Representing Data Quality for Streaming and Static Data

Anja Klein, Hong-Hai Do,
Gregor Hackenbroich
SAP Research CEC Dresden
SAP AG, Germany
<anja.klein>, <hong-hai.do>,
<gregor.hackenbroich>@sap.com

Marcel Karnstedt
Department of Computer
Science and Automation,
TU Ilmenau, Germany
marcel.karnstedt@tu-ilmenau.de

Wolfgang Lehner
Database Technology Group
TU Dresden, Germany
wolfgang.lehner@tu-dresden.de

Abstract

In smart item environments, multitude of sensors are applied to capture data about product conditions and usage to guide business decisions as well as production automation processes. A big issue in this application area is posed by the restricted quality of sensor data due to limited sensor precision as well as sensor failures and malfunctions. Decisions derived on incorrect or misleading sensor data are likely to be faulty. The issue of how to efficiently provide applications with information about data quality (DQ) is still an open research problem.

In this paper, we present a flexible model for the efficient transfer and management of data quality for streaming as well as static data. We propose a data stream metamodel to allow for the propagation of data quality from the sensors up to the respective business application without a significant overhead of data. Furthermore, we present the extension of the traditional RDBMS metamodel to permit the persistent storage of data quality information in a relational database. Finally, we demonstrate a data quality metadata mapping to close the gap between the streaming environment and the target database. Our solution maintains a flexible number of DQ dimensions and supports applications directly consuming streaming data or processing data filed in a persistent database.

1. Introduction

In smart item environments data during and about product usage (product lifecycle) and environmental data (e.g. humidity) can be captured using a multitude of sensors (e.g. pressure, temperature, mileage). This data is exploited to guide and optimize production automation processes as well as complex business decisions. Some applications directly consume streaming data, where the knowledge about current data and data quality (DQ) is critical. On the other hand, in many cases the sensor data needs to be stored in a database for further processing.

In the context of sensor data, a big problem lies in restricted data quality. There are sensor inherent, physical restrictions, like limited resolution and precision. Further, sensor data quality is decreased by sensor failures and malfunctions due to real world application environments like the industrial shopfloor or mobile devices. It has to be stated, that the data quality restrictions resulting directly from system components and environment cannot be avoided without significant cost increase for better sensors (with higher precision) or sensor shielding.

If the sensor data are incorrect or misleading, deduced decisions are likely to be faulty. Thus the data quality restrictions in sensor data streams must not be ignored, but handled carefully. Information about data quality has to be collected from the sensors and streamed along with the measured data to prevent inappropriate decisions due to incomplete and/or incorrect data. Moreover, the information on data quality must be stored together with the captured sensor data in the target database. Only then, the evaluation of data with restricted quality will be possible.

Consider the hydraulic brake system of a truck equipped with pressure sensors to detect sudden pressure loss in case of a pipe rupture to send a warning to the driver. However, the disregard of a restricted sensor precision could end in a disaster. In case of a small leakage, the sensor could not detect the slow increase of pressure loss. During a hard brake, it could burst and the truck driver would have no chance to stop the vehicle. A more complex yet not so drastic scenario involving several dozens of sensors is the forecast of the residual lifetime of the engine of this truck. With this knowledge, the maintenance of the truck can be optimized. However, if the durability is overestimated due to the imprecise sensors, the truck may break down on the road and an expensive towing will be necessary.

Sensors allow for the automatic collection of a huge volume of data. To compete with the severe resource constraints posed by data streams (i.e. restricted processing power, memory and communication capacity) the large-scale data has to be reduced by data pre-processing. The raw sensor data is combined, summarized and aggregated to reduce the data volume without or with only admissible loss of information. Lest data quality information is lost, it has to be propagated through the pre-processing steps analog to the sensor data.

The propagation of data quality information results in an overhead for data transfer and management. Due to the large amount of stream data, this may shape up as very expensive. Furthermore, quality information present additional metadata on sensor data. Yet, the management of data quality is addressed neither in data stream nor in relational metadata models.

Our contributions in the context of data quality capturing, propagation and storage are as follows:

- We propose the use of jumping windows to efficiently collect and propagate data quality in data stream environments. There-upon we derive a metadata extension to allow for the uniform modelling of an unlimited number of data quality dimensions in data streams.
- We present a metadata model extension for the relational database schema to offer the possibility to store and manage data quality information in a relational dabase management system (RDBMS).
- We demonstrate the mapping between these two metadata models to close the gap between streaming system and target database.

The paper is organized as follows: The following section discusses related work in the context of data quality in data streams and databases. Section 3 gives an overview over the presented solution. Section 4 introduces the flexible data stream metamodel extension of jumping data quality windows. In Section 5 the focus lies on the efficient storage of data quality information in a relational database. The mapping of the presented metadata models is illustrated in Section 6. The paper concludes with a summarization of our contributions and an outlook on future work.

2. Related Work

Data quality in databases is discussed in several publications [10], [14] from a conceptual point of view. They focus on different definitions of the term data quality and distinguish different sets of relevant data quality dimensions [9]. Due to their high visibility, accuracy [5] and completeness [4], [12] are the quality dimensions most often referred to.

Despite the fact, that data quality is identified of high importance in the context of relational databases as well as data warehouse environments [3], [9], prior work in this context suffer from two major drawbacks. Firstly, the presented approaches refer to a (set of) reference data source(s) containing the true data to calculate the data quality. It is obvious, that in case of sensor measurement data, no such reference is present. A high precision sensor as reference is nonsensical. If such a sensor were available, it would be taken from the start. Hence, the problem of data quality capturing and propagation for sensor data streams is firstly discussed in this paper.

Furthermore, the estimation of data quality based on reference true data takes place online during query processing. Thus, in that scenario, no persistent storage of data quality information is addressed. Neither a reasonable metadata model nor an exemplary table schema to store data quality is presented.

Quality of Service (QoS) is an important issue in data stream management systems (DSMS). For example, the DSMS Aurora [1] defines the QoS dimensions latency, importance and approximation of DSMS query processing. TelegraphCQ [6] as well declares the reliability-based QoS dimension latency combined with the notion of uncertainty of workload information. However, these QoS dimensions characterize the service or processing quality of the DSMS. They are used to drive policies for scheduling and load shedding. Data quality restrictions due to sensor specifications are not addressed in this context.

The concept of window operators can be found in manifold work concerning data stream processing [2], [13]. Due to restricted memory capacity, joins of data streams are often processed window-wise [7]. Furthermore, sequence matching [11] is evaluated in moving windows of two streams. Based on these ideas, we devised jumping windows for data quality propagation to benefit from the inherent possibilities of resource saving.

Last, but not least, the mapping between data stream quality information and DQ tables in a target database was never discussed without the ability to propagate data quality in a sensor data stream and the possibility to store data quality in a database.

3. Solution Overview

This section gives an overview over the application area as well as our proposed solution illustrated in Figure 1. The real world environment, for example a manufacturing area, is monitored with the help of sensors. The measured sensor data is streamed towards the target applications, where the data is processed and actions or decisions are derived. There are two modes of data processing. On the one hand, the data is consumed directly from the stream for basic data analysis in the automatic process control, e.g. during production processes. On the other hand, many business applications require data spanning a wider time interval aggregated in a persistent database. Here, complex data mining and knowledge discovery is executed. Both application scenarios are supported by the generic and flexible solution presented in this paper.

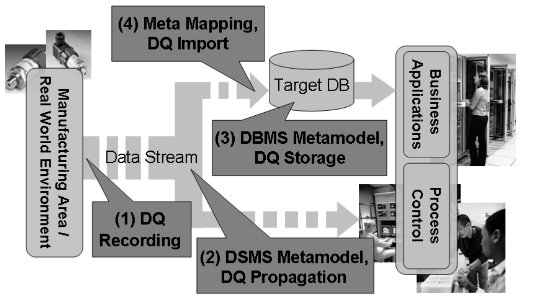

(4) Meta Mapping, DQ Import
Manufacturing Area / Real World Environment
Target DB
Business Applications
Data Stream
(3) DBMS Metamodel, DQ Storage
(1) DQ Recording
(2) DSMS Metamodel, DQ Propagation
Process Control

Figure 1. System Overview

Our solution for data quality transfer and management consists of (1) data quality recording, (2) the DSMS extension for data quality propagation, (3) the DBMS extension for persistent data quality storage and (4) the metadata mapping for data quality import from the stream into a database. During the data quality recording, DQ information is captured from the sensor. Therefor, the sensor has to be anaylised in detail as shown in Figure 2.

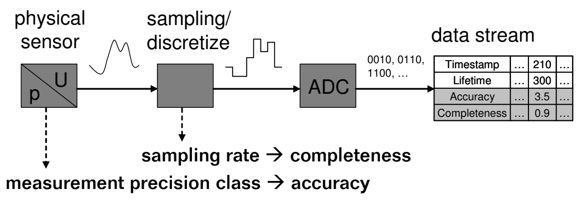


Figure 2. Design of a Sensor

The output of a sensor is a discretized and digitized data stream representing the measured physical values. The characteristics of the sensor define the data quality dimensions of the outgoing data stream. The metadata models presented in this paper are designed to cope with an unlimited number of data quality dimensions. Without loss of generality, we focus on the two important DQ dimensions accuracy and completeness.

The accuracy describes the numerical precision of a data value. It is stated in the absolute or relative error of a physical value. To initialize the propagation of the DQ dimension accuracy, the accuracy of the sensor is retrieved from the measurement precision class in the manufacturer's technical specification. A pressure sensor in our example is characterized by the precision class 1.5, which indicates the absolute measurement error as 1.5% from the maximal value range of 16 bar. Thus, the accuracy is described by the absolute error of 0.24 bar.

The completeness addresses the problem of missing values due to sensor failures or malfunctions. Multiple strategies exist to deal with missing values in ETL processes and data cleansing [8], whereas the estimation or interpolation of missing values is aspired in the majority of the cases. The data quality dimension completeness helps to distinguish between measured data items and estimated or interpolated ones. The sampling rate of the discretization defines the stream rate r (e.g. $100/s$, $1/10min$), which determines the stream length m dependant on the time t and thus serves as reference for the stream completeness c.

$$c = 1 - \frac{count(missingvalues)}{m} \tag{1}$$

$$c = 1 - \frac{count(missingvalues)}{r \cdot t} \tag{2}$$

The contributions (2) DQ propagation, (3) DQ storage and (4) DQ import are presented in detail in the following sections.

4. Data Quality Propagation in Data Streams

In this section the metadata model extension for data stream systems will be introduced. We present a straightforward approach of so-called naive data quality annotations as a first solution to the problems arising from restricted sensor DQ. To overcome the explosion of data volume given by this naive approach, we then propose the incorporation of jumping windows in the data stream metamodel. Finally, the DQ calculation and propagation in the presented jumping stream windows is described in detail.

4.1. Naive DQ Annotations

The naive approach of data quality annotations consists in streaming the data quality information for each DQ dimension (gray) with the same stream rate as the measurement stream (white) as shown in (see Figure 3). The data item is not only defined by its numerical values, but further described by its DQ information.

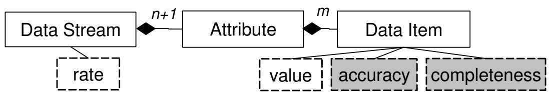


Figure 3. DSMS Metamodel with Naive DQ Annotations

A sensor data stream D of length m and rate r consists of *n+1* Attributes A_i $(0 \leq i \leq n)$, where A_0 represents the timestamp t of the sensor data stream. Each timestep t_j $(0 \leq j \leq m)$ indicates a tuple T_j with n measurement values v_{ij}.

For the naive DQ annotations every measurement value v_{ij} is enhanced with a data quality vector $\vec{q_{ij}}$ enclosing d data quality dimensions (e.g. accuracy a_{ij} and completeness c_{ij}).

$$v'_{ij} = \{v_{ij}, \vec{q_{ij}}\} \tag{3}$$

$$\vec{q_{ij}} = \begin{pmatrix} a_{ij} \\ c_{ij} \end{pmatrix} \tag{4}$$

Figure 4 shows a data stream extract of the residual lifetime of the truck's engine from the beginning. Every 10 days, the residual lifetime is estimated. It is calculated based on several sensors (i.a. oil pressure, oil temperature, mileage, number of coldstarts) with the given data quality dimensions accuracy and completeness. Analog to the sensor measurements, the sensors' data quality information were combined and aggregated to compute the quality of the residual lifetime.

Timestamp	...	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	...
Lifetime	...	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	...
Accuracy	...	3.5	3.5	2.9	2.1	3.0	2.7	4.2	5.1	3.8	3.7	2.3	2.6	1.9	3.7	3.4	2.9	2.7	3.6	3.2	1.9	...
Completeness	...	0.9	0.9	0.8	1	0.9	0.85	0.8	0.75	0.8	0.8	0.9	0.9	0.9	0.8	1	1	1	1	1	1	...

Figure 4. Lifetime with Naive DQ Annotations

Obviously, this approach significantly increases the data volume, which is multiplied by the number of considered DQ dimensions. The additional data volume S to transfer data quality results in $S = m \cdot n \cdot d$. Hence, this approach is not suitable for those applications with stringent resource constraints and should only be employed in case that communication costs for data transmission are not significant.

4.2. Extended DSMS Metadata Model

To reduce the additional data volume to transfer data quality information in a data stream, we propose the usage of jumping data quality windows. The concepts of our solution focuses on flexibility, represented by an unlimited number of supported DQ dimensions, a variable window size and adaptable aggregation functions to summarize the window data quality.

The widely used approach of sliding windows is not applicable in the context of efficient data quality propagation, because the tuple-wise update would require the tuple-wise DQ information propagated along with the data stream. Thus, no volume reduction for the data transfer would be achieved.

In this section jumping DQ windows are introduced in the data stream metamodel. Thus, the traditional DSMS metadata model has to be extended. As already mentioned, a sensor data stream D consists of n attributes A_i $(1 \leq i \leq n)$ representing sensor measurements. In the traditional metadata model, each attribute A_i is associated with an unrestricted number of data value items v_{ij}.

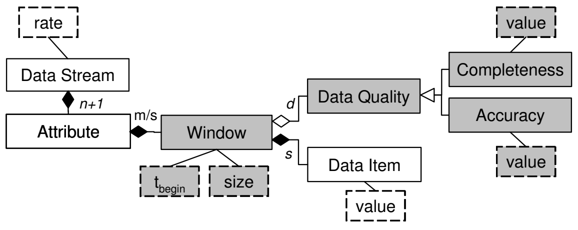


Figure 5. DSMS Metadata Model Extension

At this place the model extension is inserted. The notion of jumping windows is interposed in the relation between attribute and data item as shown in Figure 5. Each measurement attribute stream is parted in an unlimited number of windows with a given size s containing sensor data items (white) and data quality information (gray). Each window is identified by its starting point $t_{begin} = t_k$. It consists of s measurement values $v_{ij}(k \leq j \leq k + s - 1)$ of a certain attribute A_i. Furthermore, the window contains one value for each DQ dimension q_{ik} (e.g. window completeness c_{ik} and window accuracy a_{ik}).

The number of data quality dimensions is not fixed but variable for each attribute. Further, the window size s can be defined independantly for each stream attribute. The additional memory space to cover d_i data quality dimensions for each of n attributes A_i depends on the attributes' window size s_i and the stream length m.

$$S = m \cdot \sum_{i=1}^{n} \frac{d_i}{s_i} \tag{5}$$

4.3. Jumping Window Based DQ Annotations

For the jumping window based annotations, the data quality information is not sent together with every single data item but window-wise for each DQ dimension. The additional data volume is reduced to an acceptable degree by aggregating the data quality for each attribute $A_i(1 \leq i \leq n)$ in jumping stream windows w_{ik} of the given size s_i starting at timestamp $t_{begin} = t_{ik}$. Thereby, the aggregation functions can be flexibly determined for each DQ dimension corresponding to the underlying application. The attribute A_0 represents the timestamp, not a sensor measurement and thus is not equiped with data quality information.

The definitions in the following hold for each attribute A_i. For the sake of simplicity, we refer to the windows w_{ik} as windows w_k of size s, etc.

$$w_k = \{v_{ij}, \vec{q_k} | k \leq j \leq k+s-1\} \quad (6)$$

$$\vec{q_k} = f\left(\vec{q_j} | k \leq j \leq k+s-1\right) \quad (7)$$

The window w_k includes s sensor data items v_{ij} as well as the data quality vector $\vec{q_k}$ describing d data quality dimensions. The vector $\vec{q_k}$ represents the aggregated data quality information $\vec{q_j}$, which was associated with each data item (see Section 4.1), whereas no correlation in the various DQ dimensions are considered. Hence, the ith vector dimensions of $\vec{q_k}$ is computed only with the help if the ith dimension of $\vec{q_j}$. The vector function f incorporates the aggregation functions $f_l\ (1 \leq l \leq d)$ for all enclosed data quality dimensions.

In the following the data quality vector is shown for $d = 2$, including the window accuracy a_k and window completeness c_k.

$$\vec{q_k} = \begin{pmatrix} a_k \\ c_k \end{pmatrix} \quad (8)$$

$$a_k = f_a\left(a_j | k \leq j \leq k+s-1\right) \quad (9)$$

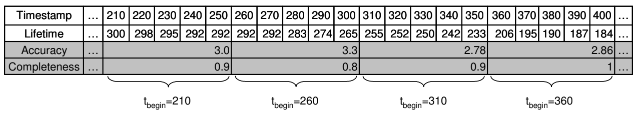
$$c_k = f_c\left(c_j | k \leq j \leq k+s-1\right) \quad (10)$$

The accuracy aggregation function f_a is not fixed, but rather can be adjusted to the application's requirements. At this point, the metadata model is configured to be as generic as possible to be adaptable for all kinds of use cases. For example, the window accuracy a_k can be calculated as the (weighted) linear or squared average as well as the maximum or minimum of the data items' accuracies a_j in the corresponding data stream window. To determine the adequate aggregation function one can revert to the knowledge acquired in the field of systematic and statistical error propagation. In our example, the linear average was used to calculate the accuracy of the truck engine's residual lifetime.

A meaningful function f_c to compute the window completeness c_k is given as the ratio of originally measured sensor data items (non-null values) to the window size s, $f_c = sum(c_j)/s$.

Figure 6 shows the resulting data quality for the residual lifetime of the truck engine. The DQ information provided for each data item (see Figure 4) are aggregated in jumping windows of size $s = 5$. Compared to the naive DQ annotation the obtained resource saving is clearly visible.

The window-wise calculation of the data quality dimensions may be executed at the embedded intelligent device the sensor is connected to or at every other point in the data stream system. However, to be as efficient as possible, the



Timestamp	…	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	…
Lifetime	…	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	…
Accuracy	…	3.0					3.3					2.78					2.86					…
Completeness	…	0.9					0.8					0.9					1					…
		t_{begin}=210					t_{begin}=260					t_{begin}=310					t_{begin}=360					

Figure 6. Lifetime DQ in Jumping Windows

DQ aggregation has to take place as near to the sensor as possible.

While the data quality is aggregated to form jumping windows information is lost, because no back-tracking to the original tuple-wise DQ information is possible. Subsequent to the DQ aggregation each tuple in the window is uniformly described by the vector $\vec{q_k}$. The error deviation introduced by the window-wise DQ aggregation is shown in the discrepancy between the window data quality q_k and the original tuple data quality q_{ij}. The error deviation in the window w_k can be estimated with the help of the standard deviation σ of the original tuple-wise DQ information $q_{ij}(k \leq j \leq k+s-1)$.

$$\sigma = \sqrt{\frac{1}{s}\sum_{i=1}^{s}(q_{ij} - \bar{q_{ij}})^2} \quad (11)$$

$$\sigma = \sqrt{\frac{1}{s}\sum_{i=1}^{s}(q_{ij} - q_k)^2} \quad (12)$$

If the aggregation function f_l is defined as the average of all tuple DQ in the respective window, the standard deviation represents equally the deviation from the window data quality as shown in equation 12. According to the Chebyshev's inequality at least 75% of the original errors lie in the interval $[q_k - 2\sigma; q_k + 2\sigma]$. If the errors are normally distributed, the fact holds for approximately 95%.

Experiments with real-world measurement data have to show, if the error deviation caused by the windowing is significantly high and has to be tracked.

Independent from this decision the error deviation in combination with the resource savings can be used as a heuristic to determine the window size s. To compare these two different domains, two cost functions C_1 and C_2 are introduced, describing the cost resulting from the error deviation and the resource saving, respectively. The goal is to minimize the overall cost C composed of these two cost functions.

$$min \quad C(s) = |C_1 \cdot \sigma - C_2 \cdot S'| \quad (13)$$

$$S' = m \cdot \sum_{i=1}^{n} d_i \cdot (1 - \frac{1}{s_i}) \quad (14)$$

The solution of the optimization problem gives the optimal window size s under this heuristic. However, the user may wish to zoom into interesting stream segments to get more detailed data quality information in smaller windows or otherwise zoom out of unspecific stream sections to reduce the data volume. The online adaption of the window size to meet changing user requirements will be discussed in future work.

5. Data Quality Management in Databases

The insertion of data quality information in relational databases is not yet supported at the metadata level. Therefore, we present the extension of the traditional relational metadata model to enable the efficient storage of data quality in a persistent database. The concept of jumping windows is also exploited in this context for efficient transmission of data quality information. Yet, the focus lies on the specific requirements posed by relational database management systems.

5.1. Extended DBMS Metadata Model

We propose to consider data quality as a new dimension in the relational metamodel. Every column in a relational table is enhanced with d data quality characteristics (a.k.a. DQ dimensions). Important thereby is the maintenance of the resource-saving window model, so that data quality information is not stored for every measurement value v_{ij}. Therefore, the database table containing sensor data is partitioned into relation windows analog to the jumping stream windows.

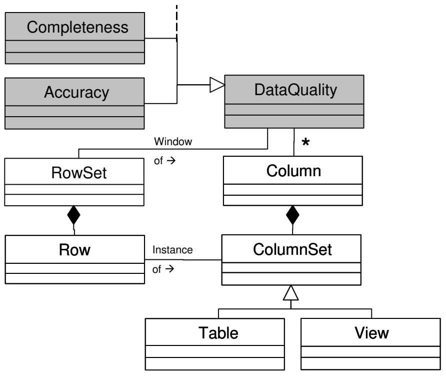


Figure 7. DBMS Metadata Model Extension

The relational metamodel extension is illustrated in the notation of the OMG standard Common Warehouse Model (CWM) as shown in Figure 7. A **Table** or **View** is composed as a **ColumnSet** of a given number of **Columns**, describing the table (or view) attributes. A **Row** represents an instance of a certain **ColumnSet** including the inserted data values.

The relational window to manage data quality information can be configured as a **RowSet** containing the sensor data of a certain time interval. The **Data Quality** associated to a certain **Column** is stored in reference to specific **RowSets** of the corresponding **ColumnSet**. As exemplary data quality dimensions Figure 7 shows the **Accuracy** and **Completeness**.

5.2. Table Schema for Data Quality

For the management of data quality in a relational database a system table (also often called catalogue table) **SysQuality** is introduced to the catalogue of the DBMS. Moreover, a new table type, the data quality table (DQ table), is introduced to store DQ information. For a relational user table containing measurement data, a DQ table describing the data quality of the mentioned measurements is automatically created. Figure 8 illustrates the schema of these new system and DQ table. For the sake of clarity, the DQ table will be named according to the table containing the corresponding sensor data <**TableName**> **_DQ**. The DQ table is easily extendable to an unlimited number of data quality dimensions, depending on the user requirements and/or availability of data quality information.

System Table for DQ management

SysQuality

Column Name	Type	Length	Nullable	Comment
QualityID	Char	36	false	Unique identifier of the data quality information
Dimension	Char	1	false	Data quality dimension 'A' – accuracy, 'C' – completeness, ...
DQTable	Char	36	false	Foreign key pointing to the table, where DQ information is stored
MeasColumn	Char	36	false	Foreign key to the column containing the measurement values
WindowSize	Integer	4	false	Size of the data quality window given by incoming data stream

DQ Table for DQ storage

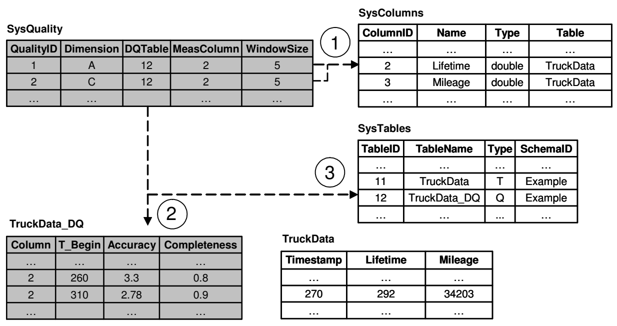
<TableName>_DQ

Column Name	Type	Length	Nullable	Comment
Column	Int	12	false	Foreign key pointing to the sensor data column
T_Begin	Int	12	false	Start timestamp of data quality window
Accuracy	Double	12	false	Value of the given window's accuracy
Completeness	Double	12	false	Value of the given window's completeness
...	...	...	...	Extendable to further DQ dimensions

Figure 8. DQ and System Table Schema

Figure 9 shows the existing user table (white) as well as the newly proposed system and DQ tables (gray) filled with measurement and quality data of the truck example. The column **SysQuality.MeasColumn** references the ID of the user table column, where the measurement data is stored (1). **SysQuality.DQTable** points to the DQ table **TruckData_DQ** containing the corresponding values of the data quality dimensions completeness and accuracy for each window starting at **T_Begin** (2). Furthermore, the tables **TruckData** and **TruckData_DQ** can also be found in the system table **SysTables** (3).

The example illustrates the consistent integration and management of data quality information in relational databases.



SysQuality

QualityID	Dimension	DQTable	MeasColumn	WindowSize
1	A	12	2	5
2	C	12	2	5
...	...		...	...

SysColumns

ColumnID	Name	Type	Table
...	...	...	...
2	Lifetime	double	TruckData
3	Mileage	double	TruckData
...	...	...	...

SysTables

TableID	TableName	Type	SchemaID
...	...	...	...
11	TruckData	T	Example
12	TruckData_DQ	Q	Example
...	...	...	...

TruckData_DQ

Column	T_Begin	Accuracy	Completeness
...	...	...	...
2	260	3.3	0.8
2	310	2.78	0.9
...		...	...

TruckData

Timestamp	Lifetime	Mileage
...	...	...
270	292	34203
...	...	...

Figure 9. Sample DQ and System Tables

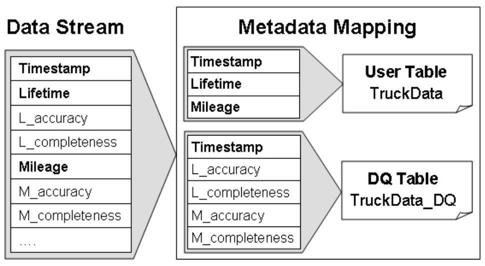


Figure 11. Mapping Procedure

6. Meta Mapping for DQ Import

This section closes the gap between the dynamic data stream system and the target database concerning the data quality transmission. The presented approach allows an easy mapping of the jumping windows in the DSMS metadata model to the relation windows in the DBMS metadata model. Only one additional insert operation per window is necessary to transfer the data quality information from the data stream into the target database.

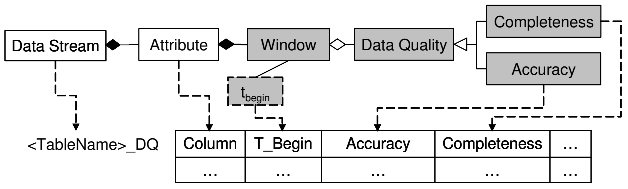


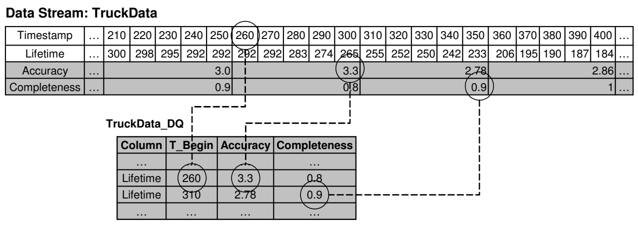
Figure 10. Metadata Mapping

Figure 10 shows the metamapping. The components of the data stream metadata model refer to the respective columns in the DQ table. For each incoming data stream, a DQ table is created and named according to the included measurements. The streaming attributes are reflected in the **Column**. The starting point **T_Begin** identifies the corresponding data quality window including **Accuracy** and **Completeness**.

In the first step of the metamapping the data stream is split into sensor and quality data as shown in Figure 11. Then, the sensor data is inserted into the corresponding user table. One relation tuple containing all measurement attributes is created for each incoming stream tuple. In the last step, the jumping stream windows are mapped to the relational windows of the database. The DQ tables are constructed to allow for the window-wise storage of data quality dimensions.

Hence, the data quality information is written to the respective data quality table for each window starting at t_{begin}. Each attribute window results in d entries, one for each data quality dimension. For n incoming attribute windows, $n \cdot d$ insert operations have to be executed. If information on a certain data quality dimension is not streamed along with the data, the missing DQ values are represented by null-values.

Figure 12 shows the mapping for the running truck example. The data quality information describing the **Lifetime** is inserted into the quality table **TruckData_DQ**.



Data Stream: TruckData

Timestamp	...	210	220	230	240	250	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	...
Lifetime	...	300	298	295	292	292	292	292	283	274	265	255	252	250	242	233	206	195	190	187	184	...
Accuracy	...					3.0					3.3					2.78					2.86	...
Completeness	...					0.9					0.8					0.9					1	...

TruckData_DQ

Column	T_Begin	Accuracy	Completeness
...	...	...	...
Lifetime	260	3.3	0.8
Lifetime	310	2.78	0.9
...	...	...	...

Figure 12. Sample Mapping

The presented mapping structure models the transfer from DSMS to DBMS. It allows the automatic generation of ETL loading procedures.

7. Conclusions

In this paper, we presented an efficient way to model data quality in data streams as well as relational databases. We introduced jumping DQ windows to enable the resource saving propagation of data quality information from the sensors through the data stream system up to a target database. The jumping window was incorporated into the traditional DSMS metadata model to allow for a uniform handling of data quality in data streams.

Because in most applications the sensor data is inserted in a persistent database for further analysis and knowledge discovery, we addressed the problem of data quality storage in relational databases. The idea of jumping stream windows was mapped to windows in a relational table. We presented an extended metadata model for the DBMS as well as the schema of required system and data quality tables to manage and store DQ information in a constistent, persistent way.

Though, we refered to accuracy and completeness as two important DQ dimensions for sensor data streams, the proposed metadata models are of generic structure to be easily extended by additional data quality dimensions. Moreover, the illustrated metadata models allow for an easy mapping to close the gap between streaming environment and target database.

Future work will be to implement the proposed metadata extensions, both for DBMS as well as DSMS, and to configure the model mapping. The goal will be to provide an end-to-end architecture from the sensors to the target database, which allows a transparent data quality capturing, propagation and storage. Thereby, special attention will lie on the determination of adequate window sizes and the adaption of the window size to changing user requirements.

Furthermore, the pre-processing of data quality information poses a challenging research topic. A DQ algebra has to be defined to transfer all operators of the data pre-processing in a data stream environment onto data quality dimensions.

References

[1] Abadi, D.; Carney, D.; etc.; *Aurora: A Data Stream Management System* SIGMOD 2003

[2] Babcock, B.; Babu, S.; etc.; *Models and Issues in Data Stream Systems*, PODS 2002

[3] Ballou, D.P.; Tayi, G.K.; *Enhancing Data Quality in Data Warehouse Environments*, ACM 1999

[4] Biswas, J.; Naumann, F.; Qiu, Q.; *Assessing the Completeness of Sensor Data*, DASFAA 2006

[5] Burdick, D.; Deshpande, P.; Jayran, T.S.; *OLAP over Uncertain and Imprecise Data*, VLDB 2005

[6] Chandrasekaran, S.; Cooper, O.; etc.; *TelegraphCQ: Continuous Dataflow Processing for an Uncertain World*, CIDR 2003

[7] Kang, J.; Naughton, J.; Viglas, S.; *Evaluating Window Joins over Unbounded Streams*, VLDB 2002

[8] Lee, M.; Lu, H.; etc.; *Cleansing Data for Mining and Warehousing*, DEXA 1999

[9] Lee, Y.; Strong, M.; Wang, R.; *Data Quality in Context*. CACM 1997

[10] Leser, U.; Naumann, F.; *Query Planning with Information Quality Bounds*, FQAS 2000

[11] Moon, Y.-S.; *Efficient Stream Sequence Matching Algorithms for Handheld Devices on Time-Series Stream Data*, DBA 2006

[12] Scannapieco, M.; Batini, C.; *Completeness in the Relational Model: a Comprehensive Framework*, IQ 2004

[13] Tatbul, N.; Zdonik, S.; *Window-aware Load Shedding for Aggregation Queries over Data Streams*, VLDB 2006

[14] Wang, R.; Ziad, W.; Lee, Y; *Data Quality*, 2001