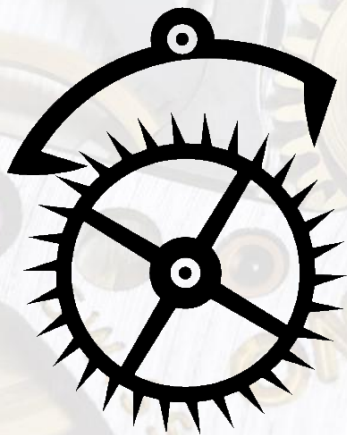# Historisation Operator

2-dimensional / bi-temporal data historisation

Arnd Wussing
Arulan Consulting

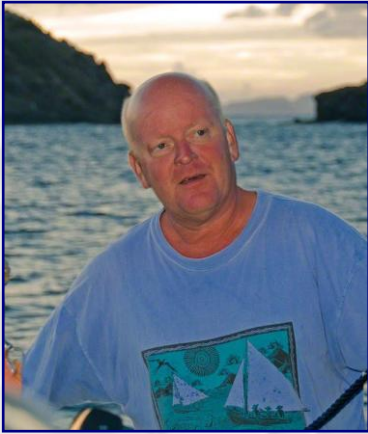# „Hist-Op": 2-dimensional data historisation for DataStage®

Figure 1 - Arnd out of the office

Arulan Consulting specializes in assisting organisations who require expertise in the following areas:

- Data warehouse design, architecture & implementation.
- ETL processing using tools such as DataStage, Informatica PowerCenter, and Pentaho PDI.
- ETL and data warehouse tuning and optimization.
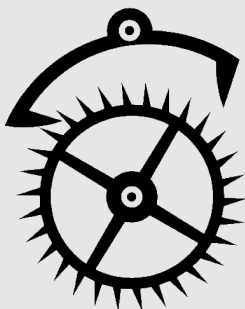- Temporal and bi-temporal processing.

Arnd Wussing, principal consultant at Arulan Consulting and developer of the "Hist-Op" has a wealth of experience with databases and data warehousing gleaned over many years of consulting work in the USA, Europe and Asia. He was awarded the coveted "IBM Information Champion" award three years running from 2010 to 2012 for work in ETL and Data Warehousing.

The image used on the title page of this document and as the icon for the "Hist-Op" is a stylized silhouette of a simple mechanical watch escapement. The escapement, in use since the late 13th century, has been the basic means for measuring time in watches, clocks and other timepieces until the advent of electronic timekeeping in 1957 and quartz watches in the 1970s. Thus it is a fitting symbol for the historisation of data - the escapement partitions the turning of the wheel into discrete units, just as historisation does with data.

Table of Contents

1.   Introduction

With the advent of data warehousing and the exponential growth in institutional data collection and analysis in recent years have come many new and demanding requirements for computer databases and the reporting systems that they support.

Two significant requirements are <u>query repeatability</u> and <u>data auditability</u>. Repeatability means that the same query returns the same results each time it is issued, and auditability stipulates that no data is ever modified or deleted without the ability to track each of those changes. In addition to the use of data warehouses for reporting purposes, these modern systems frequently underly some form of regulation and must therefore comply with auditing and data governance rules. Although bi-temporal data storage and historisation has grown from data warehousing, it is widely applicable to all manners of databases and storage systems. Therefore this document will use general database terminology rather than the rather arcane language often found in data warehousing.

These diverse new requirements are difficult to implement while keeping the database functioning efficiently. The implementation of a bi-temporal data storage model offers a very good compromise between functionality, maintainability, accessibility and compactness. If certain rules are adhered to when inserting and updating data in a bi-temporal database (chapter 4.7) then this system is quite efficient and results in a minimum of overhead in order to deliver repeatable results, plus it is possible to superimpose retroactively onto an existing database structure without necessitating a complete redesign.

The first four chapters present a model for bi-temporal data storage in databases. The remainder of the document describes the implementation of the DataStage "Hist-Op" ("Historisation Operator"), a single operator within the Parallel framework of IBM DataStage. The terms "operator" and "stage" in the context of DataStage parallel jobs and the "Hist-Op" are synonymous.

The "Hist-Op" uses the built-in parallelism of the DataStage Orchestrate framework and is platform independent, running identically on all supported hardware and OS platforms for DataStage versions 8.1 and above. The "Hist-Op" stage uses the DataStage Designer standard GUI to present a seamless and intuitive development environment.

Although DataStage is an extremely flexible application; the complexities involved in data historisation are not directly implementable within the tool as delivered. Historisation requires that all records belonging to a surrogate key are read into memory and then processed together iteratively and then recursively; DataStage remains a record-oriented tool and is not recursive on groups of records. The "Hist-Op" operator performs the complex historisation of data according to standard rules along with a set of user-specified options to achieve maximum flexibility. All of the frequently used data constellations are supported to allow the operator to be quickly implemented with likely configurations and data transformation processes requiring bi-temporal historisation.

The bi-temporal design presented in this document keeps development and maintenance overhead to a reasonable level, particularly when a package such as the "Hist-Op" performs the computations involved. It also offers the advantage that an existing database or data warehouse can be retrofitted to become a bi-temporal one; merely adding the required temporal columns (with default values for existing records) and changing the queries is sufficient to start, although only the new data will be truly bi-temporal.

 http://Hist-Op.com   16/09/2012

2.   Definition of Terms

Before delving into a description of 2-dimensional or bi-temporal data and historisation rules, it is necessary to ensure that core concepts are used unambiguously in the document, particularly those terms which often have multiple meanings depending upon context.

2.1.   Bi-Temporal and 2-Dimensional

For purposes of database historisation both terms have an identical definition and denote using 2 distinct dimensions to represent two different ranges of time and may be used interchangeably, this document will use the first term for the sake of brevity.

2.2.   Dates and Timestamps

At the heart of historisation, bi-temporal databases, bi-temporal models, and the "HistOp" lies the concept of time and how to measure, describe, define, determine and specify it. One would assume that such an important facet of our lives (and data processing) would be easy to define, but that is much easier said than done. We bandy about terms such as "time", "date", and "interval", "period" with interchangeable and contradictory meanings, and the more exotic "timestamp", "real-time", " timeslice", and "wall-clock time" can be even more confusing. Add the fact that each database vendor re-defines the words "datetime", "date", "timestamp" in their own manner and it is easy to see how confusion can reign.

Take the term "date" – in common language usage this refers to a day, i.e. "2012-08-05". But, depending upon the context, this could mean either the whole range of time during that calendar day, or a single point in time during that day such as midnight or noon.

This document will use the term **date** throughout to refer to a single point in time; depending upon the data types and context this is shown with varying levels of detail, ranging from "2012-08-05" as the least granular to "2012-08-05 18:46:22.923645" at the finest level of resolution. DataStage is limited to a precision of 6 digits to the right of the decimal as fractions of a second, so that will arbitrarily be our most accurate depiction of time; the atomic and most granular unit of time.

When specific data types are mentioned, they will be quoted; i.e. a "date" is a DataStage data type which contains only the day information while a "datetime2" is a SQL-Server specific data type.

2.3.   Inclusive & Exclusive temporal ranges

Chapter 4.5 discusses these two definitions in detail. An "inclusive" date range includes the terminus points, i.e. Mon-Fri is range of 5 days which includes the beginning day and the end day. An "exclusive" date range would not include the two terminus points. Thus "midnight to midnight" would only include one of the end points in the range as a half-inclusive, half-exclusive range. By default and unless explicitly stated otherwise, all date ranges in this document are assumed to be "inclusive". This is the more commonly used definition in any case, both in spoken English as well as in database systems.

2.4.   Use of "between" in queries

The SQL-92 standard first regulated the "between" clause and by now most databases have this encoded. The SQL DML clause "Where X between Y and Z" can be converted to "Where X >= Y AND X <= Z" for those databases where there is no "between". Note that this range is an inclusive one.

      http://Hist-Op.com   16/09/2012

2.5. Date Ranges - Tangent & Overlapping

When comparing the end point of one temporal range with the start point of the subsequent range there are 3 basic relationships as depicted below:

Separate	2001-01-01 2005-12-31		2012-01-01 9999-12-31
Tangent	2002-01-01 2005-12-31	2006-01-01 9999-12-31	
Overlapping	2002-01-01 2006-12-31		
	2005-01-01 9999-12-31		

Figure 2 - Date ranges

The top example shows two date ranges which are completely separate from each other and the bottom shows an overlapping range. These are relatively straightforward while the middle example can be a bit tricky.

The centre example shows two date ranges which are tangent to each other. This means that there are no possible dates between the end of the earlier one and the beginning of the later one; i.e. by adding the smallest temporal unit to the endpoint of the earlier range we would get the starting point of the later range. The more skeptical amongst the readers look at the center ranges and argue that there are many points in time between "2005-12-31" and "2006-01-01" and they would be right, at least partially. Whether two values are tangent to each other depends upon which units are being used.

If we use "timestamp" with 3 microseconds precision then "2012-08-05 18:30:00.123" and "2012-08-05 18:30:00.124" would be considered tangent; yet if our precision were 6 digits then "2012-08-05 18:30:00.123" and "2012-08-05 18:30:00.123999" would be considered tangent.

In Figure 2 - Date rangesFigure 2 above we used a "date" data type which contains no time component and since the smallest unit of time is a single day in that data type then two values are indeed tangent to each other.

2.6. Date Representation

Another common source of confusion is in the representation of dates. With so many national and international variations to choose from it is difficult to satisfy a global audience. This document will represent dates in the basic form YYYY-MM-DD HH:NN:SS.mmmmmm, modified to the level of precision required. This lists the components of a date in descending order of magnitude; the advantage of using this text representation of a numeric date is that the date order is the same whether sorted as an ASCII string or as the actual internal binary date representation in a timeline.

 16/09/2012

2.7. Null Values

Few fundamental concepts in the information technology field have caused as much confusion and grief as has the <null> value. In the strictest definition as used in databases, the <null> is used to represent a value of "unknown" and is distinct from any other possible value. <Nulls> can only be evaluated as true or false in an expression and cannot even be compared to themselves (i.e. one <null> is not equal to another <null>) to say nothing of adding or appending them to numbers/strings or performing any operation on them. Unfortunately the common use of the term "null string" causes problems, as this usually means an empty string of length 0, which is very different from a string that is <null>.

When dealing with date ranges either, or both, of the range terminus points may be <null> values in the database. A <null> used in a column representing the beginning of a range means "I don't know when this started" and when used at the end of a range means "The end isn't known but currently ranges to infinity". In this document these <null> values will be treated as if they were equal to the lowest possible representable date and as the highest possible date, respectively. Using DataStage limitations, this means that low-date and high-date are represented by the values "0001-01-01 00:00:00.000000" and "9999-12-31 23:59:59.999999" – then truncated to whatever level of precision the relevant data type allows.

Now that we've got the preamble and core concept definitions behind us, we can proceed to the heart of the material and ensure that we aren't using terms in a potential confusing or contradictory manner.

2.8. Naming the 2 temporal axes

The first temporal dimension, and the one most commonly used and found in many applications, is that of the data validity. This axis is represented by two dates denoting the beginning and the end of the data validity. In this document these two columns will be named "ValidFrom" and "ValidTo". An example of this axis is shown in Table 6.

The second temporal dimension concerns itself not with the data contents but with the effective technical period of records in the database. This range denotes when this record is considered to be active in the database and thus we will use the terminus points called "TechnicalFrom" and "TechnicalTo". It is deemed that the term "Effective" (which can be seen in some of the bi-temporal literature) is too easily equated to "Valid" and thus the "Technical" descriptor is preferable in order to avoid confusion.

 http://Hist-Op.com 16/09/2012

3.   "Historisation" defined

Historising data is a broad concept used to describe the process of keeping previous versions of data accessible at the same time, and in the same place (i.e. the same database table), as current data; historical data can be thought of as "old" data which has been superseded but remains accessible. A typical example of data which needs to be historised is customer address information; if the original address is merely overwritten in the database by a changed one then any information about the old address would be irrevocably lost. There are several goals in historising data, and many methods are available to reach them. The bi-temporal historisation implemented in this document allows fulfilment of the following goals:

- Every query yields identical results each time it is made, regardless of data changes.
- Data changes are auditable. Each change in data state is documented and traceable.
- Query access times for historised data are generally quite fast when using indices.
- Existing queries can be adapted by adding temporal ranges to the SELECT clause.
- Data redundancy is kept to a minimum.
- Broken records with invalid historisation periods are detectable and may be healed.
- Erroneous and partial loads as well as aborted processes can be rolled back seamlessly.
- Selecting data for offline archival is simple.
- Data partitioning is supported.
- Programming overhead for implementing the 2 temporal dimensions is reasonable.
- Historisation rules are understandable for the lay person.

The first goal, query repeatability, is perhaps the most important, as well as being the most difficult one to implement and deserves further explanation. A well-formed query in a temporal database must include 2 specific points in time in order to return correct results. In natural language, a query would be of the form:

"What address did Mr. Fudd live at on <u>May 1st, 2011</u> as we knew it on <u>January 10th, 2012?</u>"

- The first date represents the validity date in the data. This implies that every record must have a column representing the *ValidFrom* and the *ValidTo* dates to denote the temporal range that the record is valid.
- The second date represents the point in time for which the data is queried. This corresponds to a snapshot of the database contents at that time, and is termed the "effective" or the "technical" date. This document will use the latter nomenclature as it avoids potential confusion, using *TechnicalFrom* and *TechnicalTo* for column names.

The query from above translated into SQL for a bi-temporal table would then look like this:

```
SELECT      Address
FROM        Customer_Table AS Cust
WHERE       Cust.Contact_LName = 'Fudd'
            AND '2011-05-01' BETWEEN Cust.ValidFrom     AND Cust.ValidTo
            AND '2012-01-12' BETWEEN Cust.TechnicalFrom AND Cust.TechnicalTo;
```

If a query in a bi-temporal system omits the technical date, it is implicitly assumed that the query is for "now", which means to select those records where the *TechnicalTo* date is greater than the current date, usually this means that it is equal to high-date, i.e. selecting the technical records that are not terminated. In order for this query to be repeatable at a later point in time, the next repeated query must specify the technical effective time used for the first query.

                    16/09/2012

4. Historisation Examples

In order to illustrate the different historisation options we shall use a limited set of sample data, queries, tables and transactions in this document. There are quite a few different historisation rules and conditions, but these samples are sufficient to illustrate how a bi-temporal database table functions and how it stores data and allows it be retrieved.

4.1. Sample Scenario

4.1.1. Database Queries

Since the type of database that implements a bi-temporal model is usually driven (and financed) by those doing the reporting on the data, we'll start off with two exemplary queries of the type most often found in such systems, with an eye towards those which illustrate the effects and pitfalls of different data historisation methods. We'll then work backwards to the data storage mechanisms used:

- How many customers did we have in Arizona at the beginning of 2012?
- How much did Mr. Fudd at Acme purchase from us?

4.1.2. Initial table Contents

Our most simplistic model has only 2 tables, "Customer" and "Sales" which start out as follows:

Customer Table				
Cust_Key	**Name**	**Address**	**State**	**Contact Person**
1	Acme Inc.	Roadrunner Lane 1	AZ	Greasy the Weasel

Table 1 - Sample Customer Table

Sales Table					
Invoice	**Cust_Key**	**Product**	**Price**	**Qty.**	**Date**
1	1	Anvils	105.32	130	2011-06-06
2	1	Widgets, small	18.84	8012	2011-12-23
2	1	Exploding cigars	6.22	40	2011-12-23

Table 2 - Sample Sales Table

4.1.3. Database Transactions

We will keep the changes simple and to a minimum. These 2 changes are sufficient to illustrate the results of the different historisation models:

1. Mr. Elmer Fudd calls customer service on 2012-03-16 at 16:08 to tell them that effective March 1st Mr. Weasel has gone into retirement and that he is the new purchaser; this information is immediately entered into the customer database.

2. Mr. Fudd calls on 2012-07-02 at 08:00 to inform us that as of May 1st the company has relocated to Coyote Court 2 in New Mexico.

 http://Hist-Op.com 16/09/2012

4.2.   Sample Results with differing historisation

4.2.1.   No Temporal Information

While no self-respecting designer would dream of implementing this solution unless under duress, the simplest data model incorporates no time elements and no historisation at all. This, in data warehousing terms, is a Type 1 slowly changing dimension. Any changes to the database are applied to the original records and the initial values are overwritten. This step is irrevocable and the only possibility of recovering historical information is to retrieve backups, shadow copies or snapshots of the database/tables prior to changes. This is generally not practicable as it would result in a large storage requirement, plus in order to retrieve the old information a copy of the relevant database tables must be restored and queried in parallel to the operational system; this is usually not possible due to the large volumes of data involved.

After applying our 2 transactions from above (changes shown in italics and underlined), the resultant customer table would be:

Customer Table				
Cust_Key	**Name**	**Address**	**State**	**Contact Person**
1	Acme Inc.	*Coyote Court 2*	*NM*	*Elmer Fudd*

Table 3 - Customer Table, no historisation

Assuming the analyst asks the questions on 2012-12-01, the queries from chapter 4.1.1 would result in:

- How much did Mr. Fudd at Acme purchase from us (references Sales Table 2)?[*]

  Since we've overwritten the original purchaser information, we would select all sales to customer 1 from the sales table. This is incorrect, as Mr. Fudd hasn't purchased anything yet.

- How many "Widgets, small" have we sold in NM (references Sales Table 2)?[†]

  The select on the "Sales" table for all customers who purchased widgets would check the Customer table for State='NM' and erroneously return 8182; but in actuality Acme purchased the widgets in 'AZ' and none in 'NM' so the correct result should be 0.

These two examples illustrate that without historisation the data is only of limited use for reporting purposes. The database has only one temporal state, the current one. Data is only valid for current queries and is absolutely useless for retrieving any data which has changed in any form over time. One knows neither when the data became valid or even whether there have been any other values in the past.

While the model has the dubious distinction and advantage of simplicity, it cannot be used for any practical applications that require any temporal component.

---

[*] `SELECT SUM(Price*Quantity) FROM   Sales JOIN Customer on Sales.Cust_Key=Customer.Cust_Key WHERE Customer.Contact_Person = 'Elmer Fudd'` returns 164886.48
[†] `SELECT SUM(Quantity) FROM Sales JOIN Customer on Sales.Cust_Key=Customer.Cust_Key WHERE Customer.State = 'NM'` returns 8182

    http://Hist-Op.com   16/09/2012

4.2.2.   1-Dimensional Temporal

With this type of data historisation, each table has two historisation columns, a *ValidFrom* date and a *ValidTo* date that delineate the validity range for that record. This is what is referred to as a "Type 2 SCD (**S**lowly **C**hanging **D**imension)"[*] in the data warehousing world but has been used, in various forms, since the advent of record keeping - be it electronic or on paper in a ledger. With these new columns, the original customer database looks like this:

Customer Table						
Cust_Key	**Name**	**Address**	**State**	**Contact Person**	**Valid From**	**Valid To**
1	Acme Inc.	Roadrunner Lane 1	AZ	Greasy the Weasel	2010-01-01	9999-12-31

Table 4 - Customer Table, SCD Type 2

Note that the *ValidTo* date, when not known, is set to a valid date far in the future. This represents an "unterminated" record, one which is currently valid and will remain valid indefinitely until such time as it is changed. This value is usually implemented by using the highest representable date, but in some implementations this column is made nullable and the <null> value is used instead of a high-date. While this is technically correct, as <null> represents an unknown or indeterminate value, it is more efficient to use a value that can be calculated with (<null>s are not less than, greater than, or equal to anything else, not even another <null>). Using an actual date makes queries much simpler and also allows for more optimal indexing and faster access in database systems.

Query without nulls	`Select * from Customer where '2012-08-05' between ValidFrom and ValidTo;`
(a) Query with <null>s	`Select * from Customer where ValidFrom is not null and ValidTo is not null and '2012-08-05' between ValidFrom and ValidTo;`
(b) Query with <null>s	`Select * from Customer where '2012-08-05' between COALESCE(ValidFrom,'0001-01-01') and COALESCE(ValidTo,'9999-12-31');`

Table 5 - Comparison of queries with <null>s

The highest date which can be represented in DataStage is "9999-12-31 23:59:59.999999" and we will use this value interchangeably with "high-date" in this document. Likewise "low-date" is equivalent to "0001-01-01 00:00:00.000000". Other implementations may use different values for either of these points in time, particularly if the minimum or maximum values of the physical data types allow for more, or less, precision.

After applying our 2 transactions from 4.1.3 the resultant Customer table (changes shown in italics and underlined) is:

Customer Table						
Cust_Key	**Name**	**Address**	**State**	**Contact Person**	**Valid From**	**Valid To**
1	Acme Inc.	Roadrunner Lane 1	AZ	Greasy the Weasel	2010-01-01	*2012-03-15*
1	Acme Inc.	Roadrunner Lane 1	AZ	*Elmer Fudd*	*2012-03-16*	*2012-04-30*
1	Acme Inc.	*Coyote Court 2*	*NM*	Elmer Fudd	*2012-05-01*	*9999-12-31*

Table 6 - Customer Table, SCD Type 2, after modification

In this type of historisation the *Cust_Key* is no longer unique and the actual physical primary unique key, if present, is another column, thus *Cust_Key* has become what is referred to as a "Surrogate Key" (using data warehousing terminology). In the table above there are now 3 distinct records for Acme Inc. showing the 3 distinct states that this customer has had over

---

[*] Kimball, Ralph; Ross, Margy (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling (Second Edition)*. Indianapolis, IN: John Wiley & Sons. ISBN 0-471-20024-7, 2002.

 http://Hist-Op.com   16/09/2012

time. The *ValidFrom* and *ValidTo* columns contain the time ranges during which each of the records is valid. It is important to note that these ranges must not overlap, i.e. if a record has a *ValidTo* 2012-07-08 then the subsequent *ValidFrom* for the same surrogate key must be at least 2012-07-09. chapter 4.5 describes "inclusive" and "exclusive" ranges and the "Hist-Op" options available for setting these types. If the *ValidFrom*/*ValidTo* columns are SQL "date" types then this implies that the shortest period of validity is one day, there cannot be 2 different values for one day. Should a finer granularity than one day be required, then the *ValidFrom*/*ValidTo* columns need to have a "timestamp" data type rather than a "date". The "Hist-Op" does cater for smaller time periods, but in this document we will stick with simple dates for these validity ranges and use "timestamp" for the technical ones.

Assuming the analyst asks the questions on 2012-12-01, the queries from 4.1 would result in:

- How much did Mr. Fudd at Acme purchase from us (references Sales Table 2)?[*]

  This query would now work correctly, as each entry in the Sales table has a date, with that we'd check the customer table and see if, for that date, Mr. Fudd was the purchaser.

- How many "Widgets, small" have we sold in NM (references Sales Table 2)?[†]

  Just as above, we would traverse the Sales table looking for widgets, then for each found item use the date of that sale to get the appropriate company record and check to see if it matches "NM".

These results are now correct, each of our sample queries works as expected. The database is compact, relatively straightforward for SQL-knowledgeable analysts to understand and for them to make ad-hoc queries with. As a 1-dimensional system it accurately portrays the data, but only for one point in "real" time. In order to cater for the additional time component we need to add another dimension to our model.

In other words, the 1-dimensional storage of data answers the question of "what" the data looks like and now we need to add the dimension of "when" things happened in order to present a complete and accurate holistic view of the data.

4.2.3. 2-Dimensional temporal (Bi-Temporal)

Let us return to the database update #3 from chapter 4.1.3 where Mr. Fudd calls on 2012-07-02 at 08:00 for the address change effective May 1st. If an analyst makes a query about Widget sales in NM on June 1st the database will show Acme's address as "AZ", but the same query on August 1st will show Acme's address as "NM" for the same *ValidDate*. This is confusing since the database, after data update on 2012-07-02, will show the address change as being effective May 1st and the analyst will wonder why the numbers (perhaps used for sales forecasts or bonus calculations) have changed strangely since the last time the database was queried.

This is just one typical example where both the data validity date as well as the technical effective date both need to be used together in order to give a consistent view of the data. As

---

[*] `SELECT SUM(Price*Quantity) FROM    Sales JOIN Customer on Sales.Cust_Key=Customer.Cust_Key WHERE Customer.Contact_Person='Elmer Fudd' AND '2012-12-01' BETWEEN Customer.Valid_From AND Customer.Valid_To`
[†] `SELECT SUM(Quantity) FROM Sales JOIN Customer on Sales.Cust_Key=Customer.Cust_Key WHERE Customer.State='AZ' AND '2012-12-01' BETWEEN Customer.Valid_From AND Customer.Valid_To`

 http://Hist-Op.com   16/09/2012

is often the case with legacy systems, the source data might not contain the additional technical dates required. In this case this *TechnicalFrom* column may be filled at insertion time from the system date. The technical range thus generated is only an approximation but is sufficient for processing legacy data in 2-dimensional retrofitted databases.

The bi-temporal historisation model guarantees that (a) no data is ever overwritten, deleted or retroactively changed and (b) regardless of data changes, the same query will always return the same results. This is also a requirement of many auditing schemes which are a component of recent regulatory requirements in Europe (Solvency II[*] for Insurance companies and Basel III[†] in the banking sector to name just two auditing requirements, in addition to the older HIPAA[‡] and Sarbanes-Oxley[§]).

In order to implement this additional dimension, we need to add yet another 2 columns to our bi-temporal tables. These 2 new columns represent the technical time range that the record is valid. We'll use the labels *TechnicalFrom* and *TechnicalTo* for clarity, representing the date at which the system stores the data and until which it is valid in the system. This is commonly called the "Effective" date but we'll avoid that term here to prevent confusion.

Customer Table

Key	Name	Address	State	Contact Person	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Acme Inc.	Roadrunner Ln. 1	AZ	Greasy the Weasel	2010-01-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59

Table 7 - Sample customer Table SCD Type 2

These two new columns are shown as "timestamp"'s and this is the most common data type for the effective technical date as it allows for multiple states to be represented per calendar day. Unless "Exclusive" (chapter 4.5) ranges are chosen these values may not overlap, thus using a "date" data type for this range would mean that multiple changes per day would not be supported.

Several implementations that the author has worked with utilize micro- or millisecond increments to ensure that historisation can occur in sub-second time slices, but this is primarily due to the source delivering this level of accuracy and a general reluctance to discard this information. Most analysts are more than happy to write a query on a technical time of "2012-01-01 08:30" rather than "2012-01-01 08:29:18.754261". As with the Valid date ranges, an unterminated status can be represented by a <null> value but it is more efficient and less error-prone to use a high-date to denote a record as being active indefinitely.

Given these new columns for bi-temporality, the results are:

Customer Table

Key	Name	Address	State	Contact Person	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Acme Inc.	Roadrunner Ln. 1	AZ	Greasy the Weasel	2010-01-01	9999-12-31	2010-01-01 12:00:00	2012-03-16 16:07:59
1	Acme Inc.	Roadrunner Ln. 1	AZ	Greasy the Weasel	2010-01-01	2012-03-15	2012-03-16 16:08:00	9999-12-31 23:59:59
1	Acme Inc.	Roadrunner Ln. 1	AZ	Elmer Fudd	2012-03-16	9999-12-31	2012-03-16 16:08:00	2012-07-02 07:59:59
1	Acme Inc.	Roadrunner Ln. 1	AZ	Elmer Fudd	2012-03-16	2012-07-01	2012-07-02 08:00:00	9999-12-31 23:59:59
1	Acme Inc.	Coyote Court 2	NM	Elmer Fudd	2012-07-02	9999-12-31	2012-07-02 08:00:00	9999-12-31 23:59:59

Table 8 - Customer Table bi-temporal historisation

Note that there is a not insignificant overhead in storage requirements, query times and the size of indices when doing a bi-temporal historisation. Instead of the original 2 non-historised or the 4 Type2 SCD records we now have 6 records in total, plus the additional historisation columns in each row.

---

[*] See http://en.wikipedia.org/wiki/Solvency_II_Directive for an overview.
[†] See http://www.bis.org/bcbs/basel3.htm for details.
[‡] Health Insurance Portability and Accountability Act, see http://www.gpo.gov/fdsys/pkg/BILLS-104s1028is/pdf/BILLS-104s1028is.pdf
[§] See http://www.soxlaw.com for details.

    16/09/2012

It may be noted that either of these two temporal ranges, taken separately, may have overlapping ranges; but for any one technical point in time coupled with one data point in time there will be either 1 or 0 matching records.

Returning to the analyst query from the beginning of this chapter, the user now has to query the database for "Show me the Widgets in NM for the effective date of June 1st" and then repeat the query with an effective date of August 1st. This time around the 2 results will be different from each other and both correctly show the state of data for their respective time periods.

By introducing the second time dimension into the database we have effectively built in a rolling temporal snapshot of the database contents. Selecting records from this table where "{date} is between *TechnicalFrom* and *TechnicalTo*" is the equivalent of asking "Show me what the database contents were on {date}". This selection, coupled with first temporal dimension as described in 4.2.2 gives full coverage of both logical and physical states for any date on temporal axes.

4.3.   *Change detection*

In many cases the source of data is not a relational database (with its inherent capabilities for strict data typing and uniqueness constraints) but some other processing system, be it a web service or an extract from a mainframe transactional program. Even when using a database as a source, it is not only possible but probable that more than one record per surrogate key is processed as part of a delivery. A typical example of this type of delivery is a host-based system which sends data records each and every time a record is accessed in the processing screens, regardless of which field, if any, has been modified.

If a customer calls up to change their address when using such a system and the data is entered, a new record R_1 is generated; if the customer calls up again to change their telephone number, another record R_2 is generated (which cumulatively contains both changes) and if the customer calls up again to change their banking information but realize halfway through the call that the information isn't complete and cancels the call, the system generates yet another record, R_3 which is identical to R_2 except that the record-accessed-date value is different. Then some external program automatically changes a column "Sales Rep" to a new responsible sales representative (triggered by the address change), generating a new record R_4.

These 4 records are then packaged up into a text extract file for use by downstream systems, one of which is our data warehouse. Let us assume that our system doesn't process sales data - we would receive 4 records, R_1 and R_2 are legitimate "delta" records, but R_3 has no changes at all and R_4 has a change which we don't recognize as one, since the column which has changed isn't processed by our warehouse system.

The "Hist-Op" operator allows the user to specify which columns in a given record are to be used for change-data-detection (CDC). These columns are compared internally among records with the same surrogate key in order to determine if two or more records have the same logical data content and can be treated as being identical.

Often records contain columns that should not be used for change detection; typically these are columns such as physical record-ids, processing dates or other system-generated unique values that are not part of the record's data payload. Columns not used by the target system but which are present in each row may also need to be excluded from the comparison, as in the "Sales Rep" field in the example above.

 http://Hist-Op.com   16/09/2012

4.4. Temporal reduction

If two or more records are identical as defined by the CDC columns (such as records R_2, R_3 and R_4 from the previous chapter) yet they have an overlap in their data validity dates (since only unterminated records with *TechnicalTo* set to high-date are processed, they will de-facto always have overlapping timeframes in their technical dates) then those records are candidates for temporal reduction, which is just a fancy technical term for merging the records together. This reduction can result in very significant processing time and storage space savings, as without temporal reduction each record would need to be handled and historised separately. Particularly when receiving transactional data with many effectively identical records adding this temporal reduction can reduce a large percentage of the incoming data.

While temporal reduction is a useful feature, there are certain cases where this temporal reduction is not desired, as would be the case if auditing rules stipulate that the warehouse or database must contain each and every input record regardless of content. The "Hist-Op" operator allows the user to turn off temporal reduction both in the source data (reduction is performed prior to any historisation actions) and/or as part of the processing; see "*Temporally reduce Input*" and "*Temporally reduce Output*" in chapters 5.3.4.4 and 5.3.5.4 respectively.

Product Table

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	2010-12-30	2010-01-01 11:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-09-11	9999-12-31	2010-01-01 12:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2011-09-10 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2012-05-23	9999-12-31	2012-05-23 14:0100	9999-12-31 23:59:59
1	Unobtanium	Metallic compound	448472	0	2012-03-12	9999-12-31	2012-03-02 09:35:00	9999-12-31 23:59:59
1	Unobtanium	Metallic compound	448472	0	2010-02-02	9999-12-31	2012-04-01 18:14:00	9999-12-31 23:59:59
1	Unobtanium	Metallic compound	448472	0	2012-07-01	9999-12-31	2012-05-01 10:35:00	9999-12-31 23:59:59
1	Unobtanium	Metallic compound	448472	0	2012-07-01	9999-12-31	2012-05-01 10:35:00	9999-12-31 23:59:59

Table 9 – Product table, not reduced

Product Table

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	2010-12-30	2010-01-01 11:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2010-01-01 12:00:00	9999-12-31 23:59:59
1	Unobtanium	Metallic compound	448472	0	2010-02-02	9999-12-31	2012-03-02 09:35:00	9999-12-31 23:59:59

Table 10 – Product table, temporally reduced

The examples above show how input data is temporally reduced. Note that the two entries for Thiotimoline are identical (in terms of CDC) but since the first record is valid until 2010-12-30 while the next entry is valid from 2011-01-01 they cannot be temporally combined, as there is a gap between the time ranges. If we now add the following new entry to this table:

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-06-01	2011-07-01 18:00:00	9999-12-31 23:59:59

Without temporal reduction during historisation processing we would get the following result:

Product Table

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	2010-12-30	2010-01-01 11:00:00	2011-07-01 17:59:59
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	2011-05-31	2010-01-01 11:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2010-01-01 12:00:00	2011-07-01 17:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-06-02	9999-12-31	2010-01-01 12:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-06-01	2011-07-01 18:00:00	9999-12-31 23:59:59

Table 11 – Product table, no output reduction

 16/09/2012

But if temporal reduction is turned on, the result would be a more legible:

Product Table								
Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	9999-12-31	2010-01-01 11:00:00	2011-07-01 17:59:59
1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2010-01-01 12:00:00	2011-07-01 17:59:59
1	Thiotimoline	Temporal substance	1382	403	2010-01-01	9999-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59

Table 12 – Product table, output reduction

One of the precepts of bi-temporal historisation as described here is that the system allows an audit trail of all changes. The temporal reduction process resides in a gray area, while the process doesn't change existing data, it does combine records from the input stream and thus breaks the audit trail for input data rule.

One additional possibility in temporal reduction is when a new data record can be temporally reduced into an existing record in the table. This means that the "Hist-Op" would automatically deal with multiple runs with identical source data (which occurs more often than any developer would care to admit) and not historise identical records. Quite often bi-temporal tables have one or two additional columns that are used for auditing/traceability and contain a reference to a run-id for the insert and a run-id for the termination. When such columns are used, the temporal reduction of duplicate records would not be allowed; one would get two identical records , the first of which would be terminated by second record, the *run-id-insert* and *run-id-terminate* column values would differ, but the data would be same.
If this functionality has not been implemented in the tables and there are no auditing constraints disallowing temporal reduction with existing data, then the "Hist-Op" option "*temporally reduce references*" (chapter) in the Input Section can be specified.

4.5.   "Inclusive" and "Exclusive" time ranges

This is a fundamental design question that needs to be addressed and irrevocably decided upon right at the outset of implementing any form of date ranges.

- Inclusive ranges
  The start and end terminus points of the range are considered to be part of the range itself. For example, Monday through Friday are working days. This inclusive list means that both Monday and Friday are part of the list, i.e. Mon-Fri encompasses 5 days.

- Exclusive ranges
  The start and end terminus points of the range are considered to be outside of the range itself. If an exclusive range is 2012-01-01 through 2012-01-31 then the date "2012-01-31" is not considered as being part of the range. An exclusive range of Mon-Fri would only encompass 3 days (Tue, Wed and Thu).

The type of range used has effects on how the "Hist-Op" operator detects and computes records with overlapping, tangent and non-overlapping ranges in both time dimensions. The "Hist-Op" allows the Validity dates and the Technical dates to have different range types. In order to illustrate the difference between the two types, we'll use the following two new records and historise them accordingly:

    16/09/2012

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	9999-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Inclusive ranges (the default and more widely implemented option) for both dimensions:

Product Table								
Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-01-09	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Table 13 – Product table, inclusive range example

Exclusive ranges for both dimensions results in the following:

Product Table								
Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:30:30
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-01-10	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Table 14 – Product table, inclusive range example

Another example, with more obvious effects, would be

Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-01-10	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

The Inclusive range results are:

Product Table								
Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-01-09	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Table 15 – Product table, inclusive range example 2

While the Exclusive range results are:

Product Table								
Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Thiotimoline	Temporal substance	1382	403	2010-06-01	2011-01-10	2011-07-01 18:00:00	9999-12-31 23:59:59
1	Thiotimoline	Temporal substance	1500	50	2011-01-10	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Table 16 - Product table, exclusive range example 2

4.6. Historising Deletions

One of the basic rules in historisation stipulates that data, once entered into the system, is not physically deleted. The only way that records can ever be removed from the database is as part of an archival process (chapter 4.8). Thus we only perform logical deletions rather than physical ones and for purposes of bi-temporal historisation the insertion and deletion logic is identical. The only difference is the resultant output code for the new record in the output link, either "I" (or the override value) for inserts and "D" (or the override value) for deletions. There are two routes to take in implementing deletions in the tables and both are chosen after the "Hist-Op" operator process the data:

- Drop the "D" type records from the output stream
  Any records that were terminated as part of this deletion are updated in the database as normal, but no new record is inserted for the deletion. This keeps the number of records in the database smaller, but the information about the actual deletion action is lost (if the deletion doesn't historise any records then nothing happens, if it historises existing records then the deletion can be inferred by the existence of those records terminated without a subsequent new record)

 16/09/2012

- Write the "D" type records but specify a status column
  This action ensures that no information is lost, but requires yet another column in the table to hold the status of the record – there are 3 states possible (a) deleted, (b) active or (c) historised. Queries on the data must also include a clause limiting results to "`WHERE HistStatus <> 'D'`". Note that a status of historised is redundant, as this is information also reflected by the *TechnicalTo* date being less than high-date. On the other hand, this status column can be indexed and may be more efficient than doing a date range check with "between" SQL logic when there are many historised entries in the table.

4.7.　Data and Query Prerequisites and Implications

Implementing a standard bi-temporal model requires that each record

- Is never updated with the exception of termination/historisation
  The only field ever changed after the initial insert is the *TechnicalTo* date. Initially this is high-date or <null> and when the record is terminated it is updated to an actual date that represents the point in time where the record's validity ends. Once this *TechnicalTo* date has been changed the record may not be modified again. Never again. No Exceptions. Manual updates of values without being bound by the rules involved in historisation would invalidate the whole scheme.
- (a) Has a *ValidFrom* date that is less than the *ValidTo* date (inclusive range), or
- (b) Has a *ValidFrom* date that is less than or equal to the *ValidTo* date (exclusive range)
  The *ValidFrom* date may never be higher than the *ValidTo* value. <Null> values are considered to be low-value in *ValidFrom* and high-value in *ValidTo* for comparison purposes. The two values may be equal, i.e. for a 1-day validity period if the *ValidFrom/ValidTo* data type is "date".
- (a) Has a *TechnicalFrom* date that is less than the *TechnicalTo* date (inclusive range), or
- (b) Has a *TechnicalFrom* date that is less than or equal to the *TechnicalTo* date (exclusive range).
  The rules are the same as those defined above for *ValidFrom/ValidTo*.

Given these prerequisites and assuming the default standard of inclusive date ranges, the SQL Pseudo-Query "`SELECT * FROM {table} where {Validity Date} between ValidFrom and ValidTo AND {Effective Date} between TechnicalFrom and TechnicalTo;`" will always return either 0 or 1 records. In other words, for any given point in "real time" (technical time) and for any given logical data point in time there will only be 1 valid record per key or none at all. This query and the result set of only 1 data record is at the heart of bi-temporal database design.

4.8.　Data archival

No matter how much storage costs drop and how efficiently mass storage can be accessed, there will always come a time when one has to let go of data and move it from the active system to an offline archive. At what point in time this pruning needs to be done is decided upon by many factors and each organization needs to make an individual decision for each application or database. Often there are regulatory restrictions that dictate how data must be available and also when data may no longer be available. Nonetheless, it is necessary to factor the archival process into the design and specifications of a database system in order to minimize disruptions in production and make sure that the archival algorithm doesn't require re-writing all of the surviving records or repartitioning of the database.

 http://Hist-Op.com　16/09/2012

As can be expected when choosing which criteria to use for deciding which data needs to be archived, the primary factor is a time-based one: age.

With a Type 2 SCD record using just one date range the methodology for archiving data before a given date could be clear-cut – "SELECT all records where the *ValidTo* is less than {archival date}".

Bi-temporal data adds another time dimension which needs to be accounted for when archiving data. One has additional flexibility, necessitating a decision as to which time axis needs to be used for the archival decision. The usual choice is that archival is done using the same date for both dimensions, resulting in a query such as

```
SELECT all records where TechnicalTo is less than {archival date}
AND where the ValidTo is less than {archival date}
```

The first part selects only records that were terminated prior to the archival date and the second part selects those whose validity period also ended prior to the archival date. The effect of archiving data is that all subsequent queries must use a point in time for the technical axis which is after the date of archival.

4.9. Historising Dependencies / Multiple tables

The true complexities and technical difficulties in a bi-temporal database system comes when ensuring temporal consistency over multiple tables. While this document doesn't cover the implementaion of these dependancies, we will offer an example:

Customer Table

Key	Name	Address_SID	Contact Person	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Acme Inc.	1	Greasy the Weasel	2010-01-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59
1	Ajax Corp.	1	Dr. Vulter	2005-06-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59

Table 17 - Sample dependancy customer table

Address Table

Address_SID	Address	State	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Roadrunner Ln. 1	AZ	2005-61-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59

Table 18 - Sample dependancy address table

 Now Ajax Corp. moves offices to Delaware on 2012-04-06. In order to correctly depict this new state of affairs, the tables would need to look as follows:

Customer Table

Key	Name	Address_SID	Contact Person	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Acme Inc.	1	Greasy the Weasel	2010-01-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59
1	Ajax Corp.	1	Dr. Vulter	2005-06-01	9999-12-31	2011-01-01 12:00:00	2012-04-05 23:59:59
1	Ajax Corp.	2	Dr. Vulter	2005-06-01	2012-04-05	2012-04-06 00:00:00	9999-12-31 23:59:59
1	Ajax Corp.	3	Dr. Vulter	2012-04-06	9999-12-31	2012-04-06 00:00:00	9999-12-31 23:59:59

Table 19 - Sample dependancy customer table after historisation

Address Table

Address_SID	Address	State	Valid From	Valid To	TechnicalFrom	TechnicalTo
1	Roadrunner Ln. 1	AZ	2005-61-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59
2	Roadrunner Ln. 1	AZ	2012-06-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59
3	Wilmington Rd. 2	DE	2005-61-01	9999-12-31	2011-01-01 12:00:00	9999-12-31 23:59:59

Table 20 - Sample dependancy address table after historisation

 http://Hist-Op.com 16/09/2012

5. "Hist-Op" Operator

5.1. Operator Overview

The "Hist-Op" historisation operator is included in job design s just as any other stage.

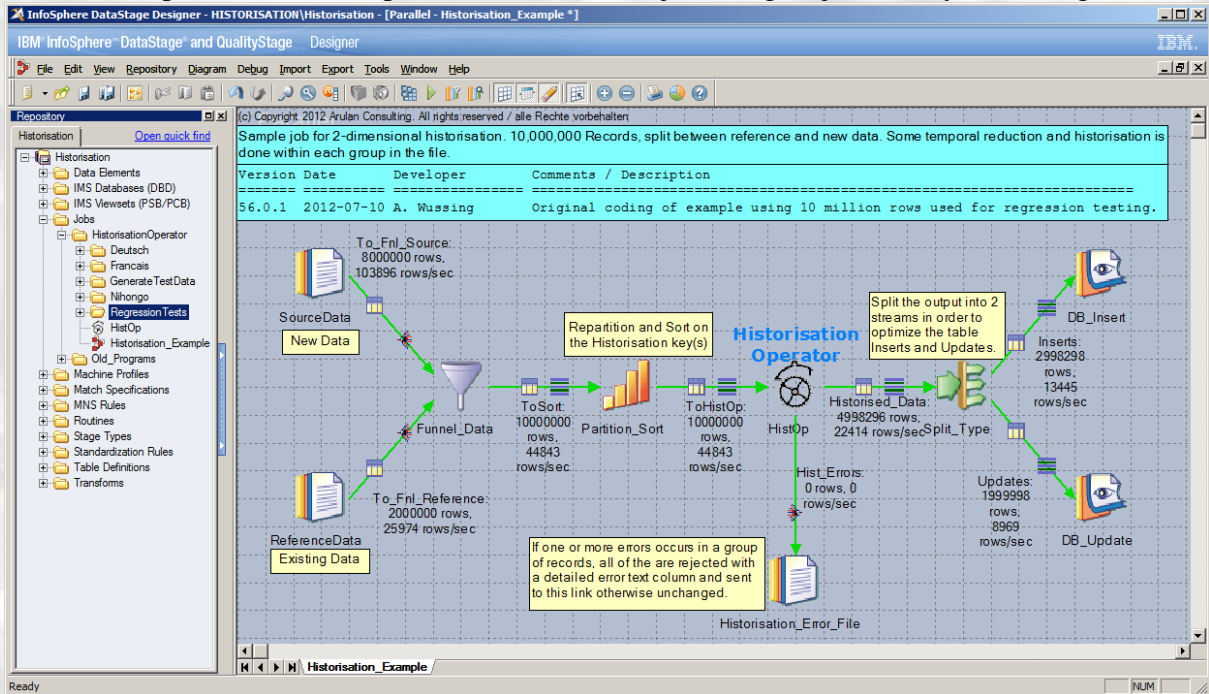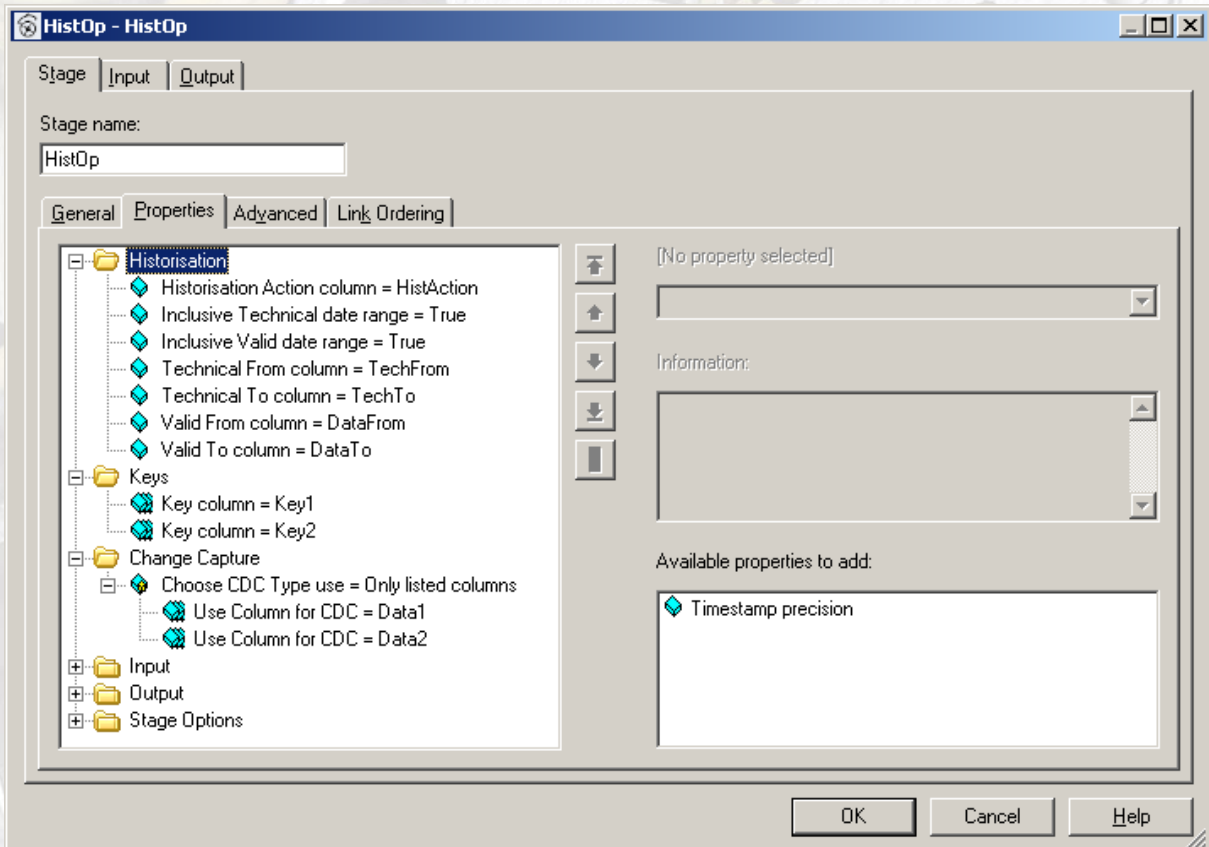**Figure 3 - Example job screenshot**

Figure 4 - Example job Operator declaration screenshot

 16/09/2012

The two screenshots shown overleaf illustrate how the "Hist-Op" stage is used in a job and what the DataStage Designer GUI interface to specify the stage options looks like at version 8.7. In this chapter we will address and explain the technical details of the "Hist-Op" and go through each of the stage options.

5.2.   Using "Hist-Op" from the Designer

The "Hist-Op" operator consists of 2 parts: One is the actual executable (in the form of a library), installed on disk in a location accessible to DataStage via the $PATH settings at runtime. The other component is the GUI declaration and is packaged as part of a DataStage export file which also includes a sample program (chapter 7). This is imported into the project repository and installs the Historisation Operator in the following location

```
Stage Types ► Parallel ► Processing ► Hist-Op
```

The "Hist-Op" operator can then be dragged-and-dropped into a Parallel job in the Designer from this repository location and used just as any other Parallel stage; alternatively it can be dragged-and-dropped into the palette "Development" or "Favorites" group for easier access in the Designer.

The path to the library file containing the executable code depends on the installation platform. The installation procedure is documented in chapter 8.
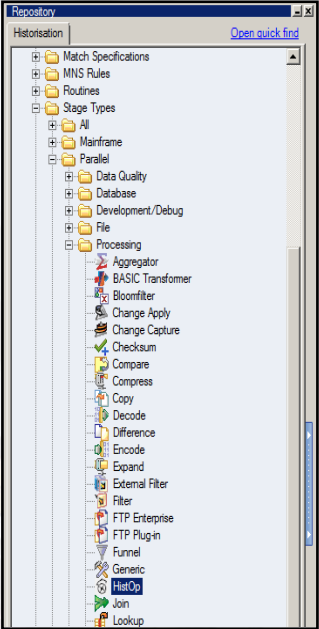
Figure 5 - "Hist-Op" location

5.3.   "Hist-Op" Stage settings

The settings and options for the "Hist-Op" are grouped into 6 categories:
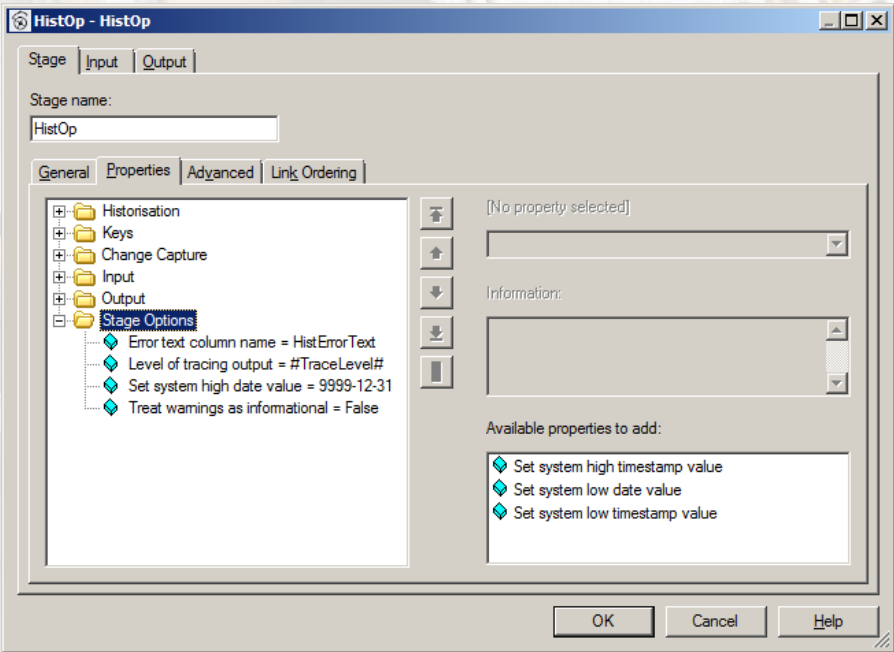
- Historisation

- Keys

- Change Capture

- Input

- Output

- Stage Options

Figure 6 – Hist-Op Categories

16/09/2012
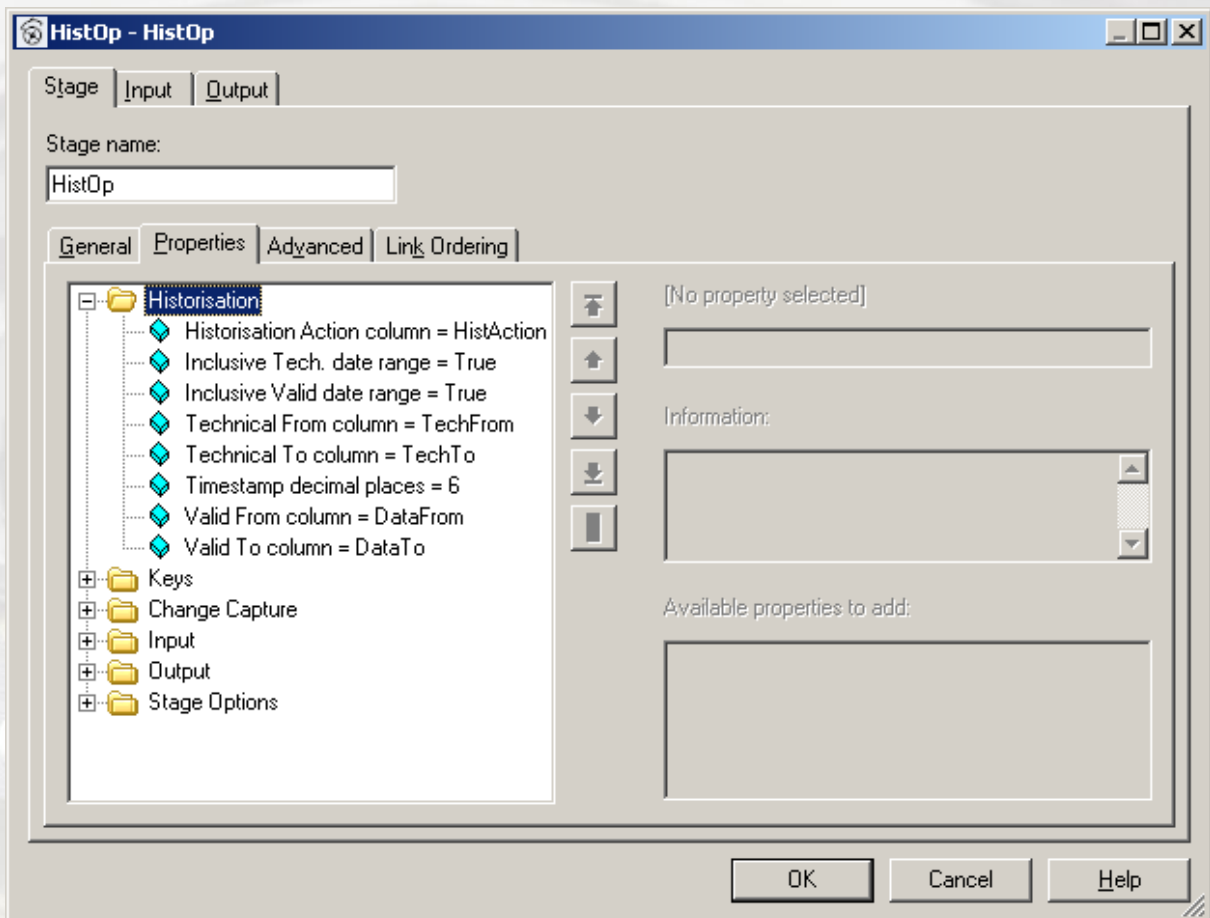
5.3.1. Historisation category

Figure 7 – Historisation Category

The historisation category defines those columns which are used for historisation or directly affect the behaviour of the historisation process. There are 7 mandatory fields and one optional field in this category.

5.3.1.1. "Historisation Action column"

The Historisation Action column specified here is the name of an input column of type string which may contain one of only 3 possible values at runtime. The default values are given here in square brackets, but these values can be overridden in the "Input" category described in chapter 5.3.4:

- Insert/Data record ["I"]
  Data records are synonymous with insert records, they represent new data which needs to be inserted into the tables and which has not yet been historised. This value is also the output string value for inserts, but the option *"HistAction insert string"* in chapter 5.3.4.2 can override the default setting.

- Reference record ["R"]
  These are reference records read from the target table. Only non-terminated records need to be read from the table (records where *TechnicalTo* is high-value or <null>) as historised records aren't processed. By default any reference records that haven't been terminated by the "Hist-Op" are not passed on to the output link. This behaviour can be modified by the option *"Output unchanged records"* (chapter 5.3.5.3) if required. By default the

 16/09/2012

HistAction column uses "R" for this value, but the option *"HistAction reference code"* (chapter 5.3.4.3) can be set to override the default setting.

- Delete record ["D"]
  This is a sub-type of Insert / Data records, these are new records which perform a logical data deletion in the historised table. A detailed description of the delete historisation action is given below. By default the HistAction column value is "D" for deletions, but the option *"HistAction delete code"* (chapter 5.3.4.1) can be set to override the default setting. Refer to chapter 4.6 for a description of the differences between deletions and insertions.

5.3.1.2. *"Inclusive Tech. range"*

This parameter, which defaults to true, defines how the *TechnicalFrom* and *TechnicalTo* range terminus values are treated.

When true, this range is considered to be inclusive of its beginning and end points; the values defining the boundaries are included within the range. This has implications on whether the operator considers two ranges to be overlapping or tangent. With inclusive ranges, *TechnicalTo* value of "2011-12-31" would be considered as being next to, or tangent, to the next record's "2012-01-01" in *TechnicalFrom.* If this range were set to "exclusive" then they wouldn't be next to each other, in fact there would be a range of "2011-12-31" - "2012-01-01" that would fit between the two.

This setting is usually determined by the database architect during the design phase and may not be changed once the system is specified. All tables loaded using the "Hist-Op" need to use the same value for this parameter. Records are temporally reduced and historised differently according to which setting is chosen. See chapter 4.5 for a detailed description of inclusive and exclusive date ranges.

5.3.1.3. *"Inclusive Data range"*

This parameter, which defaults to true, defines how the *ValidFrom* and *ValidTo* terminus values are treated. See chapter 5.3.1.2 *"Tech. range is inclusive"* for a description of the settings.

5.3.1.4. *"Technical From column"*

This is set to the name of the column that contains the *TechnicalFrom* value for the data. The value of the column represents the beginning of the effective period for the record and this time dimension represent the real-time axis, i.e. times that relate to when the data was actually present in the database. The column may be either "date" or "timestamp", can be nullable and "timestamp"'s may be declared with microseconds.

The column is paired with the "*Technical To column*" and both must be of the same type. When "timestamp"'s are used if one contains microseconds and the other does not, the operator uses no microseconds for both columns. When microseconds are specified, the value of "*microsecond decimal places*" is used to determine how many digits of precision the operator will work with. <null> values are treated as being equivalent to "0001-01-01 00:00:00.000000" (low-date) and reduced to the precision and data type has been specified.

5.3.1.5. *"Technical To column"*

This is set to the name of the column that represents the *TechnicalTo* value for the data. This is the end of the period of validity for the record. See "*Technical From Column*" description above for the details on this paired column. <null> values are treated as being equivalent to "9999-12-31 23:59:59.999999" (high-date) and reduced to the precision and data type has been specified.

 16/09/2012

5.3.1.6. "Timestamp decimal places"

This is another parameter that has an impact on how records are historised. If the columns are of type "date "or "timestamp" with no microseconds then this setting is ignored, but when a "timestamp" with microseconds is utilized then this setting specifies how many of the available 6 microsecond digits of precision are to be used for historisation computations. The processing will create times which are "adjacent" to each other, the value of "2012-07-01 11:30:00.123" is considered adjacent to "2012-07-01 11:30:00.124" when the *timestamp decimal places*" setting is 3, but is not considered adjacent when the setting is 4 (the second value would need to be "2012-07-01 11:30:00.1231" when using 4 digits).

5.3.1.7. "Valid From column"

This is set to the name of the column that contains the *ValidFrom* value for the data. This is the beginning of the period of validity for the record. The column may be either "date" or "timestamp", can be nullable and may be declared with microseconds.

The column is paired with the "*Valid To column*" and both must be of the same type. When "timestamp"s are used if one contains microseconds and the other does not, the operator uses no microseconds for both columns. When microseconds are specified, the value of "*microsecond decimal places*" is used to determine how many digits of precision the operator will work with. <null> values are treated as being equivalent to "0001-01-01 00:00:00.000000" (low-date) and reduced to the precision and data type has been specified.

5.3.1.8. "Valid To column"

This is set to the name of the column that represents the *ValidTo* value for the data. This is the beginning of the period of validity for the record. See "*Valid From Column*" description above for the details on this paired column. <null> values are treated as being equivalent to "9999-12-31 23:59:59.999999" (high-date) and reduced to the precision and data type has been specified.

 16/09/2012
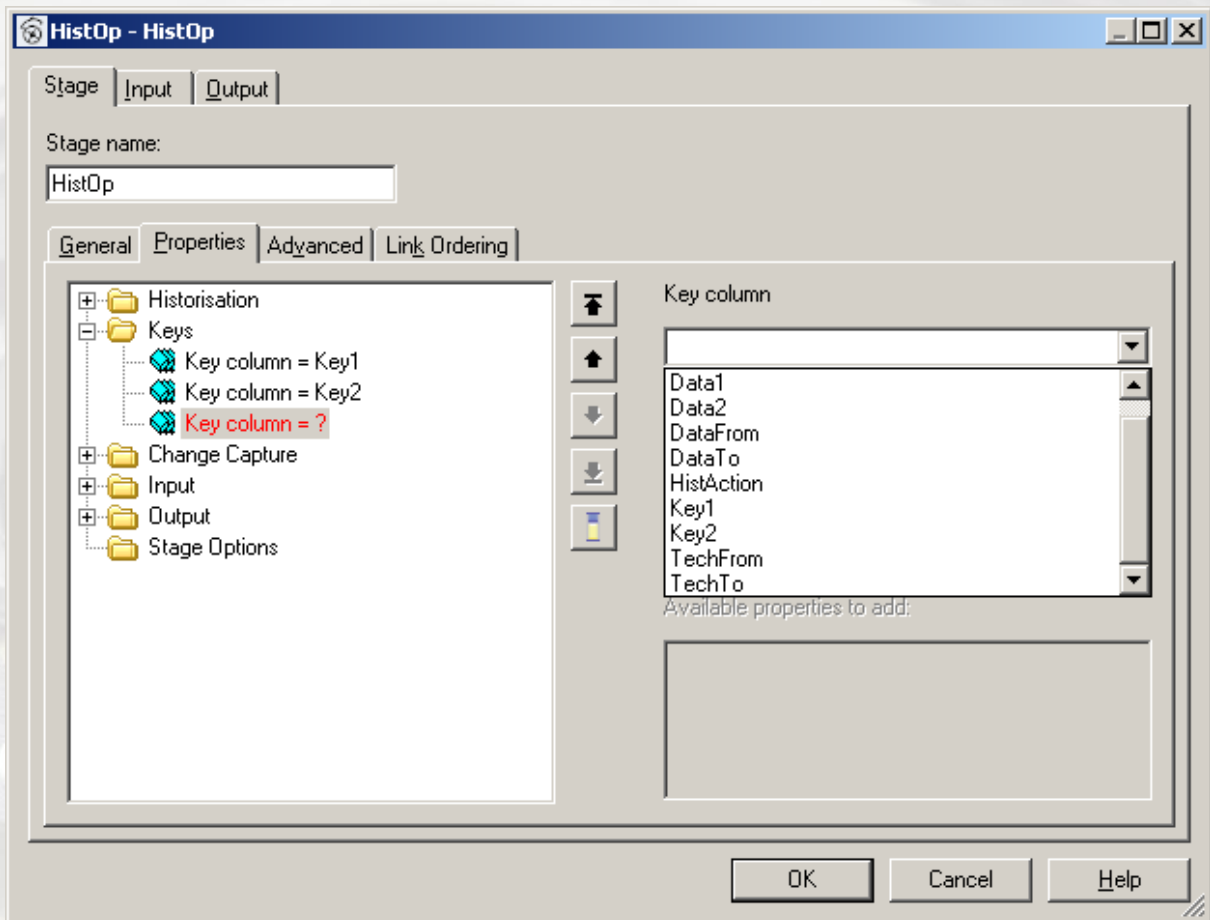
5.3.2. Keys Category

Figure 8 -- Keys category

The "Hist-Op" groups records together for processing which have the same key. While most tables use just one key column, the "Hist-Op" allows as many keys as there are columns to be declared as being part of the combined surrogate key used for grouping records together. This key is often referred to as the TIK, or "Time Invariant Key", as it remains the same regardless of changes to the data contents and the historisation status.

Key columns can be of any DataStage data type and may contain <null> values. The "Hist-Op" operator internally converts all parts of the key to a string which is concatenated together and used for comparison and null values are represented by a special internal string so that <null> and "" (empty) strings can be differentiated.

The "Hist-Op" does not repartition or sort the data streams. It is up to the developer to ensure that each processing node gets all the records belonging to a given key; the data must be partitioned on at least the first key column (further partitioning the keys make no difference to the outcome) and the data must be sorted so that all records for a given surrogate key come sequentially in group. The sort order within a group of records is not important. The "Hist-Op" will re-sort within each group and output records sorted by ascending *TechnicalFrom* then by ascending *ValidFrom*.

5.3.2.1. "Key column"

Specifies the name of the column to be used as the surrogate key / TIK; this is a repeatable entry but must occur at least 1 time. Columns can be of any data type and may be nullable,

 16/09/2012

and <null> values with multiple key columns are treated as not invalidating the whole key. As many keys as columns in the input may be specified.

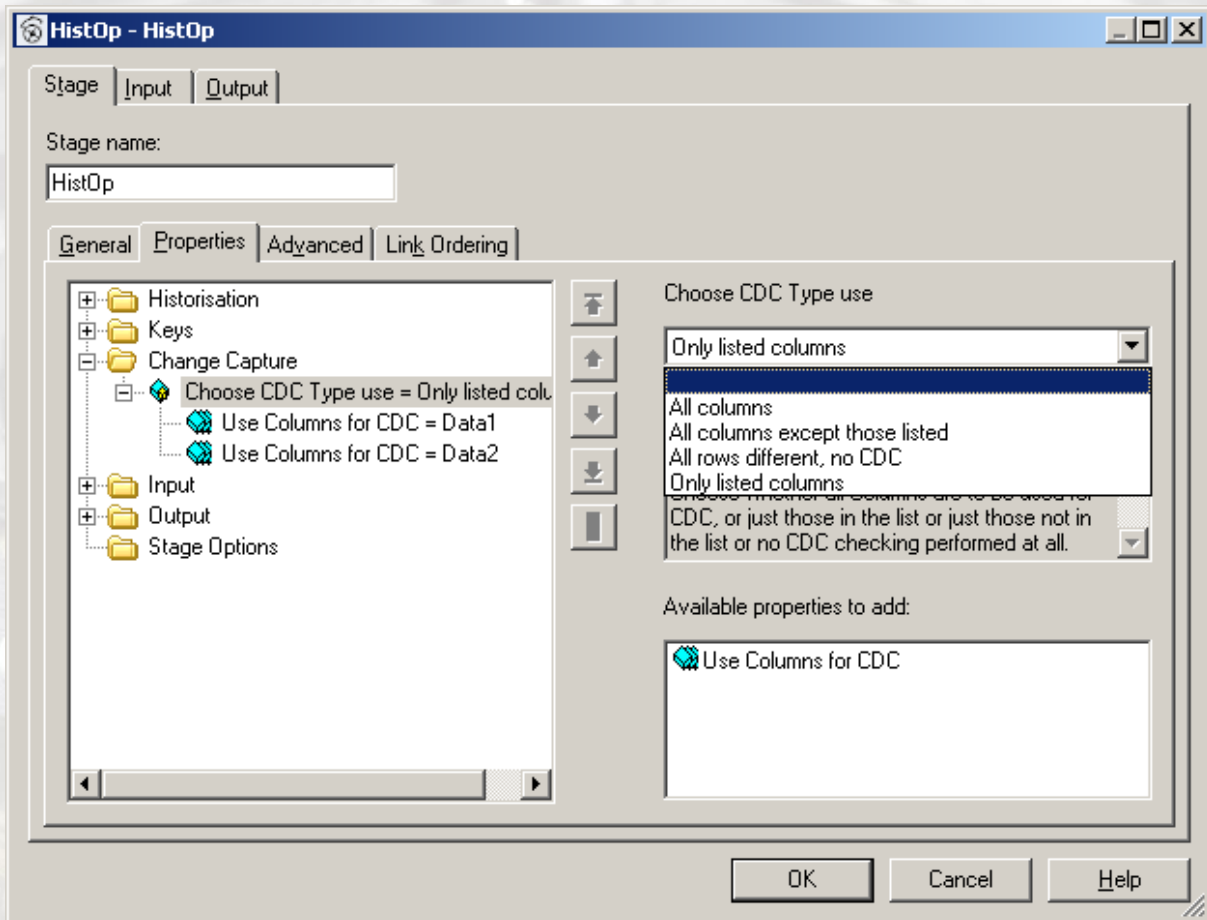### 5.3.3.   *Change Capture Category*

Figure 9 - Change Capture Category

This category allows the selection of what type of change-data-detection the "Hist-Op" is to perform and what columns in the input stream are to be used to compare records within a group (as defined by the keys declared in 5.3.2)

5.3.3.1.   *"Choose CDC Type use"*

The option selector for CDC, "Choose CDC Type use" lets the developer specify one of the following 4 options:

- **All Columns**
  When this option is chosen the selection of CDC key columns is disabled and every column in the record is used to compute the CDC - except for the 5 historisation columns (*ValidFrom*, *ValidTo*, *TechnicalFrom*, *TechnicalTo* & *HistAction*).
- **All columns except those listed**
  This option uses all columns for CDC (minus the 5 historisation columns *ValidFrom*, *ValidTo*, *TechnicalFrom*, *TechnicalTo* & *HistAction*) but, unlike the option above, allow columns to be specified that are not to be used for CDC. This option is useful when there are many columns that are part of the CDC but only a few that need to be ignored.
- **All rows different, no CDC**
  When this option is chosen the selection of CDC key columns is disabled. The "Hist-Op" will treat each record within a group as different from the others, regardless of contents.

 16/09/2012

Using this option can reduce computational overhead when it is certain that no duplicates can occur in the input data stream.

- **Only listed columns**
  when this option is chosen the list of CDC key columns is activated and the developer can enter the column names that are to be used for CDC computation. As many columns as desired can be declared.

Unlike the method used for comparing keys, the "Hist-Op" uses an internal CRC32 algorithm to determine a checksum for the CDC value. Using a compressed representation is generally superior to comparing the full data contents of the CDC columns; often almost all the columns in a record are used and this would be quite a bit of memory overhead versus the small computational one for calculating the CRC32. The CRC32 algorithm is an accepted industry standard for simple checksums and is both efficient and reliable. While there is a slight chance of two different records returning the same CRC32 value, the odds are infinitesimally small considering the small size of the groups and homogenous field data layout.

5.3.3.2. "Use Columns for CDC"

Specify the name of the column to be used for CDC. This is a repeatable entry but must occur at least 1 time. Columns may be any data type and may be nullable, and <null> values with multiple key columns are treated as not invalidating the whole key. As many keys as there are columns in the input may be specified. In this case a <null> in a column in one row is considered equal to a <null> in the same column of another row.

5.3.3.3. "Skip Columns for CDC"

Specify the name of the column to be skipped for CDC, any columns not in this list will be used for CDC. This is a repeatable entry but must occur at least 1 time. Columns may be any data type and may be nullable, and <null> values with multiple key columns are treated as not invalidating the whole key. As many keys as there are columns in the input may be specified. In this case a <null> in a column in one row is considered equal to a <null> in the same column of another row.

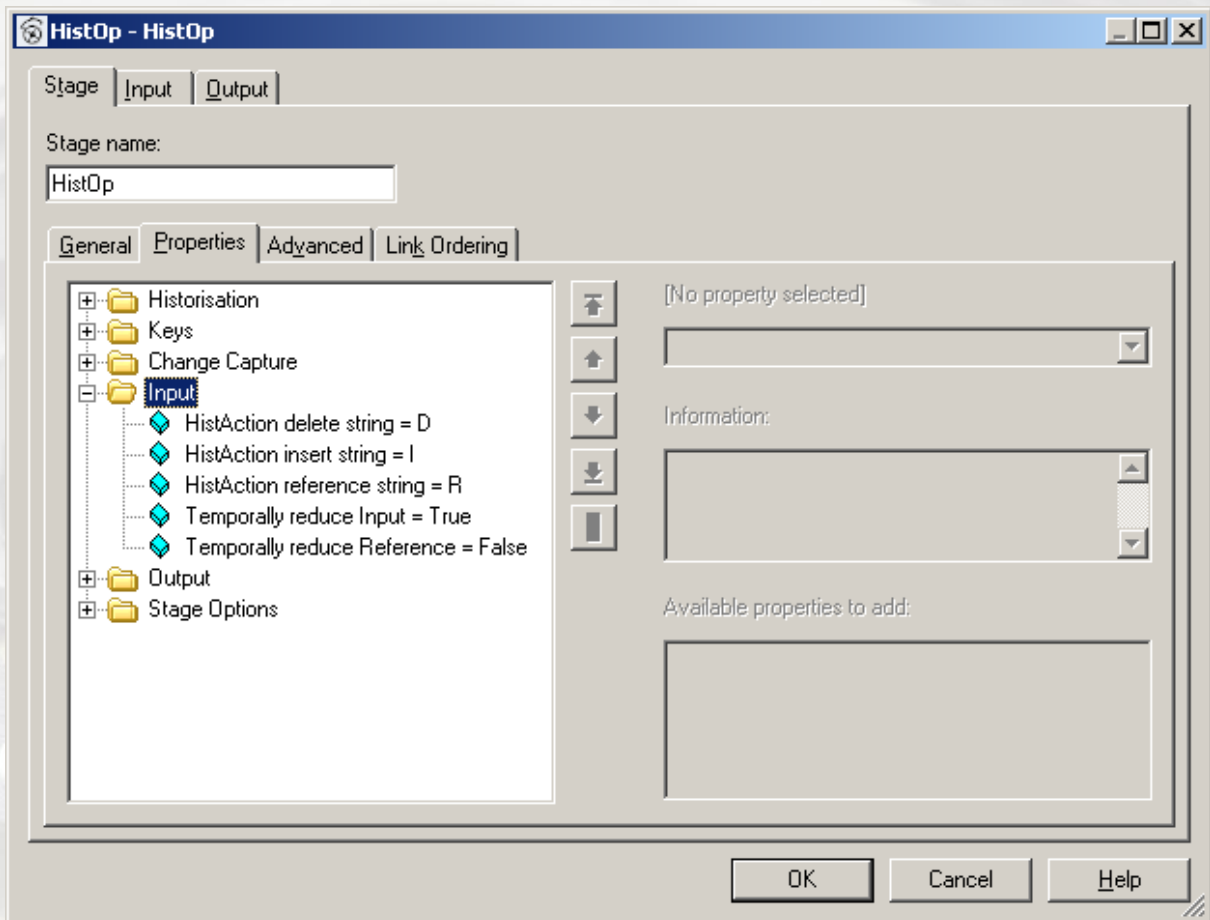   16/09/2012

5.3.4. Input Category

Figure 10 - Input Category

All 5 settings in this category are optional.

5.3.4.1. "HistAction delete string"

This is the string value in the "HistAction" column (as defined in chapter 5.3.1.1) that corresponds to a record coming from the input stream that is to be deleted. The default value is "D".

5.3.4.2. "HistAction insert string"

This is the string value in the "HistAction" column (as defined in chapter 5.3.1.1) that corresponds to a record coming from the input stream. The default value is "I".

5.3.4.3. "HistAction reference string"

This is the string value in the "HistAction" column (as defined in chapter 5.3.1.1) that corresponds to a record coming from the reference stream, i.e. a record coming from the table. Note that these records are historised in the operator and the resultant output rows with action codes of "I" and "U" (the defaults for "I"nsert into database and "U"pdate database) expect that the rows actually do exist in the source/target table. The default value is "R".

5.3.4.4. "Temporally reduce Input"

If not specified the default value is "true", meaning that the historisation operator will examine the input records as a group and attempt to temporally reduce them (chapter 4.4).

         16/09/2012

The reduction is performed prior to performing any historisation and is affected by the range settings and that of the CDC switch "*No CDC*" defined in chapter 5.3.3.

5.3.4.5. "Temporally reduce reference"

This option turns on further temporal reduction in addition to that above. While normal reduction compares records in the input data with each other, setting this option will make the "Hist-Op" check the reference data as well to see if the new record can be temporally reduced. See chapter 4.4 for details on this option.

5.3.5. Output Category
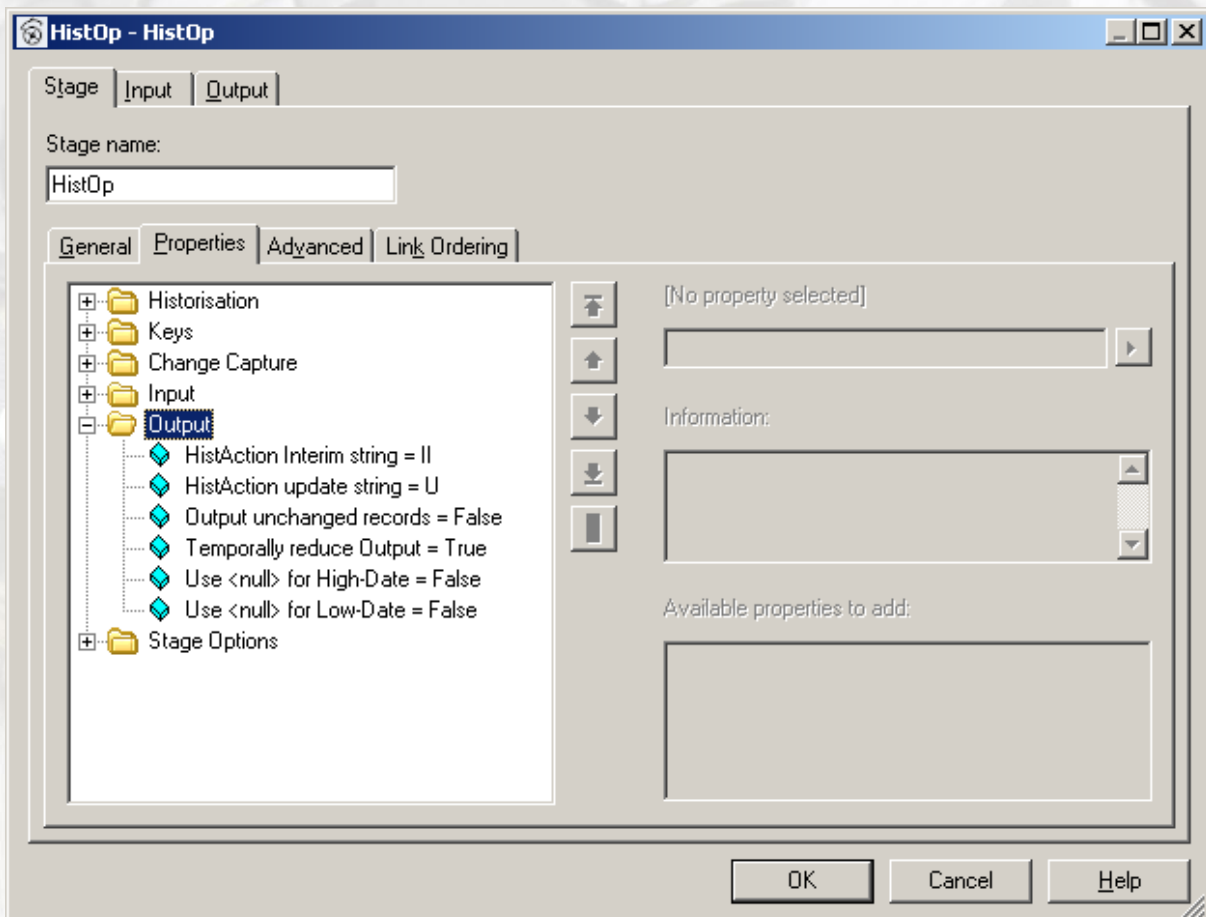
Figure 11 - Output Category

All 6 settings in this category are optional.

5.3.5.1. "HistAction interim string"

The Hist-Op will use the HistAction Insert code (chapter 5.3.4.2) for output records that result in a database insert. There are two types of records which generate this code, the original insert row and any number of new rows generated within the Hist-Op as part of the historisation process. These "interim" rows might need to be differentiated from the input rows and this option allows these rows to use a different HistAction string.

5.3.5.2. "HistAction update string"

The output HistAction value is modified from the original input value. In order to optimize the database changes made by historisation this value is set to either "I"nsert (or the override value), "D"elete (or the override value) and "U"pdate – the default setting for this parameter. It denotes records from the reference stream that need to have database updates performed on

 16/09/2012

them. The historisation operator will only ever modify the *TechnicalTo* date for an existing record, no other changes are made. This allows database "I"nserts to be treated differently from the other row types which perform database updates.

5.3.5.3. "Output unchanged records"

By default this setting is "false", unchanged records would just waste valuable database resources if they were to be written back to the database. In some cases it might be beneficial to keep these unchanged records in the output stream for further processing and when this option is set to "true" these records are kept and are output with a HistAction column value of "R"eference (or the override value as set in chapter 5.3.4.3).

5.3.5.4. "Temporally reduce Output"

This option, which defaults to "true", checks the interim data during historising to see if it might have created a record which can be temporally reduced according to the rules in chapter 4.4..

5.3.5.5. "use <null> for High-Date"

Within the "Hist-Op" <null> values in the *ValidTo/TechnicalTo* columns, when present, are treated as high-value using "9999-12-31 23:59:59.999999" reduced to whatever precision the column is defined at. It is possible for incoming data to contain both this high-value and <null> values; the operator treats them the same internally and keeps the original representation on output. When this switch is set to true, any high-value dates will be converted to <null> values on output.

5.3.5.6. "use <null> for Low-Date"

Within the "Hist-Op" <null> values in *the ValidFrom/TechnicalFrom* columns, when present, are treated as low-value using "0001-01-01 00:00:00.000000" reduced to whatever precision the column is defined at. It is possible for incoming data to contain both this low-value and <null> values; the operator treats them the same internally and keeps the original representation on output. When this switch is set to true, any low-value dates will be converted to <null> values on output.
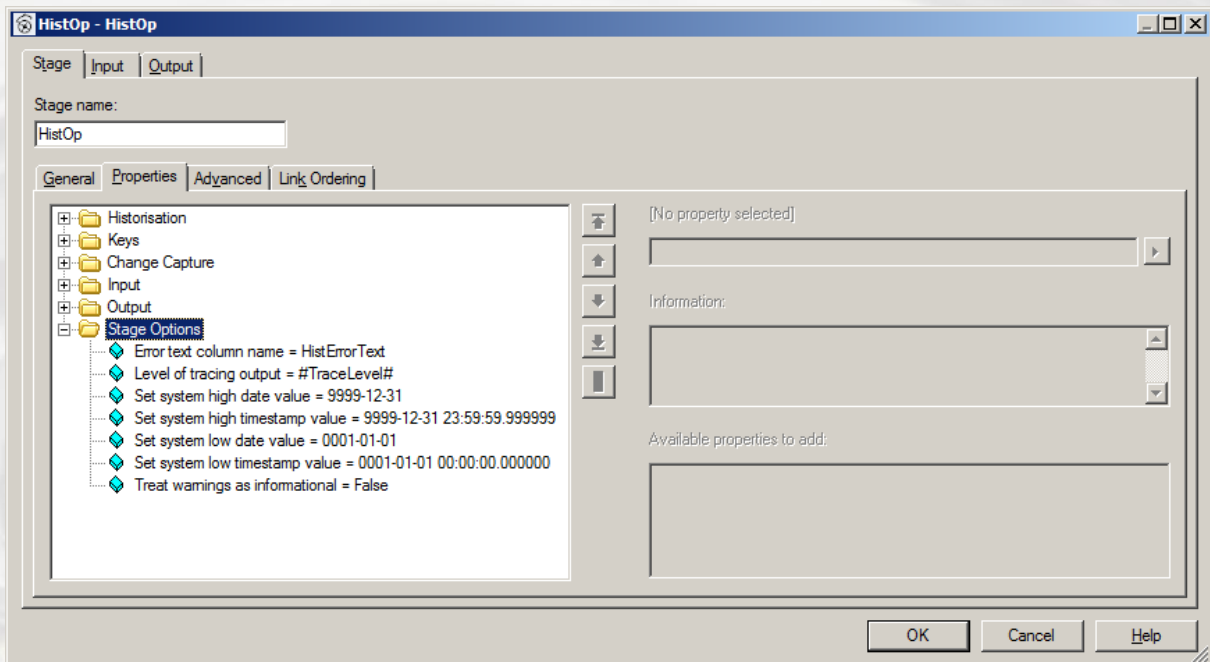
 16/09/2012

5.3.6. Options Category

Figure 12 - Options Category

This category contains 7 values, all of which are optional.

5.3.6.1. "Error text column name"

The error output link of the "Hist-Op" stage contains a copy of the input link plus an additional string column which contains the text describing why the row was rejected. The name of this column can be modified from the default of "HistorisationErrorText". Note that if one record in group contains an error, all the records from that group are rejected. The error text for rows which are in an erroneous group but don't contain any errors is a standardised one to the effect that other records in the group have triggered this row to be rejected.

5.3.6.2. "Level of tracing output"

This is an integer value from 0 (the default) to 100. The higher this number, the more tracing and informational output is generated by the "Hist-Op" and written to the job log file. Any value above 0 generates sufficient logging information to slow the job down significantly when processing large numbers of records and this option should only be used for tracing and debugging purposes during the development cycle and never in production or where performance is a factor.

    0   –   No tracing
   10   –   Basic group level tracing
   50   –   Basic record level tracing
  100 –   Detailed tracing

5.3.6.3. "Set system high date value"

Allows the default system high date "9999-12-31" to be set to a different value. This affects the historisation process as the operator checks for this value in date type fields to determine if the range is terminated or open.

 16/09/2012

5.3.6.4. "Set system high timestamp value"

Allows the default system high timestamp "9999-12-31 23:59:59.999999" to be set to a different value. This affects the historisation process as the operator checks for this value in timestamp type fields to determine if the range is terminated or open.

5.3.6.5. "Set system low date value"

Allows the default system low date "0001-01-01" to be set to a different value. This affects the historisation process as the operator checks for this value in date type fields to determine if the range is terminated or open at the beginning of the period.

5.3.6.6. "Set system low timestamp value"

Allows the default system high date "0001-01-01 00:00:00.000000" to be set to a different value. This affects the historisation process as the operator checks for this value in timestamp type fields to determine if the range is terminated or open at the beginning of the period.

5.3.6.7. "Treat warnings as informational"

There are a number of conditions that are not considered fatal to processing and are written to the log file as warning messages (e.g. if an invalid *HistAction* code is encountered a warning is written to the log and all records in the group are sent down the error link while processing continues). Although the author believes strongly that no job should be designed to produce warning messages during regular processing, there may be cases where these types of errors may be present in the data stream and this option allows the operator to deprecate messages that would normally be warnings to informational level so that the log may remain free of warnings without resorting to job-level or project-level message handling.

 http://Hist-Op.com 16/09/2012

6. Historisation Rules and Examples

While the possibilities of data combinations for historisation are too numerous to exhaustively cover, in this chapter we will cover the most common historisation cases encountered in normal processing and show the behaviour of the operator. Where the result is not necessarily intuitive, a detailed explanation of the logic is be included. All of the examples, unless otherwise noted, use the default settings; in particular this means that temporal ranges are considered to be "inclusive".

The time periods shown are for the *ValidTo* and *ValidFrom* dates, the ranges in dark blue represent the initial contents of the database (the reference record), red represents historised records, green represents the new record and light blue represents records created and inserted by the "Hist-Op"

Cases 6.1 through 6.4 are labelled "A" through "D". These letters represent the basic types of historisation performed and correspond to the tracing texts and values displayed when tracing level is set to high (chapter 5.3.6.2).

6.1.   A - New record overlaps at beginning
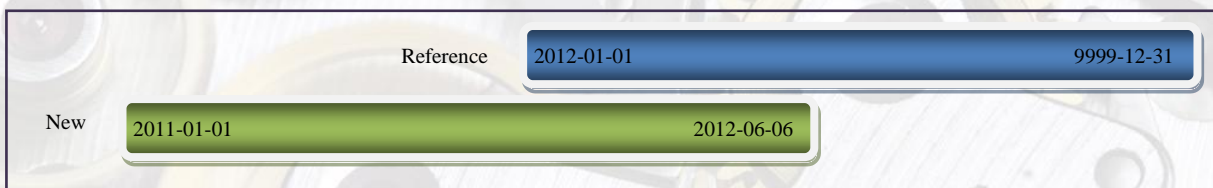
Figure 13 - Before Historisation: New record begins before reference record

Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
R	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	9999-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2011-01-01	2012-06-01	2011-08-01 15:30:00	9999-12-31 23:59:59

| Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
I	1	Thiotimoline	Temporal substance	1382	403	2012-06-02	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2011-01-01	2012-06-01	2011-08-01 15:30:00	9999-12-31 23:59:59
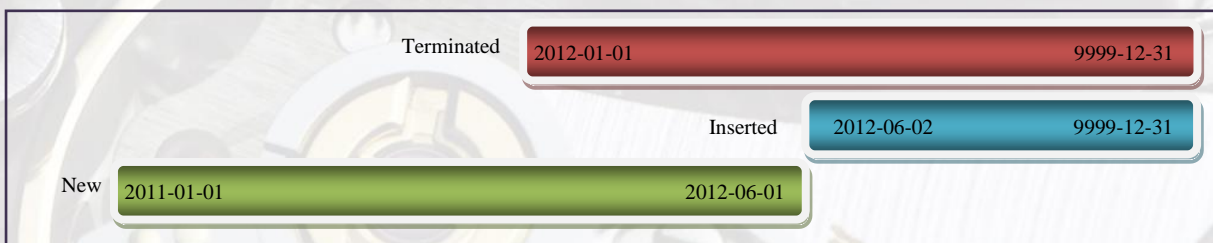
Figure 14 - After Historisation: New record begins before reference record

The first figure illustrates the temporal ranges of the existing reference record and the new record and during which times they overlap. The next figure shows the results of historisation, where the original reference record has been terminated, the new record inserted and the non-overlapping range from the original has been covered by a record inserted by the "Hist-Op".

 http://Hist-Op.com   16/09/2012

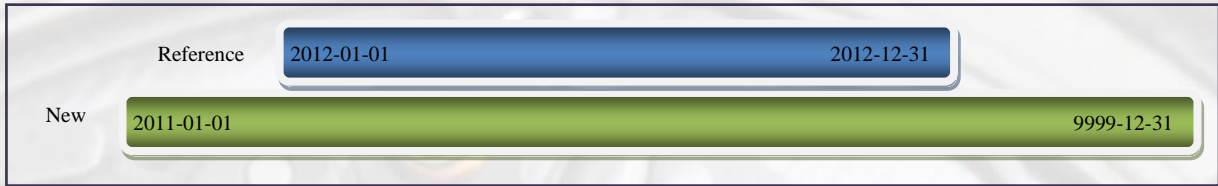## 6.2. B - New record overlaps completely

Figure 15 - Before Historisation: New record overlaps reference

Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
R	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	2012-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2011-01-01	2012-06-01	2011-08-01 15:30:00	9999-12-31 23:59:59

Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
I	1	Thiotimoline	Temporal substance	1500	50	2011-01-01	2012-06-01	2011-08-01 15:30:00	9999-12-31 23:59:59

Figure 16 - After Historisation: New record overlaps reference

This is the simplest historisation, since the new record has a time span which completely covers the reference data; thus all that needs to be done is to terminate the reference record and insert the new one.

 16/09/2012

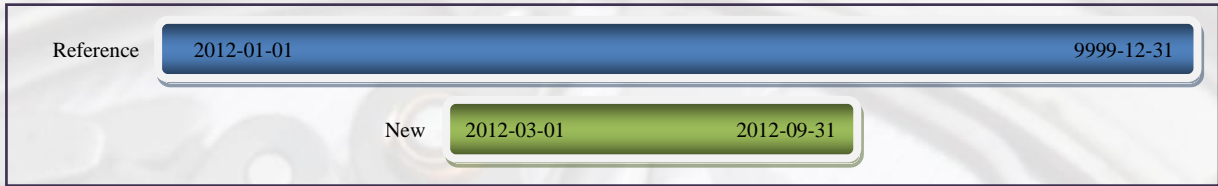## 6.3. C - New record overlaps at end
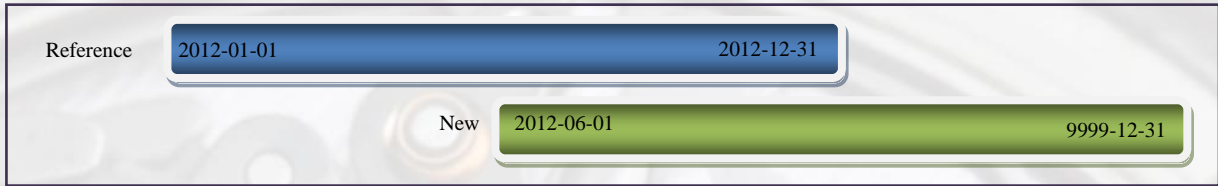
Figure 17 - Before Historisation: New record begins after reference record

Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
R	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	2012-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2012-06-01	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	2012-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
I	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	2012-05-31	2011-08-01 15:30:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2012-06-01	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59
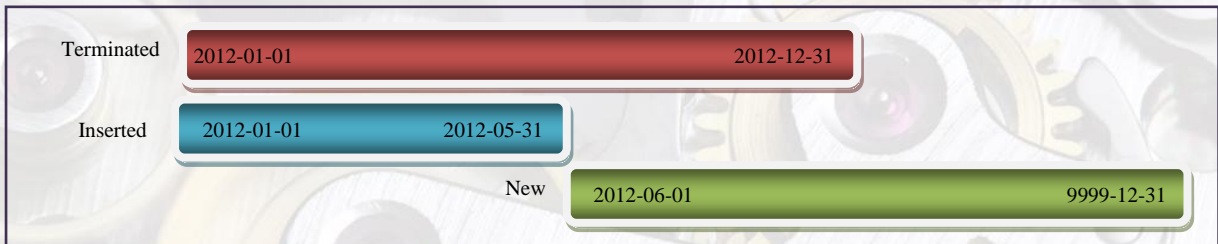
Figure 18 - After Historisation: New record begins after reference record

The historisation in this case is similar to that done in 6.1, except that here the reference record period ends before the new record's period ends, so the historisation of the reference data is done at the beginning of the period.

 16/09/2012

6.4. D - New record inside period of reference

Figure 19 - Before Historisation: New record begins after and ends before reference record

Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
R	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	9999-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2012-03-01	2012-09-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	9999-12-31	2011-07-01 18:00:00	2011-08-01 15:29:59
I	1	Thiotimoline	Temporal substance	1382	403	2012-01-01	2012-02-29	2011-08-01 15:30:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2012-10-01	9999-12-31	2011-08-01 15:30:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1500	50	2012-03-01	2012-09-31	2011-08-01 15:30:00	9999-12-31 23:59:59

Figure 20 - After Historisation: New record begins after and ends before reference record

The first figure graphically shows that the new record lies temporally inside the period of the reference record. This means that we need to insert a new interim record both at the beginning and at the end of the reference record period in addition to terminating the reference record and inserting the new one.

 16/09/2012

6.5. Temporal Reduction

Figure 21 - Before Historisation: Temporal reduction

Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
R	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	2012-12-31	2011-01-01 00:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	2011-02-28	2011-07-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-03-01	2011-05-31	2011-08-01 18:00:00	9999-12-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-06-01	9999-12-31	2011-09-01 18:00:00	9999-12-31 23:59:59

Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2011-01-01 00:00:00	2011-06-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2011-07-01 18:00:00	9999-12-31 23:59:59

Figure 22 - After Historisation: Temporally reduced

In this case three items are part of the delivery as new records, all of them with identical values (i.e. if we were use "*Price*" and "*Stock*" as the CDC columns). Since their respective validity ranges are tangent to each other, they are temporally reduced to one input record. This is done before historisation with the reference record. Without temporal reduction, the result would be as below

Product Table									
Action	Key	Name	Description	Price	Stock	Valid From	Valid To	TechnicalFrom	TechnicalTo
U	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	9999-12-31	2011-01-01 00:00:00	2011-06-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-01-01	2011-02-28	2011-07-01 18:00:00	9999-12-31 23:59:59
U	1	Thiotimoline	Temporal substance	1382	403	2011-03-01	9999-12-31	2011-07-01 18:00:00	2011-07-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-03-01	2011-05-31	2011-08-01 18:00:00	9999-12-31 23:59:59
U	1	Thiotimoline	Temporal substance	1382	403	2011-06-01	9999-12-31	2011-07-01 18:00:00	2011-08-31 23:59:59
I	1	Thiotimoline	Temporal substance	1382	403	2011-06-01	9999-12-31	2011-09-01 18:00:00	9999-12-31 23:59:59

Table 21 - non-temporally reduced output

 http://Hist-Op.com 16/09/2012

7. "Hist-Op" Sample data and sample Job

The "Hist-Op" package includes a sequential file named "Hist-OpRegressionTestData.txt" in DOS format along with a sample DataStage Parallel job called (unimaginatively) "SampleHistOpJob" in the distribution package.

The sample file contains a number of test cases used in the development of the "Hist-Op" to ensure that processing is executed as expected and as the basis for regression testing of changes and potential differences when compiling on a new platform. While some of the examples are repetitive, they can be used to experiment with different historisation scenarios and the file format is the preferred method of
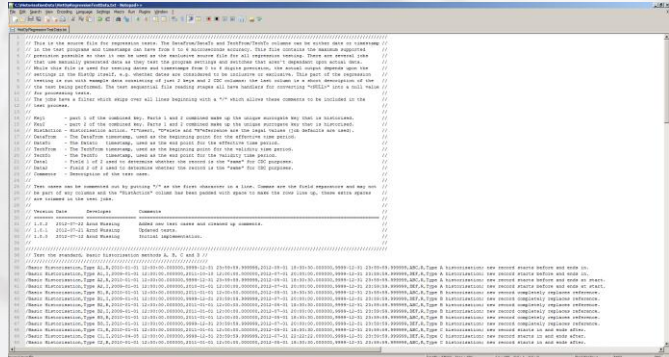
Figure 23 Hist-OpRegressionTestData.txt file

providing examples when contacting support with questions, enhancement suggestions and bug reports. All lines in the file beginning with a "/" character are skipped on import so that the file can be commented, and null values for the various fields are represented with "<NULL>". The SampleHistOpJob shown below is a deceptively simple one, but it is sufficient to illustrate a "Hist-Op" implementation. The two output links go to PEEK stages rather than to tables, but the job is designed to run quickly so it is up to the user to add the appropriate database stages if required. With the small amounts of data that are meant to be processed by this job the PEEK output is generally sufficient.
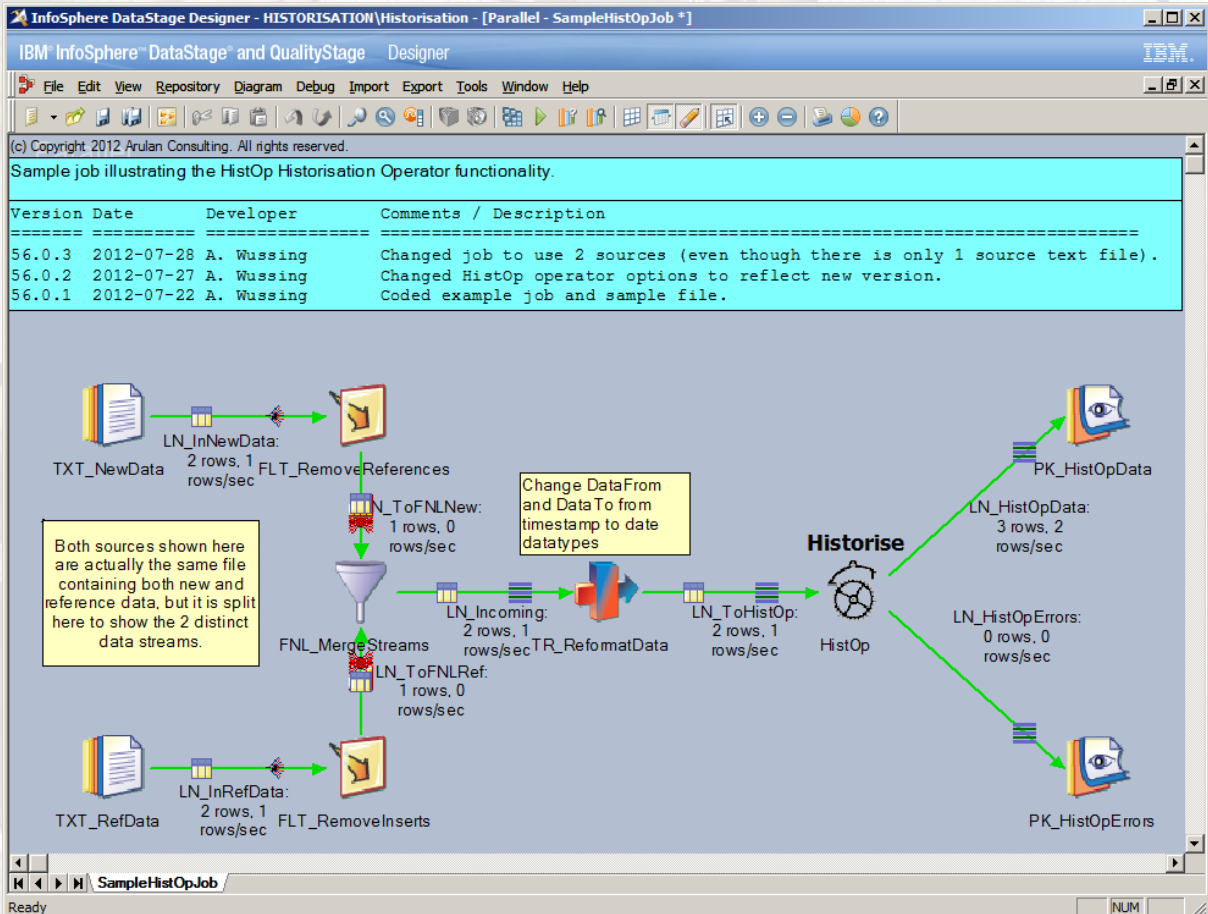
Figure 24 - SampleHistOpJob viewed in the Designer

 http://Hist-Op.com 16/09/2012

The job has the path, the filename and the tracing level parameterized, the settings may need to be updated to reflect the installation path. The tracing level (chapter 5.3.6.2) defaults to 100 (the maximum value) in this program. With this setting the amount of output into the DataStage Director log can be significant and will slow down job execution, but as this job is designed to illustrate the functioning of the "Hist-Op" with small amounts of data this does not pose a problem in this job.

When the "Level of tracing output" is set to a non-zero number there are at least two informational entries in the log file. The first entry shows the static operator settings and how the command line has been parsed and what the job settings are:
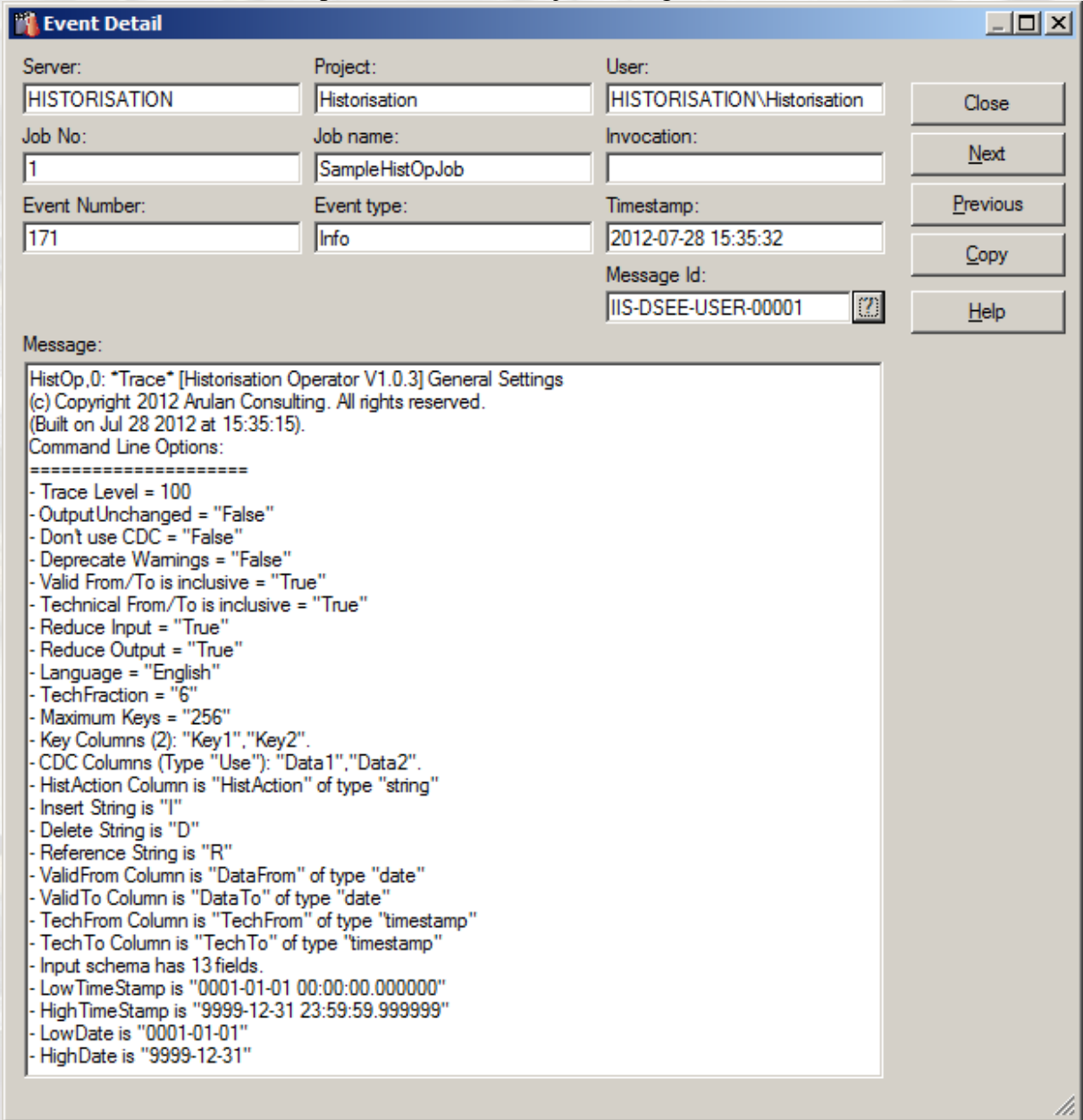
Figure 25 - Trace output, General Settings

After the single general settings log entry come the data-driven messages, their contents and frequency depend upon the trace level and the amount of data. Each group of records as defined by the key(s) is processed together and the log entry will show the input data for the group (both the reference records and the new data):

 16/09/2012
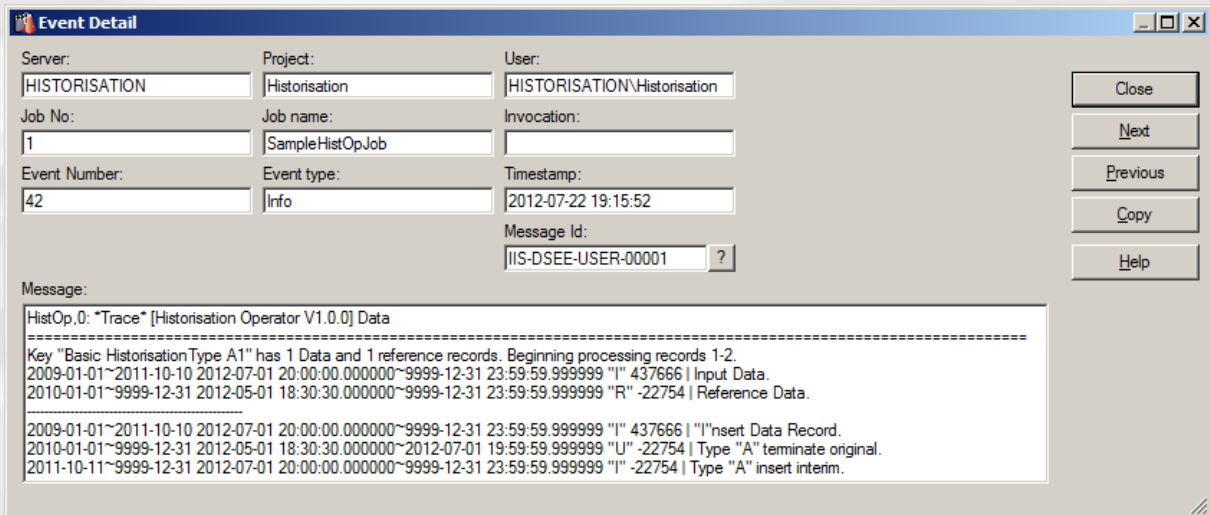
Figure 26 - Trace output, Group data output

The font used to display messages in the DataStage Director is a variable-width one, which makes displaying the output in an formatted fashion impossible. While the output is legible in the window as shown in the screenshot above, it is often simpler to cut-and-paste the text data into a text editor (i.e. notepad.exe, notepad++.exe or vi) with a fixed width font to analyze the output:

```
Hist-Op,0: *Trace* [Historisation Operator V1.0.0] Data
===============================================================================
Key "Basic HistorisationType A1" has 1 Data and 1 reference records. Beginning processing records 1-2.
2009-01-01~2011-10-10 2012-07-01 20:00:00.000000~9999-12-31 23:59:59.999999 "I" 437666 | Input Data.
2010-01-01~9999-12-31 2012-05-01 18:30:30.000000~9999-12-31 23:59:59.999999 "R" -22754 | Reference Data.
------------------------------------------------
2009-01-01~2011-10-10 2012-07-01 20:00:00.000000~9999-12-31 23:59:59.999999 "I" 437666 | "I"nsert Data
2010-01-01~9999-12-31 2012-05-01 18:30:30.000000~2012-07-01 19:59:59.999999 "U" -22754 | Type "A"
```

Figure 27 – Trace output in a text file

The output is formatted as follows:

ValidFrom"~"ValidTo TechFrom"~"TechTo HistAction CDC "|" Comment/Action

Where the data types and precision displayed depends upon the "Hist-Op" settings. The CDC is an unsigned 32 bit integer, so quite often there will be negative values. The actual CDC is a bigger value, only the first 6 digits are displayed in order to save space; that is sufficient to see if the CDC values of two records are identical or different. Note that the values themselves are irrelevant, they are just used for comparative purpose.

The first part of the output, between the group separator line with "====" values and the action separator line with "------", shows the "I"nput data records sorted in order of *ValidFrom* and then by *TechFrom*. Then comes the list of reference records in the same sort order (terminated records will show up in this list, but they are not part of processing and will be discarded unless the option *"Output unchanged records"* (chapter 5.3.5.3) is specified.

The second part of the output shows, for each "I"nput record in the stream, the historisation actions performed on the reference list. Note that once an "I"nput record has been processed; it too becomes part of the reference list and may, in turn, be modified by the historisation rules.

 16/09/2012

8. Installation

8.1. Windows Client

The "Hist-Op" installation process is performed both from the Client Windows system as well as on the server side. The Client-side installation is done using the DataStage Designer program. The download file is in the compressed ZIP format that can be uncompressed using one of many different decompression programs, the contents are shown using WinZip in Figure 28.
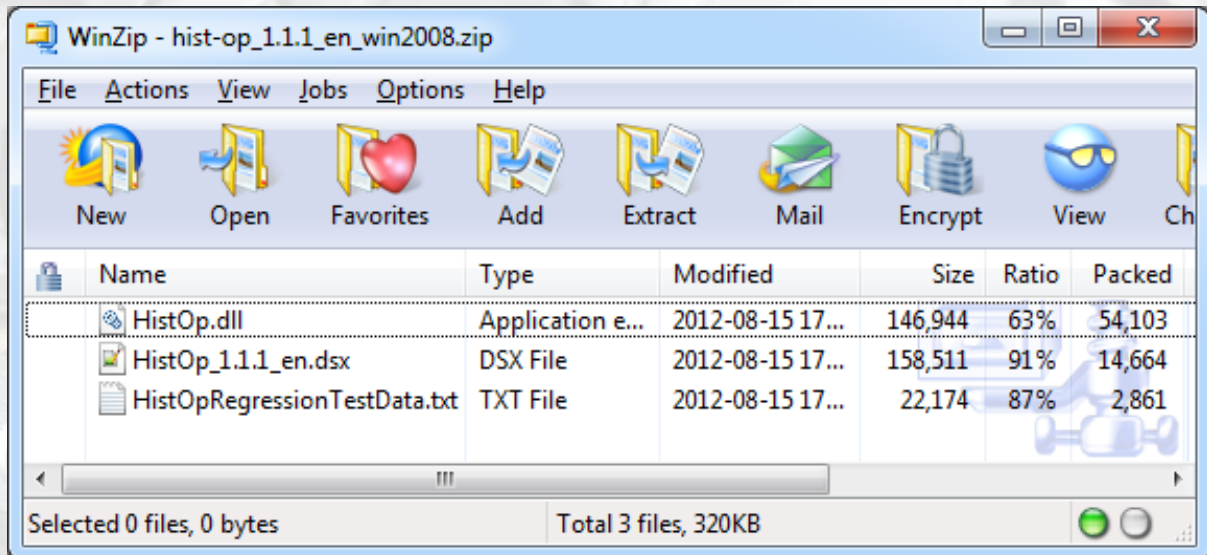
Figure 28 Distribution file contents

This document is downloaded as a separate file and is not included in the software download. The three files in the distribution are:

HistOp.dll
This is the Windows dll (dynamic linked library) file which contains all of the executable code for the historisation process. In UNIX systems the downloadable zip file will contain a "Hist-Op.so" file instead of the dll file. This file is used for the server-side installation and that portion of the installation is described below.

HistOp_1.1.1_en.dsx
This contains the DataStage GUI definition for the "Hist-Op" as well as the sample program described in chapter 7.

HistOpRegressionTestData.txt
This is the sample data file for the sample program and is also the reference for sample test cases that are used in regression testing the software and contains a complete list of standard historisation examples.

8.2. Unpack the .zip file

Using the decompression tool of your choice, unpack the files in the downloaded .zip file on the client PC where the DataStage Designer program is installed.

           16/09/2012

8.3. Import the .dsx file into DataStage

Start the DataStage Designer on your client PC and choose

```
Import ► DataStage Components
```

That will open up the window shown to the right in Figure 29 which allows the extracted .dsx file from the previous step to be specified in the "Import from file:" textbox.
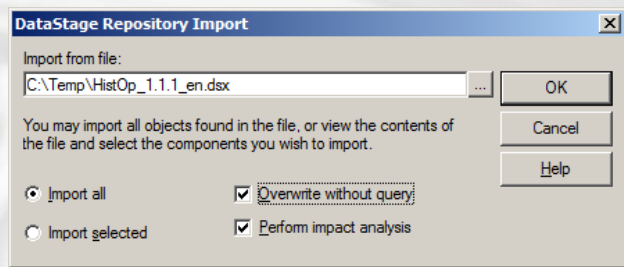
Figure 29 Designer Repository import selection.

When the OK button is pressed, the objects included in the .dsx file are loaded into the central DataStage repository. As shown in Figure 31 to the right, the two objects have been installed, the "SampleHistOpJob" has been placed in the "Jobs" branch at the top level and the "Hist-Op" definition has been placed into the "Stage Types" at

```
► Stage Types
  ► Parallel
    ► Processing
      ► Hist-Op
```

The operator may be dragged-and-dropped from this location or copied into the Designer Palette category for "Processing" or "Favorites" in order to make the operator more accessible during development work.

The sample job is ready for compilation and execution and both objects are now globally accessible for all developers in the project. Note that the path specified in the sample program may need to be adjusted according to where the sample text file is placed.

All that remain before the "Hist-Op" can be used are the simple steps on the server side.
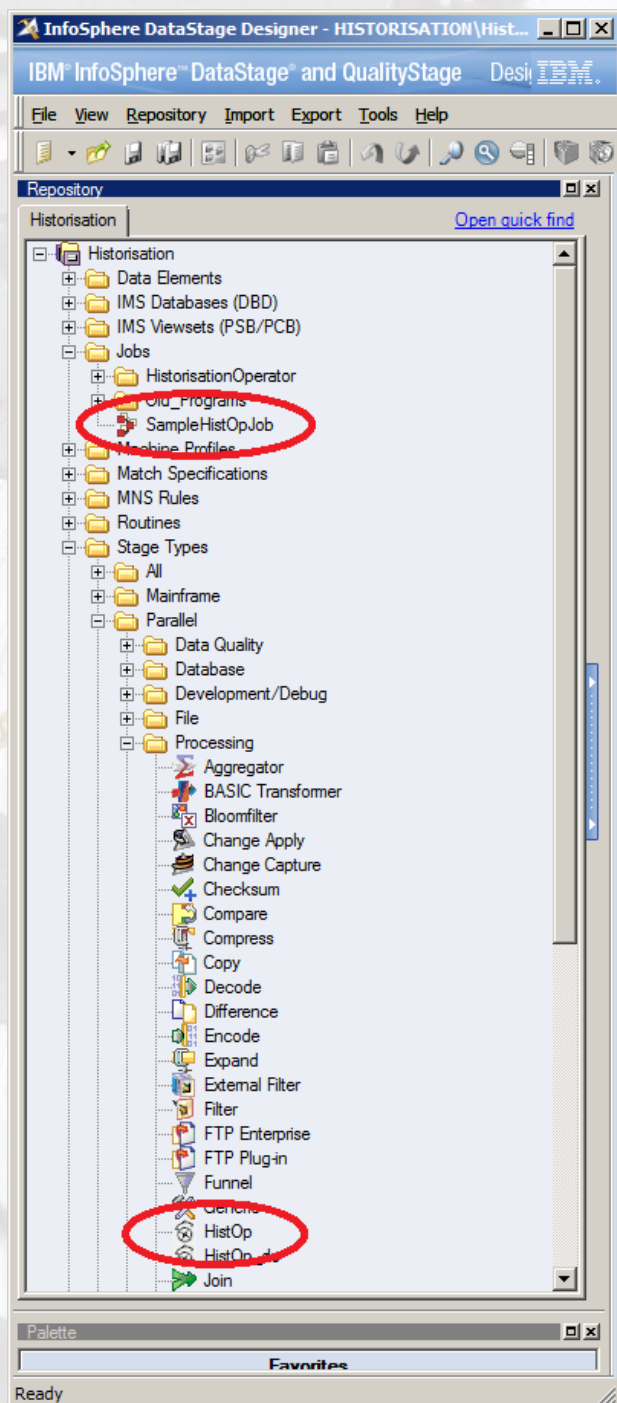
Figure 30 Designer Repository Palette view

 16/09/2012

8.4. Windows Server Installation

1. The "HistOp.dll" library needs to accessible to DataStage jobs at runtime. This means that the .dll must be in a directory that is specified by the runtime $PATH environment variable. The easiest method of doing this is to place the .dll into a directory already included in the runtime $PATH. Since the $PATH used by DataStage may be different from the system default settings the best way to check this value is to run any DataStage job, then look at one of the first entries in the log file (from the DataStage Director) titled "Environment variable settings:"
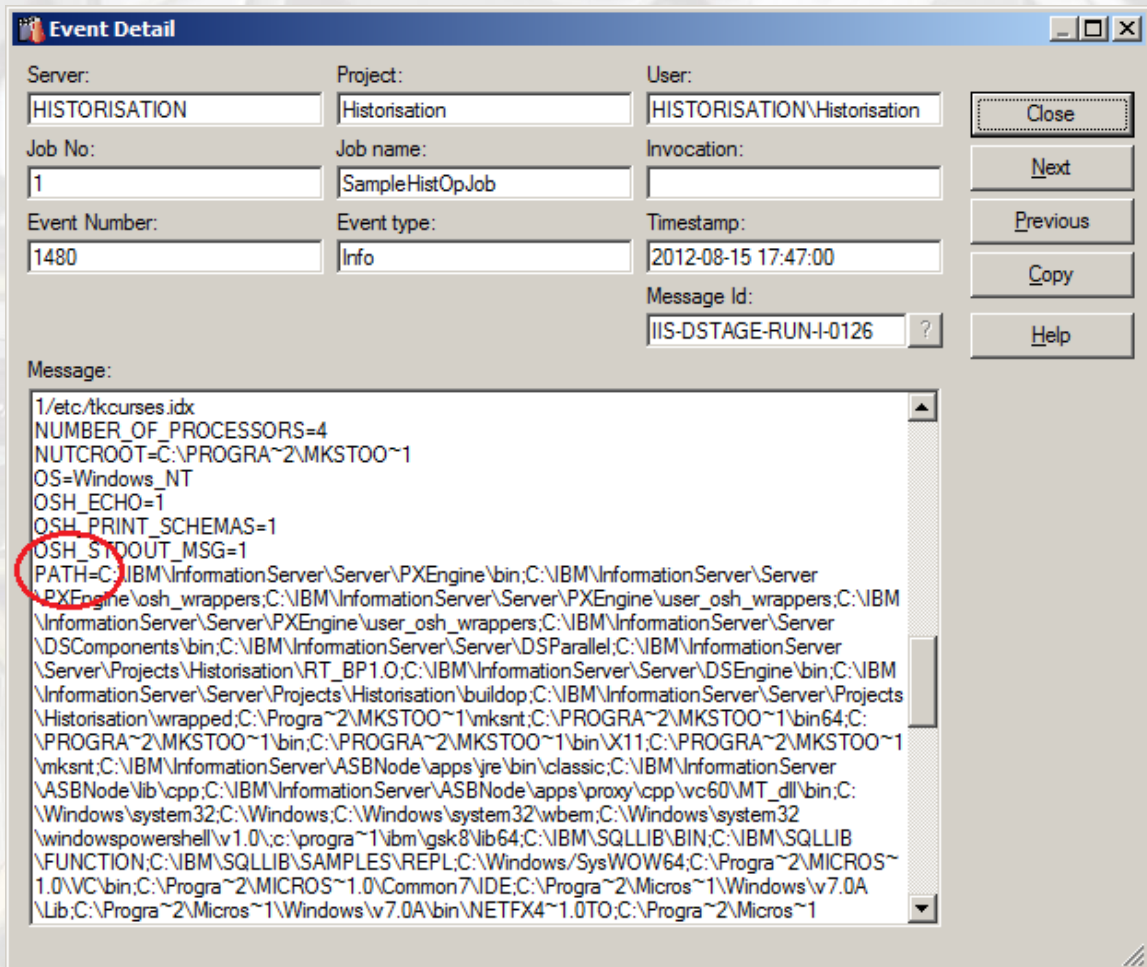
Figure 31 - Environment variable settings

Once the appropriate directory has been selected from this list, copy the "HistOp.dll" into that directory and make sure that users have read-execute access to it.

Another other method that can be used is to amend the DataStage $PATH setting to include the directory where the .dll has been copied. This can be done in the DataStage Administrator program.

```
► Choose the project
 ► Properties
  ► "Environment" button
   ► Select the "General" category
```

 http://Hist-Op.com 16/09/2012
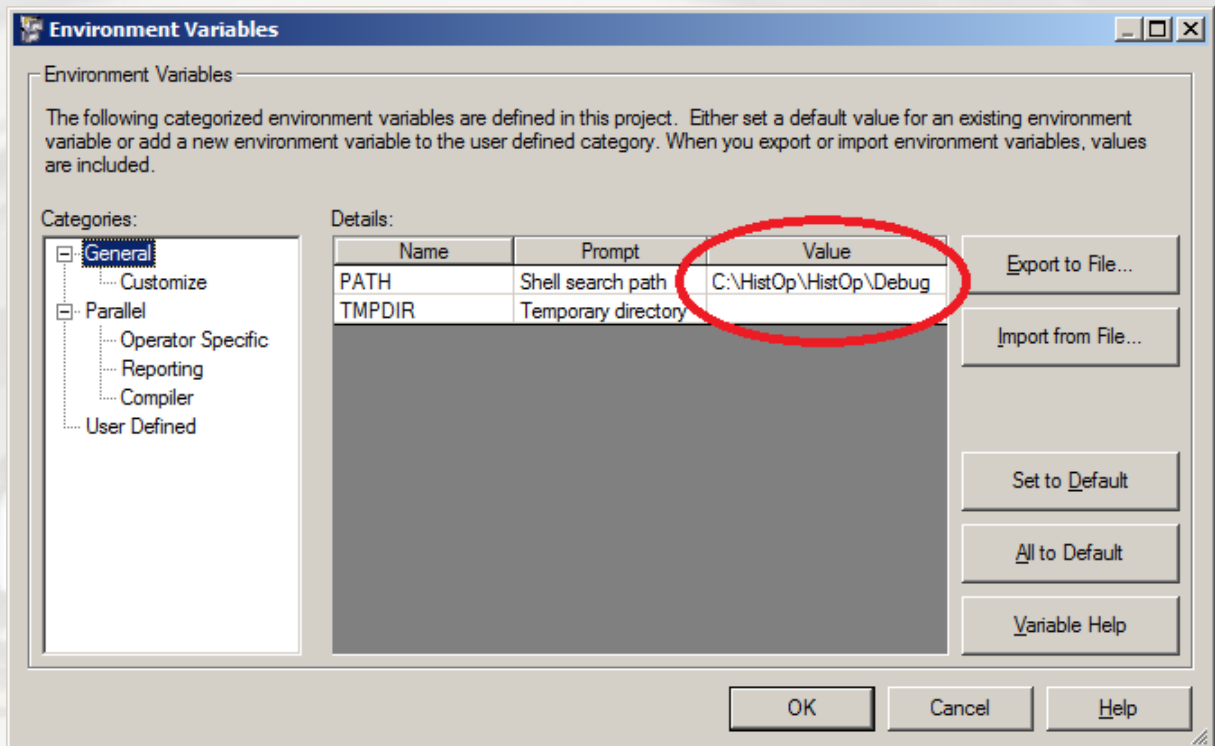
Figure 32 - Administrator Environment Variables

Add the path to the directory containing the .dll file and save the changes, they will take effect immediately.

8.5. Operator Registration

The final step is to register the new operator with DataStage. This is done by editing the file "operator.apt". If the "Hist-Op" is to be installed for only one project, then the file in the sub-folder "buildop" of the project directory needs to be modified, otherwise the "Hist-Op" can be registered system-wide by editing the file located at "{InstallPath}/InformationServer/Server/ PXEngine/etc". Append the line "HistOp HistOp 1" to the end of the operator.apt file in order to let DataStage know that the logical name "HistOp" calls the library routine "HistOp" and the number 1 means that this mapping rule is currently active.
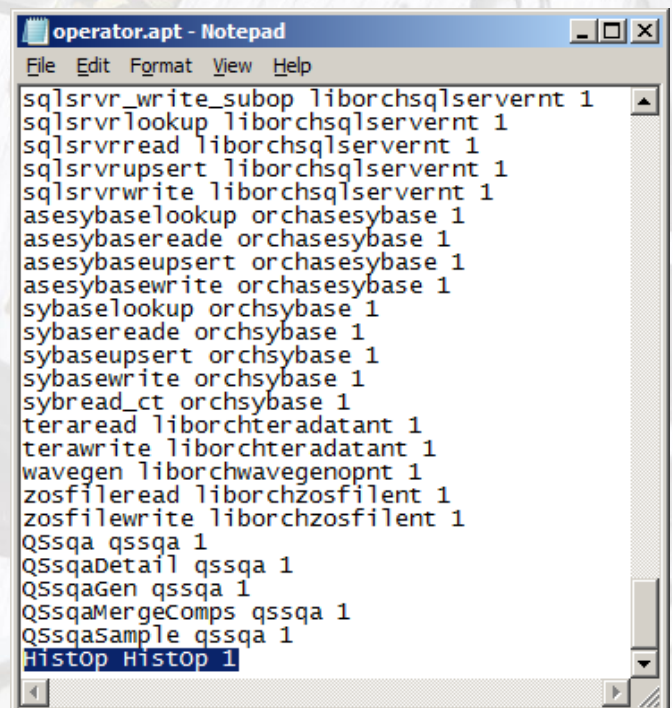
Figure 33 Editing Operator.apt file

9. History of Changes

Version	Date	Author	Description
1.1.6	2012-09-16	Arnd Wussing	Added chapter 4.8. Minor grammatical corrections.
1.1.5	2012-09-09	Arnd Wussing	Added chapter 2.8. Minor grammatical fixes.
1.1.4	2012-08-31	Arnd Wussing	Smaller documentation changes.
1.1.3	2012-08-26	Arnd Wussing	Further refinement of document contents.
1.1.2	2012-08-18	Arnd Wussing	Corrected numerous typographical errors.
1.1.2	2012-08-18	Arnd Wussing	Corrected numerous typographical errors.
1.1.1	2012-08-16	Arnd Wussing	Corrected typographical errors. Added Low/High Date and Timestamp options.
1.1.0	2012-08-11	Arnd Wussing	Initial shipment version.
1.0.11	2012-08-09	Arnd Wussing	Corrected layout.
0.0.10	2012-08-08	Arnd Wussing	Proofread, minor corrections.
0.0.9	2012-08-03	Arnd Wussing	Added option "InterimString" with new screenshot. Added "Temporally reduce Reference" with new screenshot. Added URL to http://Hist-Op.Com website.
0.0.8	2012-07-25	Arnd Wussing	New screenshots, as texts in options have changed. Corrected layout.
-	-	-	Older versions removed from list.

Table 22 - Document change history

10. References

C. Adamson and M. Venerable. *Data Warehousing Design Solutions*. Wiley, New York, 1 edition, 1998.

C.J. Date, Hugh Darwen, Nikos Lorentzos (2002). *Temporal Data & the Relational Model, First Edition* (The Morgan Kaufmann Series in Data Management Systems); Morgan Kaufmann; 1st edition; 422 pages. ISBN 1-55860-855-9.

C. J. Date, *An Introduction to Database Systems*, Eighth Edition, Addison-Wesley, ISBN 0-321-19784-4, 2003.

W.H. Inmon, *Building the Data Warehouse*, John Wiley and Sons, 1992.

C. S. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes and S. Jajodia (eds.), "A Glossary of Temporal Database Concepts," ACM SIGMOD Record: 23(1), 52-64, March, 1994.

Christian S. Jensen and Richard T. Snodgrass, *Temporal Data Management*. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 11, NO. 1. http://raptor.cs.arizona.edu/people/rts/pubs/TKDEJan99.pdf

 16/09/2012

11. List of Figures

12. List of Tables

16/09/2012

13. Glossary

2-Dimensional Data

Storage of data in records that utilize 2 time dimensions to store and access data. The first dimension is the data validity time period and the second dimension is the effective technical validity. These are usually represented as date/time ranges, i.e. *ValidFrom - ValidTo* and *TechnicalFrom - TechnicalTo*

Basel III

The third part of the Basel Accords which supersede the previously implement Basel II regulations; these are international banking regulatory standards geared to ensuring that banks have sufficient capitalization on hand. Since the standards require visibility (both internal and external) to key indicators and numbers the underlying database systems used to present this data now fall under regulatory supervision and accountability standards.

Bi-Temporal

See "*2-dimensional Data*"

CDC

An acronym for "C"hange-"D"ata-"C"apture. Encompasses the methods and software used to determine if and how a record changes over time. In this document's context it is used to determine whether 2 images of the same record are identical using just a subset of all the columns in the record for comparison.

CRC32

Stands for a 32-bit cyclical redundancy check. This type of check converts the data to be checked into an unsigned 32-bit number. The algorithms used (of which there are several) are designed so that small changes in source data result in large differences in the resultant number and that the distribution of results is spread as widely as possible.

DataStage®

A software product for ETL initially developed by VMark Software and now owned and further developed by IBM DataStage.

Data Warehouse

A Data Warehouse is a database construct used for analysis and reporting. While most databases attempt to adhere to a 3NF (third normal form) representation, data warehouses often have different structures (e.g. snowflake, star schema, etc.) in order to make reporting more efficient.

Escapement

A device used in keeping time which limits the motion of a wheel and delivers mechanical impulses to a watch or clock in order to keep time. The anchor escapement is used as the logo for the "Hist-Op".

ETL

An acronym for the processes (and their order) involved in data migration and data warehousing: "E"xtract, "T"ransform, and "L"oad. The term is used most frequently in data warehousing but the concept applies to any migration of data.

Exclusive Range

This is a type of date range where the beginning and end terminus points of the range are considered to not be part of the range. This has implications on both how the historisation of records is performed and also on how the queries on data need to be performed. See "*Inclusive Range*".

Generic Stage

All of the DataStage Operators can be called from the Orchestrate scripting language at runtime, including the "Hist-Op". The Generic Stage is a GUI stage on the DataStage Designer canvas which allows these operators to be called via the command-line interface directly.

GUI

Acronym for "G"raphical "U"ser "I"interface. In the context of this document this refers to the user interfaces of the DataStage Designer and DataStage Director programs as well as the stage definitions within the Designer.

High-Date

The high-date represents the highest possible value in a date column. The base value of "9999-12-31 23:59:59.999999" is used in this system and then reduced to whatever data type and precision is being used, e.g. a date with 2 microseconds would have a high-value of "9999-12-31 23:59:59.99" while a date would have a high-value of "9999-12-31". If the target column is nullable, then <null> also represents the high-date. Since DataStage "timestamp" fields are limited to 6 microseconds of accuracy no higher value can be represented. In databases such as SQL-Server and the data type datetime2 which has up to 7 microseconds it might be necessary to modify the high-date in a subsequent stage or revert to using <null> values to represent the highest possible date.

Inclusive Range

This is a type of date range where the beginning and end terminus points of the range are considered to be part of the range; for instance the range from 2012-07-01 through 2012-07-31 are the days in July and 07-01 and 07-31 are part of that month. This type of range is the more commonly used on and the choice has implications on both how the historisation of records is performed and also on how the queries on data need to be performed. See "*Exclusive Range*".

Low-Date

The low-date represents the smallest possible value in a date column. The base value of "0001-01-01 00:00:00.000000" is used in this system and then reduced to whatever data type and precision is being used, e.g. a date with 2 microseconds would have a low-value of "00001-01-01 00:00:00.00" while a date would have a low-value of "0001-01-01". If the target column is nullable, then <null> also represents the low-date.

Null

This term is used in this document to represent the SQL-Type <null> value. A <null> value has no value and represents an "unknown" value for the field. <Null> values cannot be compared to other values, not even to another <null>. Note that an empty string, "", with length of 0, is different from a <null> string.

Operator

DataStage has evolved over time through a combination of product development and acquisition and integration of new technologies. The parallel engine is an example of a product assimilated into DataStage. The original Orchestrate engine did not have a graphical front-end and used a script-based methodology. The programmer would design the ETL process by including Orchestrate "operators" in the script; these would perform operations on the input and output data streams. The DataStage Designer has a graphical design paradigm which lets the programmer place "stages" upon the design canvas and links those together to denote data flow. The original "operators" get presented as "stages" in the designer canvas and this is why the two terms are used interchangeably in this document although; under the covers they are differentiated as two different types.

Slowly Changing Dimensions

This concept partially covers the temporal aspects of Historising data and comes from the Data Warehousing world. Type II slowly changing dimensions are roughly equivalent to the 1-dimensional model presented here and further SCD dimensions are documented at the Wiki page located at http://en.wikipedia.org/wiki/Slowly_changing_dimension.
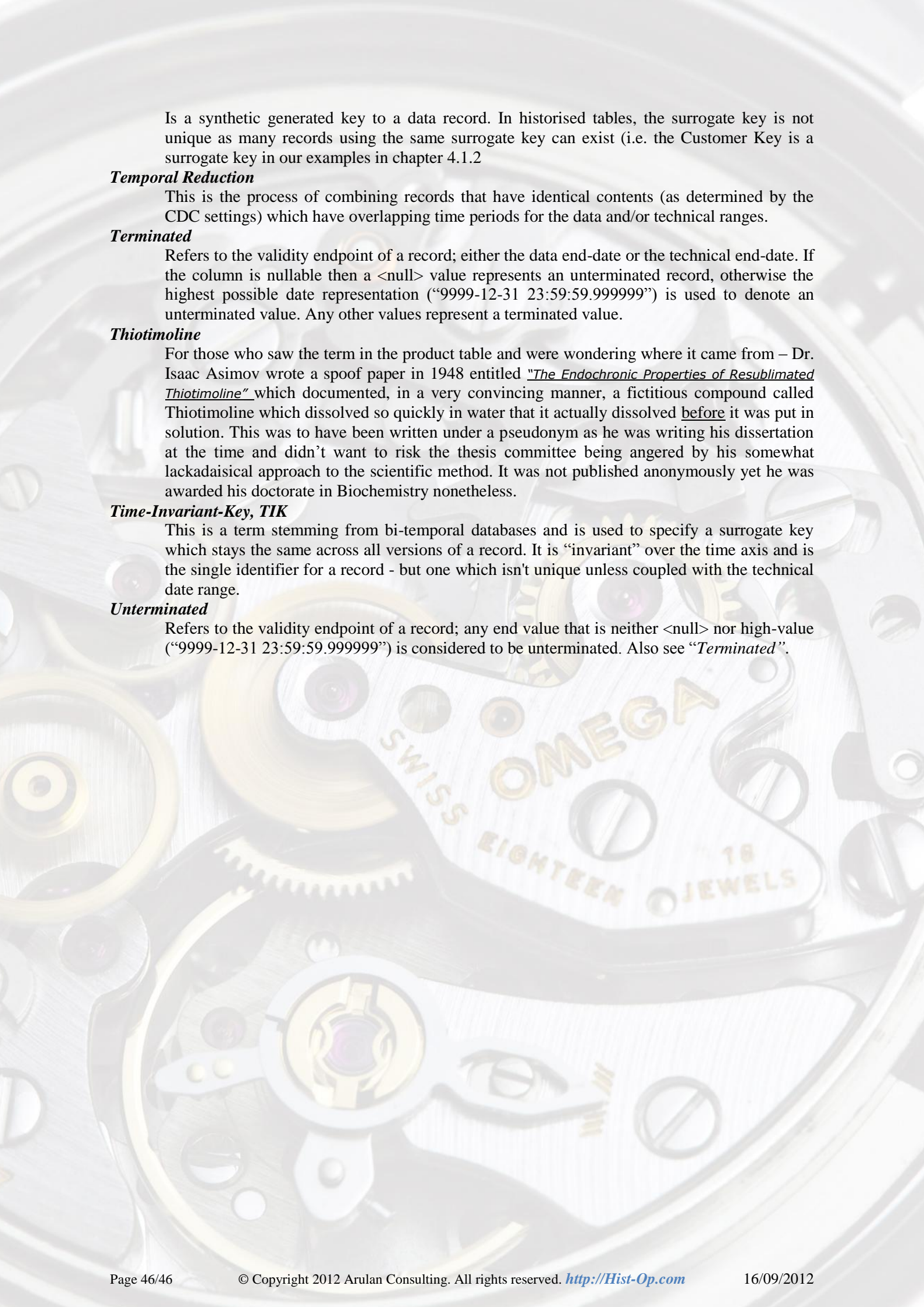
Solvency II

This is EU Directive 2009/138/EC which codifies and harmonizes the EU insurance regulations. While its primary concern is the amount of capital that insurance companies must hold to reduce the risk of insolvency, it has many regulatory implications on how financial and transactional data is stored and accessed. Solvency II is scheduled to come into effect on 1 January 2014.

Stage

See "*Operator*" for the definition of both terms.

Surrogate Key

 http://Hist-Op.com 16/09/2012

Is a synthetic generated key to a data record. In historised tables, the surrogate key is not unique as many records using the same surrogate key can exist (i.e. the Customer Key is a surrogate key in our examples in chapter 4.1.2

Temporal Reduction

This is the process of combining records that have identical contents (as determined by the CDC settings) which have overlapping time periods for the data and/or technical ranges.

Terminated

Refers to the validity endpoint of a record; either the data end-date or the technical end-date. If the column is nullable then a <null> value represents an unterminated record, otherwise the highest possible date representation ("9999-12-31 23:59:59.999999") is used to denote an unterminated value. Any other values represent a terminated value.

Thiotimoline

For those who saw the term in the product table and were wondering where it came from – Dr. Isaac Asimov wrote a spoof paper in 1948 entitled *"The Endochronic Properties of Resublimated Thiotimoline"* which documented, in a very convincing manner, a fictitious compound called Thiotimoline which dissolved so quickly in water that it actually dissolved before it was put in solution. This was to have been written under a pseudonym as he was writing his dissertation at the time and didn't want to risk the thesis committee being angered by his somewhat lackadaisical approach to the scientific method. It was not published anonymously yet he was awarded his doctorate in Biochemistry nonetheless.

Time-Invariant-Key, TIK

This is a term stemming from bi-temporal databases and is used to specify a surrogate key which stays the same across all versions of a record. It is "invariant" over the time axis and is the single identifier for a record - but one which isn't unique unless coupled with the technical date range.

Unterminated

Refers to the validity endpoint of a record; any end value that is neither <null> nor high-value ("9999-12-31 23:59:59.999999") is considered to be unterminated. Also see *"Terminated"*.

 16/09/2012