SensorCloud

DBAP 5 Schema Monitor

15.01.2015, Henning Budde

Version 2.0

1. Anforderungen und Ziele des Schema Monitors

1.1 Motivation

Im Rahmen des Projekts SensorCloud werden von verschiedensten Sensoren und Aktoren deren Konfigurationen, deren Messwerte und Steuerbefehle in unterschiedlichen Datenbanken, mal in einer lokalen Datenbank eines LocationMasters, mal auf einem Cloud Knoten des föderierten Datenbanksystems abgelegt. Um auf Änderungen wie beispielsweise neue Sensor- oder Aktortypen reagieren zu können, soll die Struktur dieser Datenbanken veränderbar sein.

1.2 Anforderungen

Das föderierte Datenbanksystem ist so aufgebaut, dass unterschiedlichste Datenbankmanagementsysteme zum Einsatz kommen können. In DBAP2[1] wurden diverse unterschiedliche Datenbankmanagement Systeme analysiert und für das FDBS der SensorCloud vorgeschlagen. Alle diese Vorschläge sollen auch für den Schema Monitor als zu überwachende Komponenten vorgesehen werden.

Um dies zu realisieren, wird zunächst ein zentrales Abbild des Datenbank Schemas benötigt. Ein solches globales Datenbank Schema für ein föderiertes Datenbanksystem liegt im JSON Format vor und soll als Ausgangspunkt für den zu entwickelnden Schema Monitor dienen. Abbildung 1 zeigt einen Überblick über das föderierte Datenbanksystem.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

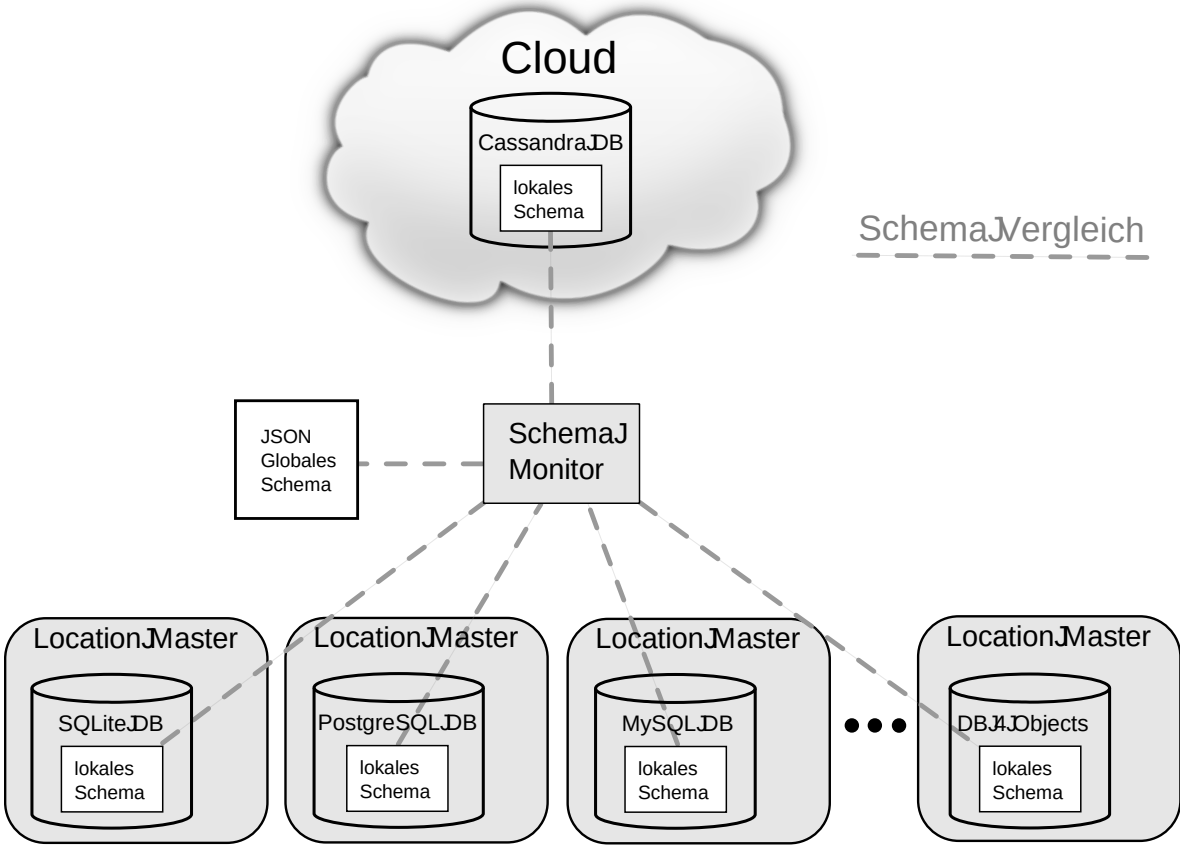

Abbildung 1: Föderiertes Datenbanksystem mit Schema Monitor

Werden Änderungen an dem JSON Globalem Schema vorgenommen, so sollen diese durch den Schema Monitor in den lokalen Schemata aller Location Master Datenbanken sowie der Cloud Datenbank übernommen werden. Zusätzlich soll es die Möglichkeit geben bestimmte Änderungen der lokalen Schemata der Cloud oder der LocationMaster Datenbanken, sowohl in das JSON Globale Schema, als auch auf die lokalen Schemata der restlichen Datenbanken, zu übernehmen. Dazu müssen Regeln aufgestellt werden, wann eine solche Änderung im globalen Schema bzw. in den Zieldatenbanken wirksam werden darf.

In dem Fall, dass das globale JSON Schema geändert werden muss, soll der aktualisierte JSON Code der betroffenen Entitäten generiert werden. Für Änderungen an lokalen Datenbank Schemata soll zunächst ein Änderungsskript in der DDL[1] der jeweiligen Zieldatenbank erstellt werden, welches manuell oder automatisiert ausgeführt werden kann, um Schema Anpassungen vorzunehmen.

Der Schema Monitor unterstützt folgende Datenbanksysteme:

- Cassandra

[1] Data Dictionary Language

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

- Postgre SQL
- MySQL
- SQLite
- DB4Objects

Auf folgende DBMS kann der Schema Monitor erweitert werden

- Dateisystem: z.B. als invertiertes Dateisystem für Cloud-Systeme

Zusätzlich sollen folgende Dienste vom Schema Monitor angeboten werden:

- Backup eines lokalen Schemas in die Cloud
- Wiederherstellen eines Backups aus der Cloud

2. Konzeptionelle Überlegungen

2.1 Schema Vergleich

Um unterschiedliche Datenbank Schemata vergleichen zu können müssen diese zunächst in eine vergleichbare Form gebracht werden. Relevante Informationen sind dabei Tabellennamen sowie Name und Datentyp der einzelnen Spalten. Um diese Informationen vergleichen zu können werden sie hierfür als Liste von Zeichenketten notiert. Eine Tabelle wird von der Trennzeichenkette „$$$“ angeführt, gefolgt vom Namen der Tabelle, sowie pro Spalte jeweils Name und Datentyp.

Die Folgende Tabelle zeigt eine Beispielliste für eine Datenbank mit den Tabellen Sensor und Aktor, mit jeweils zwei Spalten:

Index	Inhalt	Bedeutung
0	**$$$**	**Trennzeichen**
1	**Sensor**	**Start der Tabelle Sensor**
2	**SenID**	**Name der ersten Spalte der Tabelle Sensor**
3	**text**	**Datentyp der ersten Spalte der Tabelle Sensor**
4	**SenBez**	**Name der zweiten Spalte der Tabelle Sensor**

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

5	**text**	**Datentyp der zweiten Spalte der Tabelle Sensor**
6	**$$$**	**Trennzeichen**
7	**Aktor**	**Start der Tabelle Aktor**
8	**AktID**	**Name der ersten Spalte der Tabelle Aktor**
9	**text**	**Datentyp der ersten Spalte der Tabelle Aktor**
10	**AktBez**	**Name der zweiten Spalte der Tabelle Aktor**
11	**text**	**Datentyp der zweiten Spalte der Tabelle Aktor**

Abbildung 2: Vergleichsformat des Schema Monitors

Besonders zu beachten ist dabei, dass auch die Datentypnamen normalisiert werden müssen, da diese sich in unterschiedlichen Datenbankmanagementsystemen unterscheiden können. Während in einer MySQL Datenbank Zeichenketten üblicherweise den Datentypnamen *VARCHAR* haben, werden sie in einer Cassandra Datenbank als *text* bezeichnet.

Für das im Schema Monitor verwendete Vergleichsformat werden ausschließlich die für das FDBS definierten Datentypen beachtet.

Name	Inhalt
text	Zeichenketten
int	Ganze Zahlen
decimal	Gleitkommazahlen
timestamp	Zeitstempel

Abbildung 3: Datentypen des Globalen Schemas des FDBS

Gemäß des konzeptionellen Schemas des FDBS[2] sind *Constraints* (Integritätsbedingungen) vorgesehen, diese sind allerdings rein formal. Sie werden zwar im JSON Globalen Schema angegeben und können genutzt werden, sie werden jedoch nicht in den lokalen Datenbanken und im Cloud Schema implementiert und müssen demnach vom Schema Monitor nicht beachtet werden.

Bei einem direkten Vergleich zweier Schemata wird erkannt welche Tabellen und Spalten in den jeweiligen Datenbanken fehlen, beziehungsweise zu viel sind. Einer Erkennung in welche Datenbank die Veränderung vorgenommen wurde ist nicht möglich.

Um festzustellen, ob zum Beispiel eine Spalte in einer Tabelle der einen Datenbank gelöscht oder ob die gleiche Spalte in der Tabelle der anderen Datenbank hinzugefügt wurde, wird zunächst das JSON Globale Schema mit der letzten gespeicherten Version des JSON Globalen Schemas verglichen um fest zu stellen, was am Schema genau verändert

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

wurde. Anschließend wird das JSON Globale Schema mit einer der anderen Datenbanken verglichen und alle Änderungen die dabei gefunden wurden, die nicht am JSON Globalen Schema festgestellt wurden, werden als Änderungen an der jeweiligen Datenbank registriert. Anhand eines Berechtigungssystems kann schließlich festgestellt werden, welche Änderungen an welcher Datenbank notwendig sind.

2.2 Berechtigungen

Um festzustellen, welche Änderungen auf alle Datenbanksysteme übernommen werden sollen, gibt es eine Berechtigungsmatrix. Diese enthält als Zeilen die Datenbanktypen: JSON Globales Schema, Cloud Datenbank und LocationMaster Datenbanken und als Spalten die Berechtigungen: Erzeugen und Löschen von Tabellen und Spalten sowie ändern des Datentyps einer Spalte. Ist eine Änderung an einer Datenbank erlaubt, so wird sie auf alle anderen Datenbanken übertragen, ist sie es nicht, wird sie rückgängig gemacht.

Die nachfolgende Tabelle veranschaulicht eine beispielhafte Berechtigungsmatrix, wenn in ein Feld ein *X* eingetragen ist, ist eine Änderung erlaubt, ist nichts eingetragen, so ist eine Änderung nicht erlaubt:

	Tabelle erzeugen	Tabelle löschen	Spalte erzeugen	Spalte löschen	Spalte ändern
JSON Schema	X	X	X	X	X
Cloud DB	X		X		
LocMa DB					

Abbildung 4: Schema Monitor Berechtigungsmatrix

2.3 Sicherheit beim Löschen

Das automatisierte Löschen von Tabellen oder Spalten einer Datenbank ist problematisch, da es im schlimmsten Falle zu einem unwiderruflichen Datenverlust führen kann. Um sicher zu stellen, dass keine wichtigen Daten verloren gehen, wäre eine Versionierung der Datenbanken mit der Hilfe von *Views* denkbar. Bei Views handelt es sich um virtuelle Tabellen, die keine eigenen Daten enthalten, sondern lediglich auf Spalten richtiger Tabellen verweisen. Soll eine Spalte oder Tabelle gelöscht werden, so würde sie in einer höheren Version einfach nicht mehr auftauchen, ohne dass die eigentliche Tabelle und somit die gespeicherten Daten betroffen sind. Ein zusätzlicher Vorteil wäre die Abwärtskompatibilität mit älteren Versionen der Datenbank. Aufgrund der heterogenen Datenbankmanagementsysteme, die der Schema Monitor unterstützen soll, ist diese Lösung nicht anwendbar. Sowohl SQLite als auch Cassandra unterstützen lediglich „*Materialized Views*“, auf die im Gegensatz zu herkömmlichen Views nur lesend

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

zugegriffen werden kann. Bei einer Cassandra Datenbank fällt nur der Versionierungsaspekt weg, da das Konzept der Datenbank ohnehin keine festen Spalten Definitionen vorsieht und man so beliebige Spalten in einzelnen Datensätzen löschen und hinzufügen kann.

Für eine SQLite Datenbank müsste jedoch eine extra Lösung entworfen werden und im Hinblick auf eine leichte Erweiterbarkeit des Schema Monitors durch weitere unterstützte Datenbankmanagementsysteme, dessen Funktionsumfang im Vorfeld nicht bekannt sind, wurde sich für eine einfachere, vom Datenbankmanagementsystem unabhängige Lösung entschieden.

Über ein Flag wird gespeichert, ob eine Datenbank als *Sicher* markiert ist. Ist dies der Fall, werden Spalten oder Tabellen nicht gelöscht, sondern durch Anhängen des Suffixes *„_backup“* an den Namen gekennzeichnet. Der Schema Monitor ignoriert alle Spalten und Tabellen mit diesem Suffix beim weiteren Vergleich der Schemata.

Zusätzlich gibt es neben dem vollautomatischen Schema Vergleich mit anschließendem Ausführen aller erkannten Änderungen die Möglichkeit nur einen Vergleich durch zu führen und sich per Email informieren zu lassen, wenn eine Änderung notwendig ist, ohne diese automatisch aus zu führen.

2.4 Groß- und Kleinschreibung von Entitätsnamen

Gemäß der Spezifikationen des Konzeptionellen Schemas[2] sind Tabellen- und Spaltennamen in der *„CamelCase“-Schreibweise*[2] zu erstellen. Nicht alle Datenbankmanagementsysteme unterstützen dies. Viele relationale Datenbankmanagementsystem sind nicht „Case Sensitive“ und es macht keinen Unterschied ob ein Entitätsname durchgehend groß, klein oder in CamelCase-Schreibweise erstellt wird. Sollten also in Datenbanken mehrere Schreibweise für ein und dieselbe Entität auftreten, so wird immer die CamelCase-Schreibweise bevorzugt.

3. Implementierung

Der Schema Monitor wurde im Rahmen eines Praxissemesterprojekts[3] von Herrn Alexander Klein als Java Programmsystem entwickelt. Eine Beschreibung der Klassen findet man in [3] S. 12ff. Der Schema Monitor kann sowohl auf einem Cloud Knoten als auch auf einem LocationMaster betrieben werden.

[2] zusammengesetzte Wörter wobei jedes Wort mit einem großen Buchstaben anfängt

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

4. Benutzung des Schema Monitors mit seinen Diensten

4.1 Inbetriebnahme des Schema Monitors

Da es sich bei dem Schema Monitor um ein Java Programm handelt muss auf dem Rechner, auf dem er ausgeführt werden soll, die Java Laufzeitumgebung installiert sein. Der Schema Monitor setzt darüber hinaus keine weitere Software voraus.

Zum ersten Start müssen die Regeln des Schema Monitors erstellt bzw. die vorhandenen Grundregeln angepasst werden.

Der Pfad zum JSON Globalen Schema muss gesetzt werden und die Zugangsdaten für die Datenbankverbindungen (Lokal und Cloud) eingetragen werden.

```
+-----------------------------------------------------+
|              Sensor Cloud Schema Monitor            |
|                                                     |
|          Hauptmenue                                 |
|          ---------                                  |
|            (R)egeln aendern                         |
|            (L)ocation Master verwalten              |
|            (C)loud DB anpassen                      |
|            (J)SON Schema anpassen                   |
|            (S)chema vergleichen                     |
|            (B)ackups                                |
|            (A)enderungsskript ausfuehren            |
|           e(X)it                                    |
+-----------------------------------------------------+
```

Abbildung 5: Menü des Schema Monitors

4.1.1 Einstellen der Regeln

Einstellen der Regeln für die automatische Änderungsskript Generierung durch Auswahl der Option „(R)egeln aendern“. Ein eingetragenes X heißt dabei, dass eine Änderung durch die jeweilige Datenbank erlaubt ist.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

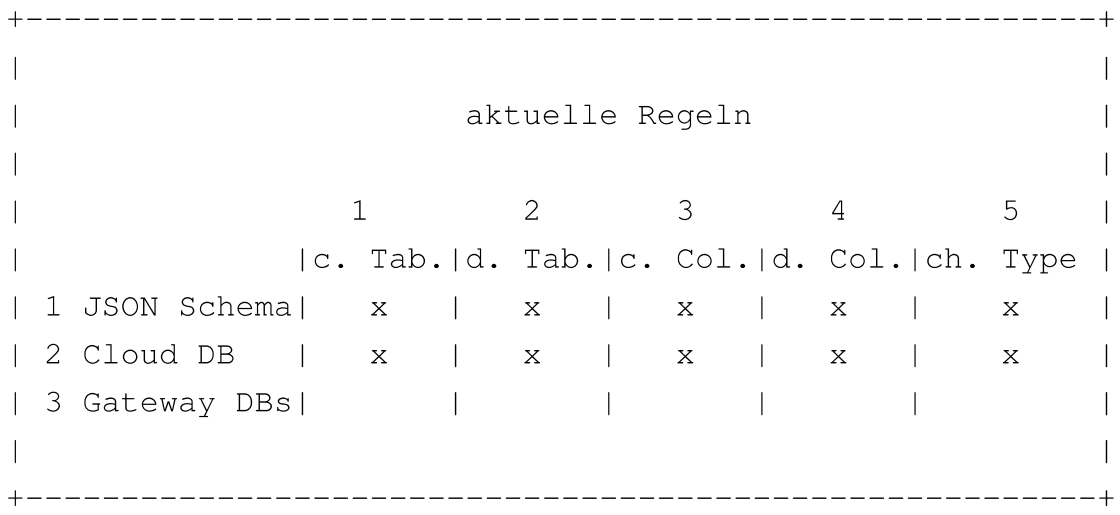

Abbildung 6: Ändern der Regeln des Schema Monitors

Zusätzlich zur GUI ist es auch möglich innerhalb einer Konfigurationsdatei die Regeln zu editieren.

4.1.2 JSON Globales Schema

Über den Punkt „(J)son Schema anpassen“ kann der Pfad zum JSON Globalen Schema gesetzt werden oder auch manuelle Änderungen vorgenommen werden. Die JSON Globale Schema Datei muss dafür im lokalen System vorhanden sein.

4.1.3 Lokale und Cloud Datenbanken

Die Datenbanken werden über die Menüpunkte „(L)ocationMaster verwalten“ und „(C)loud DB anpassen“ ausgewählt. Es können dabei mehrere zu überwachende LocationMaster ausgewählt werden. Die Cloud DB ist dabei auf eine Datenbank beschränkt, was sich aber mit dem Konzept der Cloud Datenbank von SensorCloud deckt (mehrere lokale DB auf mehreren LocationMastern, eine Cloud Datenbank mit mehreren Knoten).

Zusätzlich zur GUI ist es auch möglich die Datenbanken innerhalb einer Konfigurationsdatei zu editieren.

4.2 Schema Überwachung

Über den Punkt „(S)chema vergleichen“ können die Schemata der zuvor ausgewählten Datenbanken miteinander verglichen werden. Hierbei können verschiedene Varianten und Richtungen in die abgeglichen werden soll ausgewählt werden.

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

```
+-----------------------------------------------------------+
|                                                           |
|              Datenbank Schemata vergleichen               |
|                                                           |
|    (V)ergleichen                                          |
|   Vergleichen und Aenderungsskript DB(1) -> DB2 erzeugen  |
|   Vergleichen und Aenderungsskript DB(2) -> DB1 erzeugen  |
|    (z)urueck                                              |
|                                                           |
+-----------------------------------------------------------+
```

Abbildung 7: Vergleichsmöglichkeiten des Schema Monitors

4.2.1 (V)ergleichen

Diese Option vergleicht die Unterschiede der ausgewählten Schemata und zeigt diese auf der Konsole an. Diese Option ist nur zum „Monitoren“ der gewählten Datenbanken gedacht. Es werden keine Änderungsskripte erzeugt oder sogar angewendet. Ein Beispiel eines Schemavergleichs zwischen dem globalen Schema und dem Schema einer lokalen sqlite-DB, das in Form einer Log-Datei vorliegt, ist in Abbildung 8 gegeben

Log

```
Vergleiche JSONSchema und sqlite_01
-----------------------------------------------
Spalten der Tabelle Aktor nur in JSONSchema vorhanden:
AktDatEin - long
AktBez - text
-----------------------------------------------
Spalten der Tabelle Aktor nur in sqlite_01 vorhanden:
AID - text
ABez - text
-----------------------------------------------
Felder mit unterschiedlichen Typen:
JSONSchema:       Sensorprodukt / SenProPLZ / text
sqlite_01:       Sensorprodukt / SenProPLZ / int
-----------------------------------------------
Tabellen die nur in JSONSchema sind:
SensorVerbund
EventMitglieder
```

Abbildung 8: Ausschnitt aus Log-Datei

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

4.2.2 Vergleichen und Änderungsskript erzeugen

Diese Option vergleicht zunächst die ausgewählten Schemata und erzeugt auch direkt ein Änderungsskript um die Schemata auf den gleichen Stand zu bringen. Dabei kann vom Benutzer selbst ausgewählt werden ob er von der Cloud zum LocationMaster oder eben von einem LocationMaster zur Cloud synchronisieren möchte. Die erzeugten Skripte werden dem Benutzer zunächst angezeigt und werden nur nach Bestätigung durch den Benutzer ausgeführt. Zusätzlich werden die Skripte als Skriptdatei im Dateisystem abgelegt so dass diese später oder von einem anderen Dienst aus ausgeführt werden können. Datenbanken die zuvor als „nicht Sicher“ in Hinsicht auf Veränderung des Schemas konfiguriert werden bekommen zunächst bei zu ändernden Entitäten einen Namensanhang „_backup“ so dass der ursprüngliche Entitätsname wieder frei ist.

4.2.3 (B)ackups

Der Menüpunkt „Backups“ erlaubt es dem Benutzer ein Backup der aktuell auf den Datenbanken befindlichen Schemata zu machen. Dies ist meist zu empfehlen bevor man etwaige Änderungen auf der Datenbank ausführt, wenn nicht zuvor eine Gesamtsicherung des Datenbank Knotens vorgenommen wurde.

4.2.4 (A)enderungsskript ausführen

Diese Option kann zuvor erstellte Änderungsskripte ausführen. Dafür muss ein anzuwendendes Skript ausgewählt werden. Hierdurch lassen sich Skripte mit den Optionen „Backups“ oder „Änderungsskript erstellen“ ausführen und in die jeweils ausgewählte Datenbank einspielen.

4.3 Automatisierter Schema Vergleich

Bei einem automatischen Schemavergleich sucht der Schema Monitor nach Unterschieden zwischen dem JSON Globalen Schema und der Cloud Datenbank sowie allen am Schema Monitor registrierten Location Master Datenbanken. Bei nicht übereinstimmenden Schemata werden anhand der vorher konfigurierten Regeln Änderungsskripte erzeugt. Dabei gibt es folgende Optionen:

4.3.1 Email Modus

Option: -b

Durch Starten des Schema Monitors durch den Befehl

java -jar SchemaMonitor.jar -b [email1 email2 ...]

werden die genannten Datenbanken verglichen und die erzeugten Änderungsskripte im Unterordner *Skripte/"timestamp"/* unter dem Namen der jeweiligen Datenbank

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

gespeichert. Zusätzlich wird im Unterordner *logs/* ein Log gespeichert, das zusätzlich an alle angegebenen E-Mail Adressen verschickt wird. An den Datenbanken selbst finden keine Schema Änderungen von Seiten des Schema Monitors statt.

4.3.2 Automatischer Modus

Option: -a

-a Wird der Schema Monitor mit dem Befehl

java -jar SchemaMonitor.jar -a [email1 email2 ...]

aufgerufen, so geschieht alles, was auch bei Aufruf mit dem Argument -b geschieht, zusätzlich werden aber alle Änderungsskripte automatisch auf der jeweiligen Datenbank ausgeführt, um alle vom Schema Monitor vorgeschlagenen Änderungen zu übernehmen.

Wird der Schema Monitor in diesem Modus regelmäßig, zum Beispiel als *Cronjob*, aufgerufen, dient er der kontinuierlichen Überwachung des Datenbankschemas des gesamten föderierten Datenbanksystems innerhalb des SensorCloud Projektes.

5. Erweiterung des SchemaMonitors

5.1 Integritätsbedingungen

In der ersten Version des Schema Monitors war es dem SchemaMonitor nicht möglich mit Integritätsbedingungen umzugehen. Sie wurden weder aus Richtung des JSON Globalen Schemas noch beim Auslesen aus einer Datenbank berücksichtigt. Da die Integritätsbedingungen für bestimmte Anwendungsfälle (Bsp.: Anlegen der Indizes in einer Cassandra Datenbank für DAL Dienste) der SensorCloud relevant sind, besteht ein Interesse daran den SchemaMonitor so zu erweitern, dass er auch mit diesen Informationen umgehen kann. Folgende Integritätsbedingungen müssen zukünftig vom SchemaMonitor unterstützt werden:

- Primary Key (Primärschlüssel)
- Foreign Key (Fremdschlüssel)
- Unique
- NotNull

Um diese Informationen verarbeiten zu können muss der SchemaMonitor an mehreren Stellen erweitert werden:

- Vergleichsschema
- Datenbankschnittstelle

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

- Schema Änderungsskripte

5.1.1 Änderungen am Vergleichsschema

Das Vergleichsschema (vgl. 2.1) muss um die Information der Integritätsbedingung erweitert werden. Diese Erweiterung lässt sich aber nicht trivial an die bisherigen Informationen anhängen ohne auch Veränderungen am Quellcode des SchemaMonitors vorzunehmen. Wie in Abbildung 2 zu sehen ist, ist das Vergleichsformat derzeit so aufgebaut, dass zunächst ein Trennzeichen kommt ($$$), darauf ein Tabellennamen folgt und daraufhin bis zum nächsten Trennzeichen abwechselnd ein Spaltenname und ein Spaltendatentyp folgen.

Eine Möglichkeit wäre die Integritätsbedingungen direkt hinter die Spaltennamen zu setzen, beispielsweise mit Leerzeichen oder anderem Trennzeichen vom Namen getrennt. Dies hätte aber einen größeren Effekt auf den umgebenden Quellcode des SchemaMonitors, da dieser dann an sämtlichen Stellen an denen das Vergleichsschema benutzt wird, mit der neu hinzugefügten Information umgehen lernen muss. Technisch sinnvoller umzusetzen ist deswegen eher die Möglichkeit die Integritätsbedingungen als zusätzliche Zeile in das Vergleichsschema mit einzubauen. Bisher gab es im Wechsel Spaltenname und Datentyp der Spalte. Nun wird in diese Rotation zusätzlich die Integritätsbedingung eingebaut. Im Quellcode des SchemaMonitors muss deshalb nun an sämtlichen Stellen nur beachtet werden, dass eine zu beschreibende Spalte nun aus 3 Zeilen besteht.

$$$
Name der Tabelle
Name der Spalte
Datentyp der Spalte
Integritätsbedingung der Spalte
Name der Spalte
Datentyp der Spalte
Integritätsbedingung der Spalte
…
$$$
Name der Tabelle
…

Abbildung 9: Erweiterung des Vergleichsschemas

Die verlangten Integritätsbedingungen des SchemaMonitors werden mit den folgenden Schlüsselwörtern bzw. Ausdrücken PRIK für Primärschlüssel, FKEY(E.a) [E=Entität, a=Primärschlüsselattribut, auf das der

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Fremdschlüssel bezogen ist] für Fremdschlüssel, UNIQUE für die Unique-Bedingung und NOTNULL für die NotNull-Bedingung in das Vergleichsschema eingetragen (siehe die nachfolgenden Beispiele für die Entität Sensor:

Primary Key (Primärschlüssel):

$$$
Sensor
SenID
text
PRIK
...

Foreign Key (Fremdschlüssel):

$$$
...
SenSenTypID
text
FKEY(SensorTyp.SenTypID)
...

Unique (Einzigartig):

$$$
...
NutStaEma
text
UNIQUE
...

NotNull:

$$$
...
SenBez
text

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

NOTNULL
...

Mehrere Integritätsbedingungen können mit Hilfe eines Semikolons voneinander getrennt werden.

$$$
Sensor
SenID
text
PRIK;UNIQUE;NOTNULL
...

5.1.2 Änderungen der Datenbankschnittstellen

Die Datenbankschnittstellen des SchemaMonitors müssen jeweils für das Lesen und das Schreiben von Schemata geändert werden. Nicht alle Datenbanken unterstützen sämtliche Integritätsbedingungen und auch nicht alle Datenbanken unterstützen dazu das Auslesen von Integritätsbedingungen (vgl. [1]).

Datenbanken, die keine Integritätsbedingungen unterstützen, müssen trotzdem in ihrer Implementierung im SchemaMonitor verändert werden. Für zu schreibende Integritätsbedingungen reicht es aber aus, die im Vergleichsschema hinzugefügte Zeile mit der Integritätsbedingung zu überspringen. Dies kann entweder über ein Einlesen und anschließendem Verwerfen geschehen oder durch das direkte Überspringen der Zeile nach der Spaltendatentyps-Zeile.

Bei den Datenbanken, bei denen entweder keine Integritätsbedingungen im Schema hinterlegt oder nicht ausgelesen werden können, muss beim Erzeugen des Vergleichsschemas die Zeilen für die Integritätsbedingungen leer gelassen werden.

5.1.3 Schema Änderungsskripte

Auch die Schemaänderungsskripte müssen für die Integration von Integritätsbedingungen angepasst werden. Das Schema Änderungsskript wird aus dem Vergleichsschema erzeugt. Je nach verwendeter Datenbank (insbesondere relationale Datenbanken) können nun die Integritätsbedingungen mit erzeugt werden.

Primary Key Änderungsskript am Beispiel für relationale DBMS

Eine als „PRIK“ gekennzeichnete Spalte sollte zu folgendem SQL Änderungsskript führen:

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Im Vergleichsschema: PRIK

ALTER TABLE *tablename* ADD PRIMARY KEY(*columnname*)

Foreign Key Änderungsskript am Beispiel für relationale DBMS

Eine als „FKEY“ gekennzeichnete Spalte sollte zu folgendem SQL Änderungsskript führen:

Im Vergleichsschema: FKEY(refTablename.refColumn)

ALTER TABLE *tablename* ADD FOREIGN KEY(*columname*) REFERENCES *refTablename*(*refColumn*)

Unique Constraint Änderungsskript am Beispiel für relationale DBMS

Eine als „UNIQUE“ gekennzeichnete Spalte sollte zu folgendem SQL Änderungsskript führen:

Im Vergleichsschema: UNIQUE

ALTER TABLE *tablename* ADD UNIQUE(*columnname*)

NotNull Constraint Änderungsskript am Beispiel für relationale DBMS

Eine als „NOTNULL“ gekennzeichnete Spalte sollte zu folgendem SQL Änderungsskript führen:

Im Vergleichsschema: NOTNULL

ALTER TABLE *tablename* ADD NOTNULL(*columnname*)

5.2 Versionierung des JSON Globalen Schemas mit Unterstützung des SchemaMoniors

Bei jeder Schema-Evolution wird entschieden, ob die Änderung des Datenbankschemas physikalisch in den internen Schemata der eingesetzten Datenbanken ausgeführt werden muss, oder ob die Änderung durch eine alleinige Änderung im globalen Schema repräsentiert werden kann (im Folgenden logische Änderung genannt). Schemaänderungen werden über den Schema-Monitor vorgenommen.

Abbildung 10 zeigt die möglichen Operationen bei einer Schemaänderung auf Entitätstypen und Attribute und die durchzuführende Änderung (logisch vs. physikalisch). Ein Löschen eines Entitätstyps oder eines Attributs wird aufgrund der Abwärtskompatibilität zu früheren Schemaversionen nur logisch durchgeführt, wohingegen das Hinzufügen neuer Entitätstypen oder Attribute physikalisch in dem internen Schema des FDBS erfolgt. Die Umbenennung eines Entitätstyps oder eines Attributs wird im JSON globalen Schema logisch vorgenommen und nicht physikalisch im

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

Schema des FDBS. Der generische DAL erkennt an der Versionsnummer die Existenz eines Entitätstyps oder eines Attributs und bildet das externe Schema auf das interne Schema des FDBS ab. Hierdurch entfällt der Aufwand einer Datenmigration in der Anwendung und auf Datenbankebene, da der DAL zur Laufzeit die Migration auf logischer Ebene vornimmt. Anwendungen können so bei einer Schemaänderung ohne Unterbrechung weiterlaufen und gegen die frühere Version des Schemas arbeiten, bis die Anwendung an das neue Schema angepasst wurde. Dies erhöht die Verfügbarkeit und Robustheit des Gesamtsystems aus Anwendungs- und Anwendersicht bei gleichzeitiger Agilität [4].

Operation / Änder-ungstyp	Löschen	Hinzufügen	Umbenennung	Constraintänderung
Physikalisch		X		
Logisch	X		X	X

Abbildung 10: Mögliche Attributoperationen und deren Auswirkung

6. Abschlussbetrachtung

Die aktuelle Version des Schema Monitors mit all seinen Diensten wird auf der FDBS Teststrecke eingesetzt. Er hilft insbesondäere bei Aktualisierung des globalen Schemas als auch bei Aktualisierung der noch ständig in der Weiterentwicklung sich befindenden Cassandra Knoten. Derzeit wird er noch nicht in der Vollautomatisierten Version eingesetzt, sondern wird weitestgehend noch manuell gestartet und Änderungen werden erst nach Interaktion mit einem Benutzer ausgeführt.

Durch die Implementierung einer allgemeinen Datenbankschnittstelle sind Anbindungen an weitere bisher nicht genutzte Datenbankmanagementsysteme möglich und man folgt somit dem allgemeinen Ansatz des FDBS, dass die Komponenten untereinander austauschbar sind und trotzdem der volle Funktionsumfang gewährleistet werden kann.

Literatur

[1] „Deploymentanalyse“, Andreas Lockermann, MS 2 FDBS Bericht, 17.01.2013

[2] „Konzeptionelles Schema“, H. Budde, et Al., MS 4 FDBS Bericht, 15.01.2014

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages

[3] „Entwicklung eines Schema Monitors im Rahmen des SensorCloud Projektes“, Praxissemesterbericht Alexander Klein, FH Köln 2013

[4] “Agile database techniques”, Ambler, S. W. (2003):, United States: Wiley

Versionshistorie

1.0 Erste Version zum Meilenstein 4

2.0 Kapitel 5 in Kapitel 6 verschoben und neues Kapitel 5 eingefügt
- Integritätsbedingungen
- Versionierung des JSON Globalen Schemas mit Unterstützung des SchemaMonitors

Gefördert durch:
Bundesministerium für Wirtschaft und Technologie
aufgrund eines Beschlusses des Deutschen Bundestages