|  |  |
| --- | --- |
| **Arbeitspaket:** | 1.6a - NoSQL als Alternative zur MySQL Datenbank evaluieren |
| AP Inhalte / Ergebnisse: | * Untersuchung einer NoSQL Datenbank in Hinblick auf Performanceengpässe (Daten von 500.000 Smartmeters werden täglich/nächtlich zugleich übertragen) als Alternative zum relationalen Datenmodell. * Vertraut machen mit der NoSQL Datenbank (Konzept, Funktionen, Vorteile und API). * Konvertierung der Dokumente der NoSQL Datenbank in Java Objekte * Prototyp in Java. * Entscheidung, ob SQL/NoSQL als Datenbank eingesetzt wird kann getroffen werden. |
| Verantwortlich: | Max |
| Mitarbeit: | Reimar, Christopher |
| **Abgenommen:** | <Datum, Name> |
| **Änderungen:** | Jänner 2017: Ergänzungen Hadoop. |

.

# Tasks

* „Einlesen“ in NoSQL Datenbanken
* Vergleich von Apache Cassandra, Mongo DB, Couch DB und Hadoop
* Lokale Installation eines NoSQL Datenbanksystems (MongoDB)
* Lokale Installation von Hadoop
* Entwurf der Dokumentenstruktur für die Messdaten (Meterstamm- und Benutzerdaten werden in einer relationalen Datenbank abgelegt).
* Einrichtung Musterdatenbank für Meterdaten
* Musterprogramm für den Zugriff und Verständnis für NoSQL erstellen

# Ergebnis

* Apache Cassandra, Mongo DB und Couch DB sind alle drei darauf ausgelegt, große Datenmengen in Dokumentenstruktur abzulegen. Hadoop stellt die Daten nicht als Dokument zur Verfügung, sondern transformiert die Rohdaten. Die Transformationsfunktion (map/reduce) wird vom Benutzer erstellt. Im Detail untersucht wurden MongoDB und Hadoop.
* Vorteile MongoDB:
  + die Daten werden als Dokument (JSON) abgelegt, die Struktur ist daher nicht von vornherein festgelegt, eine Erweiterung um zusätzliche Attribute ist daher einfach möglich (Vorteil gegenüber der starren Struktur eines RDBMS).
  + Zusätzliche Indizes ermöglichen den performanten Zugriff aus „anderem Blickwinkel“ als die primäre Sortierung (meter\_Id), zum Beispiel eine Gruppe, wie eine Siedlung oder alle Meters eines Besitzers.
* Vorteile Hadoop:
  + Kann riesige Datenmengen verwalten, und auf mehreren Rechnern verteilt laufen.
* Lokale Installation: MongoDB unter Windows. Das Softwarepaket ist unter Linux ebenso verfügbar, des weiteren sind die Client Libraries für Java, C++ und C# verfügbar, dh keine Einschränkung auf Betriebssystem oder Programmiersprache.
* Lokale Installation: Hadoop unter Linux, Import per in Java implementierter Map Funktion, Hauptproblem ist die Reduce Funktion, für das API sind die Daten in der Urform notwendig, da nicht im Vorhinein „gewusst“ werden kann, welche Abfragen gegen die Datenbank gemacht werden wollen, insbesondere die Differenz-Abfrage stellt die Reduce Funktion vor Probleme (welche Daten in welcher Auflösung abzulegen).
* Performance Untersuchungen „Massendaten in kurzer Zeit laden“: Ergebnis ist für den hochauflösenden REDD Datensatz nicht so überragend, als dass die Nachteile durch den Verzicht auf ein RDBMS aufgewogen werden können (Test auf Laptop, Ladeprogramm nicht besonders optimiert)
* Struktur „Meterdaten“ als Entwurf fertiggestellt.
* Musterprogramm für MongoDB Zugriff: grundlegende Funktionen implementiert. Skripts teilweise erstellt.
* Integration relationale Benutzerverwaltung mit NoSQL Meterdaten, Analyse ob handhabbar: wurde nicht mehr durchgeführt, da die Entscheidung zu Gunsten des RDBMS gefällt wurde.

# Weitere Schritte

Keine weiteren Schritte mehr.

Musterdatendatensatz „Meterdaten“

{

"\_id" : ObjectId("58443ae4e649551c0d66ea7e"),

"meterId" : "4568",

"timestamp" : NumberLong("1480866531981"),

“count\_total” : 17890.5,

“count\_ register1” : 6540.2,

“count\_ register2” : 6857.1,

“count\_ register3” : 4493.2,

"power\_p1" : 669,

"power\_p2" : 2532,

"power\_p3" : 3708,

"voltage\_u1" : 223,

"voltage\_u2" : 211,

"voltage\_u3" : 206,

" work\_p1" : 3,

" work\_p2" : 12,

" work\_p3" : 18,

"frequency" : 50.002

}

Der Musterdatensatz ist noch mit dem DLMS/COSEM Objektmodell abzugleichen, und evtl zu erweitern (vgl BlueBook http://dlms.com/documents/Excerpt\_BB12.pdf)

|  |  |
| --- | --- |
| **Arbeitspaket:** | 1.6b - Erweiterungen am bestehenden ERM |
| AP Inhalte / Ergebnisse: | * Analyse des bestehenden Datenmodells * Feststellen, welche Entitäten/Relationen/Attribute fehlen * Erweitern |
| Verantwortlich: | Max |
| Mitarbeit: | Reimar, Christopher |
| **Abgenommen:** | <Datum, Name> |

# Ergebnis

Folgende Erweiterungen erscheinen als notwendig:

* Entität: „Owner“, als derjenige der Messdatenwerte in die Datenbank importiert, als Grundlage für die Rolle „Owner“ um Zugriff in der höchsten (gespeicherten) Auflösung zu erhalten.
* Relation „owner“ abgebildet als Fremdschlüssel zu meter\_data.
* Entität „Gruppe“ als Bündelung von Smartmeter Geräten die zum Beispiel von einem Verteiler versorgt werden.
* Relation „Teil der Gruppe“: m:n Beziehung zwischen „Gruppe“ und „meter\_management“
* Meter\_management: erweitern um die gültigen Spalten von meter\_data in Verbindung mit meter\_data: enthält „alle“ relevanten Werte in je einem Spalte.

# Offene Punkte

Zu „owner“ könnte sich auf Grund der Anforderungen noch eine Änderung ergeben: abhängig vom Import der Messdaten kann diese Verbindung am „meter\_management“ besser aufgehoben sein, abhängig vom „Eigentümer“ der Messdaten: gehören die Messdaten demjenigen der sie in die Datenbank importiert (da er sie ja zweifellos besitzt) oder gehören sie dem Besitzer (customer) des Smart Meter? -> klären.

# Weitere Schritte

* Konsens für SQL erreichen.
* Datenmodell entsprechend erweitern (zusätzliche Entitäten und Relationen, weitere Attribute bei meter\_data).
* Datenbank installieren und alle Tabellen anlegen.