PL/SQL basierte Datenanonymisierung und -pseudonymisierung

Daniel Günther
SHS VIVEON AG
München

Schlüsselworte
Anonymisierung, Pseudonymisierung, PL/SQL, Testdaten

Einleitung

Gesetzliche Vorgaben und ein risikobewusstes Handeln machen es notwendig, kundenbezogene Daten im Data Warehouse Umfeld zu anonymisieren oder zu pseudonymisieren. So soll sichergestellt werden, dass nur wenige Personen für bestimmte Zwecke Zugriff auf produktive Kundendaten erhalten. Alle anderen Nutzer können diese Daten ohne Personenbezug auswerten. Durch Anonymisierung der Produktivdaten kann ein konsistenter Testdatenbestand erstellt werden. Gerade dort, wo Massendaten von mehreren Terabyte verarbeitet werden, stellt diese Anforderung das Systemdesign immer wieder vor Herausforderungen.

Der Vortrag stellt einen metadatengetriebenen, leicht anzupassenden, PL/SQL-basierten Ansatz zur Anonymisierung bzw. Pseudonymisierung vor. Dabei wird auf die verschiedenen Anforderungen wie z.B. Erhaltung der Datentypen oder Erhaltung der Foreign Key Constraints und der sich daraus ergebenden Schwierigkeiten eingegangen. Die Anonymisierung bzw. Pseudonymisierung lässt sich mit verschiedenen Ansätzen wie z.B. Hashen, Dynamisches Joinen, Anlegen/Pflegen von Referenztabellen usw. umsetzen. Der Vortrag geht auf die einzelnen Umsetzungmöglichkeiten ein und zeigt die Vor- und Nachteile auf. Es wird besonderes Augenmerk auf eine performante Implementierung der Lösung gelegt.

Im Folgenden wird zwischen den Data Masking Verfahren Anonymisierung und Pseudonymisierung unterschieden. Bei der Anonymisierung werden Werte so geändert, dass nicht mehr auf den Originalwert geschlossen werden kann, z.B. die Abbildung der Emailadresse von Daniel.Guenther@SHS-VIVEON.com auf Xxxxxx.Xxxxxxxx@XXX-XXXXXX.com . Ein Wert gilt als pseudonymisiert, wenn nur mit erheblichem Aufwand auf den Originalwert geschlossen werden kann. Zu Pseudonymisierungsmethoden zählen z.B. Übersetzungslisten.

Motivation / Anforderung

In DWH Projekten besteht immer wieder die Notwendigkeit, konsistente Test- und Entwicklungsdatenbestände zur Verfügung zu stellen. Der hohe Aufwand konsistente, synthetische Testdaten für komplexen DWH Umgebungen zu generieren, verstärkt den Wunsch, die vorhandenen Produktionsdaten dafür zu nutzen. Für einen ETL-Entwickler besteht kein Unterschied wenn z.B. für ein Telefongespräch der Teilnehmervorname statt des Originalnamens *Daniel* den Namen *Scott* enthält. Der Austausch aller personenbezogenen Daten gegen andere Werte führt zu einem ansonsten konsistenten Testdatenbestand. Dies ist die Grundidee für das im nachfolgenden beschriebenen PL/SQL basierten Anonymisierungs- und Pseudonymisierungsframework.

Zu dieser einfachen Idee kamen insbesondere von den zukünftigen Nutzern der Testdaten eine ganze Reihe von Anforderungen hinzu. An dieser stellen seien nur die wichtigsten aufgeführt:

- der Datentyp bleibt erhalten
- die Datentyplänge bleibt erhalten

- bestehende Unique Key Constraints werden nach einer Anonymisierung/Pseudonymisierung nicht verletzt
- Referentielle Integritäten bleiben erhalten
- Besondere Fachlichkeit bleibt erhalten, z.B. 1. Stelle KontoNr.
- Ein Originalwert wird immer zum gleichen Wert anonymisiert bzw. pseudonymisiert sowohl zeitlich als auch örtlich (PROD, UAT, TEST, DEV)

Konzept

Der erste Lösungsansatz, ein geschütztes Schema mit Übersetzungstabellen für jedes zu anonymisierende Attribut aufzubauen, wurde aus Performancegründen verworfen. So müsste vor jedem Anonymisierungslauf sichergestellt werden, dass für jeden neuen Satz die notwendigen Übersetzungswerte generiert werden. Alternativ könnte das Füllen und Generieren mit Triggern gelöst werden. Die hohen Datenmengen im DWH Umfeld verbieten den Einsatz von zeilenbasierten Triggern jedoch.

Ebenfalls als nicht zielführend erwies sich der Ansatz, jedes personenbezogene Attribut durch einen Hashwert zu ersetzen. Abgesehen von der recht hohen CPU Last, die beim Generieren von z.B. MD5 Hashes auftritt, kann es zu Hash Collisions kommen. Außerdem ist ein Hashwert wie z.B. *19654581E9C563CE5A40C93521385634* für einen Entwickler schlecht als Name vorstellbar. Datentypen und Längen können bei diesem Ansatz nicht identisch gehalten werden, weshalb die Datenstrukturen aufgeweicht werden müssten.

In einem Proof of Concept wurde untersucht, inwieweit sich das Oracle Datamasking Pack des Oracle EM 11g für diese Aufgabe nutzen lässt. Grundsätzlich konnten mit dem Werkzeug anonymisierte bzw. pseudonymisierte Testdaten aus dem Produktionsdatenbestand abgeleitet werden. Dazu bringt des Oracle Datamasking Pack einen soliden Umfang an Funktionen mit. Der Aufwand, die Funktionen an die eigenen Anforderungen anzupassen, bleibt jedoch. Verworfen wurde die weitere Umsetzung mit dem Data Masking Pack im Wesentlichen aus zwei Gründen. Erstens wegen der Eigenschaft für jede zu anonymisierende Tabelle immer eine temporärere Tabelle zu erzeugen, in welche unter Zuhilfenahme weiterer Zwischentabellen das Ergebnis der Anonymisierung geschrieben wird. Anschließend wird die Originaltabelle gelöscht. Im regulären Ladebetrieb ist das eine sehr ressourcenintensive und gefährliche Prozedur. Zweitens werden für die Durchführung des Anonymisierungsprozesses unnötigerweise DBA Rechte benötigt.

Aus diesen Erfahrungen heraus, wurde die Entscheidung getroffen, eine eigene Lösung zu entwickeln. Zentraler Bestandteil der eigenen Lösung ist ein **Metadatenmodell,** in welchem die Tabellennamen und Spaltennamen mit personenbezogenen Inhalten erfasst werden. Das Metadatenmodell löst im Wesentlichen zwei Probleme:

1. Gleichartige Attribute wie z.B. Vorname oder First_Name aus verschiedenen Vorsystemen werden über ein sogenanntes **Businessattribut** zusammengefasst. Diesem Businessattribut wird eine Anonymisierung- bzw. Pseudonymisierungssmethode zugeordnet. So wird später sichergestellt, dass gleiche Inhalte aus verschiedenen Systemen mit einer identischen Methode bearbeitet werden.
2. Tablesets fassen Tabellen zusammen, deren Daten gemeinsam anonymisiert werden soll. So können Abhängigkeiten hinterlegt werden, auch wenn zwischen Tabellen kein Foreign Key Constraint besteht.

Ein **Generator** erzeugt aus diesen Metadaten **PL/SQL Packages**, die die eigentliche Anonymisierung/Pseudonymisierung umsetzen und **Views,** die den Zugriff auf Original- bzw. anonymisierten Wert steuern. Für jede zu anonymisierende Tabelle wird ein Package erzeugt. Die Packages greifen wiederum auf die Anonymisierungs- bzw. Pseudonymisiserungs**library** zu, die die Methoden bereitstellt, um ein einzelnes Attribut zu modifizieren.
Der **Controller** steuert anhand der Tablesets den Ablauf der Anonymisierung.
Im nachfolgenden wird auf die Hauptbestandeile L**ibrary, Package, View und Controller** im Detail eingegangen.

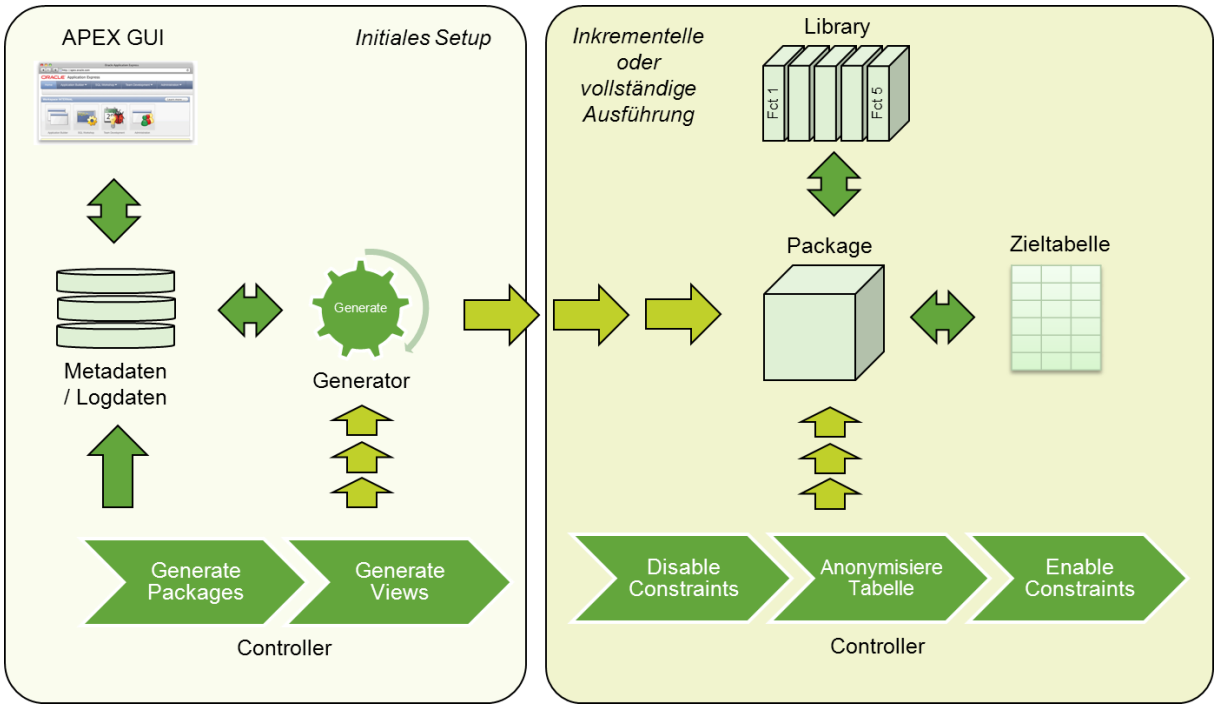

Abbildung 1 Architektur

Library

Die Library ist der wichtigste Bestandteil der entstanden Anonymisierungs- und Pseudonymisierungsengine. Sie enthält die implementierten Funktionen, die für jedes Businessattribut unter Angabe des Originalwertes einen anonymisierten Wert zurückgeben. Neben der eigentlichen Anonymisierung/Pseudonymisierung werden dabei spezielle Regeln beachtet, wie z.B. das Erhalten von Präfixen und Suffixen oder das Erhalten des Geburtsjahres aus dem Geburtsdatum. Einige Beispiele sind in folgender Tabelle aufgelistet.

Business Attribut	Original Wert	Anonymisierter Wert	Regel
E-Mail Adresse	hans.m@abc.de	fdsvfm@uzt.de	- Erhalten der TLD (.de) - Wähle Zufallsnamen
Geburtstag	15.08.1973	03.12.1973	- Geburtsjahr bleibt erhalten - Zufallstag und Zufallsmonat

Vorname	Daniel	Scott	- Wähle per Hashfunktion aus vorgegebener Liste von Namen
Number_1 (default)	12345	96835	- Generiere Zufallszahl gleicher Länge
...	...	...	...

Tabelle 1 Beispiele der Anonymisiserungsbibliothek

Da im DWH Umfeld häufig mit großen Datenmengen umgegangen werden muß, wurde Wert auf eine möglichst performante Implementierung gelegt. Deshalb arbeiten alle Funktionen, die Werte „übersetzen“, mit PL/SQL Tabellen. Vorteil der PL/SQL Tabellen: Sie können im Session RAM gehalten werden. Die Zugriffzeit ist also sehr kurz.

Non-Unique Attribute
Für jedes Non-Unique Attribut wird eine eigene PL/SQL Lookup Tabelle INDEX BY BINARY INTEGER mit den Übersetzungswerten in den RAM gelegt. Während des Anonymisierungslaufs wird der Originalwert nicht direkt in der PL/SQL Tabelle gesucht, sondern per ORA_HASH Funktion nur eine Slotnummer ermittelt. Mit dieser Slotnummer wird der zu anonymisierende Wert aus der PL/SQL Tabelle ausgelesen und zurückgegeben. Dieses Verfahren hat den Vorteil, ein konstantes Set an Übersetzungsdaten nutzen zu können. Damit entfällt der Aufwand, für jeden in einer Tabelle neu hinzugekommenen Originalwert einen Übersetzungswert vor dem Anonymisierungslauf generieren zu müssen.

Unique Number Attribute
Schwieriger ist das Verfahren bei Unique Number Attributen. Ein Problem ist der zur Verfügung stehende Wertebereich. Dieser ist mit zehn Ausprägungen ([0-9]) deutlich kleiner als bei zeichenbasierten Attributen ([a-Z],[0-9]). Das größte Problem stellt jedoch die Anforderung der Eindeutigkeit nach der Pseudonymisierung dar. Insbesondere IDs wie Konto- oder Versicherungsnummern sind sehr lang, Werte größer 10^{12} sind eher die Regel als die Ausnahme. Um alle potenziell auftretenden Werte pseudonymisieren zu können, sind dann 10^{12} oder mehr Werte erforderlich. Tabellen solcher Größenordnung lassen sich mit heutiger Technologie nicht effizient joinen. Eine PL/SQL Tabelle nach dem Vorbild der Non-Unique Attribute kann mangels Memorylimits nicht aufgebaut werden. Deshalb müssen die IDs in kleine Teilstücke zerlegt werden. Eine sinnvolle Maximalgröße eines Stückes ergibt sich bei 10^{6}. So benötigt die PL/SQL Tabelle für die Pseudonyme von 0 bis 999.999 ca. 51 MB PGA.
Der Orginalwert 37624578900 wird in die beiden Stücke 37624 und 578900 aufgeteilt. Jedes Stück wird separat pseudonymisiert. Anschließend werden die Pseudonyme wieder zusammengesetzt.

Folgende Vorteile der vorgestellten Implementierung der Library ergeben sich:

- Kapselung der Anonymisierung/Pseudonymisierung in einem Package
- Flexible Einsetzbarkeit der Funktionen
  Der Einsatz ist nicht auf die generierten Packages beschränkt, sondern die Funktionen können beispielsweise auch in Verbindung mit dem REMAP_DATA Operator von Data Pump genutzt werden.
- Vorberechnung der Pseudonyme
- In-memory Verarbeitung

Es ergeben sich aber auch Einschränkungen. Die Grenze des nutzbaren Session Memory limitiert die Anzahl der möglichen Pseudonyme.

Die Nutzung von PL/SQL + PL/SQL Tabellen ist in erster Linie CPU abhängig. Bei DWH Systemen stellt dies kein Problem dar, da diese in der Regel io-begrenzt sind. Dennoch sind einige Regeln zu beachten, um möglichst performante Anonymsisierungs- bzw. Pseudonymisierungsfunktionen bereitzustellen, wie z.B.:

- PARALLEL_ENABLE + DETERMINISTIC, auch für Funktionen, die zufällige Werte zurückgeben
- Komprimierung in wenige Zeilen Code
- effiziente Kodierung, z.B. möglichst wenige Stringoperationen nutzen
- native Compilation, auch für DBMS_RANDOM, STANDARD
- PL/SQL DEBUG ausschalten
- PL/SQL Optimization Level 3, bzw. Inline Pragma für geschachtelte Prozedur- / Funktionsaufrufe
- Vorberechnen von Konstanten, wie z.B. 10^1, 10^2 ... 10^6

Package

Haupteinsatzgebiet der Library ist die Anonymisierung bzw. Pseudonymisierung on-the-fly während der regulären Beladung. Jedoch muss jede Tabelle einmal initial anonymisiert werden bzw. besteht die Anforderung, für die Test- und Entwicklungssysteme die Daten auch unabhängig von den Ladeläufen anonymisieren zu können. Für diesen Fall wird je zu anonymisierender Tabelle ein PL/SQL Package generiert. Dieses enthält Methoden für die vollständige und inkrementelle Anonymisierung bzw. Pseudonymisiserung. Zusätzlich werden diverse Supportmethoden implementiert, die z.B. die Foreign Key Constraints ein- und ausschalten, eine strukturgleiche temporäre Tabelle inklusive Indizes, Constraints, Triggern, Kommentaren und Grants anlegen und Funktionalität für das Logging bereitstellen.

Auch bei den Methoden des Packages ist auf eine möglichst performante Implementierung Wert gelegt worden. So wird die Anonymisierungs- bzw. Pseudonymisierungsfunktion gar nicht aufgerufen, wenn der Originalwert NULL oder ein in den Metadaten hinterlegter Defaultwert ist. In diesem Fall wird NULL bzw. der Defaultwert in der Spalte belassen. Dadurch wird der Aufruf der PL/SQL Engine eingespart. Das dafür notwendige CASE / DECODE Statement arbeitet wesentlich effizienter.
Ziel ist es, die PL/SQL Engine so selten wie möglich aufzurufen. Für die inkrementelle Anonymisierung / Pseudonymisierung führen folgende Maßnahmen zu einer performanteren Verarbeitung:

- Update via Join Update
- Prüfung ob Originalwert ungleich NULL und zu anonymisierender Wert noch nicht gefüllt
- Update, Sortieren über Source RowId, um physical IO zu minimieren

Beispielhaft seien die Maßnahmen an folgender einfachen inkrementellen Anonymsierungsmethode gezeigt, die für eine Tabelle das Geburtsdatum anonymisiert:

```
update "D_CUSTOMER"
   set select "BIRTH_D_A" =
PKG_DA_ANONYMIZATION_LIB.ANONYMIZE_BIRTHDAY_DAT("BIRTH_D");
 where "BIRTH_D_A" is null;
```

SQL Statement 1 übliche Implentierung

```
update ( select "BIRTH_D_A",
                "BIRTH_D",
                …
           from "D_CUSTOMER"
          where ("BIRTH_D_A" is null and "BIRTH_D" is not null)
                …
          order by rowid )
   set "BIRTH_D_A" =
       case when BIRTH_D = TO_DATE('01.1900', 'MM.YYYY')
            then BIRTH_D
            when "BIRTH_D_A" is null and "BIRTH_D" is not null
            then PKG_DA_ANONYMIZATION_LIB.ANONYMIZE_BIRTHDAY_DAT("BIRTH_D")
            else "BIRTH_D_A" end
       …;
```

SQL Statement 2 performante inkrementelle Implementierung

Controller

Ein zentrales Packages, der Controller, steuert das Generieren aller notwendigen Packages und Views sowohl für ausgewählte Tabellen als auch Tabellengruppen (Tableset). Neben den eigentlichen Update- und Insertstatements werden auch Anweisungen zum Setzen der Parallelisierung der ausführenden Session in das Package generiert. Die Hauptaufgabe des Controllers besteht jedoch in dem metadatengesteuerten Ablauf eines initialen oder inkrementellen Anonymisierungslaufes.

Während eines Anonymiserungs- bzw. Pseudonymisierungslaufs stellt der Controller sicher, dass eine Tabelle bzw. eine Tabellengruppe nur von einem Lauf manipuliert wird. Das Starten eines weiteren Laufs wird verhindert, um Deadlocks und Inkonsistenzen in den anonymisierten Daten zu verhindern.

Es hat sich gezeigt, dass der Zyklus *Create Temp Table – Insert Into Temp Table – Drop Original Table – Rename Temp Table zu Original* für kleine Tabellen (< 1.000.000 Records, konfigurierbar) keine besonderen Vorteile bringt. Deshalb wird bei kleinen Tabellen die vollständige Anonymisierung auch über ein Update umgesetzt. Das Controllerpackage ermittelt anhand der hinterlegten Metadaten, ob eine Tabelle klein ist und ruft die entsprechende Methode dazu auf.

Um die Performance weiter zu steigern, können neben der Oracle Parallel Option alle Tabellen eines Tablesets auch unabhängig voneinander also zum gleichen Zeitpunkt bearbeitet werden. Der Controller stellt dazu sicher, dass in einem ersten Schritt die Foreign Key Constraints aller beteiligten Tabellen und sofern betroffen auch die Primary Key / Unique Key Constraints abgeschaltet werden. Im Anschluß erfolgt das zeitgleiche Anonymisieren bzw. Pseudonymisieren und der Wiederaufbau der Indizes sowie Primary Key / Unique Key Constraints. Ist dieser Prozess für jede Tabelle aus der Gruppe abgeschlossen, erfolgt das Wiederanschalten der Foreign Key Constraints.

Das Controllerpackage ist der zentrale Aufrufpunkt für APEX GUI.

Anonymisierung im DWH Kontext

Ziel der Entwicklung der Library war die Anonymisierung während des regulären Befüllens des DWH durchzuführen. Aus diesem Grund werden zusätzlich zu den Spalten mit den Originalwerten die anonymisierten bzw. pseudonymisierten Werte im Staging Layer berechnet.

Um den Aufwand für die Datenbank zu minimieren, werden die Funktionen nur auf das Inkrement zwischen zwei Ladeläufen angewendet.

Die Weitergabe der anonymisierten bzw. pseudonymisierten Daten in die oberhalb des Staging Layers liegenden DWH Layer unterscheidet sich je nach Environment (PROD, TEST, DEV). In der Development- und Testumgebung werden oberhalb des Staging Layer ausschließlich anonymisierte Daten weitergegeben. In der Produktionsumgebung besteht jedoch die Anforderung, bestimmten Benutzern je nach Rolle und Funktion im Unternehmen Zugriff auf die Echtdaten zu ermöglichen. Deshalb werden bis in die Data Mart Schicht sowohl anonymisierte Werte als auch die Originalwerte weitergereicht.

Für jede Data Mart Tabelle wird eine View erzeugt, die mittels einer dem selektierenden Benutzer zugeordneten Rolle entscheidet, ob der Originalwert oder der anonymisierte Wert zurückgegeben wird.

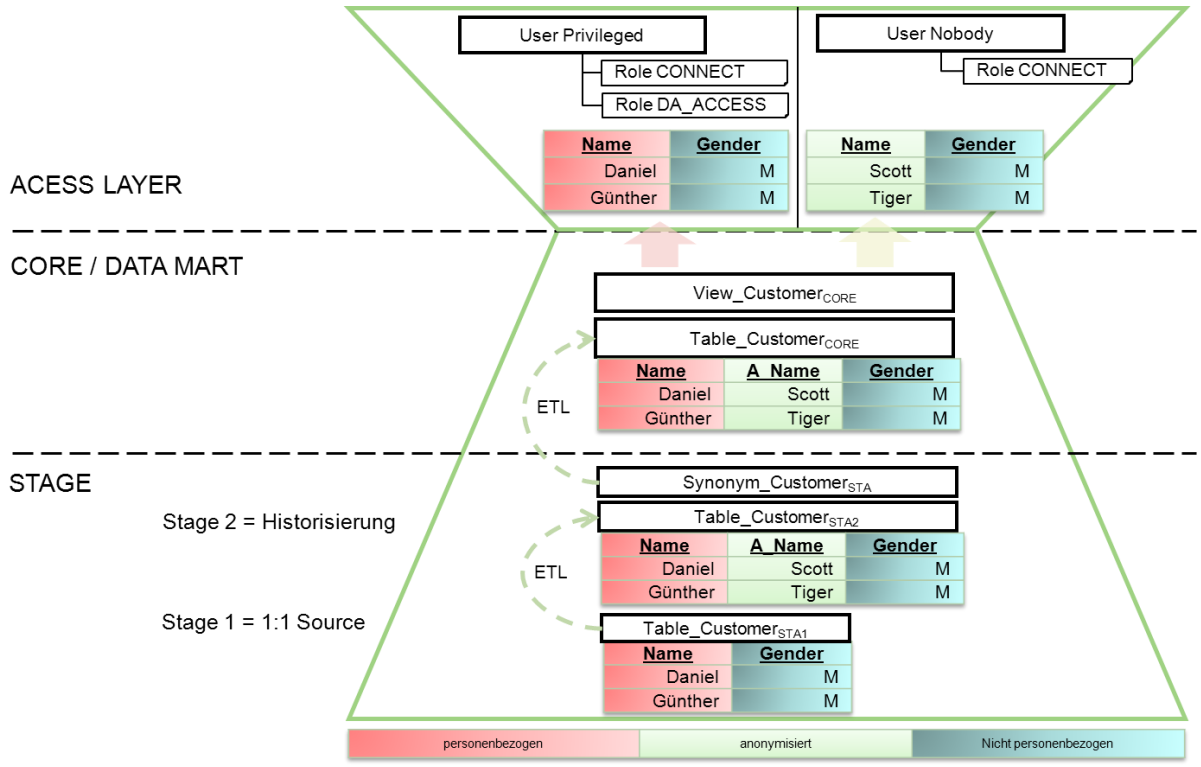

Abbildung 2 Anonymisierung innerhalb der DWH Layer in der Produktionsumgebung

Mit dem vorgestellten Ansatz sind die wichtigsten technischen Bestandteile eines Anonymisierungs- und Pseudonymisierungsframeworks angesprochen. Nicht betrachtet wurde der nicht zu unterschätzende fachliche Teil für jedes Attribut festzulegen, ob es personenbezogene Daten enthält und mit welcher Methode es zu anoymisien bzw. pseudonymiseren ist.

Kontaktadresse:
Daniel Günther
SHS VIVEON AG
Clarita Bernhardstraße 27

D-81249 München

Telefon: +49 (0) 89 747 257 0
Fax: +49 (0) 89 747 257 900
E-Mail Daniel.Guenther@SHS-VIVEON.com
Internet: www.SHS-VIVEON.de