timestamp = new DateTime(2010, 1, 1).AddHours(i)});

                prices.Add(new Price {Symbol = "GOOG", Value = rnd.NextDouble(), Timestamp = new DateTime(2010, 1, 1).AddHours(i)});

            }

            //Add a new duplicates FOR UPDATE

            for (int i = 0; i < 10; i++)

            {

                prices.Add(new Price {Symbol = "AAPL", Value = rnd.NextDouble(), Timestamp = new DateTime(2010, 1, 1).AddHours(i)});

                prices.Add(new Price {Symbol = "GOOG", Value = rnd.NextDouble(), Timestamp = new DateTime(2010, 1, 1).AddHours(i)});

            }

            //Make a temp table in sql server that matches our production table

            string tmpTable = "create table #Prices (symbol nvarchar(10), Price decimal(18,4), Timestamp DateTime)";

            //Create a datatable that matches the temp table exactly. (WARNING: order of columns must match the order in the table)

            DataTable table = new DataTable();

            table.Columns.Add(new DataColumn("Symbol", typeof (string)));

            table.Columns.Add(new DataColumn("Price", typeof (double)));

            table.Columns.Add(new DataColumn("Timestamp", typeof (DateTime)));

            //Add prices in our list to our DataTable

            foreach (Price price in prices)

            {

                DataRow row = table.NewRow();

                row["Symbol"] = price.Symbol;

                row["Price"] = price.Value;

                row["Timestamp"] = price.Timestamp;

                table.Rows.Add(row);

            }

            //Connect to DB

            string conString = ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString;

            using (SqlConnection con = new SqlConnection(conString))

            {

                con.Open();

                //Execute the command to make a temp table

                SqlCommand cmd = new SqlCommand(tmpTable, con);

                cmd.ExecuteNonQuery();

                //BulkCopy the data in the DataTable to the temp table

                using (SqlBulkCopy bulk = new SqlBulkCopy(con))

                {

                    bulk.DestinationTableName = "#Prices";

                    bulk.WriteToServer(table);

                }

                // merge command to upsert from the temp table to the production table

                string mergeSql = "merge into ProductionPrices as Target " +

                                  "using #Prices as Source " +

                                  "on " +

                                  "Target.Symbol=Source.Symbol " +

                                  "and Target.Timestamp = Source.Timestamp " +

                                  "when matched then " +

                                  "update set Target.Price=Source.Price " +

                                  "when not matched then " +

                                  "insert (Symbol,Price,Timestamp) values (Source.Symbol,Source.Price,Source.Timestamp);";

                cmd.CommandText = mergeSql;

                cmd.ExecuteNonQuery();

                //Clean up the temp table

                cmd.CommandText = "drop table #Prices";

                cmd.ExecuteNonQuery();

            }

        }

    }

    internal class Price

    {

        public string Symbol { get; set; }

        public double Value { get; set; }

        public DateTime Timestamp { get; set; }

    }

}