Technical Report CS-01-16

Privacy Protection through Query Rewriting in Smart Environments

Hannes Grunert, Andreas Heuer

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik
Lehrstuhl für Datenbank- und Informationssysteme

Long version of a 2-page conference paper for the 19th International Conference on Extending Database Technology (EDBT) 2016,
published as ISBN 978-3-89318-070-7,
on OpenProceedings.org

erhältlich unter:
www.ls-dbis.de/digbib/dbis-tr-cs-01-16.pdf

Privacy Protection through Query Rewriting in Smart Environments

Hannes Grunert, Andreas Heuer
Database Research Group
University of Rostock
18051 Rostock
(hg|ah)(at)informatik.uni-rostock.de

Abstract: By the events in the past years, the integration of data protection mechanisms into information systems becomes a central research problem again. In this technical report for our poster, we show how query rewriting can be used to maintain privacy of users in smart (or assistive) environments. We developed a privacy respecting query processing and a vertical fragmentation of queries, processing maximal parts of the query as close to the sources of the data (e.g. sensors) as possible.

1 Introduction to Assistive Systems

Assistive systems are designed to support the user at work (Ambient Assisted Working, AAW) and at home (Ambient Assisted Living, AAL). Active and passive RFID tags, cameras, microphones, but also sensors on light switches and power sockets capture the current situation in the ubiquitous environments, up to 100 times per second. These data are stored by the system and linked to other data, for example social network profiles.

Our main application scenarios are smart environments. These systems are equipped with a variety of sensors which collect a lot of information. In our Smart Meeting Room at the Graduate School MuSAMA[1] (see Figure 1) we try to support the users to perform a lecture or a meeting by calculating their intentions and future actions.

To achieve this, we have to find out what happens during the meeting by recognizing the current actions in the room. For this, the room is equipped with the following sensors and devices:

- **lamps**: multiple, dimmable lamps to enlighten the room.
- **screens**: multiple screens, which can be turned up and down.
- **powersocket**: multiple electrical outlets, which tracks how much milliamperes are used.
- **pensensor**: For each pen at the Smart Board it is tracked, which one has been taken.

[1] Multimodal Smart Appliance Ensembles for Mobile Applications

Figure 1: The Smart Appliances Lab at the University of Rostock. Through a variety of sensors, meetings and lectures are supported by the automatic, intention-driven control of the room.

- **thermometer**: a thermometer, which checks the temperature in (Unit: °C).
- **ubisense-tags and -sensor** [Cad03]: One tag per user; the sensor supplies a coordinate (x, y, z) in the room's Cartesian system and a lot of other information (e. g. whether the position is valid or not).
- **sensfloor** [SL11]: the center of the room has integrated sensors in the floor, which checks where someone stands and how much pressure he or she exerts.
- **extron** and **vgasensor**: These devices track, which (video) port is connected to a specific projector.
- **eibgateway**: A device to control the blinds of the Smart Meeting Room.

Besides of the meeting room scenario, other MuSAMA-related projects apply these techniques to AAL-systems supporting elderly people to be able to live longer in their own apartment on their own (instead of having to go to a nursing home).

While the assistive capabilities of such smart systems are useful, there is a problem in this massive amount of processed data that not only enables the intended support such as fall detection in an apartment for the elderly, but also enables other big data analytics procedures to calculate a complete personal profile of the person in the apartment.

2 Introduction to Privacy

In the digital world steadily more and more information about ourselves and our environment is collected. Besides "classical" personal information, such as the name, age or gender, a plurality of sensors records our activities and inclinations. Based on the obtained information, preferences, patterns of behavior and future events can be calculated. Furthermore, intentions and future actions of the users are derived, so that the smart environment can react independently to satisfy their needs.

Especially assistive systems using activity and intention recognition [KNY+14] are a possible cause of privacy violations, especially if the query realizing the recognition query is performed on a cloud server. The activity and intention recognition is a typical Big Data Analytics technique [Heu15] that enables other uses of the smart system and its data than the intended one, resulting in privacy violations.

To reduce privacy violations, it is necessary

- to decrease the amount of collected personal information, i.e. to apply the principle of data avoidance (except where data are required),
- to process data with personal references as less as possible or — at least — as close to the local data sources (sensors) as possible and
- to anonymize, pseudonymize and delete personal data, unless it is used for further processing and necessary to realize the aim of the assistive environment.

Data minimization and data avoidance are therefore prescribed, indispensable requirements for the design of smart systems. This requirement can be achieved in databases by transforming both queries and query results, as well as using views [HL98].

3 Privacy-aware Query Processing

The PArADISE[2]-approach aims to withdraw the burden of respecting privacy constraints from the assistive systems by adding privacy protection mechanisms to those systems storing and analyzing the data: database systems on different levels. The PArADISE approach combines performance aspects of big data analytics (by using massively parallel database technology [MH15]) and privacy protection. Our privacy aware query processor (see Figure 2) generates anonymized result sets. These data maintain a high degree of value for the initial query generated by the assistive system. On the opposite, additional knowledge can hardly be derived.

The preprocessor allows the analysis and the rewriting of database queries regarding user-defined privacy policies [GH15]. During the execution of the request, it is decided whether the request will be answered and anonymized directly on the current network peer, or

[2] Privacy Aware Assistive Distributed Information System Environment

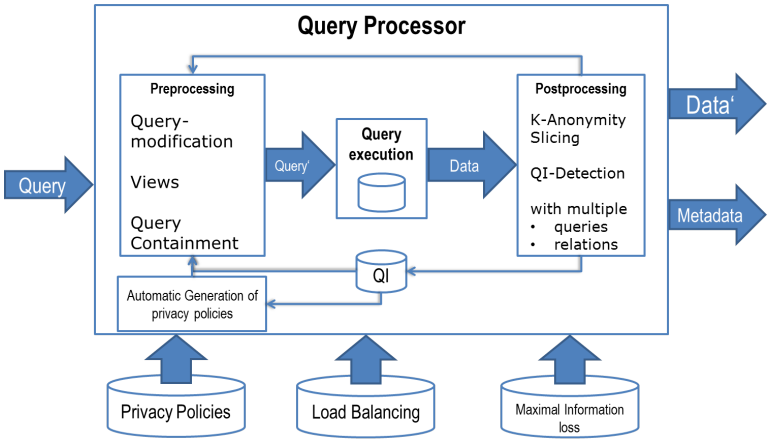

Figure 2: The concept of the privacy-aware query-processor.

is sent to lower nodes (vertical fragmentation, see below). The postprocessor executes the anonymization of the query results, taking into account various criteria of quality and privacy. For this, various data protection metrics and algorithms are provided. The module for the automatic generation of privacy settings produces and adapts existing user-defined privacy policies to new devices and changing requirements and queries.

3.1 Preprocessing

During the preprocessing stage, the preliminary query is analyzed and checked against the privacy policy of the affected user. The involved personal information queried by the system is monitored, whether it is uncovered by the user at all (projection) and if it can be used under user defined constraints. These constraints can be used to decide if the revealed information will be preselected or aggregated. Furthermore, it is also checked if the processing node has enough capacity (sufficient CPU-power, free main memory). Also, it is tested if the information system could gain enough information to produce satisfactory results. To determine how satisfactory the result is, we are using the information loss metric based on the Kullback Leibler divergence [KL51], which has been shown to be a good approximation to determine how much information remain [HS10].

Finally, the constraints are used for a modification of the query which fulfills all needs. For modification, we analyze the different parts of the (SQL:2011-)queries. Attributes in the *SELECT* clause are removed, if the user does not want to reveal specific information. If one sensor releases too much information, another sensor is queried by changing the relation in the *FROM* clause. In case that not all tuples shall appear in the result, the WHERE condition is combined with the user's integrity constraints and the system query conjunctively.

3.2 Postprocessing

Taken the preliminary result from the intermediate stage, the postprocessor checks the required information of the system and the privacy settings of the user. At this time, the result is modified with privacy-preserving algorithms like k-anonymity [Sam01] or data slicing [LLZM12], if and only if the processing unit has enough power. Thus, the modified result is sent back to the system. In case that a unit does not have enough power, the raw data will be sent to a more powerful node and anonymized later.

Under due consideration of information loss metrics, like the Kullback-Leibler-Distance [KL51] or the *Direct Distance* (DD), the quality difference between the original and the anonymized data can be calculated. It must apply that the loss of information for the intended queries should be kept to a minimum while the loss of information for the unintended query should be as high as possible. A simple metric to calculate the information loss is the Direct Distance which is defined by

$$DD(R, R') = \sum_{i=1}^{n} \sum_{j=1}^{m} distance(i, j),$$

where n is the number of tuples and m the number of attributes in the original relation R resp. in the anonymized relation R', and

$$distance(i, j) := \begin{cases} 0, \text{ if } value[R_{i,j}] = value[R'_{i,j}] \\ 1 \text{ else} \end{cases}$$

calculates the number of unequal attribute values between R and R'.

The ratio of different values in R' to the total number of values in R ($m * n$) defines the quality of the anonymized result of the query. Now, the goal is to find the "Golden Path" between information loss and a high privacy level.

In the last years, several algorithms based on k-anonymity [Sam01], differential privacy [Dwo11] and similar concepts have been developed. But there exists no *one-size-fits-all* solution which solves all problems. By choosing an appropriate anonymization concept for the intended analysis, it is possible to purposefully anonymize the data.

The decision, which concept is used in the later implementation, must be closely done with the developers of analysis functionalities.

By prior analysis and rewriting of the queries it can thereby be determined which attributes can be used for anonymization, and whether the anonymization should be done column- (e.g. Slicing [LLZM12]) or tuple-wise (e.g. k-anonymity [Sam01]).

3.3 Privacy policies for smart environments

Our data privacy policy is based on the draft of the W3C for Privacy Preferences Platform [W3C07], but leaves out browser-specific details, such as the management of cookies. In

return, our policy provides additional information for configuring data streams, such as the allowed query interval and possible aggregation levels.

Hitherto, security models and privacy policies aim at protecting whole data records from unauthorized access. Adding a more fine-granular access control allows the user to specify which information is revealed and how the data are processed. Data records can be modified by

- selection,
- projection,
- compression,
- aggregation and
- feature extraction (e.g. on motion images)

to reduce the amount of personal information sent towards the tools for evaluation and analysis.

Our privacy policy is based on the draft of the W3C for Privacy Preferences Platform [W3C07], but leaves out browser-specific details, such as the management of cookies. In return, our policy provides additional information for configuring data streams, such as the allowed query interval and possible aggregation levels.

4 Query Rewriting by Vertical Fragmentation

The smart environment or assistive system sends a query request Q to the database d integrating the entire sensor data recorded in our environment. The result of Q is needed to perform the activity and intention recognition. The data sources are sensors being located in appliances in apartments and buildings which sum up to the mentioned (distributed) database, being d the subset of this information in a specific environment such as one single apartment.

4.1 Vertical Fragmentation of Queries

Instead of shipping d to the cloud server sending the request, maximal parts of Q will be evaluated as close to the sensor as possible. As can be seen in Figure 3, instead of performing $Q(d)$ in the cloud, the maximal subquery Q_j will be shipped to the next lower node of the processing chain, in the case of the example a PC located in our apartment. While Q performs an iterative machine learning algorithm implemented in R and SQL, and Q_j being a complex SQL query with recursion, the lowest node in the processing chain (the sensors) can only compute some filter mechanisms (simple selections) and some simple aggregations over the last values generated (window function: average of last minute).

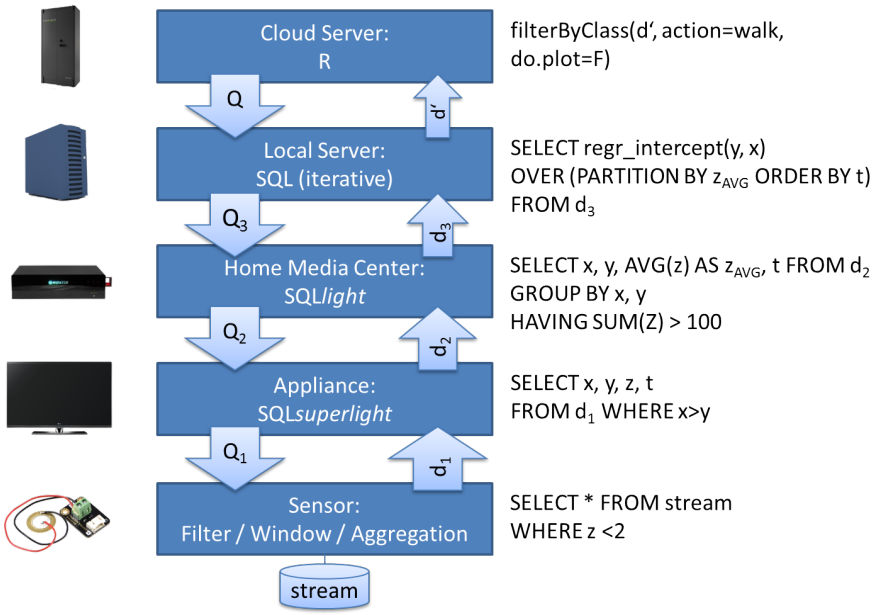

Figure 3: Vertical Query Fragmentation: Query and query result transformation on different peers.

Each of the nodes will ship the query result d_j to the node sending the request. After a final anonymization step A, the data "leaving our apartment" d' will only be a small subset of the original data d.

We assume that the lower nodes will each have less query computing power than the higher nodes: while sensors are only performing simple filters / selections and aggregations, an appliance like a TV or a smart network music player will be able to perform simple (SQL-superlight) database queries, an Android-based home media server even more complex SQL queries. This situation is shown in Table 1: besides having more computer power on higher levels of our vertical architecture, and more capabilities for expressing machine learning algorithms, we will have more computers / processors per person on the lower nodes of the architecture. Normally, one assistive system in an apartment of a person will have hundreds of sensors combined in ten to fifty appliances (like Home Media Centers, TVs, network music players and even carpets with fall detection).

The whole query processing procedure

$$Q(d) := d_j = Q_j(\dots d'_i = A(d_i = Q_i(\dots(d_1 = Q_1(d))\dots)$$

is transformed in a way, that the cloud server will perform a remainder query Q_δ on d' instead of performing Q on d, hence resulting in the privacy protecting query rewriting $Q(d) \longrightarrow Q_\delta(d')$. In other words, the query Q is fragmented in queries Q_j (that can be performed at a lower node) and a remainder query Q_δ that can only be performed at the more powerful node of our vertical fragmentation of query processors.

Level	System	Capability	Number of nodes
E1	cloud	complex ML algorithm in R, SQL:2003 with UDF	n for m person, $n \ll m$
E2	PC in apartment	SQL-92	1 for 1 person
E3	appliance in apartment	SQL "light" with joins	10 - 50 for 1 person
E4	sensor in appliance (TV), in environment (carpet)	filter / window, simple selection, aggregates on streams (over the last seconds)	$\gg 100$ for 1 person

Table 1: Overview on different systems relating to their capability to execute SQL queries.

Since recognizing the maximal possible SQL queries in an R machine learning algorithm is undecidable in general, we try to detect some larger "SQLable" patterns [Weu16] in the activity and intention recognition procedures described in [KNY+14]. Other aspects of our privacy-aware query-processor are described in, e.g. [Gru14a].

By rewriting the query Q into Q_j and Q_δ and only performing Q_δ outside our "privacy protected" appliance ensemble in our apartment, we hope to automatically prevent the service provider of our assistive system to use our personal data in a way we did not consider to be possible when starting to use his smart service. A remaining open problem is to decide whether a privacy-violating query $Q_\downarrow$ can be performed even on d' instead of d. In this case, we have to extend the anonymization step A already performed. This open problem results in a query containment problem[3] of $Q, Q_\downarrow$ and Q_j that will be part of our further research.

4.2 Use Case Scenario

While smart environments and assistive systems are calculating the result of analysis on a central hub, the required data is normally distributed on multiple databases. To perform the analysis, a database query Q is send to the database d which integrates the whole raw data collected by the sensors of the environment. The result of the query Q is needed to perform particular activity and intention recognition algorithms. In the following example, we assume that multiple sensors are placed in our apartement.

Let us imagine that we ordered an extensive assistance service provided by the company *Poodle* for our apartement. Poodle, once a pure provider of a search engine, today is also a provider of smart home technologies like intelligent, sensor based heating thermostats, Smart TVs, kitchenware, carpets and furniture. From Poodle we expect certain services, such as an automated problem identification like fall detection. To offer these services, Poodle's cloud server has to send queries (like Q mentioned above) to the sensor data in our apartment.

Because Poodle offers their assistive environment with low cost, the company wants to sell the derived personal profiles to third parties. In contrast to Poodle, we don't want that

[3] For more information regarding Query Containment Problem, see [Chi09] or chapter 8 in [KSS14]

to be done. Therefore, we are glad that Poodle offers this service also with a PArADISE based option, which is a little more expensive but is privacy-aware.

Instead of transfering all data from d to the cloud server of Poodle that executes the query, maximal parts of Q are extracted and performed as close as possible at the sensor level. In Figure 3, we can see that $Q(d)$ will not be executed in the cloud, but the maximal subqueries Q_j will be transmitted to the next node in the process chain. In our example, a computer placed in our apartment is assumed at this position.

Our example is based on the following R-code for the analysis of sensor data (excerpt of a Kalman filter):

```
1 filterByClass(sqldf(
2 SELECT regr_intercept(y, x)
3 OVER (PARTITION BY z ORDER BY t)
4 FROM(SELECT x, y, z, t
5 FROM d')
6 ), action=''walk'', do.plot=F)
```

The function *filterByClass* filters the data of a DataFrame (R data type) to detect if the executed activity was equal to "walk". There will be no graphical output of the result of the analysis (*do.plot=F*). Taking the result of the nested SQL query, the function *sqldf* transforms it to a R-readable format. The SQL query realizes a complex regression analysis with partitioning, aggregates and *ORDER BY* clauses. The data originate from a source d' which is part of a peer-to-peer network (see Figure 3).

Our network consists of cloud server, local server, as well as embedded computer and small sensors with different performance characteristics regarding computational power, available memory and compliance to different SQL standards. In our example, the raw data is extracted from a small motion recognition sensor from the lowest level. The original query Q is transmitted over several nodes towards the sensor. In the process, all data from the sensor are queried (*SELECT * FROM* d_i respectively *SELECT * FROM stream*).

In addition to the query, there exists a privacy [Gru14b, Mei15] for the assistive system w.r.t. the activity recognition filter (see Figure 4; simplified description). In this policy, two privacy claims are given: The x-value of the sensor has to be greater than the y-value at any time (x>y). Considering the z-value, it has to be always less than 2 (z<2) and can only appear in an average-aggregation, grouped by the corresponding x- and y-values, having the sum of z less than 100.

For the transformation process, the query is rewritten with respect to the given privacy policy. Thereby, the additional conditions will be inserted as *WHERE* and *HAVING* clauses in the innermost possible part of the nested SQL query. Regarding aggregated values, new attribute names are inserted and, if necessary, delegated to the out queries.

```
1 filterByClass(sqldf(
2 SELECT regr_intercept(y, x)
3 OVER (PARTITION BY zAVG ORDER BY t)
4 FROM(SELECT x, y, AVG(z) AS zAVG, t
5 FROM d'
6 WHERE x>y
```

```
<module module_ID="ActionFilter">
 <attributeList>
  <attribute name="x">
   <allow>true</allow>
   <condition>
    <atomicCondition>
     x&gt;y
    </atomicCondition>
   </condition>
  </attribute>
  <attribute name="y">
   <allow>true</allow>
  </attribute>
  <attribute name="z">
   <allow>true</allow>
   <condition>
    <atomicCondition>
     z&lt;2
    </atomicCondition>
   </condition>
   <aggregation>
    <aggregationType>
     AVG
    </aggregationType>
    <groupBy>x, y</groupBy>
    <having>SUM(z)&gt;100</having>
   </aggregation>
  </attribute>
  <attribute name="t">
   <allow>true</allow>
  </attribute>
 </attributeList>
</module>
```

Figure 4: Privacy policy for the ongoing example.

```
7  AND␣z<2
8  GROUP␣BY␣x,␣y
9  HAVING␣SUM(z)␣>␣100)
10 ),␣action=''walk'',␣do.plot=F)
```

Afterwards, the transformed query will be distributed to the particular computation nodes. While the R-code will still be executed on the cloud server, the local server has enough power to perform the regression analysis part of the SQL query on its own.

```
1 filterByClass(d',
2 action=''walk'',␣do.plot=F)
```

The result of the regression analysis d' will be provided as an R-DataFrame on the cloud server. On the local server, it is possible to separate the inner SQL query in order to outsource it to the home media center. The inner query performs the regression analysis over the defined partition.

```
1 SELECT regr_intercept(y, x)
2 OVER (PARTITION BY zAVG ORDER BY t)
3 FROM d3
```

The media center itself takes only the aggregation part of the SQL query. In our example, the analysis categorizes movement activities based on the height (z). Based on the privacy policy, the z-coordinate is restricted to its average value grouped by x and y. To ensure that enough values are considered for this aggregation, the sum of z has to be more than 100 per grouping set.

```
1 SELECT x, y, AVG(z) AS zAVG, t
2 FROM d2
3 GROUP BY x, y
4 HAVING SUM(z) > 100
```

The *WHERE*-clause is shifted to an appliance. Basic comparison operations, like *less-than* or *equals* between two attributes, are executed on the appliance. The rest of the *WHERE*-condition is pushed down towards the sensor. Unnecessary attributes are also filtered out. For the regression analysis, only the x-, y- and z-coordinate and the timestamp are remaining.

```
1 SELECT x, y, z, t
2 FROM d1
3 WHERE x>y
```

At the sensor, it is not possible to select single attributes, so all attributes are queried at once. While the appliance was able to compare two attributes against each other, the sensor can only compare an attribute against a constant.

```
1 SELECT *
2 FROM stream
3 WHERE z<2
```

After retrieving the data from the sensor the result set, will be sent through the processing chain back to the cloud server. Meanwhile, the data will be anonymized and processed step by step.

5 Summary and Outlook

In this paper we presented a vertical architecture to fragment queries against sensor data in a smart environment such that large portions of this query can be pushed down as close to the source of the data as possible. By this vertical fragmentation of queries into pushed-down and remainder queries we hope to support privacy constraints of the users of these smart environments such as assistive systems.

As a second (post-processing) technique, we are additionally able to anonymize the result sets of the data by detecting quasi-identifiers and using coumn-wise or tuple-wise anonymization.

We applied our concept in a smart home scenario as a use case and demostrated how a machine learning algorithm written in R can be transformed and pushed down in several steps resulting in some simple filter operations at the sensor level.

The data used for this purpose consists of sensor data that has been recorded in the Smart Appliance Lab of the graduate school MuSAMA [BK12].

A remaining open problem is to decide whether a privacy-violating query can be performed even on reduced data set. In this case, we have to extend the anonymization step already performed. This open problem results in a query containment problem that will be part of our further research.

References

[BK12] Sebastian Bader and Thomas Kirste. An overview of the helferlein-system. *Institut für Informatik, Universität Rostock, Rostock, Germany, Tech. Rep. CS-03-12*, 2012.

[Cad03] Jay Cadman. Deploying commercial location-aware systems. In *Proceedings of the 2003 Workshop on Location-Aware Computing (held as part of UbiComp 2003)*, pages 4–6, 2003.

[Chi09] Rada Chirkova. Query Containment. In *Encyclopedia of Database Systems*, pages 2249–2253. Springer US, 2009.

[Dwo11] Cynthia Dwork. Differential Privacy. In Henk C. A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 338–340. Springer, 2011.

[GH15] Hannes Grunert and Andreas Heuer. Generating Privacy Constraints for Assistive Environments. In *Proceedings of the 8th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015*. ACM, 2015.

[Gru14a] Hannes Grunert. Distributed Denial of Privacy. In *INFORMATIK 2014: Big Data Komplexität meistern*, pages 2299–2304. Springer, 2014.

[Gru14b] Hannes Grunert. Privacy Policy for Smart Environments. `http://www.ls-dbis.de/pp4se`, 2014. last accessed on 27.07.2015.

[Heu15] Andreas Heuer. METIS in PArADISE — Provenance Management bei der Auswertung von Sensordatenmengen für die Entwicklung von Assistenzsystemen. In Norbert Ritter, Andreas Henrich, Wolfgang Lehner, Andreas Thor, Steffen Friedrich, and Wolfram Wingerath, editors, *Datenbanksysteme für Business, Technologie und Web (BTW 2015) - Workshopband, 2.-3. März 2015, Hamburg, Germany*, volume 242 of *LNI*, pages 131–136. GI, 2015.

[HL98] Andreas Heuer and Astrid Lubinski. Data Reduction - an Adaptation Technique for Mobile Environments. In *Interactive Applications of Mobile Computing (IMC'98)*, 1998.

[HS10] Ayça Azgin Hintoglu and Yücel Saygın. Suppressing microdata to prevent classification based inference. *The VLDB Journal*, 19(3):385–410, 2010.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[KNY+14] Frank Krüger, Martin Nyolt, Kristina Yordanova, Albert Hein, and Thomas Kirste. Computational State Space Models for Activity and Intention Recognition. A Feasibility Study. *PLOS ONE*, November 2014. 9(11): e109381. doi:10.1371/journal.pone.0109381.

[KSS14] Veit Köppen, Gunter Saake, and Kai-Uwe Sattler. *Data Warehouse Technologien, 2. Auflage*. MITP, 2014.

[LLZM12] Tiancheng Li Li, Ninghui Li, Jian Zhang, and Ian Molloy. Slicing: A New Approach for Privacy Preserving Data Publishing. *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD)*, 24(3):561–574, March 2012.

[Mei15] Marc-Eric Meier. Implementierung eines Data-Mining-Verfahrens für adaptive Datenquellen. Bachelorarbeit, Universität Rostock, 2015.

[MH15] Dennis Marten and Andreas Heuer. A Framework for Self-managing Database Support and Parallel Computing for Assistive Systems. In *Proceedings of the 8th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015*. ACM, 2015.

[Sam01] Pierangela Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010–1027, 2001.

[SL11] Axel Steinhage and Christl Lauterbach. Sensfloor and Navifloor: Large area sensor systems beneath your feet. *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives*, 2:41–55, 2011.

[W3C07] W3C. Platform for Privacy Preferences (P3P) Project. `http://www.w3.org/P3P/`, 2007.

[Weu16] Dennis Weu. Erkennung, Übersetzung und parallele Auswertung von Operatoren und Funktionen aus R in SQL. Bachelorarbeit, Universität Rostock, 2016.

Impressum

Universität Rostock
Institut für Informatik
Lehrstuhl für Datenbank- und Informationssysteme
Albert-Einstein-Straße 22
18059 Rostock

Vertreten durch: Prof. Dr. rer. nat. habil. Andreas Heuer