LinksPlatform's Platform.Data.Doublets Class Library

./UInt64LinksTransactionsLayer.cs

```csharp
using System;
using System.Linq;
using System.Collections.Generic;
using System.IO;
using System.Runtime.CompilerServices;
using System.Threading;
using System.Threading.Tasks;
using Platform.Disposables;
using Platform.Timestamps;
using Platform.Unsafe;
using Platform.IO;
using Platform.Data.Doublets.Decorators;

namespace Platform.Data.Doublets
{
    public class UInt64LinksTransactionsLayer : LinksDisposableDecoratorBase<ulong> //-V3073
    {
        /// <remarks>
        /// Альтернативные варианты хранения трансформации (элемента транзакции):
        ///
        /// private enum TransitionType
        /// {
        ///     Creation,
        ///     UpdateOf,
        ///     UpdateTo,
        ///     Deletion
        /// }
        ///
        /// Или
        ///
        /// public struct TransitionHeader
        /// {
        ///     public ulong TransactionIdCombined;
        ///     public ulong TimestampCombined;
        ///
        ///     public ulong TransactionId
        ///     {
        ///         get
        ///         {
        ///             return (ulong) mask & TransactionIdCombined;
        ///         }
        ///     }
        ///
        ///     public UniqueTimestamp Timestamp
        ///     {
        ///         get
        ///         {
        ///             return (UniqueTimestamp)mask & TransactionIdCombined;
        ///         }
        ///     }
        ///
        ///     public TransactionItemType Type
        ///     {
        ///         get
        ///         {
        ///             // Использовать по одному биту из TransactionId и Timestamp,
        ///             // для значения в 2 бита, которое представляет тип операции
        ///             throw new NotImplementedException();
        ///         }
        ///     }
        /// }
        ///
        /// private struct Transition
        /// {
        ///     public TransitionHeader Header;
        ///     public Link Source;
        ///     public Link Linker;
        ///     public Link Target;
        /// }
        /// </remarks>
        public struct Transition
        {
            public static readonly long Size = StructureHelpers.SizeOf<Transition>();

            public readonly ulong TransactionId;
            public readonly UInt64Link Before;
            public readonly UInt64Link After;
            public readonly Timestamp Timestamp;

            public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId, UInt64Link before, UInt64Link after)
            {
                TransactionId = transactionId;
                Before = before;
                After = after;
                Timestamp = uniqueTimestampFactory.Create();
            }

            public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId, UInt64Link before)
                : this(uniqueTimestampFactory, transactionId, before, default)
            {
            }

            public Transition(UniqueTimestampFactory uniqueTimestampFactory, ulong transactionId)
                : this(uniqueTimestampFactory, transactionId, default, default)
            {
            }

            public override string ToString() => $"{Timestamp} {TransactionId}: {Before} => {After}";
        }

        /// <remarks>
        /// Другие варианты реализации транзакций (атомарности):
```

```
114         ///         1. Разделение хранения значения связи ((Source Target) или (Source Linker
↪   Target)) и индексов.
115         ///         2. Хранение трансформаций/операций в отдельном хранилище Links, но
↪   дополнительно потребуется решить вопрос
116         ///             со ссылками на внешние идентификаторы, или как-то иначе решить
↪   вопрос с пересечениями идентификаторов.
117         ///
118         /// Где хранить промежуточный список транзакций?
119         ///
120         /// В оперативной памяти:
121         /// Минусы:
122         ///         1. Может усложнить систему, если она будет функционировать
↪   самостоятельно,
123         ///     так как нужно отдельно выделять память под список трансформаций.
124         ///         2. Выделенной оперативной памяти может не хватить, в том случае,
125         ///     если транзакция использует слишком много трансформаций.
126         ///             -> Можно использовать жёсткий диск для слишком длинных транзакций.
127         ///             -> Максимальный размер списка трансформаций можно ограничить /
↪   задать константой.
128         ///         3. При подтверждении транзакции (Commit) все трансформации
↪   записываются разом создавая задержку.
129         ///
130         /// На жёстком диске:
131         /// Минусы:
132         ///         1. Длительный отклик, на запись каждой трансформации.
133         ///         2. Лог транзакций дополнительно наполняется отменёнными транзакциями.
134         ///             -> Это может решаться упаковкой/исключением дублирующих операций.
135         ///             -> Также это может решаться тем, что короткие транзакции вообще
136         ///                 не будут записываться в случае отката.
137         ///         3. Перед тем как выполнять отмену операций транзакции нужно дождаться
↪   пока все операции (трансформации)
138         ///             будут записаны в лог.
139         ///
140         /// </remarks>
141     public class Transaction : DisposableBase
142     {
143         private readonly Queue<Transition> _transitions;
144         private readonly UInt64LinksTransactionsLayer _layer;
145         public bool IsCommitted { get; private set; }
146         public bool IsReverted { get; private set; }
147
148         public Transaction(UInt64LinksTransactionsLayer layer)
149         {
150             _layer = layer;
151             if (_layer._currentTransactionId != 0)
152             {
153                 throw new NotSupportedException("Nested transactions not supported.");
154             }
155             IsCommitted = false;
156             IsReverted = false;
157             _transitions = new Queue<Transition>();
158             SetCurrentTransaction(layer, this);
159         }
160
161         public void Commit()
162         {
163             EnsureTransactionAllowsWriteOperations(this);
164             while (_transitions.Count > 0)
165             {
166                 var transition = _transitions.Dequeue();
167                 _layer._transitions.Enqueue(transition);
168             }

169                 _layer._lastCommitedTransactionId = _layer._currentTransactionId;
170             IsCommitted = true;
171         }
172
173         private void Revert()
174         {
175             EnsureTransactionAllowsWriteOperations(this);
176             var transitionsToRevert = new Transition[_transitions.Count];
177             _transitions.CopyTo(transitionsToRevert, 0);
178             for (var i = transitionsToRevert.Length - 1; i >= 0; i--)
179             {
180                 _layer.RevertTransition(transitionsToRevert[i]);
181             }
182             IsReverted = true;
183         }
184
185         public static void SetCurrentTransaction(UInt64LinksTransactionsLayer layer,
↪   Transaction transaction)
186         {
187             layer._currentTransactionId = layer._lastCommitedTransactionId + 1;
188             layer._currentTransactionTransitions = transaction._transitions;
189             layer._currentTransaction = transaction;
190         }
191
192         public static void EnsureTransactionAllowsWriteOperations(Transaction
↪   transaction)
193         {
194             if (transaction.IsReverted)
195             {
196                 throw new InvalidOperationException("Transation is reverted.");
197             }
198             if (transaction.IsCommitted)
199             {
200                 throw new InvalidOperationException("Transation is commited.");
201             }
202         }
203
204         protected override void DisposeCore(bool manual, bool wasDisposed)
205         {
206             if (!wasDisposed && _layer != null && !_layer.IsDisposed)
207             {
208                 if (!IsCommitted && !IsReverted)
209                 {
210                     Revert();
211                 }
212                 _layer.ResetCurrentTransation();
213             }
214         }
215
216         // TODO: THIS IS EXCEPTION WORKAROUND, REMOVE IT THEN
↪   https://github.com/linksplatform/Disposables/issues/13 FIXED
217         protected override bool AllowMultipleDisposeCalls => true;
218     }
219
220     public static readonly TimeSpan DefaultPushDelay = TimeSpan.FromSeconds(0.1);
221
222     private readonly string _logAddress;
223     private readonly FileStream _log;
224     private readonly Queue<Transition> _transitions;
225     private readonly UniqueTimestampFactory _uniqueTimestampFactory;
226     private Task _transitionsPusher;
227     private Transition _lastCommittedTransition;
228     private ulong _currentTransactionId;
```

```csharp
229        private Queue<Transition> _currentTransactionTransitions;
230        private Transaction _currentTransaction;
231        private ulong _lastCommitedTransactionId;
232
233        public UInt64LinksTransactionsLayer(ILinks<ulong> links, string logAddress)
234            : base(links)
235        {
236            if (string.IsNullOrWhiteSpace(logAddress))
237            {
238                throw new ArgumentNullException(nameof(logAddress));
239            }
240            // В первой строке файла хранится последняя закоммиченную транзакцию.
241            // При запуске это используется для проверки удачного закрытия файла лога.
242            // In the first line of the file the last committed transaction is stored.
243            // On startup, this is used to check that the log file is successfully closed.
244            var lastCommitedTransition =
       ↪ FileHelpers.ReadFirstOrDefault<Transition>(logAddress);
245            var lastWrittenTransition =
       ↪ FileHelpers.ReadLastOrDefault<Transition>(logAddress);
246            if (!lastCommitedTransition.Equals(lastWrittenTransition))
247            {
248                Dispose();
249                throw new NotSupportedException("Database is damaged, autorecovery is not
       ↪ supported yet.");
250            }
251            if (lastCommitedTransition.Equals(default(Transition)))
252            {
253                FileHelpers.WriteFirst(logAddress, lastCommitedTransition);
254            }
255            _lastCommitedTransition = lastCommitedTransition;
256            // TODO: Think about a better way to calculate or store this value
257            var allTransitions = FileHelpers.ReadAll<Transition>(logAddress);
258            _lastCommitedTransactionId = allTransitions.Max(x => x.TransactionId);
259            _uniqueTimestampFactory = new UniqueTimestampFactory();
260            _logAddress = logAddress;
261            _log = FileHelpers.Append(logAddress);
262            _transitions = new Queue<Transition>();
263            _transitionsPusher = new Task(TransitionsPusher);
264            _transitionsPusher.Start();
265        }
266
267        public IList<ulong> GetLinkValue(ulong link) => Links.GetLink(link);
268
269        public override ulong Create()
270        {
271            var createdLinkIndex = Links.Create();
272            var createdLink = new UInt64Link(Links.GetLink(createdLinkIndex));
273            CommitTransition(new Transition(_uniqueTimestampFactory,
       ↪ _currentTransactionId, default, createdLink));
274            return createdLinkIndex;
275        }
276
277        public override ulong Update(IList<ulong> parts)
278        {
279            var beforeLink = new UInt64Link(Links.GetLink(parts[Constants.IndexPart]));
280            parts[Constants.IndexPart] = Links.Update(parts);
281            var afterLink = new UInt64Link(Links.GetLink(parts[Constants.IndexPart]));
282            CommitTransition(new Transition(_uniqueTimestampFactory,
       ↪ _currentTransactionId, beforeLink, afterLink));
283            return parts[Constants.IndexPart];
284        }
285

286        public override void Delete(ulong link)
287        {
288            var deletedLink = new UInt64Link(Links.GetLink(link));
289            Links.Delete(link);
290            CommitTransition(new Transition(_uniqueTimestampFactory,
       ↪ _currentTransactionId, deletedLink, default));
291        }
292
293        [MethodImpl(MethodImplOptions.AggressiveInlining)]
294        private Queue<Transition> GetCurrentTransitions() =>
       ↪ _currentTransactionTransitions ?? _transitions;
295
296        private void CommitTransition(Transition transition)
297        {
298            if (_currentTransaction != null)
299            {
300                Transaction.EnsureTransactionAllowsWriteOperations(_currentTransaction);
301            }
302            var transitions = GetCurrentTransitions();
303            transitions.Enqueue(transition);
304        }
305
306        private void RevertTransition(Transition transition)
307        {
308            if (transition.After.IsNull()) // Revert Deletion with Creation
309            {
310                Links.Create();
311            }
312            else if (transition.Before.IsNull()) // Revert Creation with Deletion
313            {
314                Links.Delete(transition.After.Index);
315            }
316            else // Revert Update
317            {
318                Links.Update(new[] { transition.After.Index, transition.Before.Source,
       ↪ transition.Before.Target });
319            }
320        }
321
322        private void ResetCurrentTransation()
323        {
324            _currentTransactionId = 0;
325            _currentTransactionTransitions = null;
326            _currentTransaction = null;
327        }
328
329        private void PushTransitions()
330        {
331            if (_log == null || _transitions == null)
332            {
333                return;
334            }
335            for (var i = 0; i < _transitions.Count; i++)
336            {
337                var transition = _transitions.Dequeue();
338
339                _log.Write(transition);
340                _lastCommitedTransition = transition;
341            }
342        }
343
344        private void TransitionsPusher()
```

```
345            {
346                while (!IsDisposed && _transitionsPusher != null)
347                {
348                    Thread.Sleep(DefaultPushDelay);
349                    PushTransitions();
350                }
351            }
352
353            public Transaction BeginTransaction() => new Transaction(this);
354
355            private void DisposeTransitions()
356            {
357                try
358                {
359                    var pusher = _transitionsPusher;
360                    if (pusher != null)
361                    {
362                        _transitionsPusher = null;
363                        pusher.Wait();
364                    }
365                    if (_transitions != null)
366                    {
367                        PushTransitions();
368                    }
369                    Disposable.TryDispose(_log);
370                    FileHelpers.WriteFirst(_logAddress, _lastCommitedTransition);
371                }
372                catch
373                {
374                }
375            }
376
377            #region DisposalBase
378
379            protected override void DisposeCore(bool manual, bool wasDisposed)
380            {
381                if (!wasDisposed)
382                {
383                    DisposeTransitions();
384                }
385                base.DisposeCore(manual, wasDisposed);
386            }
387
388            #endregion
389        }
390    }
```

fmt.sh

```bash
#!/bin/bash
set -e # Exit with nonzero exit code if anything fails

echo """
\\documentclass[7pt,a4paper,fleqn]{report}
\\usepackage[left=6mm,top=5mm,right=5mm,bottom=7mm,landscape]{geometry}
\\textwidth=283mm
\\pagestyle{plain}
\\usepackage[utf8]{inputenc}
\\usepackage[T1]{fontenc}
\\usepackage[T2A]{fontenc}
\\usepackage[gray]{xcolor}
\\usepackage{minted}

\\makeatletter
\\let\\xUTFviii@two@octets\\UTFviii@two@octets

\\def\\UTFviii@two@octets#1#2{%
\\ifx\\FancyVerbBreakAnywhereBreak#2%
\\expandafter\\xUTFviii@two@octets\\expandafter#1%
\\else
\\xUTFviii@two@octets#1#2%
\\fi
}
\\makeatother

\\usepackage{multicol}
\\usepackage{makeidx}
\\usepackage[columns=2]{idxlayout}
\\makeindex
\\renewcommand{\\thesection}{\\arabic{chapter}.\\arabic{section}}
\\setcounter{chapter}{1}
\\setcounter{section}{0}
\\usepackage[tiny]{titlesec}
\\titlespacing\\chapter{0mm}{0mm}{0mm}
\\titlespacing\\section{0mm}{0mm}{0mm}
```

```
37  \\DeclareUnicodeCharacter{221E}{\\ensuremath{\\infty}}
38  \\DeclareUnicodeCharacter{FFFD}{\\ensuremath{ }}
39  \\usepackage{fancyhdr}
40  \\pagestyle{fancy}
41  \\fancyhf{}
42  \\fancyfoot[C]{\\thepage}
43  \\renewcommand{\\headrulewidth}{0mm}
44  \\renewcommand{\\footrulewidth}{0mm}
45  \\renewcommand{\\baselinestretch}{0.7}
46  \\begin{document}
47
48  \\newminted{csharp}{
49      breaklines,
50      breakanywhere
51  }
52
53  \\sf
54  \\noindent{\\Large LinksPlatform's Platform.Data.Doublets Class Library}
55  \\begin{multicols}{2}
56  """
57
58  # CSharp
59  #find * -type f -iname '*.cs' -exec sh -c 'enconv "{}"' \;
60  find . -type f -iname '*.cs' | sort -b | python fmt.py
61
62  echo """
63  \\end{multicols}
64  \\begin{section}{fmt.sh}
65  \\vspace{2mm}
66  \\inputminted[tabsize=2,breaklines,linenos=true]{bash}{fmt.sh}
67  \\end{section}
68  \\begin{section}{fmt.py}
69  \\vspace{2mm}
70  \\inputminted[tabsize=2,breaklines,linenos=true]{python}{fmt.py}
71  \\end{section}
72  \\printindex
73  \\end{document}
74  """
```

fmt.py

```
1   #!/usr/bin/python
2   # -*- coding: utf-8 -*-
3   import sys
4   reload(sys)
5   sys.setdefaultencoding('utf-8')
6   for line in sys.stdin.readlines():
7       line = line.strip()
8   print "\\index{%s}" % (line.replace('_','\\_'))
9   print "\\begin{section}{%s}" % (line.replace('_','\\_'))
10  #print "\\inputminted[tabsize=2,breaklines,linenos=true]{csharp}{%s}" % (line)
11  print "\\begin{minted}[tabsize=2,breaklines,breakanywhere,linenos=true,xleftmargin=7mm,framesep=4mm,fontsize=\\small,fontfamily=NimbusMono]{csharp}"
12  f = open(line,"rt")
13  c = "\n".join([x.strip("\n") for x in f.readlines()])
14  f.close()
15  c = c.replace(u'\ufeff','')
16  print c
17  print "\\end{minted}"
18  print "\\end{section}"
19  print
```

Index